

5-2017

# Virtual Teaching Assistant: A Web Tool (for C++)

Santosh Basnet

St. Cloud State University, [basa0801@stcloudstate.edu](mailto:basa0801@stcloudstate.edu)

Follow this and additional works at: [https://repository.stcloudstate.edu/csit\\_etds](https://repository.stcloudstate.edu/csit_etds)



Part of the [Computer Sciences Commons](#)

---

## Recommended Citation

Basnet, Santosh, "Virtual Teaching Assistant: A Web Tool (for C++)" (2017). *Culminating Projects in Computer Science and Information Technology*. 17.

[https://repository.stcloudstate.edu/csit\\_etds/17](https://repository.stcloudstate.edu/csit_etds/17)

This Thesis is brought to you for free and open access by the Department of Computer Science and Information Technology at theRepository at St. Cloud State. It has been accepted for inclusion in Culminating Projects in Computer Science and Information Technology by an authorized administrator of theRepository at St. Cloud State. For more information, please contact [rswexelbaum@stcloudstate.edu](mailto:rswexelbaum@stcloudstate.edu).

**VIRTUAL TEACHING ASSISTANT:  
A WEB TOOL (FOR C++)**

by

Santosh Basnet

A Thesis

Submitted to the Graduate Faculty of

St. Cloud State University

In Partial Fulfilment of the Requirements

for the Degree of

Master of Science

in Computer Science

March 2017

Thesis Committee:  
Dr. Jie Hu Meichsner  
Dr. Omar Al-Azzam  
Dr. Qingjun Jim Chen

## ABSTRACT

With the advancements in technology and popularity of online education, the need for virtual teaching assistance has suddenly risen. Students prefer to get virtual help from teachers and tutors at their convenience and time. A virtual tutor – web tool is an effective way to meet this requirement, which is convenient for both students and teachers.

The students at St. Cloud State University expect to have more one-on-one human tutoring, which is hard in regular classroom settings. Especially, this expectation seems more serious for students who take CSC1 201 – Computer Science I (C++). To solve this problem, a virtual tutor - web tool has been proposed to help the students. The virtual tutor is emotive, which can help distant students enhance their understanding of C++. Although this is a blueprint which is used to learn CSC1 201 at St. Cloud State University, this can be applied to any other classes. The tool allows the user to use speech recognition to ask questions to the tool which returns useful answers; to work online with others; and to interact with teachers. In addition, a screen sharing option is also offered to allow users to share the work with each other. Additional services are offered.

This tool introduces an easy and better approach to offer extra help to the students in need instead of relying 100% on the traditional approach. In addition, this might reach wider target audiences including senior students who are ready to offer help to the junior students without setting up any traditional face-to-face meetings. This could be a great tool for students with low self-esteem as well.

In this document, overall tool design is explained in details, and the major steps in building the tool are outlined. The steps include motivation, background, problem descriptions, scope, research and analysis, design and implementation and testing carried out to develop the virtual tool – web tool.

## ACKNOWLEDGEMENT

I would first like to thank my thesis committee: Dr. Jie Hu. Meichsner, Dr. Qingjun Jim Chen and Dr. Omar Al-Azzam of the Department of Computer Science at St. Cloud State University. The constructive feedbacks and suggestions were abundant whenever I ran into a trouble spot or had a question about my research or writing. I am thankful for their guidance, constructive criticism and advices during the project.

I would also like to thank all the students who were involved in the initial survey for this research project. Without their enthusiastic cooperation and input, the survey couldn't have been effectively conducted.

I would also like to use this opportunity to express my gratitude to all my lecturers at the St. Cloud State University (SCSU), Minnesota, USA for providing me the best of practical knowledge such that I was able to use them while developing this tool.

I would also like to acknowledge Mr. Muaz Khan whose tutorials and online notes have assisted me successfully completing the project, and I am gratefully indebted to his valuable support to the overall virtual engineer committee in GitHub.

Finally, I must express my exceptionally significant appreciation to my friend Ritu Tamang for furnishing me with constant support and encouragement during my time of study and through the process toward researching, developing, and writing this theory. This achievement would not have been conceivable without her. Thank you.

Author

Santosh Basnet

## Table of Contents

	Page
Abstract .....	2
Acknowledgement .....	3
List of Figures .....	10
List of Tables .....	13
List of Abbreviations .....	15
<b>Chapter 1: Introduction .....</b>	<b>16</b>
1.1.Chapter Overview .....	16
1.2.Motivation.....	16
1.2.1. Popularity of Online Education.....	16
1.2.2. Student’s Issues and Lack of Services to Acknowledge Them .....	17
1.3.Problem Statement .....	17
1.4.Project Scope .....	17
1.5.Main Objectives .....	17
1.6.Approach.....	20
1.6.1. Current Solution.....	20
1.6.2. Proposed Solution .....	21
<b>Chapter 2: Literature Evaluation.....</b>	<b>23</b>
2.1.Chapter Overview .....	23
2.2.System Analysis.....	23
2.3.Similar Tool with Similar Feature Analysis .....	24
2.3.1. Speech Recognition Tools .....	24
2.3.2. Screen Sharing Tools .....	26
2.3.3. Online Communication Tools.....	28
2.4.Similar Tool Analysis .....	30
2.4.1. Treehouse .....	30
2.4.2. Udemy.....	30
2.5.Technical Analysis.....	31

2.5.1. Speech Recognition.....	31
2.5.2. Screen Sharing .....	32
2.5.3. Online Chat Communication .....	33
2.5.4. Other Features .....	34
<b>Chapter 3: Requirements and Specifications .....</b>	<b>35</b>
3.1.Chapter Overview .....	35
3.2.Web Tool User Roles.....	35
3.3.Functional Requirements .....	35
3.4.Non-Functional Requirements .....	37
3.5.Resource Requirements .....	37
3.5.1. Software Requirements .....	37
3.5.2. Hardware Requirements.....	38
3.5.3. Web Server Requirements .....	38
3.5.4. Database Requirements.....	38
3.6.Language Requirements for Building the Tool .....	38
3.6.1. Scripting Language .....	38
3.6.2. Markup Languages.....	38
<b>Chapter 4: System Design .....</b>	<b>39</b>
4.1.Chapter Overview .....	39
4.2.System Architecture Design .....	39
4.2.1. High-level Architecture Design Diagram .....	39
4.3.Site Map.....	41
4.3.1. Student Site Map.....	41
4.3.2. Teacher Site Map .....	42
4.4.Use Case Diagrams .....	42
4.4.1. Overall Complete Use Case Diagram .....	42
4.4.2. Use Case Diagram for Student.....	44
4.4.3. Use Case Diagram for Teacher .....	44
4.4.4. Use Case Diagram for Guest.....	45
4.5.Extracting Interfaces, Controllers and Services Scripts.....	46
4.5.1. Interface Designing .....	46

4.5.2. Extracting Controllers and Services.....	73
4.5.3. Extracting Database Tables and File Storage Necessity .....	80
4.6.Use Cases Realization.....	82
4.6.1. Log in .....	83
4.6.2. Go to FAQ Page.....	86
4.6.3. View Lab Assistant Hours .....	89
4.6.4. Contact Admin or Teacher.....	92
4.6.5. Ask a Question.....	94
4.6.6. View Chapters.....	98
4.6.7. Go to Online Chat .....	100
4.6.8. Start Share Screen .....	109
4.6.9. Join Room .....	113
4.6.10. Manage Members.....	115
4.6.11. Manage Messages .....	121
4.6.12. Manage Lab Assistant Hours .....	126
4.6.13. Manage Content in the Website .....	131
4.6.14. Log Out .....	133
4.6.15. Search Term in the Page .....	134
<b>Chapter 5: System Implementation .....</b>	<b>136</b>
5.1.Chapter Overview .....	136
5.2.Design Decisions .....	136
5.3.Development Environment .....	136
5.3.1. Installation and Configuration of Tools .....	136
5.3.2. Language Environment .....	137
5.4.Code Implementations and Major Source Codes.....	138
5.4.1. Login Feature Implementation.....	138
5.4.2. Ask a Tutor Page Implementation Using Speech Recognition.....	140
5.4.3. Online Chat Implementation.....	143
5.4.4. Status Controller .....	147
5.4.5. Screen Sharing Implementation .....	148
5.4.6. Manage Members Implementation .....	155

5.4.7. Displaying and Modifying Lab Assistant Hours' Table Implementation.....	161
5.4.8. Search Controller (for FAQ and Chapters Page) .....	164
5.4.9. Contact Message Sending Implementation.....	164
5.4.10. Contact Message Management Implementation .....	166
5.4.11. Resume Session Controller .....	169
5.4.12. Additional Services .....	170
<b>Chapter 6: Testing .....</b>	<b>173</b>
6.1. Chapter Overview .....	173
6.2. Unit Testing .....	173
6.2.1. User Verification.....	173
6.2.2. Speech Recognition Activation.....	174
6.2.3. Speech Recognition.....	174
6.2.4. Database Search for Answer .....	175
6.2.5. Change Status.....	177
6.2.6. Send Message in Online Chat .....	177
6.2.7. Receive Messages Sent in Online Chat.....	178
6.2.8. Save Online User Details .....	178
6.2.9. Open Screen Sharing room .....	179
6.2.10. Sending Invitation .....	180
6.2.11. Lab Assistant Hours' Update Verification.....	181
6.2.12. Send Contact Message .....	181
6.2.13. Add a New Member to the System .....	182
6.2.14. Edit a Member to the System.....	183
6.2.15. Delete a Member from the System .....	183
6.2.16. Delete a Contact Message from the System.....	184
6.2.17. Search Content in the Page .....	184
6.3. Integration Testing.....	185
6.3.1. 'Searching the Database' Integrated with 'Speech Recognition' .....	185
6.3.2. 'Send/Receive Messages' Integrated with 'Status Changing Mechanism' .....	186
6.3.3. 'Open Screen Sharing Room' Integrated with 'Send Invitation' .....	187
6.3.4. User Verification Integrated with Save Online User's Details .....	188



6.4.Overall System Testing.....	189
6.4.1. System URL Validation.....	189
6.4.2. Home page – Guest (After launching the System in the Browser).....	190
6.4.3. Log in.....	192
6.4.4. Contact.....	198
6.4.5. Logout.....	201
6.4.6. Home Page – Student.....	202
6.4.7. Ask a Question Page – Student.....	204
6.4.8. Online Chat Page – Student.....	209
6.4.9. Screen Sharing Page – Student.....	217
6.4.10. Chapters Page – Student.....	235
6.4.11. Lab Assistant Hours Page – Student.....	239
6.4.12. FAQ Page – Student.....	240
6.4.13. Contact Page – Student.....	243
6.4.14. Home Page –Teacher.....	244
6.4.15. Ask a Question Page – Teacher.....	245
6.4.16. Online Chat Page – Teacher.....	245
6.4.17. Screen Sharing Page – Teacher.....	245
6.4.18. Chapters Page – Teacher.....	246
6.4.19. Lab Assistant Hours Page – Teacher.....	246
6.4.20. FAQ Page – Teacher.....	247
6.4.21. Manage Page – Teacher.....	248
6.4.22. Message Page – Teacher.....	260
<b>Chapter 7: Deployment.....</b>	<b>264</b>
7.1.Chapter Overview.....	264
7.1.1. Deployment Settings.....	264
<b>Chapter 8: Evaluation.....</b>	<b>265</b>
8.1.Chapter Overview.....	265
8.2.Accomplishments.....	265
8.3.Limitations.....	265
8.4.Future Enhancements.....	266

8.5.Maintenance .....	267
8.6.Conclusion .....	268
References .....	270
Appendix A – Source Codes .....	273

## List of Figures

Figure 1: MS Bing API Speech Recognition trial .....	25
Figure 2: Google Web Speech API Speech Recognition trial .....	26
Figure 3: Google Hangouts Chat UI .....	29
Figure 4: Google Hangouts Video call UI .....	29
Figure 5: High –Level System Architecture Diagram .....	40
Figure 6: Site Map for Student.....	41
Figure 7: Site Map for Teacher .....	42
Figure 8: Overall Use Case diagram .....	43
Figure 9: Student Use Case diagram.....	44
Figure 10: Teacher Use Case diagram .....	45
Figure 11: Guest Use Case diagram.....	46
Figure 12: Home page for Guest.....	47
Figure 13: Home page for Student.....	49
Figure 14: Drop down menu for <i>Ask a Tutor</i> menu.....	51
Figure 15: Drop down menu for <i>Study Materials</i> menu .....	51
Figure 16: Home page for Teacher .....	52
Figure 17: Ask a Question interface .....	54
Figure 18: Mic design when listening to the user .....	55
Figure 19: Online Chat interface: online users box on left, chat messages box on right.....	56
Figure 20: Different status available for users .....	56
Figure 21: Status of message sent as Message sent in circle .....	58
Figure 22: Screen Sharing interface final design (before screen share starts) .....	59
Figure 23: Screen Sharing interface final design (after screen share starts) .....	59
Figure 24: Chapters interface with collapsible titles.....	61
Figure 25: Lab Assistant Hours interface designed for students/guests .....	63
Figure 26: Lab Assistant Hours interface designed for teachers/admins.....	64
Figure 27: FAQ interface with a complete question set .....	65
Figure 28: Manage interface .....	67
Figure 29: Add a new member form.....	68

Figure 30: Edit a member form.....	69
Figure 31: Login screen Interface design.....	70
Figure 32: Log out interface .....	71
Figure 33: Contact interface (Only available for student/guest) .....	72
Figure 34: Message interface for Teachers.....	73
Figure 35: Log in Use Case diagram .....	83
Figure 36: Use login full action flow diagram.....	85
Figure 37: Go to FAQ page Use Case diagram .....	86
Figure 38: Go to FAQ page full action flow diagram.....	88
Figure 39: View lab Assistant hours' page Use Case diagram.....	89
Figure 40: View lab assistant hours' full action flow diagram.....	91
Figure 41: Contact admin or teacher page Use Case diagram .....	92
Figure 42: Ask a Question Full Use Case diagram.....	94
Figure 43: Ask a Question full action flow diagram.....	97
Figure 44: View Chapters Full Use Case diagram.....	98
Figure 45: View Chapters full action flow diagram .....	100
Figure 46: Go to Online Chat Full Use Case diagram .....	100
Figure 47: Go to Online Chat full action flow diagram.....	102
Figure 48: Send Message full action flow diagram .....	105
Figure 49: Change Status full action flow diagram .....	108
Figure 50: Share Screen Full Use Case diagram .....	109
Figure 51: Start Screen Share and invite users full action flow diagram.....	111
Figure 52: Join Room Full Use Case diagram .....	113
Figure 53: Manage Members Full Use Case diagram .....	115
Figure 54: Manage Members full action flow diagram .....	117
Figure 55: Manage Members Full Use Case diagram .....	121
Figure 56: Manage Messages full action flow diagram.....	123
Figure 57: Manage Lab Assistant hours' full Use Case diagram .....	126
Figure 58: Manage Lab Assistant hours' full action flow diagram .....	128
Figure 59: Manage Content on the website full Use Case diagram.....	131
Figure 60: Log Out Full Use Case diagram .....	133

Figure 61: Communication between scripts .....	138
Figure 62: Login mechanism full action flow.....	139
Figure 63: Ask a Tutor full action flow .....	141
Figure 64: Sending a Message full action flow .....	146
Figure 65: Screen Sharing full action flow .....	149
Figure 66: Members Management full action flow .....	156
Figure 67: Lab Assistant hours' management full action flow .....	162
Figure 68: Contact Message sending full action flow .....	165
Figure 69: Manage Contact messages full action flow.....	167

## List of Tables

Table 1: Property matrix for the default Home page (without logging in).....	48
Table 2: Property matrix for the default Home page logged in as Student.....	49
Table 3: Property matrix for the default Home page logged in as Teacher.....	52
Table 4: Property matrix for Ask a Question interface.....	55
Table 5: Property matrix for Online Chat interface.....	57
Table 6: Property matrix for Screen Sharing interface.....	60
Table 7: Property matrix for Chapters' interface.....	62
Table 8: Property matrix for Lab Assistant Hours' interface for students/guests.....	63
Table 9: Property matrix for Lab Assistant Hours' interface for teachers/admins.....	64
Table 10: Property matrix for FAQ interface.....	66
Table 11: Property matrix for Manage interface.....	67
Table 12: Property matrix for Adding a new member form.....	68
Table 13: Property matrix for Editing a member form.....	69
Table 14: Property matrix for Log in interface.....	70
Table 15: Property matrix for Contact interface.....	72
Table 16: Property matrix for Message interface for teacher.....	73
Table 17: Log in Use Case realization table.....	83
Table 18: Go to FAQ page Use Case realization table.....	86
Table 19: View lab Assistant hours' page Use Case realization table.....	89
Table 20: Contact admin or teacher page use case realization table.....	92
Table 21: Ask a Question Use Case realization table.....	94
Table 22: Chapters Use Case realization table.....	98
Table 23: Go to Online Chat Use Case realization table.....	101
Table 24: Send Message (in chat) Use Case realization table.....	103
Table 25: Change Online Status Use Case realization table.....	106
Table 26: Start Screen Sharing Use Case realization table.....	109
Table 27: Invite Users Use Case realization table.....	112
Table 28: Join Room Use Case realization table.....	114
Table 29: Manage Members Use Case realization table.....	115

Table 30: Add Members Use Case realization table.....	118
Table 31: Edit Members Use Case realization table.....	119
Table 32: Delete Members Use Case realization table .....	120
Table 33: Manage Messages Use Case realization table .....	122
Table 34: Read Messages Use Case realization table.....	124
Table 35: Delete Messages Use Case realization table.....	125
Table 36: Manage Lab Assistant Hours Use Case realization table .....	126
Table 37: Add new schedule Use Case realization table .....	129
Table 38: Modify or Delete Scheduled Hours Use Case realization table .....	130
Table 39: Manage Content in the website Use Case realization table .....	132
Table 40: Log Out Use Case realization table .....	133
Table 41: Search term in the page Use Case realization table.....	134

### List of Abbreviations

- API - Application Program Interface
- GUI - Graphical User Interface
- RTC - Real-Time Communication
- MVC - Model View Controller
- VNC - Virtual Network Computing
- SSH - Secure Socket Shell
- RDBMS - Relational Database Management System
- HTTP - Hyper Text Transfer Protocol
- JS - JavaScript
- HTML - HyperText Markup Language
- CSS - Cascading Style Sheets
- URL - Uniform Resource Locator
- SCSU - Saint Cloud State University
- CSCI - Computer Science
- FAQ - Frequently Asked Question



## **Chapter 1: INTRODUCTION**

### **1.1 Chapter Overview**

This chapter summarizes the motivation behind the research and implementation of the project, the main objectives, and the methodology followed to complete the project. This section highlights the need of virtual teaching assistant: a web tool that could potentially enhance the performance of the students.

### **1.2 Motivation**

This section is dedicated to describing the motivations behind the research and implementation of the project.

#### **1.2.1 Popularity of Online Education**

Online communication tools have been increasingly popular in the modern world, and virtual classrooms are becoming common [1][2]. Students can now interact with classmates and faculty virtually and do various tasks of college life, such as turning in homework, discussing classroom activities, or even taking quizzes through the internet [3]. There are many cases where online tools and applications have helped students improve their academic performance. This kind of virtual tools has improved students' learning and has allowed both students and faculty to use time more efficiently.

Of the many popular virtual tools, courses offered on YouTube and applications produced by Udemy are some of the examples which have produced positive results. They are helping students learn different programming languages and use of 3d tools like Unity. For example, Udemy currently claims that it offers around 40,000+ courses, and there are over 10 million + students taking courses in everything from programming to yoga to photography to much more [2]. For a college student, these resources might be helpful to boost grades as well. However, these are the courses which have no link to the courses offered at colleges. The order of the class contents may not synchronize with the class contents at a university. In addition, these courses need a membership to use resources.

Nonetheless, virtual tools have a great deal more to offer students. Use of them can ideally save time. There are chances of more interaction between students, which might not be possible in a regular classroom setting. Therefore, it would be helpful if such kind of tool could be built to help college students with their university.

### **1.2.2 Student's Issues and Lack of Services to Acknowledge Them**

It is common for students to lose attention in a classroom setting. Missing one or more lecture classes may be a bigger issue. In this case, one-on-one human tutoring could be effective, enabling higher success for students. However, providing this service to each student would be an unrealistic goal. Lab assistant hours is another useful option. For example, currently, the Computer Science Department at SCSU offers Lab consulting hours for those who need help with assignments as well as labs with CSCI 201 class. Using this service is proving to be helpful to students. However, not all students are able to benefit from this service. For a working student, the lab hours may not be the best time. Also, some students may have other classes during the Lab consulting hours. Another huge issue for some students is low self-esteem, which may hinder one-on-one interaction with lab assistants or even with a professor.

### **1.3 Problem Statement**

To overcome various issues mentioned in section 1.2.2 and to add additional support to lab assistants and professors for CSCI 201 course, a tool is proposed to develop as a teaching assistant where students can ask questions on C++. It would also be able to answer all the major topics and related questions listed in the course contents. Students will have access to this tool at any time and at their convenience, potentially being helpful to all students.

Besides students, teachers would also benefit from this system. Online screen sharing and virtual communication (online chat service) could be done to help students virtually and even organize the class in need. The teacher can even post useful materials for students in the web tool.

This tool, however, would not answer unrelated queries (strictly to CSCI 201 class), and it is not meant to do students' homework assignments. The tool's only purpose is to help students with some examples/samples and videos related to the labs and provide services such as voice-driven technology to answer the related queries and provide screen sharing options to share knowledge with other students or with teachers.

#### **1.4 Project Scope**

The proposed solution is limited to CSCI 201 class offered at Saint Cloud State University (SCSU). This web tool has its own user interfaces for different actions such as asking a question using voice recognition or by typing, screen sharing, looking at lab assistant schedule, etc. The scope of web tool will be limited to following aspects:

- The web tool for students who are taking CSCI 201 (C++) class.
- Speech recognition recognizes voice input in English from the user when microphone is allowed.
- Screen sharing when microphone (for speech) and camera (if video cam is to be shared) are enabled.
  - In order to use screen sharing, an extension is to be downloaded which is currently available only for Google Chrome and Firefox. However, an extension is not needed to view a shared screen.
- Online communication (chat service) when logged in.
- Platform: Desktop or Laptop computers (for optimum performance). However, other devices are also useful to surf the web tool but may not give full service.
- Users of the tool: students and teachers (included tutors).

#### **1.5 Main Objectives**

The main objective to be accomplished is to build a web-tool (website) to assist CSCI 201 students to understand the basic concepts of C++, which will be an effort to boost the academic performance of students in the class. To meet this objective, described below are the sub-tasks that need to be completed:

- Development of a website interface that allows students and teachers to use different functions offered by the tool.

The proposed tool will have its own user interface which is slightly different for two kinds of users: students and teachers. The user should be able to easily navigate around and make use of all the functions offered. The functions offered are limited for this project, and each will have its own interface which will be discussed more in detail later.

- Log in system

The tool will be able to offer login system for two different kinds of users: student and teachers. Each will have their own interface to navigate around the tool. Only partial services are offered if the user is not logged in.

- Speech recognition.

The tool will be able to offer the speech recognition system to be able to ask questions and provide the answers. The system will be able to successfully recognize the question asked by the user when enabled and perform the function accordingly. The user can speak in a natural way, which should not be mandatory and only be enabled when the user wants to make use of it.

- Screen sharing

The tool will offer screen sharing options wherein the user will be able to organize their own room and invite other users to the room. As an audience, the user will be able to join the room and attend the meeting without any issues. While screen sharing, the user should have control over audio/video and only enabled when the user wants to make use of it.

- Online communication – chat service

The tool will offer online chat service. Group communication will be available where students can share the confusion with each other. Students will be able to

help each other in need. All the available students and teachers will be listed and can start communication. A separate interface will be available solely for this purpose.

- Real-time modifiable table for lab assistant hours  
The tool will offer modifiable table to update lab assistant hours in the tool for users logged in as a teacher.
- Contents of chapters, FAQ, and contact.  
The tool will list chapter and sections required for the class. FAQ section will include all the important C++ FAQs. A quick searching mechanism will be able to search through the page making it easier for the user to navigate through the chapters and FAQs. Contact page will allow the user to connect with teacher/admin of the tool.
- User-friendly  
The tool will be designed such that all the implementation complexities are hidden from the user and will be very easy to navigate through the tool to make use of all the important functionalities.

## **1.6 Approach**

### **1.6.1 Current Solution**

As mentioned earlier in section 1.2.1, there are multiple online tools and websites that are offering similar kind of services to the students. YouTube lectures, Udemy, TreeHouse are some of the examples. But, services like answering machine using speech recognition, online communication tools and screen sharing may not be available in a single tool.

Yet, there are various other tools, software applications, and websites that already support speech recognition. One such example is Google [4], which supports speech recognition for

searching. The goal of speech recognition could be quite different in other cases depending on their primary goals.

Similarly, there are many other software applications currently available that offer screen sharing. TeamViewer, Skype, Viber, WebEx are some of the examples. These services, however, are only focused on the screen sharing aspect and may be only paid service. Likewise, there are tools that offer online chat services but are only focused on online chatting which is only a part of this project.

However, using these individual tools for some individual features may come up with some extra costs. And so far, there are not any tools that are available with these many features combined together and targeted only to CSCI 201 students at SCSU. Currently, SCSU has D2L system to help students in many ways like accessing lecture notes, assignments, and other things but D2L still lacks the features mentioned in this project.

### **1.6.2 Proposed Solution**

Using different tools and software applications for different services might be painful, time-consuming and expensive as well. The solution to this problem may be building a web tool that supports all services mentioned above. Having a web tool with separated interfaces for each functionality for CSCI 201 students and teachers could be time-saving, easier and manageable.

So the ultimate solution is to build a website (also referred as web-tool) that would be accessible to registered students at any given time. The web-tool will contain detailed information on different topics in C++. In addition, each topic will be linked to the helpful video(s) and the related lecture. This tool will virtually provide helpful answers to student's questions, which could also be done using speech recognition technology.

As mentioned above, web-tool will also be facilitated with a voice recognition system where a student can verbally ask questions. Students can also directly go to their desired content in

a new window, which will have a detailed description of the chosen content along with the video demonstration of how that particular topic in C++ works. A separate page for the lab assistant will also be available in the web tool which is editable for teachers.

The tool will also be facilitated with screen sharing option where teachers/students can have a separate room and start sharing the screen. A separate window will be available for the screen sharing. It can even be possible to organize a small class with organizer sharing the screen from his endpoint.

The tool will also provide an online chatting service where students can ask questions to an available tutor/lab assistant or other students. This can help students get the lab assistant's service from anywhere during lab assistants' hours. This can highly improve the performance of students because they will now have the flexibility to ask questions and gain optimum help virtually. This can also be a great tool for those students who do not have enough confidence and are fearful of raising questions when they have no clear concepts about contents.

## Chapter 2: LITERATURE EVALUATION

### 2.1 Chapter Overview

This chapter is more focused on the literature review of the project dividing into three main categories: system analysis, feature analysis and technical analysis.

### 2.2 System Analysis

System analysis plays a vital role in identifying and gathering the system requirements. This is needed to successfully know the exact needs of the users and functionalities of the system.

The main source of gathering the requirements for this system was through the survey, brainstorming, observation, user's suggestions, and document analysis. The area of research started with brainstorming a potential idea of making web tool to help the new students with C++. The brainstorming was followed by a small survey which showed the positive results to start with the project. In addition, observation from teachers and lab assistants suggesting struggle of students helped in gathering the major requirements. The suggestions from the thesis committee to include features like online communication tool and screen sharing also helped in determining some of the major requirements of the project. Finally, document analysis was done reviewing different projects which offered similar features and functionalities which helped gathered and finalized more requirement for this project.

### Survey Analysis

- a. How often do you lose concentration in class?
  - Never: 10%
  - Sometimes: 40%
  - Most of the time: 50%
  
- b. What is the primary reason behind that?
  - Missing class: 25%
  - Unable to understand the concept: 40%
  - Being late to the class and not enough practice after classes: 20%



- Not applicable: 15%
- c. Do you think a tool (software or online) can help you with the class in order to help you understand the basic concepts of CSCI 201 class?
- Yes: 73%
  - No: -
  - Maybe: 27%

### **2.3 Feature Analysis**

Different features are offered by different tools. It is important to do the analysis of existing tools that offer similar features that are intended to be included in this project. For example, there are many applications that offer screen sharing but do not necessarily offers other services that this project is offering. Analysis of those projects is necessary to know the major motives behind building the project, identifying the best practices and features offered by the project. Studying of similar kind of tools offering similar features also helps in suggesting a better and fastest path in starting the project. Furthermore, finding flaws in those projects will help in building the project with new technology free of those flaws. This will ultimately help in building a more effective product with enhancements.

In correlation to this project, only some of the major features have been analyzed such as speech recognition, online communication, and screen sharing.

#### **2.3.1 Speech Recognition Tools**

This section is dedicated to quickly analyze some popular speech recognition tools to know important aspects while integrating the similar tools in the project.

##### **2.3.1.1 Microsoft Bing Speech API**

Microsoft has developed Bing Speech API that provides the service of speech recognition converting spoken audio to text. The API can recognize audio and displays the converted text in real-time. Once the audio is recognized, it is sent to the server, and the result is returned

and displayed [6]. Microsoft has provided a speech recognition demonstration page that exhibits the power of the tool (<https://www.microsoft.com/cognitive-services/en-us/speech-api>) [7]. A button with the microphone image, on-click, triggers the action of recording the voice which is then sent to the server for converting and displaying the result in the text format as shown in Figure 1.

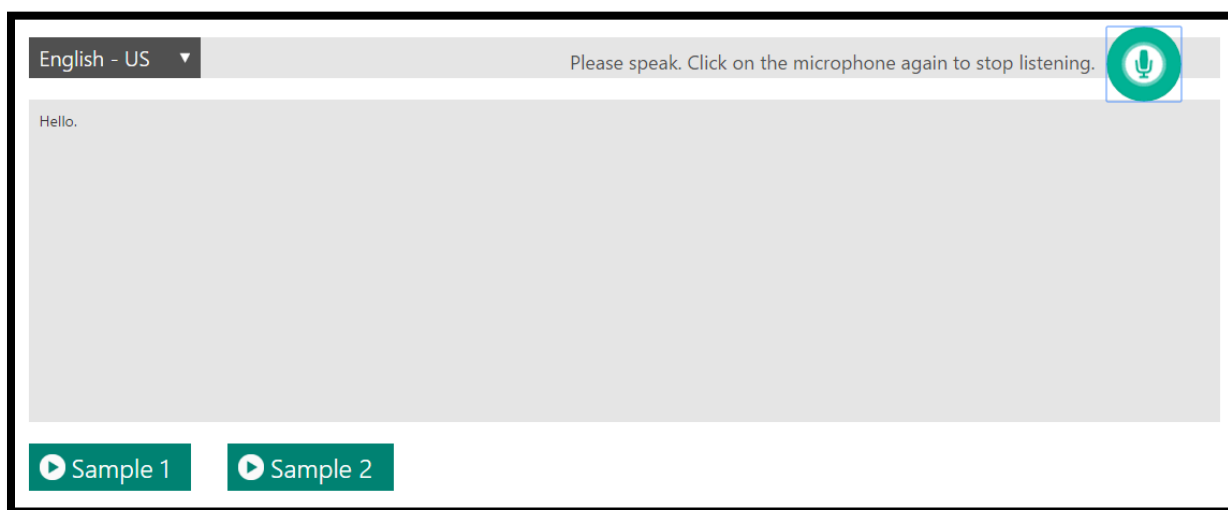


Figure 1: MS Bing API Speech Recognition trial (adapted from [7])

However, this feature is not free to integrate and very complicated. Hence, this API may not be of much help to this project. It is built for the commercial purpose. Nonetheless, this gives an idea of how the feature can be designed and built. This could be a great help in starting to design and write the speech recognition tool for the project.

### 2.3.1.2 Google Web Speech API

This API developed by Google provides the service of speech recognition capability in latest google chrome versions. This is new web speech API which makes easier for a developer to add speech recognition to the web page. This API claims to recognize the text as soon as the user speaks and displays the recognized text [4].

Google has provided a sample web speech API demonstration page (<https://www.google.com/intl/en/chrome/demos/speech.html>) [5] with simple user interface to demonstrate the feature. The user can use the web speech feature by clicking on the ‘microphone icon’ which appears on the right top end of the textbox as shown in Figure 2. Once the icon is clicked, speech recognizer activates which starts capturing audio. The final audio is then sent to servers for transcription, and transcribed text is displayed in the text box. It also can detect user input in multiple languages.

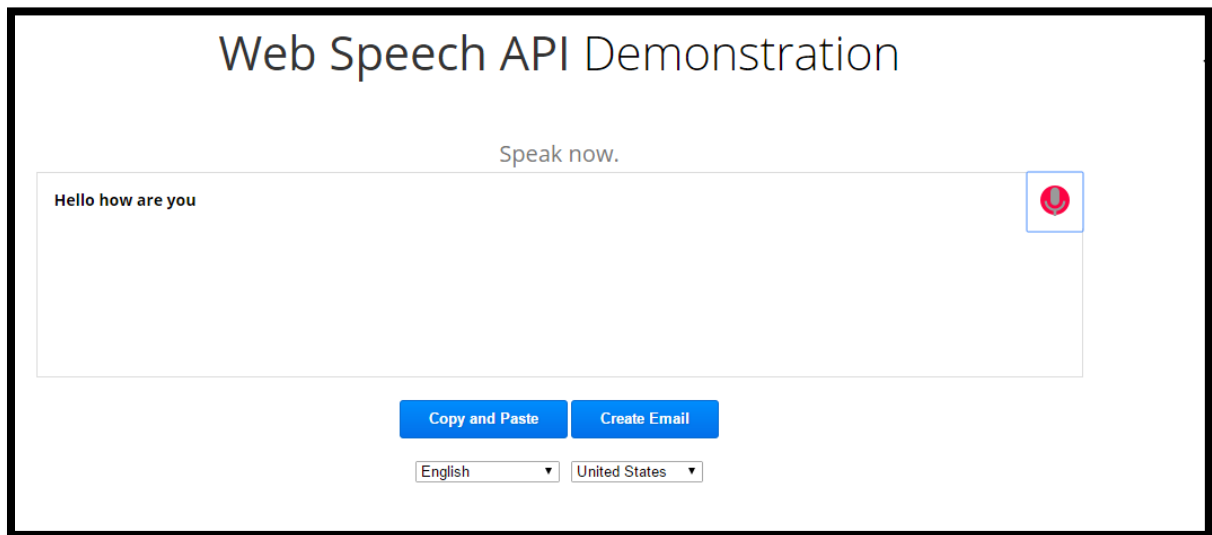


Figure 2: Google Web Speech API Speech Recognition trial (adapted from [5])

This feature seems to fit the speech recognition requirement of the project which is one of the major components. Based on the UI requirement and technical complexity, implementation of similar feature can be a good candidate to be included in this project. Especially, the demo source code is available to use, and this could give a great head start on the project.

### 2.3.2 Screen Sharing Tools

This section is dedicated to analyzing some widely-used screen sharing tools. The features that these tools offer could be very important. They may be the ‘must have’ feature to include in the project while integrating similar screen sharing tools in the project.

### **2.3.2.1 Show My PC API [8]**

This API is developed by Show My PC group. It combines Virtual Network Computing (VNC) remote access technology with an open-source Secure Socket Shell (SSH) forwarding client to provide screen sharing option. The main goal of this API is to support the integration of screen sharing into any kinds of custom applications or websites. In addition, this API also supports following features:

- a. System can send alerts to the users for a session by email or phone with no password exchange required.
- b. If the password is to be used, system can generate a password with a specific length.
- c. Much like team viewer application, this API also supports the auto start of the application with a predefined password.

This API claims to be one of the excellent providers of screen sharing solutions. For this project, similar kind of API can be built and integrated into the main project. However, to use this service, the user needs to download and install the application. Integrating this tool to this project could also be a solution but, this API is not free to use and very complicated. Nonetheless, this API gives an idea regarding interface designing and on what features the screen sharing can offer.

### **2.3.2.2 screen leap API [9]**

Screenleap is a very popular screen sharing tool which claims to be the fastest and simplest way to share screen using any kinds of web-enabled devices. This offers services such that no additional software must be downloaded. Screenleap also offers the service of chrome extension. This extension can be installed in the web browser and the user can start sharing screen. Although this tool is popular and offers a variety of services, it is very complicated and may take a longer period to build similar tools and requiring more resources. This tool, however, can give the basic idea of how screen sharing can be built, both from UI and tool's features perspective.

### **2.3.3 Online Communication Tools**

There are various online chatting tools available to make the online communication better, especially for team projects. This section is dedicated to analyzing some similar tools so that some important features can be learned and possibly help in integrating to the project depending on the requirements.

#### **2.3.3.1 Google Chat/Hangouts**

Google hangouts as shown in Figure 3 and Figure 4, have become very popular in terms of online chatting and even in screen sharing/video calling aspect. This includes instant messaging, video chat, SMS and VOIP features [10]. In addition, this is free to use. The only criteria to use this service is to have a Gmail account. The tool, however, is heavy and complex. The amount of manpower and technology used in the project is huge. Even though web tool project is targeting to have similar feature as offered by Google hangouts, same amount of manpower and technology is not available. Hence, integrating similar feature is not feasible in web tool project. However, it is possible to have a simpler version of the feature which does not require more manpower and is economical too.

In conclusion, a similar feature is good to have in the web tool project but with much simplicity. Nonetheless, Google hangout can be a good reference when the similar feature is to be integrated.

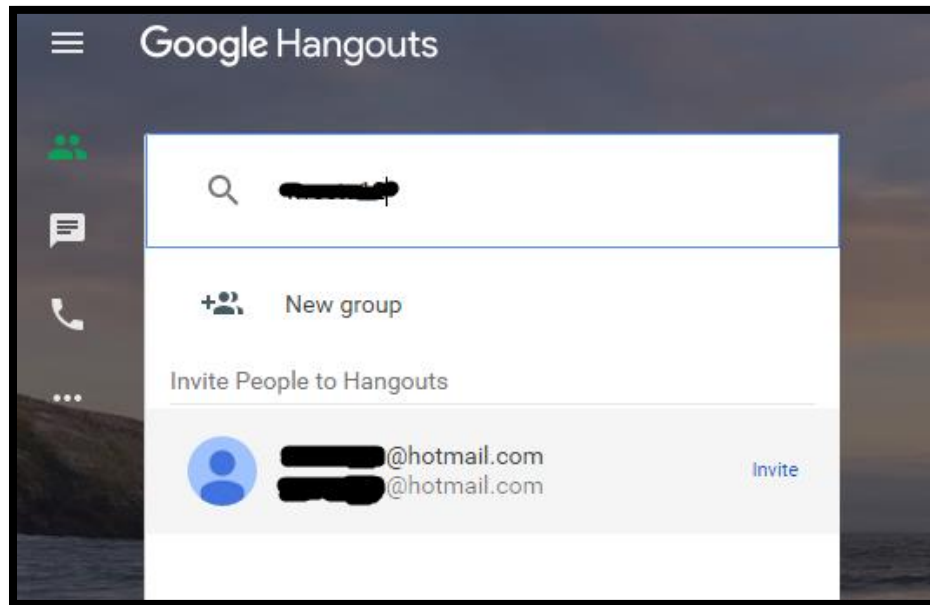


Figure 3: Google Hangouts Chat UI – Google Inc., n.d

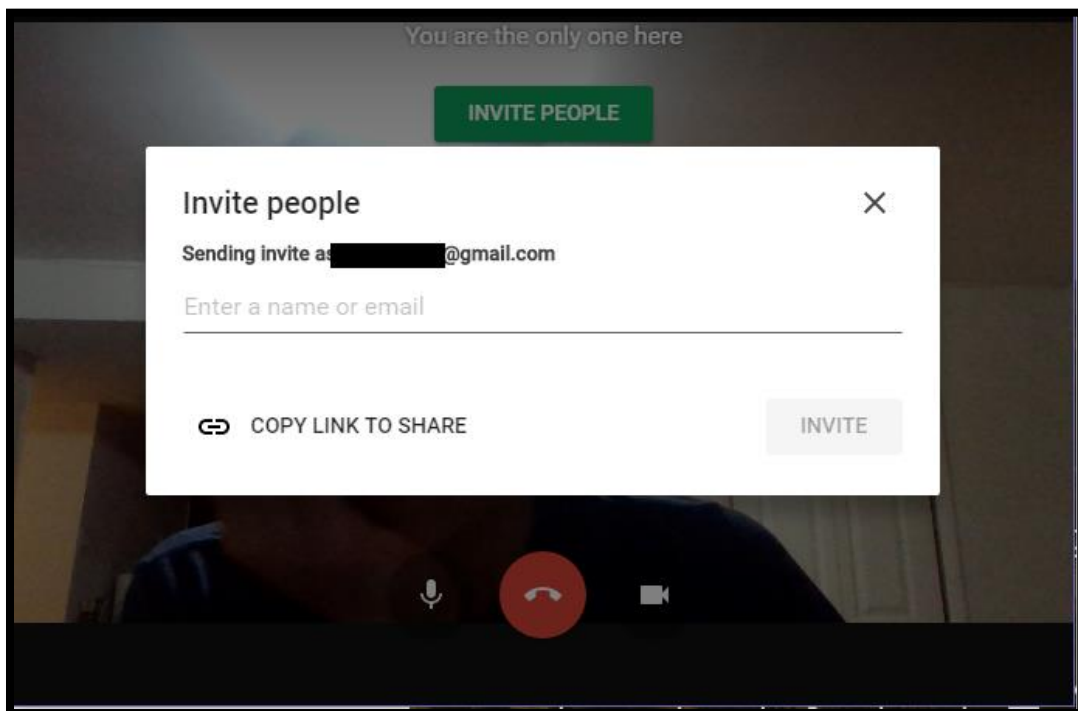


Figure 4: Google Hangouts Video call UI – Google Inc., n.d

### **2.3.3.2 HipChat [11]**

HipChat is another popular tool offering both online communication and screen sharing features. HipChat also offers free services but is only limited to messaging services. This is designed for more security communication targeted toward IT teams. The Hipchat team says that it is persistent, searchable, and loaded with goodies such as group chat, video chat, and screen sharing. However, all the services offered by this tool is not free to use. Hence, this tool may not be of great interest to this project. This tool, however, like Google Hangouts, can be a great reference since this tool offers the similar features the project is targeting to integrate.

## **2.4 Similar Tool Analysis**

Besides the important features mentioned in section 2.3, there are tools that offer the virtual tutoring services such as Treehouse and Udemy. Similar tools can be analyzed to identify and gather the best practices and requirements that could be crucial for this project. Features offered by these tools can be studied and may be considered to involve in the current project.

### **2.4.1 Treehouse [1]**

Treehouse is one of the famous online tools that is getting very popular among students. This tool contains a numerous number of tutorial videos that cover many topics and areas from web design, coding, business to much more. The tool also offers different quizzes and interactive code challenges. These features can be very critical for students. However, these services are only available for subscribed users and not free to the user for all. For the project, integrating tutorial videos can be an important aspect like in Treehouse.

### **2.4.2 Udemy [2]**

Udemy is another online tool that offers a variety of services virtually to the students. This tool, similar to Treehouse offers tutorial videos and exercises related to different topics. In addition, the tool offers different certification course packages. However, the features are only available for subscribed users. Even though these services are very different and seem to

be out of scope for this project, most certainly, these features can be considered for future enhancements.

Even though features listed in Treehouse and Udemy are important features to have, they are unlikely to be included in the project. The goal of the project is to target students taking CSCI 201 and achieve success in the class. Services offered in Treehouse and Udemy seems to be targeted to self-learners who are interested in certain kind of study packages. Therefore, all the features listed in these tools may not be considered to include in this project.

## **2.5 Technical Analysis**

Features analysis has been done studying different tools offering major features. However, there are many technical decisions to be made before the implementation of features like speech recognition, online communication, and screen sharing.

### **2.5.1 Speech Recognition**

After feature analysis, it is clear that speech recognition can offer different kinds of services to the users. However, it is very important to have a deep technical understanding of how to implement this feature.

There are libraries and APIs available which can offer the functionality of the speech recognition. One of such libraries is Speech Recognition API, which is getting very popular since speech recognition software's are getting more and more important and becoming major integral part in every software. The HTML Speech Recognition API allows JavaScript to have access to a browser's audio stream with the user's permission and finally converts them to the text [13]. Speech recognition can be implemented either continuous or discrete way [12].

1. Discrete speech recognition is done when the user speaks slowly, and after each word, there is some pause. This is relatively easy implementation approach since the



system is able to recognize each word at a time. However, from user's point of view, users should speak such that there is a pause after each word.

2. Continuous speech recognition allows the user to speak normally and the result is displayed when the user stops talking. It lets the user fluently dictate words without any pause between words/sentences. This, in turn, makes the implementation somewhat complex since the user can speak at any pace and there are multiple words/sentences to be recognized.

For this project, the speech recognition feature can be used extensively and therefore continuous recognition is needed. Given the fact that user will be asking continuous question, a continuous speech recognition is necessary. Discrete speech recognition can consume some time in recognizing the user's input which can be a negative aspect of the tool. Below is the sample code implementation of continuous speech recognition using HTML Speech Recognition API [13].

```
var recognition = new webkitSpeechRecognition();
recognition.continuous = true;
recognition.onresult = function(event) {
    console.log(event)
}
recognition.start();
```

### 2.5.2 Screen Sharing

Several libraries are available, which can be used when implementing the screen sharing. This is the most complex feature that is to be implemented in the system. More technical analysis and decisions are to be made before the implementation of this feature.

As mentioned in section 2.3.2.2, Screenleap [9] is one of the available tools that offer the services of screen sharing. Screenleap offers the service of chrome extension which once it is installed, it allows the user to start sharing screen using any web browser. A similar approach can be implemented in this project. The goal is to use a web browser to start sharing the screen. An extension can be developed for web browsers and make them available in

extension store. This lets the user to access the extension easily in extension store and not make them download and install the standalone tool in the system which can be considered serious security threat.

To implement this feature, a widely available open sourced RTCMulticonnection, which is a WebRTC library, can be considered. This library in JavaScript allows sharing both screen and audio with the help of a signaling server and an extension [14][15]. This tool allows to capture the desktop windows, but given that other end user in the screen, sharing session is also using the WebRTC capable browser. This library is easy to implement and need to be considered to implement screen sharing portions of the tool.

### **2.5.3 Online Chat Communication**

Like speech recognition, it is clear from the feature analysis on kinds of services that are needed to be offered when online chat communication is implemented. During online chat communication, multiple users are involved, and responses are live to each user. It is very necessary to understand the involvement of web servers in processing the messages such that it is visible to each user using the system. Implementing this feature, there are multiple decisions to be made.

A server should be created which not only processes each sent messages but also stores the messages for future purposes. All the sent messages can be saved in the file on the server which can be retrieved whenever needed. A good interface design is needed, and code implementation must be done such that there is good and fast communication between the interfaces and the web servers processing the message. To meet this purpose, PHP or Perl can be very helpful which can run the code in the server generating necessary HTML which is eventually sent back to clients. A thorough understanding of PHP or Perl is therefore very necessary since a lot of features in the tool will be communicating with servers. [17]

However, unlike in Perl or even C, PHP requires much less amount of codes and commands to execute similar functions. A PHP page can contain HTML with embedded code to perform

server side jobs. The similar job can be very complex and time-consuming when written in Perl or even in C. Hence, for this project, PHP can be very important in implementing almost all the service side coding and implementation. Especially this can be very crucial in implementing the online chat communication which requires a lot of file opening, writing, saving and closing.

A sample PHP page implementation is shown below [18].

```
<!DOCTYPE HTML>
<html>
  <head>
    <title>Sample</title>
  </head>
  <body>
    <?php
      echo "Hello World !";
    ?>
  </body>
</html>
```

Sample PHP code for file content reading, opening, and writing [19].

```
// Reading the file content
$fileContent = file($filename);

// Opening a file for writing
$fp = fopen($filename, "w+");

// Writing the file
fwrite($fp, "Some text");
```

#### 2.5.4 Other Features

To implement and integrate most of the features into the web tool, it requires the knowledge of HTML, CSS, PHP, JavaScript including Bootstrap and Angular JS framework. jQuery is another library which is hugely used in implementing different features in the web tool.

Client-side implementation could be done by using HTML, CSS, and JavaScript, and server-side implementation could be handled by PHP.

## Chapter 3: REQUIREMENTS AND SPECIFICATIONS

### 3.1 Chapter Overview

A full description of the requirements, intended purpose and the specifications for the web-tool to develop are described in detail in this chapter. Other topic such as what services the web-tool will offer, how it will interact with the users, and how it will be expected to perform will be discussed in this chapter as well.

### 3.2 Web Tool User

There are two types of web tool users.

- a. Admin/Teacher: They will be responsible for updates, management of contents and other users.
- b. Students: They are the actual user of the website's functionalities. Invalid users do not have a valid user-id and/or password. They will not be able to use major services offered by the website such as online chat service, speech recognition equipped searching mechanism and screen sharing.

### 3.3 Functional Requirements

Since there are two kinds of users who will be using the tool, there will be some similar and some different interfaces depending on the type of user.

- a. First, a logging in mechanism should be integrated so that tool can only be used by dedicated users and to differentiate the kind of users. This should be Login page which should ask for user's credential to log into the system.  
Once logged in, for both **Student** and **Teacher**, the website should provide the following major functionalities with their own interfaces.
- b. A *home* page describing the tool with the main menu bar that will help in navigating around the tool.
- c. *Ask a question* page that should allow the user to ask a course related question and should be able to answer with related description and videos. The user should be allowed to do this action by either manually typing question or using speech recognition technology.

- d. *Ask a Tutor* page that should allow the user to use two main services: online chat and screen sharing. Each will have their own interfaces.
  - i. *Online Chat* page should allow student/teacher to post the queries and see all the available online users. This will be a public chat room where everyone should be able to see the communication and be able to react to them.
  - ii. *Screen Sharing* page should allow the user to start/open screen sharing and be able to invite others to the room. Once a room is opened, the user should be able to share the unique room URL to others. Screen sharing should enable both audio/video as per user's requirement.
- e. *A Study Materials* page should offer following three main functionalities:
  - i. *A Chapters* section should list all the major chapters with necessary notes and description. This can also include related videos.
  - ii. *A Lab Assistant Hours* page should allow students to look at the latest lab assistant hours' schedule for the week. This should have its own interface, and they should be modifiable for Teachers but read-only for students.
  - iii. *A FAQ* page should contain all the major frequently asked questions in C++ (more specifically related to CSCI 201).  
*Chapter* and *FAQ* pages should also have a search mechanism that should allow the user to find the user's interesting content in the page quickly.
- f. *A Logout* page similar to a login page should be available for the user to log out of the system.

Only for the user logged in as Teacher, following additional features should be offered by web tool.

- g. *A Manage* page that should allow admin/teacher to manage the users of the tool. This page should allow admin/teacher to add new users and delete/edit the existing users.
- h. *A Message* page should give admin/teacher ability to check the messages sent by other users of the tool with concerns and queries. This should provide an option to manage the messages as needed.

Only for the user logged in as Student, following feature should be available.

- i. A *Contact* us menu that should allow the user to notify the admin whenever needed with concerns and queries related to the tool.

### **3.4 Non-Functional Requirements**

#### a. Availability and Reliability

The tool should be available to students almost all the time. The features should also be accurate and reliable when giving the search result. Logging mechanism should be error free as well. In terms of speech recognition, it should accurately take the user input and produce the right results. In the case of online chat, the tool should have good response time and offer seamless communication.

#### b. Security

The overall session in the website should be secured especially since screen sharing is offered.

#### c. Maintainability

The tool should be designed such that maintenance should be easy. The tool should be written such that it is consistent throughout the tool. Simplicity, conciseness, self-descriptiveness and modularity should be maintained to support maintainability.

#### d. Scalability

The tool should be built such that new features could be added in future when needed.

#### e. Usability

The tool should be easy to use for all kinds of user roles. The tool should be made user-friendly such that user should be able to locate and easily perform all the functionalities.

### **3.5 Resource Requirements**

#### **3.5.1 Software Requirement**

- Sublime Text
- Google Chrome - Inspector
- WinSCP - For secure file transfer between local and remote computer

### **3.5.2 Hardware Requirement**

- Windows Desktop Machine or Laptop equipped with microphone and camera
- Linux Operating System

### **3.5.3 Web Server Requirement**

- Apache HTTP Server
- XAMPP - For initial design and testing

### **3.5.4 Database Requirement**

- MySQL Relational Database Management System (RDBMS)

## **3.6 Language Requirements for Building the Tool**

### **3.6.1 Scripting Languages**

- JavaScript
  - AngularJS - JavaScript-based front-end web application framework
  - JQuery - JavaScript library
- PHP - Server-side scripting language for web development

### **3.6.2 Markup Languages**

- HTML
- CSS
- Bootstrap - HTML, CSS and JavaScript-based front-end web application framework for developing responsive design

## Chapter 4: SYSTEM DESIGN

### 4.1 Chapter Overview

In this chapter, the overall designing phase of the web tool, which includes the architectural designing, flowcharts, use cases and diagrams, will be described.

### 4.2 System Architecture Design

The software design pattern used to develop the virtual assistant web tool is 'MVC' design pattern. MVC pattern divides the whole system into three main categories as Models, Views, and Controllers [20]. However, in this project, Controller contains two main parts as '**controller**' to control the actions and '**services**' which perform the services triggered by the controller.

#### 4.2.1 High-level Architecture Design Diagram

There are two kinds of primary user of the tools. The only input to the tool is from the primary users. First, Views will help to gather the inputs from the user. Then, Controllers will process the input and modify the Models as needed to reflect the results of actions. The result is then displayed by the Views which is the only interface between the system and the users.

On a high level, tools offer a variety of services to the users that include:

- Asking a question to the web tool (either using a voice recognition or not)
- Asking the tutor using online chat services or screen sharing
- Look at chapters, notes and FAQ sections
- Look at the lab assistant hour schedule
- Managing the users
- Sending offline messages to admin.

Each of the listed actions goes through series of steps to get the desired result. First, the user tries to perform an action and depending on action type; the action is either processed in client side or sent to the server side. If the process request is sent to the server side, the server



performs the necessary service that may involve the database transactions and send the result back to the client side. Views in client side then display the result.

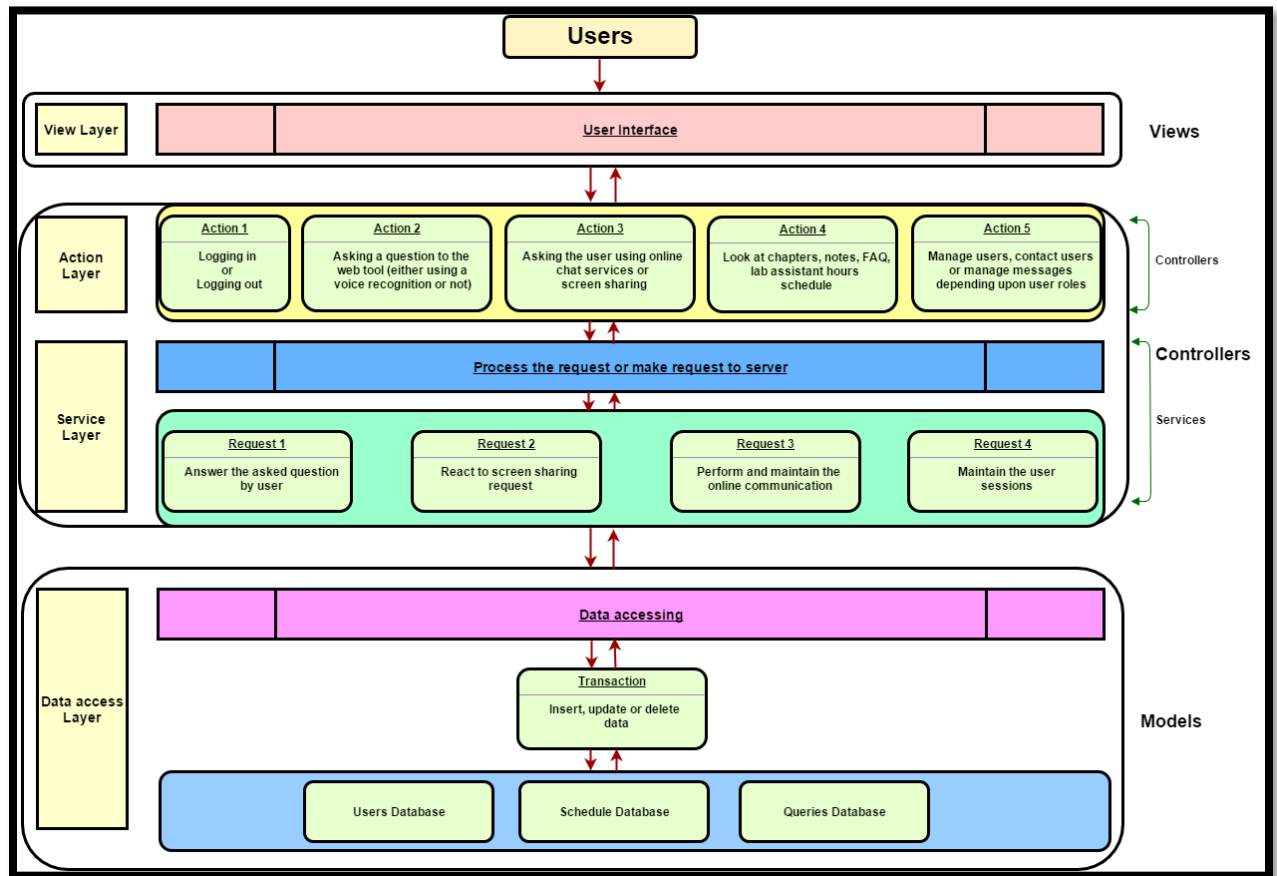


Figure 5: High –Level System Architecture Diagram

As mentioned earlier, there are three major categories; Models, Controllers, and Views used to design overall system. All the interfaces and web pages that users see are included under Views. The user gives the input to the system through views which are recorded using Models and Controller decides the course of actions to perform. The modified models are returned after the actions are processed by the controller which are then finally displayed in the Views to the users.

Notice in Figure 5, categories are represented in different layers. **View Layer** contains all the UI page design of the project. Controllers contain two major layers as **Action Layer** and **Service Layer**. Action Layer initiates the processing of action which is mentioned as actions 1 through 5. The actions are then fully processed in service layer depending on the requirements. The major services are listed in this layer. Finally, **Data Access Layer** is for accessing the data from the database on the server side using a different kind of data models as needed.

### 4.3 Site Map

There are two kinds of primary users. The site map demonstrates the overall web tool map for two different kinds of primary users. It lists all the pages of the tool accessible to the users.

#### 4.3.1 Student Site Map

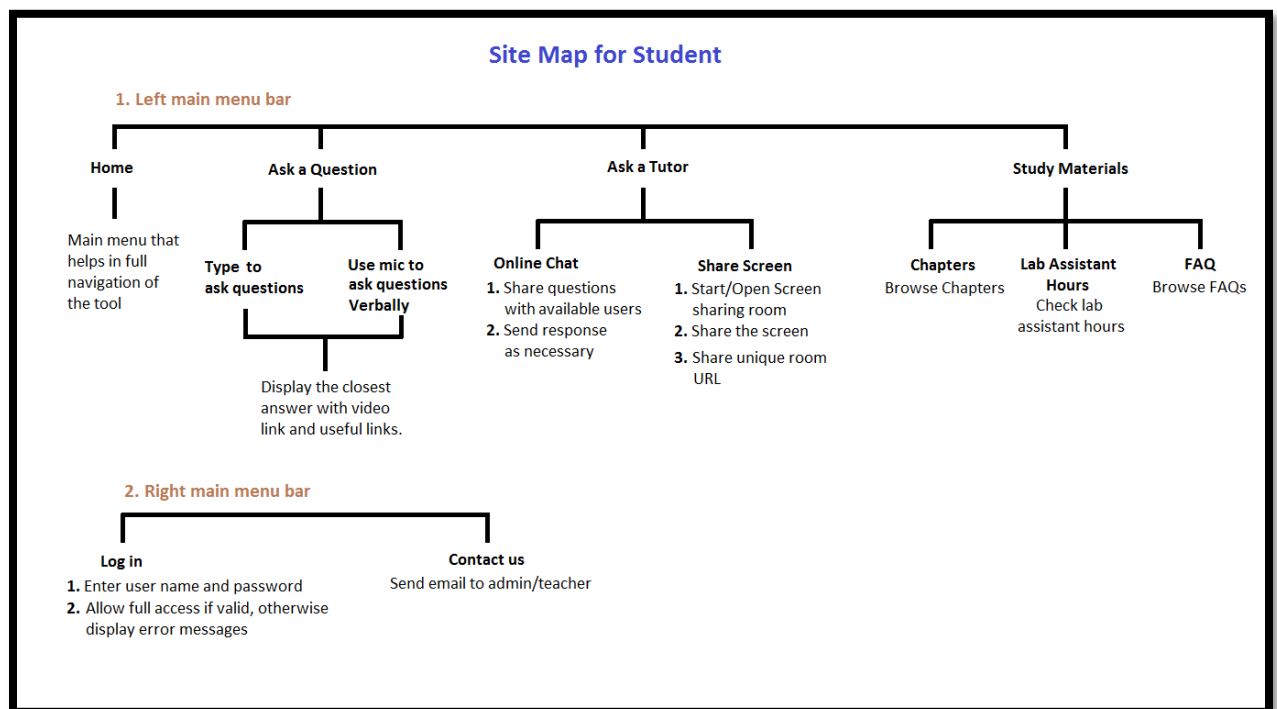


Figure 6: Site Map for Student

### 4.3.2 Teacher Site Map

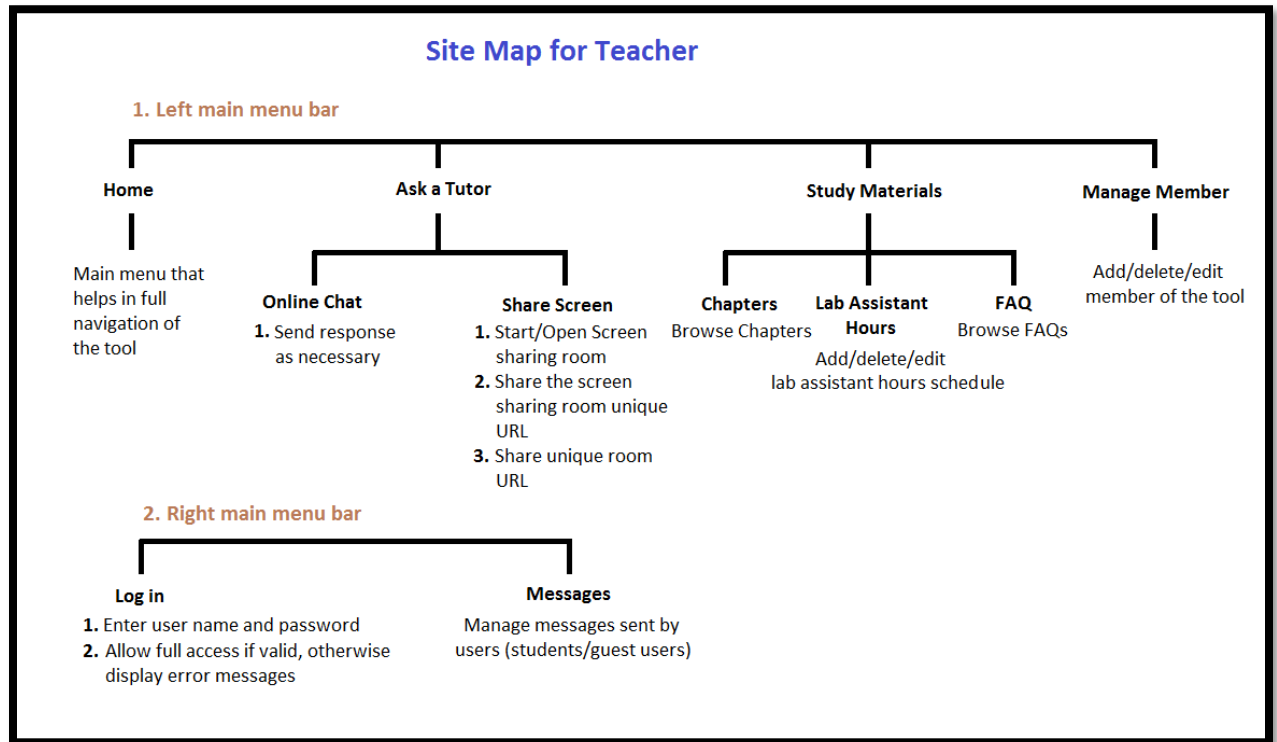


Figure 7: Site Map for Teacher

## 4.4 Use Case Diagrams

There are two kinds of primary users: Teacher and Students. However, a guest can also be considered as a user who can still use some of the functions offered by the system.

Considering all cases, an overall complete use case diagram, followed by separate use case diagrams for different types of user roles will be discussed in this section.

### 4.4.1 Overall Complete Use Case Diagram

An overall use case diagram including all different types of user roles with major functionalities is shown in Figure 8.

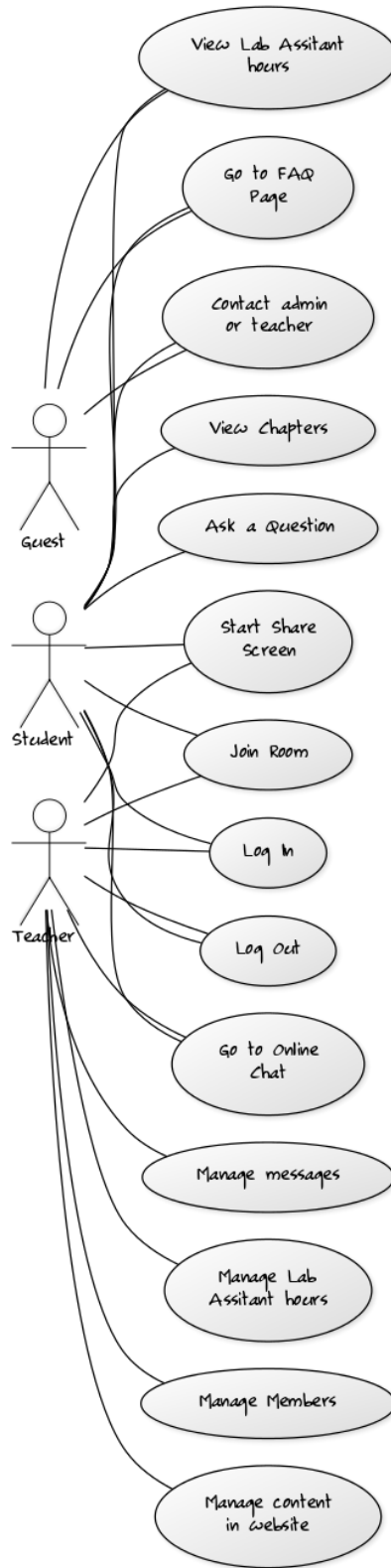


Figure 8: Overall Use Case diagram

#### 4.4.2 Use Case Diagram for Student

A detailed use case diagram for Student is shown in Figure 9.

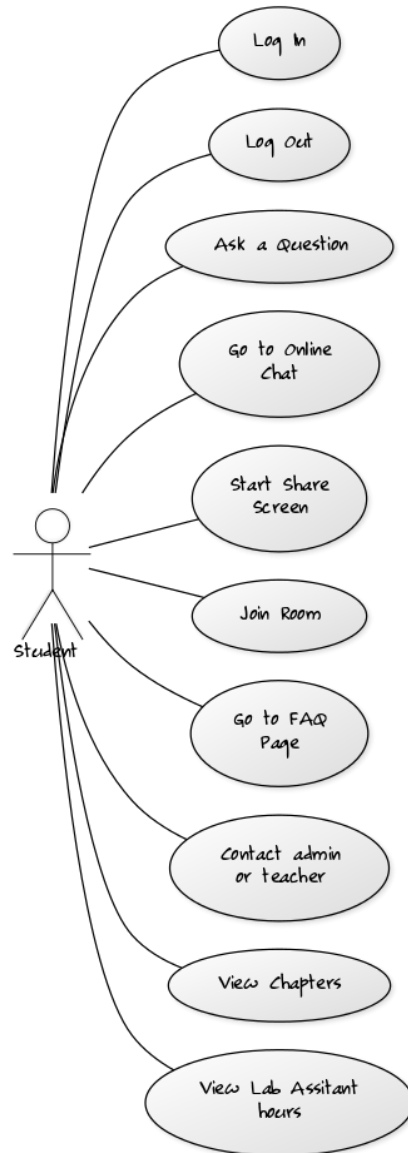


Figure 9: Student Use Case diagram

#### 4.4.3 Use Case Diagram for Teacher

A detailed use case diagram for Teacher is shown in Figure 10.

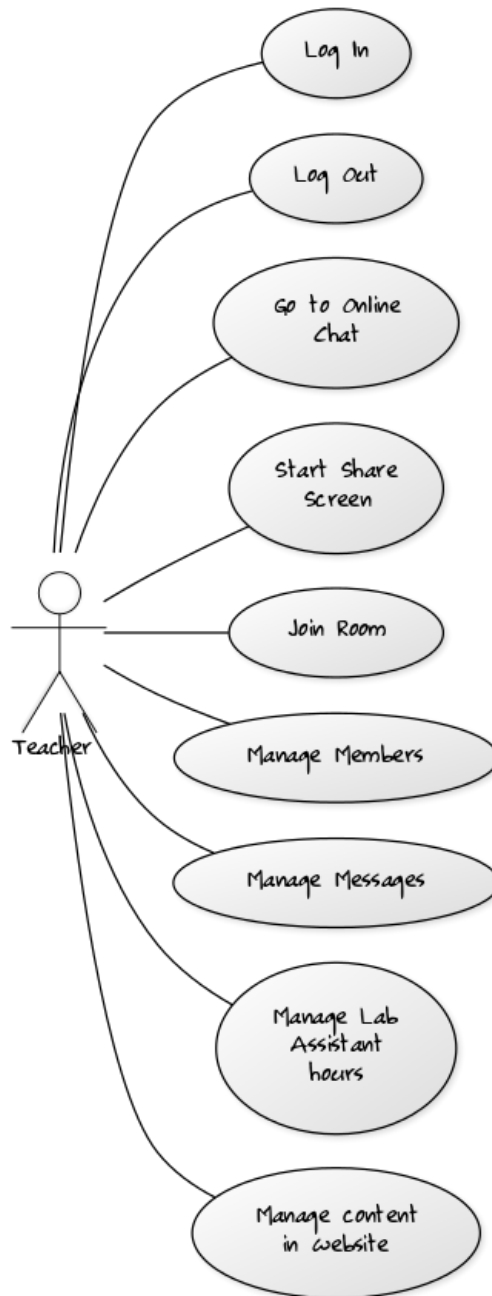


Figure 10: Teacher Use Case diagram

#### 4.4.4 Use Case Diagram for Guest

A detailed use case diagram for guest who doesn't have any credential to use all the functionalities of the system is shown in Figure 11.

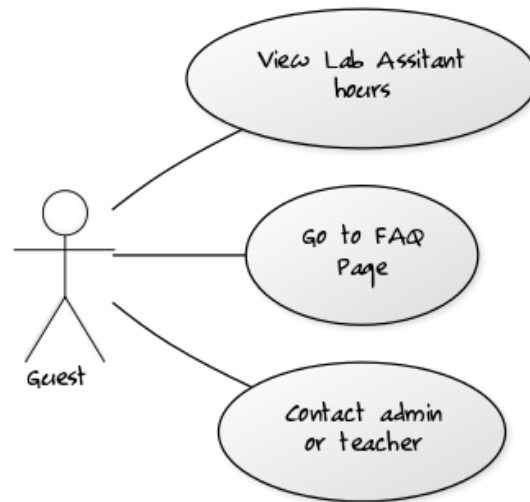


Figure 11: Guest Use Case diagram

## 4.5 Extracting Interfaces, Controllers and Services Scripts

To successfully offer all the services, a good implementation approach is needed. First, a good interface design is necessary to communicate with different type of users of the system. Once interfaces are designed, major controller and services which direct and implements the function need to be identified along with database and file storage needs.

### 4.5.1 Interface Designing

Depending on user type, there are many functions that need to be handled. For a user to successfully perform any function, the interface should be built and should be as user-friendly as possible and easy to navigate. Below are the decided minimum interfaces needed to successfully offer all the functional requirements of the project.

#### 4.5.1.1 Homepage Interface

This page is the main startup page which has main menu bar to navigate throughout the tool. The home page is designed to be slightly different for different users. Each user has their own specific set of main functions. The available menus in the main menu bar differs depending on those set of functions.

#### 4.5.1.1.1 Default Homepage and Property Matrix (Without Logging in)


**Virtual Tutor**

vt [HOME](#) [ASK A QUESTION](#) [ASK A TUTOR](#) [FAQ](#) [LAB ASSISTANT HOURS](#) [LOGIN](#) [CONTACT](#)

**DON'T THINK  
DO IT.**

**MAKE A *difference.***

Apply for *Computer Science*



Apply for Computer Science today. Click on picture to know more.

**Welcome to Class *201!***  
**I am your *Virtual Tutor.***

Here is some stuff.

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

**Column 1**  
Lorem ipsum dolor sit amet, consectetur adipisicing elit...  
Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris...

**Column 2**  
Lorem ipsum dolor sit amet, consectetur adipisicing elit...  
Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris...

**Column 3**  
Lorem ipsum dolor sit amet, consectetur adipisicing elit...  
Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris...

Figure 12: Home page for Guest

This page is the default page of the web tool. This page has the main navigation bar. It also contains information about the web tool. It is designed to have a slider with useful images hyperlinked to the useful links. Lastly, it contains three columns with a different set of useful information to the users.



Table 1: Property matrix for the default Home page (without logging in)

	<b>Field Name</b>	<b>Field Type</b>	<b>Display Text</b>	<b>Input Type</b>	<b>Required</b>	<b>Condition</b>	<b>Explanations</b>
1.	Home	Button	Home	Mouse Click	Yes		Clicking this button should take the user to home page from any screen (default screen).
2.	Ask a Question	Button	Ask a Question	Mouse Click	Yes	The user must log in to use this service	Clicking this button should take the user to log in screen.
3.	Ask a tutor	Button	Ask a tutor	Mouse Click	Yes	The user must log in to use this service drop down buttons.	Clicking this button should take the user to log in screen.
4.	FAQ	Button	FAQ	Mouse Click	Yes		Clicking this button should take to <i>FAQ</i> page where the users can look through all the FAQ related to C++.
5.	Lab Assistant Hours	Button	Lab Assistant Hours	Mouse Click	Yes		Clicking this button should take the users to a page where they can look at tutors' schedule.
6.	Log in	Button	Login	Mouse Click	Yes		Clicking this button should take the user to login screen with a form to enter the user and password.
7.	Contact	Button	Contact	Mouse Click	Yes		Clicking this button should take the user to a page with a form to contact the teacher/admin of the tool.

#### 4.5.1.1.2 Homepage and Property Matrix When Logged in as Student

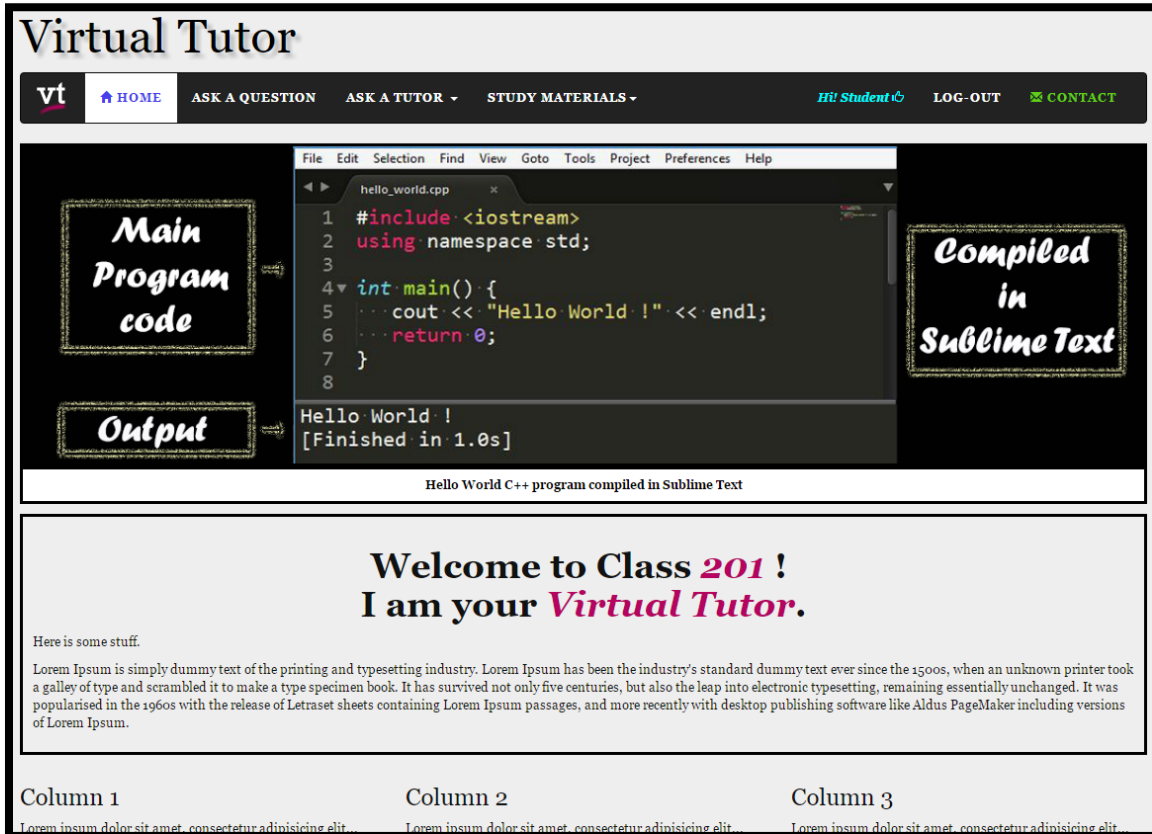


Figure 13: Home page for Student

This page contains navigation bar designed for students. Below is the condition that is applied to all the fields listed in table 2.

1. Logged in as Student to the web-tool.

Table 2: Property matrix for the default Home page logged in as Student

	Field Name	Field Type	Display Text	Input Type	Required	Condition	Explanations
1.	Home	Button	Home	Mouse Click	Yes		Clicking this button should take the user to default home page from any screen.
2.	Ask a Question	Button	Ask a Question	Mouse Click	Yes		Clicking this button should take the user to a page which allows

							them to ask questions (C++ related) verbally or manually to the system.
3.	Ask a tutor	Drop down button	Ask a tutor	Mouse Click	Yes	This should contain two additional drop-down buttons.	This drop button should display two buttons to go to online chat or share screen page. (Figure 14 below)
3.1.	Online Chat	Button	Online Chat	Mouse Click	Yes	One of the options in Ask a Tutor drop-down menu	Clicking this button should take the user to a page that allows them to perform online chat/communication.
3.2.	Share Screen	Button	Share Screen	Mouse Click	Yes	One of the options in <i>Ask a Tutor</i> drop-down menu	Clicking this button should take the user to a page where they will be able to share the screen.
4.	Study Materials	Drop down button	Study Materials	Mouse Click	Yes	This should contain three additional drop-down buttons.	This drop-down button should display three buttons to go to chapters, lab assistant hours or FAQ page. (Figure 15 below)
4.1.	Chapters	Button	Chapters	Mouse Click	Yes	One of the options in <i>Study Materials</i> drop-down menu	Clicking this button should open a page where the users can browse through different chapter notes.
4.2.	Lab Assistant Hours	Button	Lab Assistant Hours	Mouse Click	Yes	One of the options in <i>Study Materials</i> drop-down	Clicking this button should open a page where the users can look at the tutors' schedule.
4.3.	FAQ	Button	FAQ	Mouse Click	Yes	One of the options in <i>Study Materials</i> drop-down	Clicking this button should open a page where the users can look through all the FAQs related to C++.
5.	Log-out	Button	Log-out	Mouse Click	Yes		Clicking this button should log out the user from the current session.
6.	Contact	Button	Contact	Mouse Click	Yes		Clicking this button should take the user to a page with a form to contact the teacher/admin.

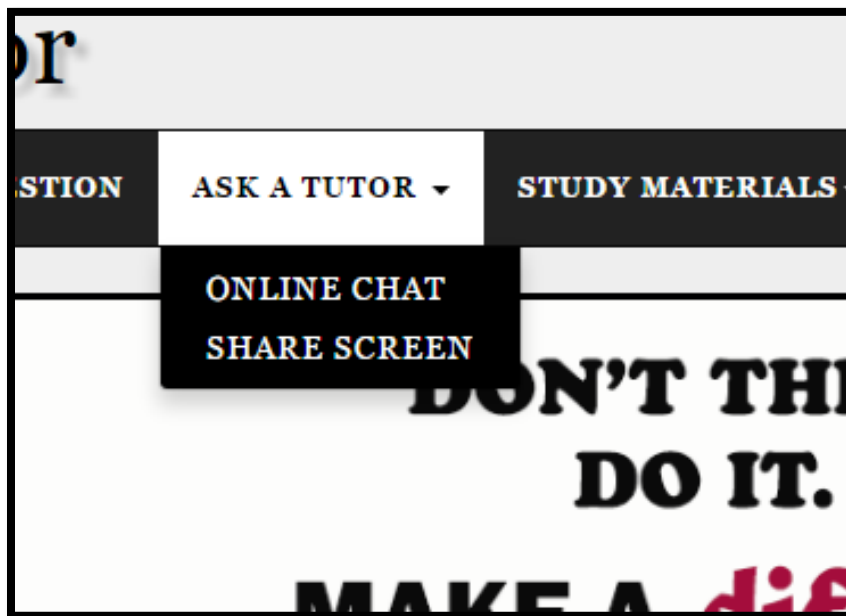


Figure 14: Drop-down menu for *Ask a Tutor* menu

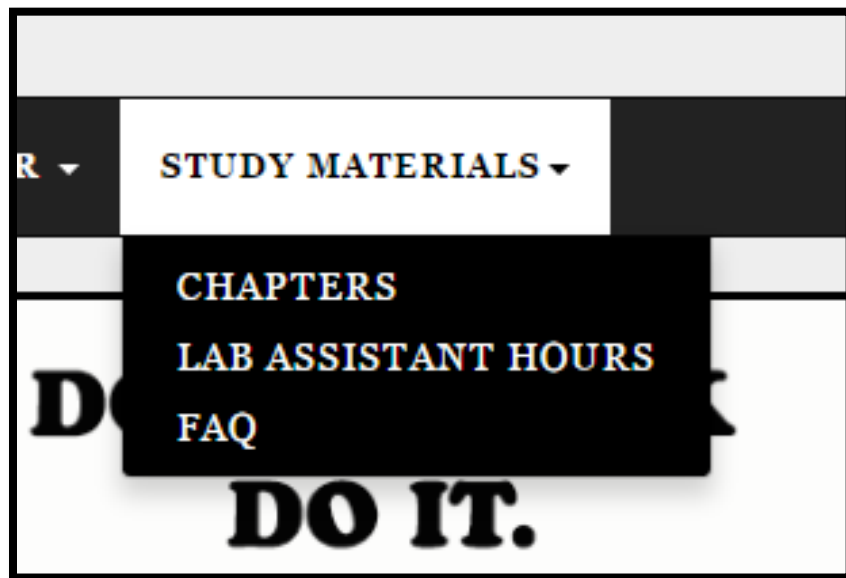


Figure 15: Drop-down menu for *Study Materials* menu

### 4.5.1.1.3 Homepage and Property Matrix When Logged in as Teacher

The screenshot shows the Virtual Tutor interface. At the top, there's a navigation bar with a 'vt' logo and links for HOME, ASK A QUESTION, ASK A TUTOR, STUDY MATERIALS, and MANAGE. A user is logged in as 'Hi! Teacher'. Below the navigation bar, the main content area is divided into three columns:

- Column 1:** Contains a box labeled 'Main Program code' and an 'Output' box showing 'Hello World!' and '[Finished in 1.0s]'.
- Column 2:** Contains a code editor window titled 'hello\_world.cpp' with the following C++ code:
 

```
1 #include <iostream>
2 using namespace std;
3
4 int main() {
5     cout << "Hello World !" << endl;
6     return 0;
7 }
8
```
- Column 3:** Contains a box labeled 'Compiled in Sublime Text'.

Below the code editor, a status bar reads 'Hello World C++ program compiled in Sublime Text'. The main content area is followed by a large box with the text: 'Welcome to Class 201 ! I am your Virtual Tutor.' Below this, there is a paragraph of Lorem Ipsum text. At the bottom of the page, there are three columns labeled 'Column 1', 'Column 2', and 'Column 3'.

Figure 16: Home page for Teacher

This page contains navigation bar designed for students. Below is the condition that is applied to all the fields listed in Table 3.

1. Logged in as Teacher to the web-tool.


Table 3: Property matrix for the default Home page logged in as Teacher

	Field Name	Field Type	Display Text	Input Type	Required	Condition	Explanations
1.	Home	Button	Home	Mouse Click	Yes		Clicking this button should take the user to default home page

							from any screen.
2.	Ask a Question	Button	Ask a Question	Mouse Click	No		If added, clicking this button should take the user to a page which allows them to ask questions (C++ related) verbally or manually to the system.
3.	Ask a tutor	Drop down button	Ask a tutor	Mouse Click	Yes	This should contain two additional drop-down buttons.	This drop button should display two buttons to go to online chat or share screen page. (Figure 14 above)
3.1.	Online Chat	Button	Online Chat	Mouse Click	Yes	One of the options in Ask a Tutor drop-down menu	Clicking this button should take the user to a page that allows them to perform online chat/communication.
3.2.	Share Screen	Button	Share Screen	Mouse Click	Yes	One of the options in <i>Ask a Tutor</i> drop-down menu	Clicking this button should take the user to a page where they will be able to share the screen.
4.	Study Materials	Drop down button	Study Materials	Mouse Click	Yes	This should contain three additional drop-down buttons.	This drop down button should display three buttons to go to chapters, lab assistant hours or FAQ page. (Figure 15 above)
4.1.	Chapters	Button	Chapters	Mouse Click	No	One of the options in <i>Study Materials</i> drop-down menu	If added, clicking this button should take to a page where the users can browse through different chapter notes.
4.2.	Lab Assistant Hours	Button	Lab Assistant Hours	Mouse Click	Yes	One of the options in <i>Study Materials</i> drop-down	Clicking this button should open a page where the teachers can add/edit/delete lab assistant lab hours to the schedule table.
4.3.	FAQ	Button	FAQ	Mouse Click	No	One of the options in <i>Study Materials</i> drop-down	If added, clicking this button should open a page where the users can look through all the FAQs related to C++.
5.	Manage	Button	Manage	Mouse Click	Yes		Clicking this button should open a page where the teachers can add/edit/delete web

							tool's users/students.
6.	Log-out	Button	Log-out	Mouse Click	Yes		Clicking this button should log out the user from the current session.
7.	Messages	Button	Messages	Mouse Click	Yes		Clicking this button should take the user to a page to check all the messages, concerns or queries sent by users/students. The user should be able to manage them.

#### 4.5.1.2 Ask a Question Interface

Since web tool is going to offer speech recognition facility, this page is designed to have a mic button  to take voice input. The user can also give input using the textbox in the page. It is designed to display the answer in the answer box as shown in Figure 17 below.

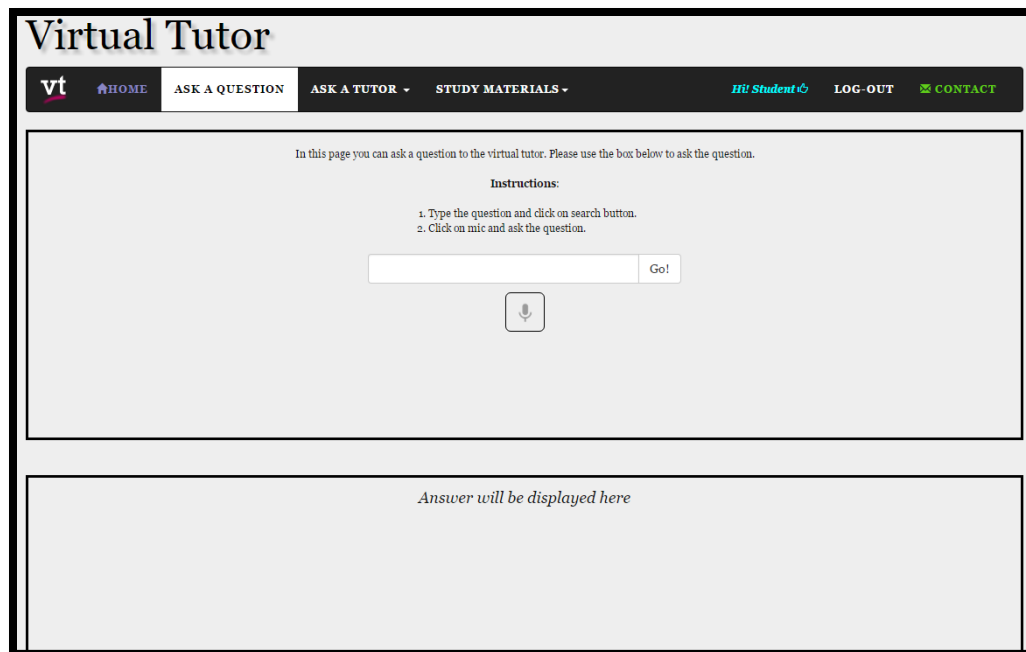


Figure 17: Ask a Question interface

The mic button should also be able to play the animation to denote the mic recording as shown in Figure 18 below.

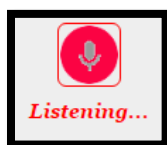


Figure 18: Mic design when listening to the user

Table 4: Property matrix for Ask a Question interface

	Field Name	Field Type	Display Text	Input Type	Required	Condition	Explanations
1.	Question display box	Input text box		Using mic or typing manually	Yes		Displays the question to be asked to the system.
2.	Ask System	Button	Go	Mouse Click	Yes/No	If the user chooses to type the question to ask the system.	Clicking this button should ask a question in the <i>question display box</i> to the system.
3.	Mic	Button	Mic image	Mouse Click	Yes/No	If the user chooses to ask a question verbally to the system.	Clicking this button should animate the mic image in button, start recording the user's voice for 5 seconds, display question in <i>question display box</i> and make an action request to the system for an answer.
4.	Answer display box	HTML div tag	Answer will be displayed here		Yes	The question has been asked to the system for an answer.	Once a question is asked to the system, the response is displayed in this box.

#### 4.5.1.3 Online Chat Interface

This is one of the main functions of the web tool. The online chat interface is designed such that it is easier to view other online users and share messages with them. The interface designed is shown in Figure 19.



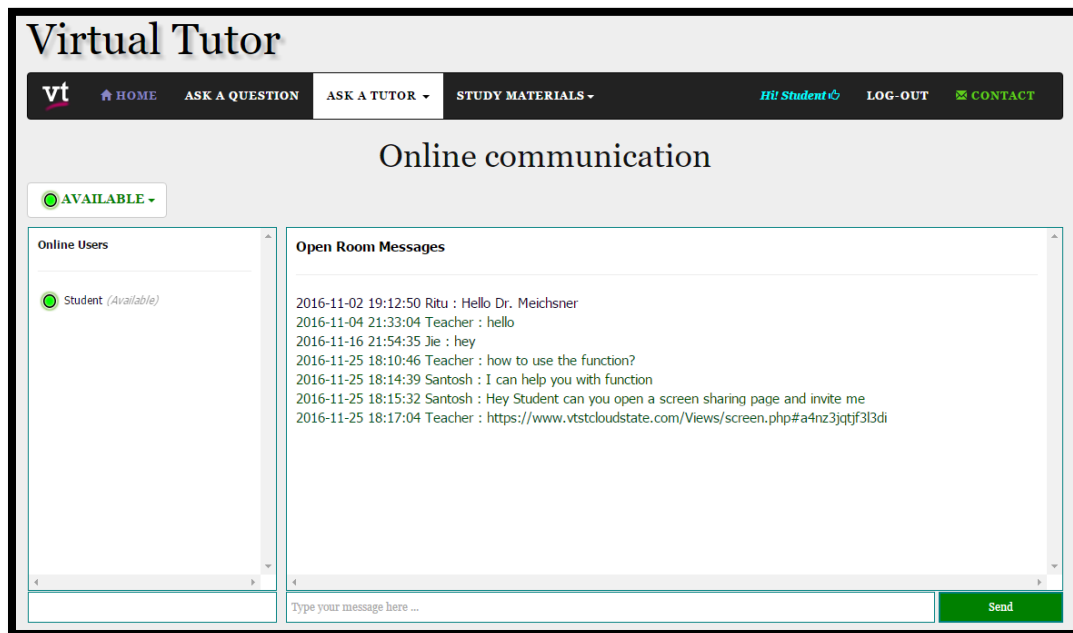


Figure 19: Online Chat interface: online users box on left, chat messages box on right

This page is designed to at least have a chat box to display the communication messages. It is also desired to have a button to show the status of the user which could be available, busy or away as shown in Figure 20 below.

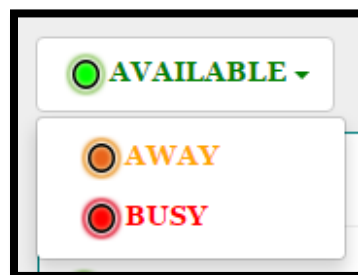


Figure 20: Different status available for users

The final design decision also includes an online users box (left side in Figure 19) where the user can see all the current online users.

Table 5: Property matrix for Online Chat interface

	Field Name	Field Type	Display Text	Input Type	Required	Condition	Explanations
1.	User Status	Button	Available (Default)	Mouse Click and select	Yes		The users should be able to either select <i>available</i> , <i>away</i> or <i>busy</i> status as shown in figure 20 above.
2.	Online Users	Text box	Online Users		Yes		This box should populate current online users of the web-tool.
3.	Open room messages	Text box	Open room messages		Yes		This box should populate all recent messages made (last 20 messages).
4.	Message box	Input text box	Type your message here	Keyboard typing	Yes		This textbox is designed to take in input message from the user to share with other users.
5.	Send	Button	Send	Mouse Click	Yes		Clicking this button should send the message to other online users and also should appear in <i>open room message</i> box.
6.	Message Status	Text box			Yes		This box should display the status of a message sent. For example, it should display <i>Message Sent Failed</i> in case message was not able to send or <i>Message Sent</i> if the message was sent successfully as shown in figure 21 below.

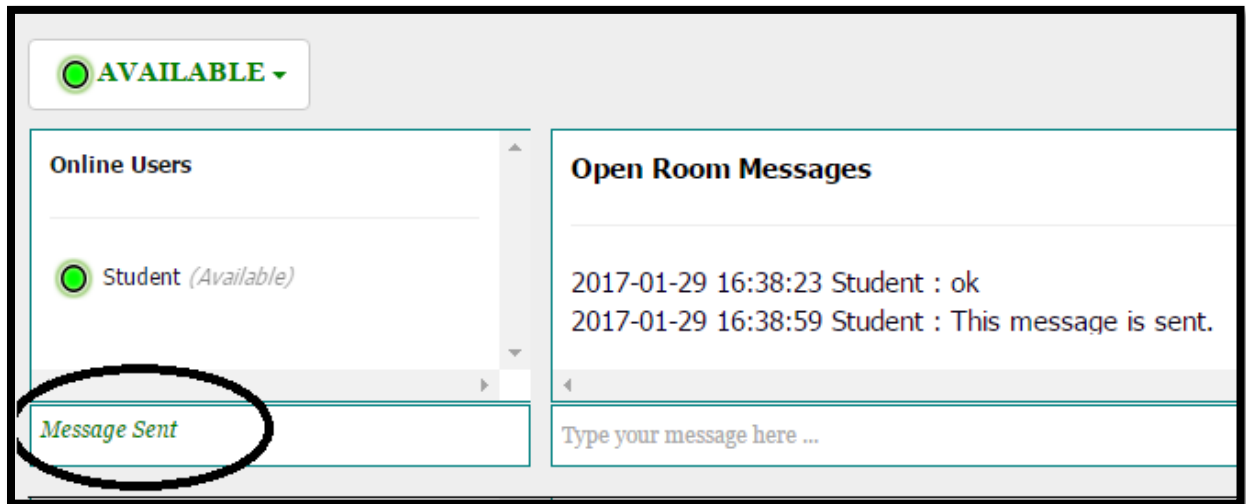


Figure 21: Status of message sent as *Message sent* in circle.

#### 4.5.1.4 Screen Sharing Interface

Screen Sharing interface is designed for the user to start screen sharing easily. This interface contains an instruction set box to guide them on how to use the screen sharing function. It is also designed to show all the users who are online like on *online chat* interface. But in this interface, the user will be able to directly invite other online users to the opened room. They can select individual online users or select all at once and start screen sharing. The final design of screen sharing interface before and after screen sharing starts are shown in Figure 22 and 23.

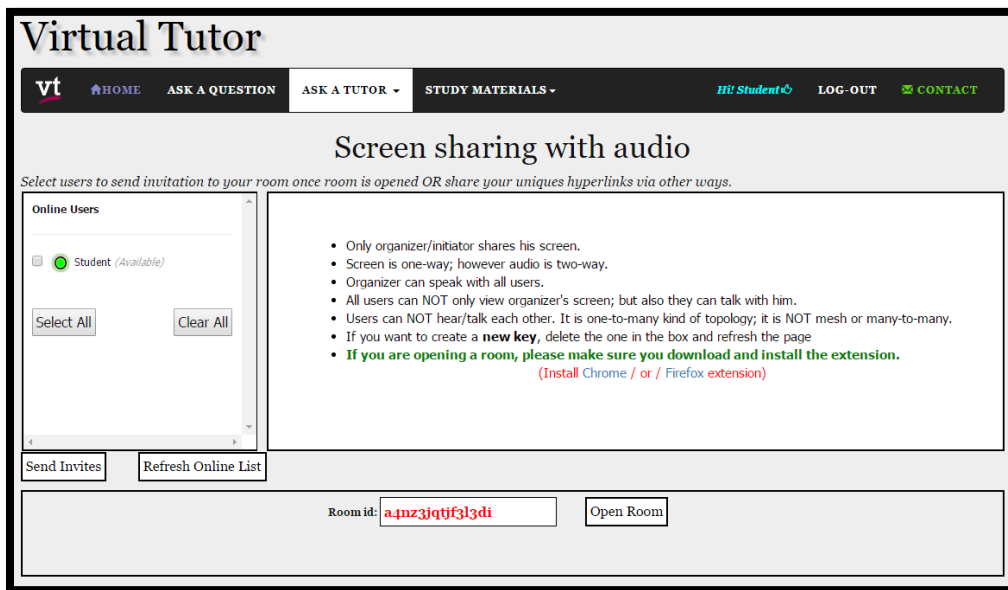


Figure 22: Screen Sharing interface final design (before screen share starts).

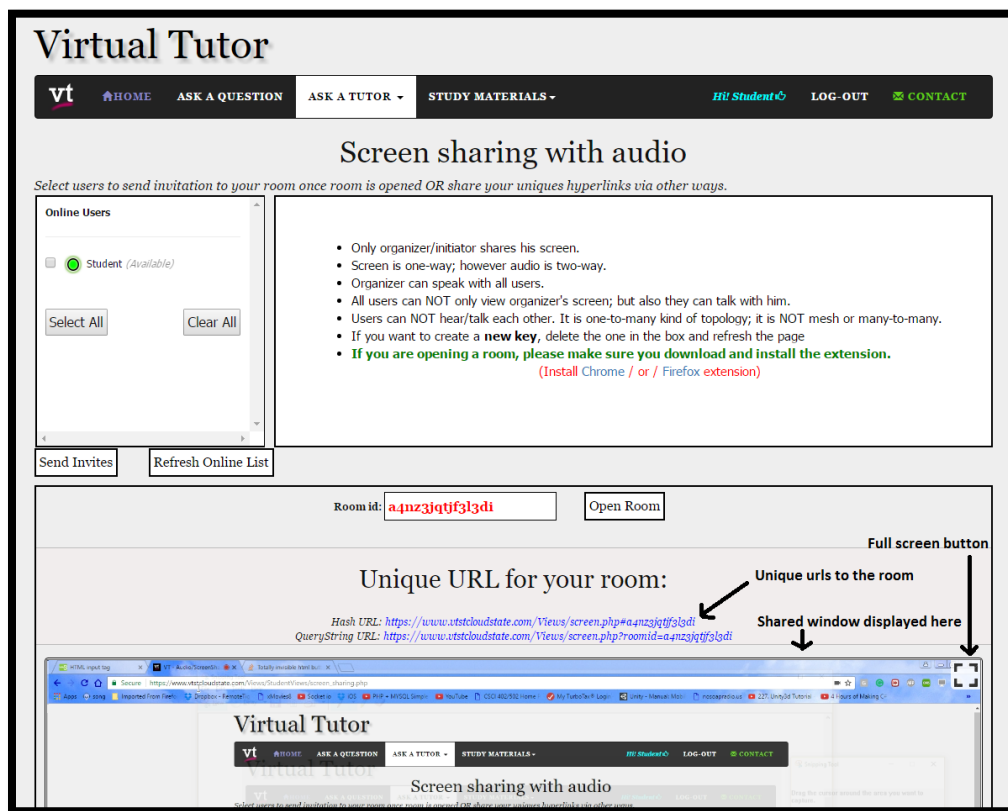


Figure 23: Screen Sharing interface final design (after screen share starts).

Table 6: Property matrix for Screen Sharing interface

	Field Name	Field Type	Display Text	Input Type	Required	Condition	Explanations
1.	Check box	Checkbox		Mouse click	Yes		Clicking should select the checkbox and clicking again should uncheck the check box.
2.	Select all	Button	Select all	Mouse Click	Yes		Clicking this should select all the check boxes.
3.	Clear all	Button	Clear all	Mouse Click	Yes		Clicking this should unselect all the check boxes.
4.	Send invites	Button	Send invites	Mouse Click	Yes		Clicking this button should invite all the selected users to the opened room.
5.	Refresh list	Button	Refresh online list	Mouse Click	Yes		Clicking this button should refresh the online user's list.
6.	Instructions box	Text box	All the instructions to start sharing screen		Yes		This text box should list the instructions needed for the user to start sharing the screen with other users.
7.	Room id	Input text box	Unique room id		Yes		This textbox should display the unique room id
8.	Open room	Button	Open room	Mouse Click	Yes		Clicking this button should initiate the screen sharing function with appropriate dialog boxes to follow the steps.
9.	Unique URLs	Text box	Unique URLs generated using unique room id		Yes	Only should appear once a room is successfully opened.	The unique URLs to the room should be displayed in this box.
10.	Video Box	HTML Div box			Yes	Only should appear once a room is successfully opened.	This box will display the shared screen window once screen sharing starts.
11.	Full screen	button	[ ]	Mouse Click	Yes	Only should appear once a room is	Clicking this button should maximize the screen sharing

						successfully opened.	window to full screen.
--	--	--	--	--	--	----------------------	------------------------

#### 4.5.1.5 Chapter Interface

This interface is designed to give chapters notes to the users. Each of the chapters is designed to have their own content which can be collapsed and seen on clicking. A detailed description is shown below in Figure 24 and property matrix table 7.

The screenshot shows the Virtual Tutor web interface. At the top, there is a navigation bar with the logo 'vt' and links for HOME, ASK A QUESTION, ASK A TUTOR, and STUDY MATERIALS. On the right side of the navigation bar, there are links for 'Hi Student', LOG-OUT, and CONTACT. Below the navigation bar is a search bar with the text 'Search for' and a magnifying glass icon. The main content area is titled 'Chapters' and contains a list of chapters with collapsible titles. The first chapter is '1. INTRODUCTION TO COMPUTERS AND PROGRAMMING', which is expanded to show three sub-chapters: '1.1 Why Program', '1.2 Computer Systems: Hardware and Software', and '1.3 Programs and Programming Languages'. The other chapters listed are '2. INTRODUCTION TO C++', '3. EXPRESSIONS AND INTERACTIVITY', and '4. MAKING DECISIONS'.

Figure 24: *Chapters* interface with collapsible titles.

Table 7: Property matrix for Chapters' interface

	Field Name	Field Type	Display Text	Input Type	Required	Condition	Explanations
1.	Search	Text box	Search for	Keyboard typing	Yes		As soon as users start typing, a search result from within the page should be displayed.
2.	Title	Collapsible Button	Title of the chapters	Mouse click	Yes		Clicking this should allow the user to display the full view and clicking again should collapse the view.

The collapsible button should be applied to all title and subtitles of the chapters as shown in Figure 24 above. The interface is also designed to have a search box wherein user can search for a different topic in the page.

#### **4.5.1.6 Lab Assistant Hours Interface**

Lab Assistant Hours interface is designed to give the user the ability to look at the current lab assistant hours. The page is designed separately depending on kind of users which could be students/guest or teachers. Figure 25 below shows the interface when logged in as a student. Similar is the interface when guest looks at this page.

However, the interface is designed little differently for teachers since they need the ability to modify the schedule table in the interface which looks like interface as shown in Figure 26 below.

# Virtual Tutor

vt [HOME](#) [ASK A QUESTION](#) [ASK A TUTOR](#) [STUDY MATERIALS](#) [HU Student](#) [LOG-OUT](#) [CONTACT](#)

## Lab Assistant Hours

Time/Day	Monday	Tuesday	Wednesday	Thursday	Friday
08:00 AM - 08:30 AM	santosh	Other	santosh	Randy	
08:30 AM - 09:00 AM	Someone				
09:00 AM - 09:30 AM		santosh		Santosh	
09:30 AM - 10:00 AM					
10:00 AM - 10:30 AM					
10:30 AM - 11:00 AM					
11:00 AM - 11:30 AM			santosh		
11:30 AM - 12:00 PM					
12:00 PM - 12:30 PM					
12:30 PM - 01:00 PM					
01:00 PM - 01:30 PM					
01:30 PM - 02:00 PM					
02:00 PM - 02:30 PM					
02:30 PM - 03:00 PM					
03:00 PM - 03:30 PM					
03:30 PM - 04:00 PM					
04:00 PM - 04:30 PM					

Figure 25: *Lab Assistant Hours* interface designed for students/guests.

Table 8: Property matrix for Lab Assistant Hours' interface for students/guests

	Field Name	Field Type	Display Text	Input Type	Required	Condition	Explanations
1.	Lab Assistant hour table	HTML table	Full schedule		Yes	For student, table is un-editable	Users should be able to look at lab assistant hours for the week.



**Virtual Tutor**

vt HOME ASK A QUESTION ASK A TUTOR STUDY MATERIALS MANAGE Hi! Teacher LOG-OUT MESSAGES

### Lab Assistant Hours

Edit/Update Save Changes

Table is now editable ...

Time/Day	Monday	Tuesday	Wednesday	Thursday	Friday
08:00 AM - 08:30 AM	santosh	Other	santosh	Randy	
08:30 AM - 09:00 AM	Someone				
09:00 AM - 09:30 AM		santosh		Santosh	
09:30 AM - 10:00 AM					
10:00 AM - 10:30 AM					
10:30 AM - 11:00 AM					
11:00 AM - 11:30 AM			santosh		
11:30 AM - 12:00 PM					
12:00 PM - 12:30 PM					
12:30 PM - 01:00 PM					
01:00 PM - 01:30 PM					
01:30 PM - 02:00 PM					
02:00 PM - 02:30 PM					
02:30 PM - 03:00 PM					

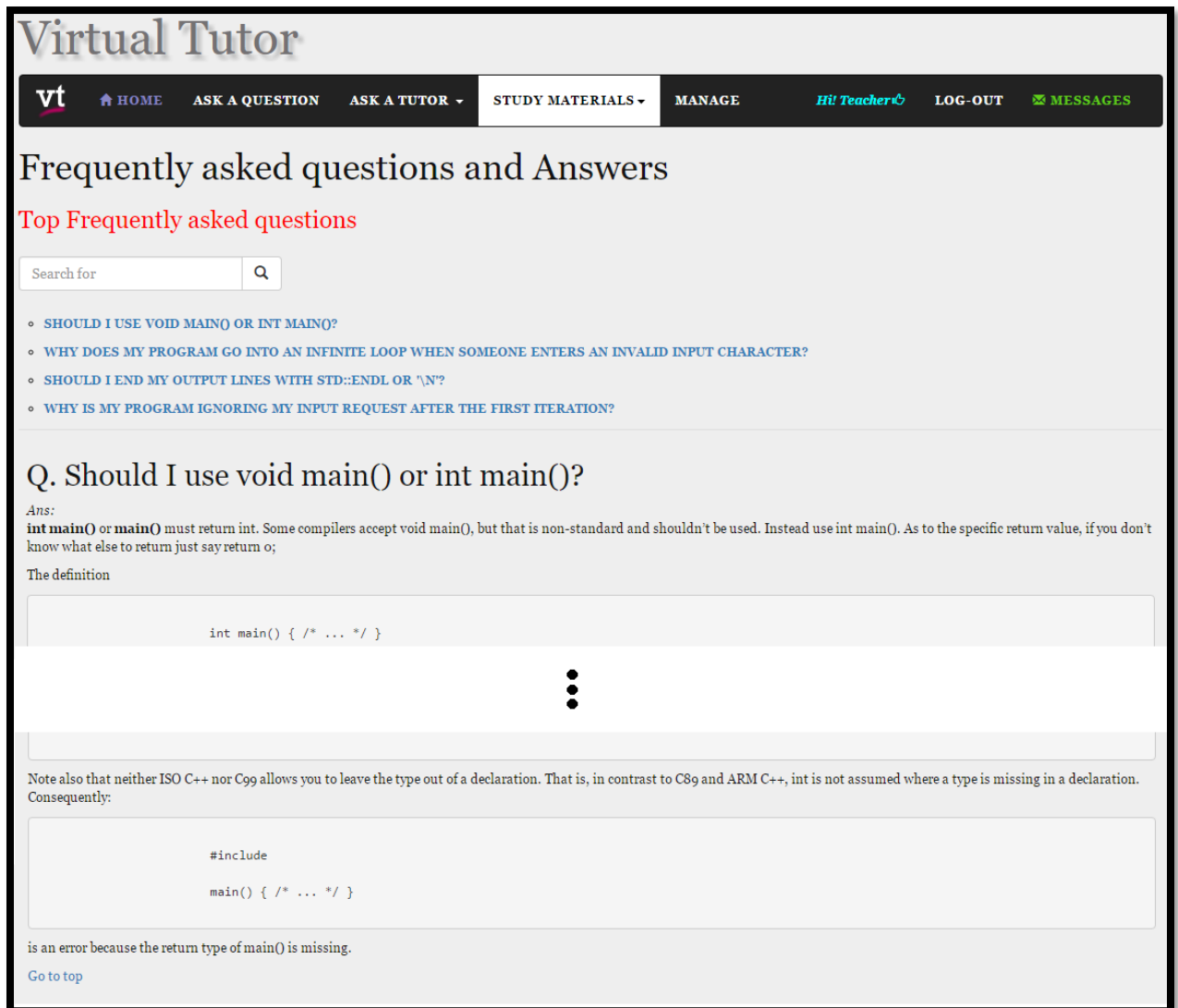
Figure 26: Lab Assistant Hours interface designed for teachers/admins.

Table 9: Property matrix for Lab Assistant Hours' interface for teachers/admins

	Field Name	Field Type	Display Text	Input Type	Required	Condition	Explanations
1.	Edit	Button	Edit/Update	Mouse Click	Yes	Only for teacher/admin	Clicking this button should allow the user to start editing the lab assistant hour table.
2.	Save	Button	Save Changes	Mouse Click	Yes	Only for teacher/admin	Clicking this button should save the changes made in the table and lock for further modification.
3.	Lab Assistant hour table	HTML Editable table	Full assistant hour schedule		Yes	Editable only for teacher/admin when edit is enabled	Users should be able to modify the table as per need.

### 4.5.1.7 FAQ Interface

FAQ page is designed to provide FAQs related to C++ to the users, especially students. This page is designed such that all the questions are listed on top and are hyperlinked to the answers to make the navigation easier. In Figure 27 below, all the heading questions in blue are hyperlinked to their respective answers in the page.



The screenshot shows the Virtual Tutor website's FAQ page. At the top, there is a navigation bar with links for HOME, ASK A QUESTION, ASK A TUTOR, STUDY MATERIALS, and MANAGE. The main heading is "Frequently asked questions and Answers". Below this, there is a section for "Top Frequently asked questions" with a search bar and a list of four blue hyperlinked questions. The first question is expanded to show its answer. The answer explains that `int main()` or `main()` must return `int`, and provides a code snippet for the correct definition. A note mentions that neither ISO C++ nor C99 allows leaving the type out of a declaration, and shows a code snippet for an incorrect definition that would result in a compiler error.

Virtual Tutor

vt HOME ASK A QUESTION ASK A TUTOR STUDY MATERIALS MANAGE Hi! Teacher! LOG-OUT MESSAGES

## Frequently asked questions and Answers

### Top Frequently asked questions

Search for

- SHOULD I USE VOID MAIN() OR INT MAIN()?
- WHY DOES MY PROGRAM GO INTO AN INFINITE LOOP WHEN SOMEONE ENTERS AN INVALID INPUT CHARACTER?
- SHOULD I END MY OUTPUT LINES WITH STD::ENDL OR '\N'?
- WHY IS MY PROGRAM IGNORING MY INPUT REQUEST AFTER THE FIRST ITERATION?

### Q. Should I use void main() or int main()?

Ans:  
**int main()** or **main()** must return int. Some compilers accept void main(), but that is non-standard and shouldn't be used. Instead use int main(). As to the specific return value, if you don't know what else to return just say return 0;

The definition

```
int main() { /* ... */ }
```

⋮

Note also that neither ISO C++ nor C99 allows you to leave the type out of a declaration. That is, in contrast to C89 and ARM C++, int is not assumed where a type is missing in a declaration. Consequently:

```
#include
main() { /* ... */ }
```

is an error because the return type of main() is missing.

[Go to top](#)

Figure 27: FAQ interface with a complete question set.

Table 10: Property matrix for FAQ interface

	Field Name	Field Type	Display Text	Input Type	Required	Condition	Explanations
1.	Search	Text box	Search for	Keyboard typing	Yes		As soon as users start typing, a search result from within the page should be displayed.
2.	Question headings	Hyperlinked texts	List of questions	Mouse Click	Yes		Clicking on the hyperlinked heading question should take the user to particular section of the page with answer
3.	Go to top	Button	Go to top	Mouse Click	Yes	It is available after the answer to each question listed in the page.	Clicking on this button should take the user to the beginning or top of the page for easy navigation.

#### 4.5.1.8 Manage (Members) Interface

This interface is only available for teachers/admins, which have the abilities to add a new member, edit or delete the existing member. Hence, this interface is only visible to teachers. The interface is designed to have a table with a list of existing members with buttons on side of each member that should allow the teacher to edit or delete the selected member. At the bottom of the page, a button is available to add a new member to the list. As soon as this button is clicked, a form should appear to fill in the member details. Figure 28 shows the full interface design and Figure 29 shows the form to add a new member to the list.

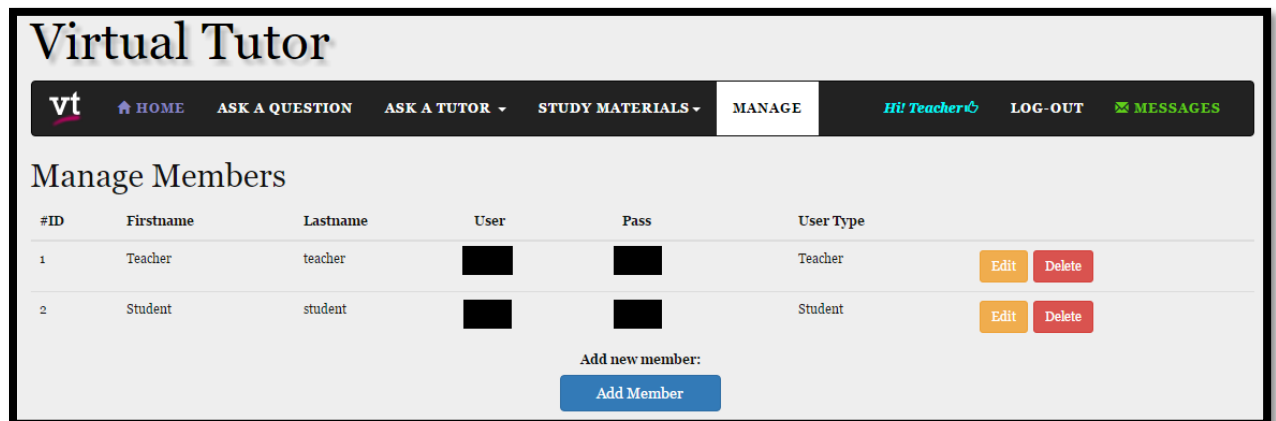
Figure 28: *Manage* interface

Table 11: Property matrix for Manage interface

	Field Name	Field Type	Display Text	Input Type	Required	Condition	Explanations
1.	Member table	HTML Table	All the members		Yes		This table displays the full details of all the members of the tool.
2.	Edit	Button	Edit	Mouse Click	Yes	Only available for teachers	Clicking this button should open a form as shown in figure 29 with the member info.
3.	Delete	Button	Delete	Mouse Click	Yes	Only available for teachers	Clicking this button should allow the user to delete the selected member.
4.	Add	Button	Add	Mouse Click	Yes	Only available for teachers	Clicking this button should open a form as shown in figure 30 to add a new member/user to the system.

The image shows a web form titled "Add New Member" with a close button in the top right corner. The form contains three input fields: "First Name" (a text input box), "Last Name" (a text input box), and "User Role" (a dropdown menu). Below these fields are two buttons: a green "Confirm Add" button and a red "Cancel" button.

Figure 29: Add a new member form

Table 12: Property matrix for Adding a new member form

	Field Name	Field Type	Display Text	Input Type	Required	Condition	Explanations
1.	First Name	Textbox		Keyboard typing	Yes		This field should take in the first name of the new member to be created.
2.	Last Name	Textbox		Keyboard typing	Yes		This field should take in the last name of the new member to be created.
3.	User Role	Drop down		Mouse Click	Yes		This drop down should let the user choose the user-type of the new member (teacher or student).
4.	Confirm	Button	Confirm Add	Mouse Click	Yes		Clicking this should create a new member and add to the member list.
5.	Cancel	Button	Cancel	Mouse Click	Yes		Clicking this button should cancel the process of creation of new member.

**Edit Member** ×

**First Name:**

**Last Name:**

**User Name:**

**Password:**

**User Role**

**Confirm Edit**

**Cancel**

Figure 30: Edit a member form

Table 13: Property matrix for Editing a member form

	<b>Field Name</b>	<b>Field Type</b>	<b>Display Text</b>	<b>Input Type</b>	<b>Required</b>	<b>Condition</b>	<b>Explanations</b>
1.	First Name	Textbox		Keyboard typing	Yes		This field should display the existing first name of the member and allow to edit it.
2.	Last Name	Textbox		Keyboard typing	Yes		This field should display the existing last name of the member and allow to edit it.
3.	User name	Textbox		Keyboard typing			This field should display the existing user name of the member and allow to

							edit it.
4.	Password	Textbox		Keyboard typing			This field should display the existing password of the member and allow to edit it.
5.	User Role	Drop down		Mouse Click	Yes		This drop down should let the user choose the user-type of the member.
6.	Confirm	Button	Confirm Edit	Mouse Click	Yes		Clicking this should successfully edit the member.
7.	Cancel	Button	Cancel	Mouse Click	Yes		Clicking this button should cancel the process of editing of the member.

#### 4.5.1.9 Login Interface

Login interface is designed to let user successfully log in to the system to use different functionalities offered by the system. A form is displayed in this display to let the user enter their user name and password.

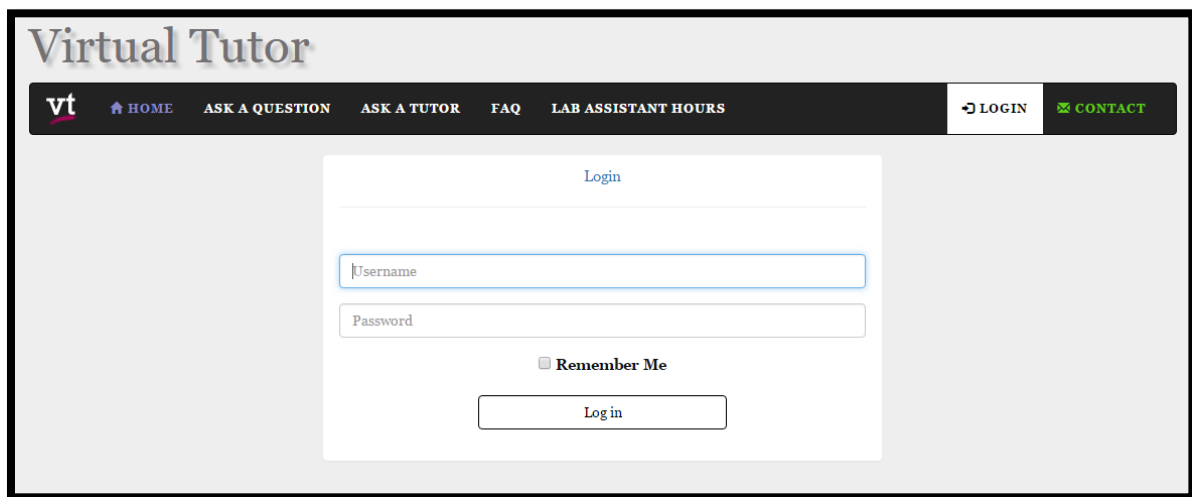


Figure 31: Login screen Interface design

Table 14: Property matrix for Log in interface

	Field Name	Field Type	Display Text	Input Type	Required	Condition	Explanations
1.	Username	Textbox	Username	Keyboard typing	Yes		This field should take in the user name.

2.	Password	Textbox (Password type)	Password	Keyboard typing	Yes	The typing in the box should be hidden or appeared as a set of dots.	This should take in the password of the user.
3.	Remember me	Checkbox		Mouse Click	Yes		This should let the user save the user name and pass in the browser for later use.
4.	Log in	Button	Log in	Mouse Click	Yes		This should trigger the authentication process and take to the respective home page on successful login.

#### 4.5.1.10 Logout Interface

This interface is designed to let the user know about the successful ending of the session.

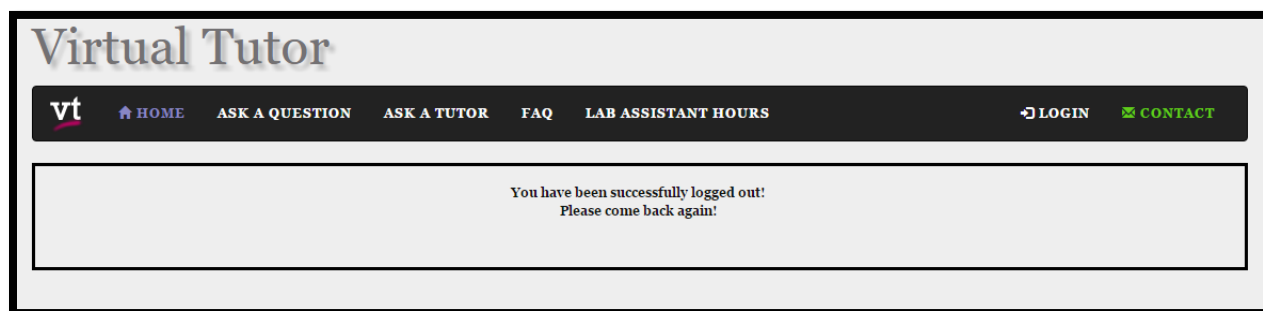


Figure 32: Log out interface

#### 4.5.1.11 Contact Interface

Contact interface represents one of the main functions of the web tool which is only available to student or guest to send the important queries and concerns to the teachers. The interface is designed as shown in Figure 33 below.



Figure 33: Contact interface (Only available for student/guest)

Table 15: Property matrix for Contact interface

	Field Name	Field Type	Display Text	Input Type	Required	Condition	Explanations
1.	First Name	Textbox	First Name	Keyboard typing	Yes	The page is only available for student/guest.	This field should take in the first name of the user writing the query or concern.
2.	Last Name	Textbox	Last Name	Keyboard typing	Yes	The page is only available for student/guest.	This field should take in the last name of the user writing the query or concern.
3.	E-Mail	Textbox	E-Mail Address	Keyboard typing	Yes	The page is only available for student/guest.	This field should take in the email address of the user writing the query or concern.
4.	Message	Textbox	Message here	Keyboard typing	Yes	The page is only available for student/guest.	This textbox should take in full message of the user to be delivered to the teacher/admin.

#### 4.5.1.12 Message Interface

The message sent using the Contact interface above is collected and shown to the teacher using Message interface. This is designed to list all the messages sent by users which are listed by date, oldest on top. The teacher should be able to read, delete or reply to the

message sent by the user. The final design decision for the interface is shown in Figure 34 below.

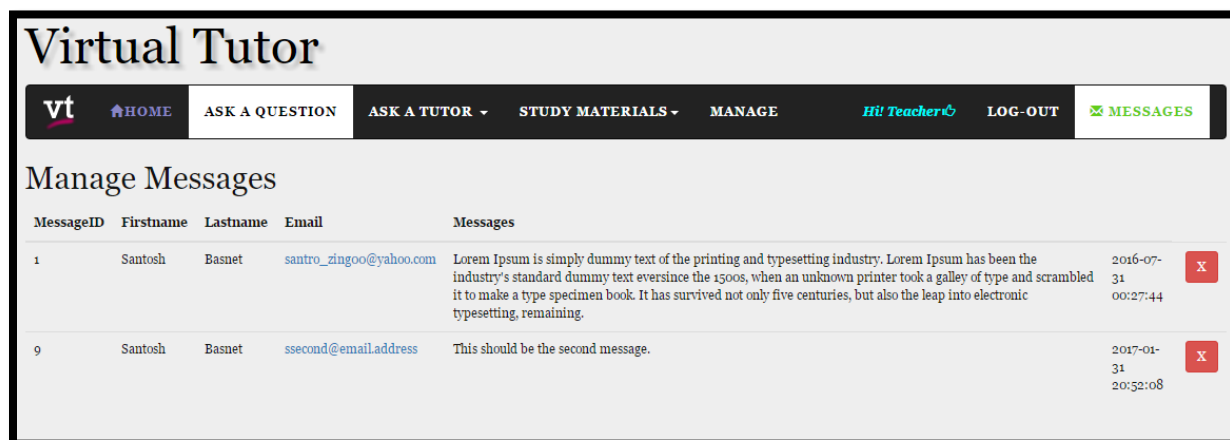


Figure 34: Message interface for Teachers

Table 16: Property matrix for Message interface for teacher

	Field Name	Field Type	Display Text	Input Type	Required	Condition	Explanations
1.	Message Table	HTML table	All the messages		Yes	The page is only available for the teacher.	This table should display all the messages to be read.
2.	Email	HTML address tag	Corresponding email address of the user sending message	Mouse Click	Yes	The page is only available for the teacher.	This tag on-click should activate the default mail client on the computer to respond to the message.

#### 4.5.2 Extracting Controllers and Services

To support all the interfaces listed above, there are many controllers and services required to process the actions requested from the interfaces. To extract all the controllers and services, the main jobs and functions from each interface could be extracted and dedicated to writing separate scripts to perform the job. The jobs could be to perform a certain action which may or may not involve communication with backend server and database. After user interactions in different interfaces (see section 4.5.1), there may be various kinds of actions that need to

be performed. *Controllers* are needed to perform those major actions and requests from all the interfaces. Depending on whether backend and/or database communication required, the controller may need to work with *services* script to complete the request. Some of the major controllers and services needed for this project are listed below.

#### **4.5.2.1 Login Controller**

This controller is responsible for processing the provided username and password entered by the user. To process and complete this request, a service script is needed to perform communication with the database to verify the user. Below is the service that should achieve the successful result.

- **User Verify Service**

This service is dedicated to verify the user by comparing the given input information with the records in the database and return the result as pass or fail.

The final response from the service is then processed by the login controller to direct the user to the proper screen.

#### **4.5.2.2 Mic Controller**

This controller controls the speech recognition that is integrated into *Ask a Tutor* interface. One input is taken using the speech recognition; a service script is needed to pull the closely related answer from the database which will be provided by the script below.

- **Search Database**

Once the question is asked, this script looks for the closest answer that is recorded in the database and return with an appropriate list of answers.

Once the script comes up with the response, the controller is responsible for displaying the answer in the proper format.

#### 4.5.2.3 Users Messages Controller

This controller displays the recent messages in the chat box. For this, a communication to the backend server is needed to collect the most recent (20) messages saved on the server which can be performed with the help of script below.

- Get file-data service

This script gets the data (messages) from the server which could be saved in a file and return to the user message controller.

In the end, the controller decorates the response from the service and display in the chat box.

#### 4.5.2.4 Chat Message Controller

This controller controls the sending and saving of the message from the user. The message is eventually saved to the file (in the backend server) once a *send* request is made in the interface. To perform the saving of the new messages, a script is needed which can access the files and/or database on the server.

- Sending message service

Once the message is typed and send request is made, this script is called by the controller. This script opens the file, saves the message and successfully closes the file. A response is then sent to the controller depending on the success or failure of the transaction.

A notification is finally shown by the controller to the user depending on the response from the script.

#### 4.5.2.5 Status Controller

To reflect the status of the user in the tool, a controller is needed. This controller can be supported with multiple scripts as shown below.

- Current status service

This service collects the current status of the user.

- Change status service

This service changes the status of the user when a change of status call is made.

#### **4.5.2.6 Screen Sharing Controller**

This is the main controller behind the screen sharing function. It takes in the request from the user, sets up a window to display the shared screen and be ready to share the window in the web browser. The script generates a separate unique room with an URL after each screen sharing session.

#### **4.5.2.7 Screen Sharing Online User List Controller**

Once screen sharing controller starts a screen sharing window, a controller is needed to share the room URL to other online users of the tool. This is fulfilled by screen sharing online user list controller. This controller gets the list of current online users and sends an invitation request to the chosen users by the organizer. Hence, a couple of service scripts are required to perform the job.

- Get file-data service

This service gathers the current online user information and returns the list of user information to the controller.

- Send invitation service

Once the list of the online user is seen, this service sends the invitation to the chosen user with the unique room URL.

In the end, the controller collects the response from the script and displays them in a readable format to the user in the interface.

#### **4.5.2.8 Detect Invitation to Screen Controllers**

The controller and services to send the invitation are discussed above. And therefore, a controller to detect and handle the invitation from the organizer of the screen sharing is needed. In order to fulfill this requirement, a service script is needed which can communicate between users of the tool.

- Detect invitation to screen service

This service script detects invitation from other users for viewing the shared screen and send the response to the controller with the appropriate data.

In addition, the controllers manipulate the response from the script and greet the user with a suitable dialog box. Depending on the response from the user, the controller handles the further request from the user.

#### **4.5.2.9 Manage Database Controller**

This controller handles the member information in the Manage interface. It displays the member information in a decorated table and is assisted by following services.

- Fetch data database service

This service fetches the data from the database and returns to the controller.

- Add, edit, or delete database service

Depending on the action request from the controller, this script adds, edits or deletes the member of the tool.

Once the database transaction is done, controllers gather the response from the script and notify the user.

#### **4.5.2.10 Lab Assistant Hours' Controller**

This controller controls the overall schedule of the lab assistant hour and displays in the interface. This controller is assisted with a service that returns the final schedule of the lab assistant hours.

- Get lab assistant hours' service

This service gets the lab assistant hours' schedule from the server and returns it to the controller to display in the decorated format.

#### **4.5.2.11 Search Controller (for FAQ and Chapters Page)**

This controller dynamically searches the closest content in the page and lists them below the search box.

#### **4.5.2.12 Contact Messages Database Controller**

Similar to member database controller, this controller handles the messages sent by the users/students in the form of queries and concerns. It displays all the contact messages in a decorated table (in the Message interface, see section 4.5.1.12) and is assisted by the following service.

- Contact messages fetch database service

This service fetches the data from the database and returns to the controller.

- Delete contact message database service

This service performs the delete transaction from the database on request.

#### **4.5.2.13 Contact Message Controller**

This controller collects the contact information and messages from the user from the *Contact* interface and uses the service below to save the new messages to the database.

- Contact message store service

This service gets the new messages from the controller and returns the success or failure response back to the controller.

The controller finally collects the response from the service and notification is shown to the user.

#### **4.5.2.14 Previous Session Controller**

This controller maintains and collects each session data once the user logs into the system. This controller is mainly responsible for detecting any previous valid session for ease of user. For example, if a user is logged in a window and user again tries to go to the tool in another window, the user should not have to log in again since the user is logged in another window already. The controller communicates with the server using following service script.

- Detect the previous session

This maintains and saves the user session data and guides the controller to perform the necessary actions.

#### **4.5.2.15 Additional Services Required**

- Timeout session check for student

This service keeps track of session for student and checks if the timeout of the session has occurred. If the timeout has occurred, it terminates the session.

- Timeout session check for teacher

Like Timeout session check student service, this service checks if timeout session has occurred for teacher and logs out if that is the case.

- Logout service

This service is responsible for logging out the user of the tool. This service is used by both services listed above.



- Image slider service

This service is responsible for having the image slider mechanism that is designed to have on the home page of the tool.

### **4.5.3 Extracting Database Tables and File Storage Necessity**

There are different kinds of data that needs to be stored for the full functionality of the tool. Based on the requirements and designed specifications above, there are three major database tables that need to be set up and some files for storing important information.

#### **4.5.3.1 Database Tables**

MySQL is a relational database management system (RDBMS) that is used in this project. A database called *VirtualTutor* is created and there are three major tables designed to have in this project which are listed below.

##### **4.5.3.1.1 Members**

This table holds the member information and has 6 defined columns.

- ID – Primary key, unique ID for every member, AUTO\_INCREMENT, integer type
- User Fame – varchar type, the first name of the user
- User Lname – varchar type, the last name of the user
- User login – varchar type, the login name of the user to log in to the system
- User pass – varchar type, the user password to log in to the system
- User type – Type of user: Teacher or Student

##### **4.5.3.1.2 Contact Messages**

This table records all the contact messages sent by the users/guests and has 6 defined columns.

- Message ID – Primary key, unique ID for every message, AUTO\_INCREMENT, integer type
- First Name – varchar type, the first name of the user who sent message
- Last Name – varchar type, the last name of the user who sent message
- Email – varchar type, the email of the user who sent message
- Message – text type, the actual message sent by the user
- Date time – timestamp, default to the message received current timestamp

#### **4.5.3.1.3 QandA**

This table records the important questions and answers with important links related to the question and has 6 defined columns.

- ID – Primary key, unique ID for every QandA, AUTO\_INCREMENT, integer type
- Question – text type, the question
- Answer – text type, the short answer to the question
- Sample Code – text type, the sample implementation code related to the question
- Video link – text type, the important video link(s) related to the question
- Useful link – text type, the important useful link(s) related to the question

#### **4.5.3.2 File Storage Needs**

For the simplicity of the project, some files are saved in the server to hold some important useful information. One example is to save all the online user names in the file which could be used by the several controllers to list the online users. Following are different files that are set up in the server for smooth execution of the tool.

##### **4.5.3.2.1 Online Users Text File**

This file lists all the current online users and is used by multiple controllers upon requirement. Every time a user logs in or logs out of the system, this file is updated, saved

and locked. Each online member information is saved using member name and unique id to avoid similar name confusion.

#### **4.5.3.2.2 Log Report Text File**

This file logs the recent 20 messages to be displayed in the chat box. Every time a new message is sent, this file is updated, saved and locked. Every message is saved with the message sent time stamp and member name sending the message.

#### **4.5.3.2.3 Lab Assistant Hours' Text File**

This file records the lab assistant hours and is updated every time lab schedule table is modified in the lab assistant hours' page.

#### **4.5.3.2.4 Invitation Text File**

This file records the information of the member who is inviting another online user to the shared screen room. It also saves the user's info that is being invited. Once the invitation is picked up, the invitation information is deleted from the file and saved.

### **4.6 Use Cases Realization**

Detailed use case realizations of each use cases with extended and refined scenarios are listed in this section. All the realizations are accompanied with a flow chart to demonstrate the overall flow of the system.

### 4.6.1 Log in

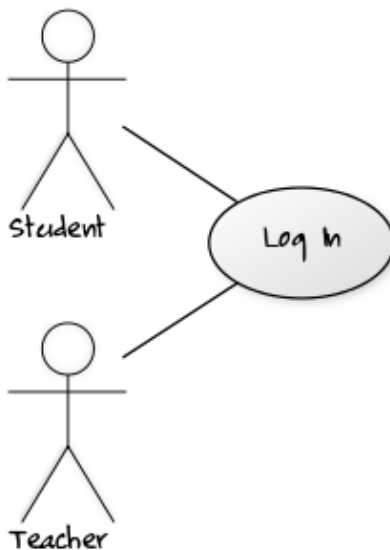


Figure 35: Log in Use Case diagram

Table 17: Log in Use Case realization table

<b>Use Case ID</b>	<b>01</b>
<b>Use Case Name</b>	<b>Virtual tutor Log in</b>
<b>Brief Description</b>	This use case enables users to log in to the system.
<b>Actors</b>	Student, Teacher
<b>Pre-condition</b>	<ol style="list-style-type: none"> <li>1. User has valid login credentials.</li> <li>2. User's information is saved and exists in the database.</li> <li>3. User has internet service available.</li> </ol>
<b>Step by Step description</b>	<ol style="list-style-type: none"> <li>1. User opens a browser (Google chrome or Firefox for full usability).</li> <li>2. User enters the virtual tutor website URL. Virtual tutor home page is displayed on the screen.</li> <li>3. User clicks <i>LOGIN</i> button in the main navigation</li> </ol>

	<p>menu and is redirected to the login page with a login form displayed on the screen.</p> <ol style="list-style-type: none"> <li>4. User enters user name and password.</li> <li>5. User hits <i>Log in</i> button to log in to the system.</li> <li>6. Use logs in to the system.</li> </ol>
<p><b>Scenarios:</b></p> <ol style="list-style-type: none"> <li><b>1. A regular scenario</b></li>   <li><b>2. Alternative scenarios</b></li>   <li><b>3. Exception Scenario</b></li> </ol>	<ol style="list-style-type: none"> <li>1.       <ol style="list-style-type: none"> <li>a. User wants to login to the system.</li> <li>b. User opens the login page.</li> <li>c. User enters user name and password.</li> <li>d. User clicks to <i>log in</i> button on the screen.</li> <li>e. User is successfully logged in.</li> </ol> </li>   <li>2.       <ol style="list-style-type: none"> <li>a. User clicks on a different button on the main menu bar displayed on the home page. (Refer use cases 02, 03, 04.)</li> <li>b. User goes back to home page clicking on HOME menu.</li> </ol> </li>   <li>3.       <ol style="list-style-type: none"> <li>a. User wants to login to the system.</li> <li>b. User opens the login page.</li> <li>c. User enters user name and password.</li> <li>d. User clicks log in button. Log in fails and is again displayed with a login form to start over.</li> </ol> </li> </ol>
<p><b>Post condition</b></p>	<p>User is successfully logged into the system and is displayed with the full navigation bar to use all the functionalities.</p>

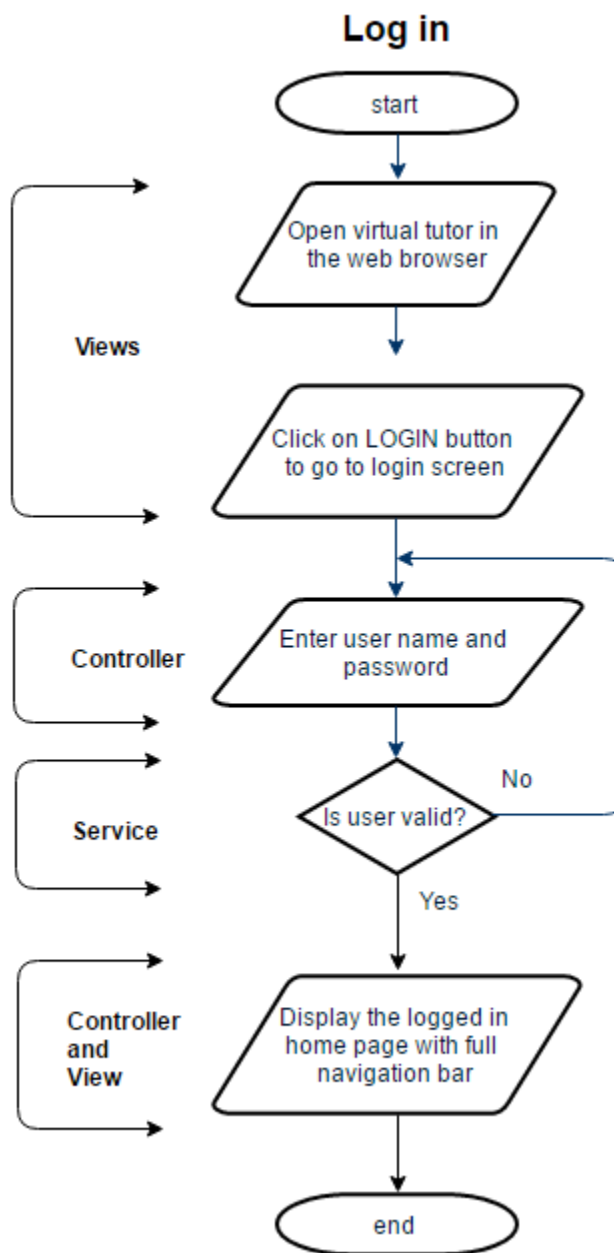


Figure 36: User login full action flow diagram

### 4.6.2 Go to FAQ Page

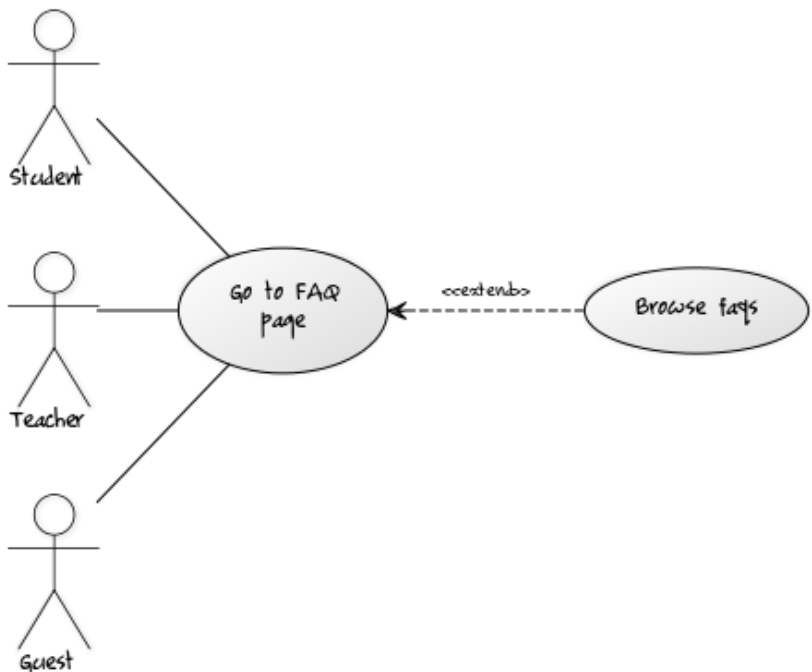


Figure 37: Go to FAQ page Use Case diagram

Table 18: Go to FAQ page Use Case realization table

<b>Use Case ID</b>	<b>02</b>
<b>Use Case Name</b>	<b>Go to FAQ page</b>
<b>Brief Description</b>	This use case enables users to go to FAQ page to browse all the FAQs related to C++.
<b>Actors</b>	Student, Teacher, Guest
<b>Pre-condition</b>	1. User has internet service available. (Optional: Use may or may not be logged in)
<b>Step by Step description</b>	1. Two major ways to get to FAQ page. <ul style="list-style-type: none"> <li>a. User is not in the tool at startup (not in use):                 <ul style="list-style-type: none"> <li>• User opens a browser (Google chrome</li> </ul> </li> </ul>

	<p>or Firefox for full usability).</p> <ul style="list-style-type: none"> <li>• User enters the virtual tutor website URL. Virtual tutor home page is displayed on the screen.</li> <li>• User clicks <i>FAQ</i> button in the main navigation menu and is redirected to the FAQ page.</li> </ul> <p>b. User is already in the tool at startup (in use):</p> <ul style="list-style-type: none"> <li>• User clicks <i>FAQ</i> button in the main navigation menu and is redirected to the FAQ page.</li> </ul> <p>(Note:</p> <p>*If the user is logged in as Student or teacher, <i>FAQ</i> button is found under <i>STUDY MATERIALS</i> drop down menu.</p> <p>*Refer to use cases 25 for search in-page functionality)</p> <p>2. User browses through the question (search for a question, Refer use case 25).</p>
<p><b>Scenarios:</b></p> <p><b>1. A regular scenario</b></p> <p><b>2. Alternative scenarios</b></p> <p><b>3. Exception Scenario</b></p>	<p>1.</p> <ol style="list-style-type: none"> <li>a. User wants to browse through FAQs.</li> <li>b. User opens the FAQ page.</li> <li>c. User successfully browses through the FAQs related to C++.</li> </ol> <p>2.</p> <ol style="list-style-type: none"> <li>a. User clicks on a different button on the main menu bar displayed on the home page. If guest, refer use cases 01, 03, 04. If logged in as Student, refer use cases 03, 04, 05, 06, 07, 10 and 24. If logged in as Teacher, refer use cases 03, 04, 05, 06, 07, 13, 17, 20 and 24.</li> <li>b. User goes back to home page clicking on HOME menu.</li> </ol> <p>3. -</p>



<b>Post condition</b>	User is successfully able to browse through all the FAQs that is available in FAQ page or search them in the page.
-----------------------	--

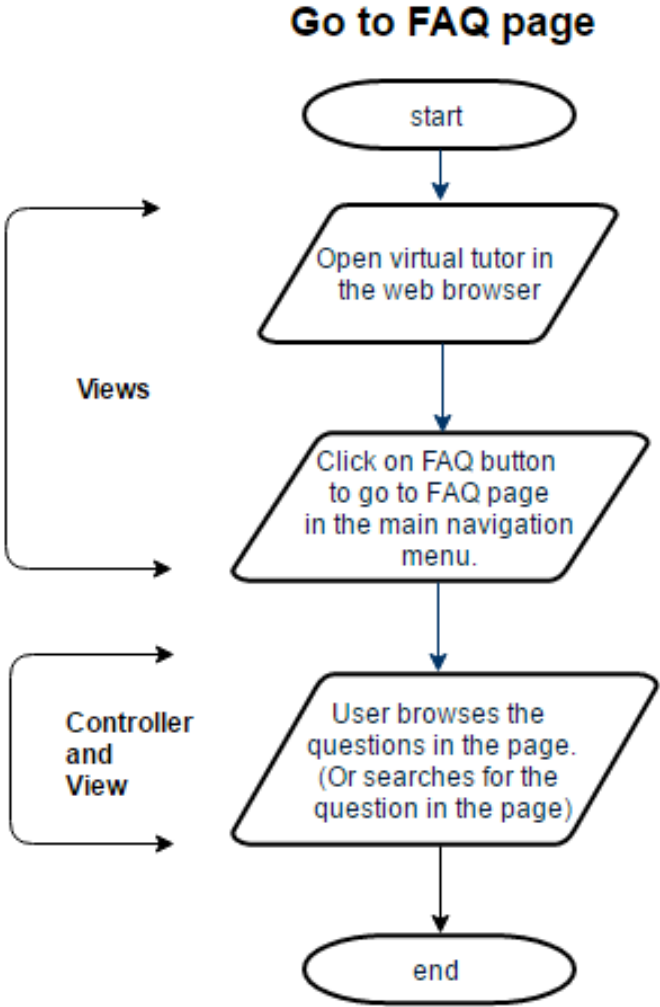


Figure 38: Go to FAQ page full action flow diagram

### 4.6.3 View Lab Assistant Hours

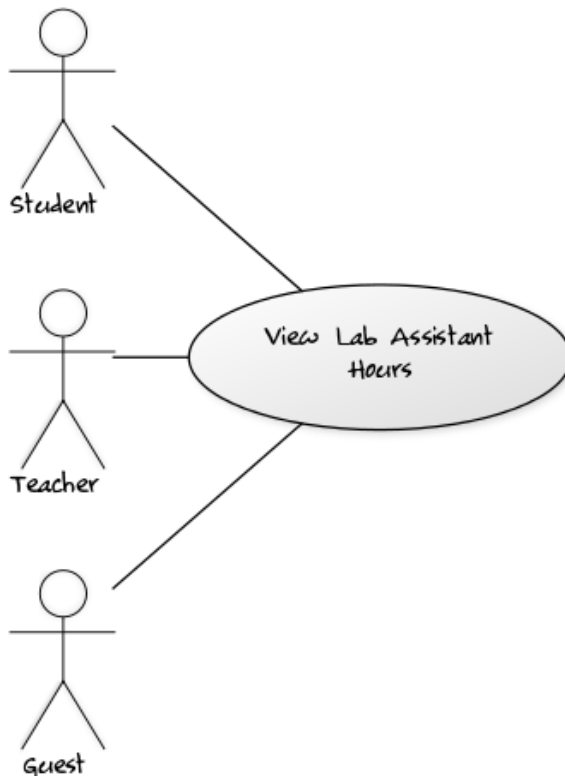


Figure 39: View lab Assistant hours' page Use Case diagram

Table 19: View lab Assistant hours' page Use Case realization table

<b>Use Case ID</b>	<b>03</b>
<b>Use Case Name</b>	<b>View Lab Assistant Hours</b>
<b>Brief Description</b>	This use case enables users to look at current lab assistant hours for the week.
<b>Actors</b>	Student, Teacher, Guest
<b>Pre-condition</b>	1. User has internet service available. (Optional: Use may or may not be logged in)

<p><b>Step by Step description</b></p>	<ol style="list-style-type: none"> <li>1. Two major ways to get to FAQ page. <ol style="list-style-type: none"> <li>a. User is not in the tool at startup (not in use): <ul style="list-style-type: none"> <li>• User opens a browser (Google chrome or Firefox for full usability).</li> <li>• User enters the virtual tutor website URL. Virtual tutor home page is displayed on the screen.</li> <li>• User clicks <i>LAB ASSISTANT HOURS</i> button in the main navigation menu and is redirected to the lab assistant hours' page.</li> </ul> </li> <li>b. User is already in the tool at startup (in use): <ul style="list-style-type: none"> <li>• User clicks <i>LAB ASSISTANT HOURS</i> button in the main navigation menu and is redirected to the lab assistant hours' page.</li> </ul> </li> </ol> </li> </ol> <p>(Note: If the user is logged in as Student or teacher, <i>LAB ASSISTANT HOURS</i> button is found under <i>STUDY MATERIALS</i> drop down menu.)</p> <ol style="list-style-type: none"> <li>2. User looks at the table of schedule of lab assistants for the week.</li> </ol>
<p><b>Scenarios:</b></p> <ol style="list-style-type: none"> <li>1. A regular scenario</li> <li>2. Alternative scenarios</li> </ol>	<ol style="list-style-type: none"> <li>1. <ol style="list-style-type: none"> <li>a. User wants to look at the current schedule of lab assistants for the week.</li> <li>b. User opens the <i>LAB ASSISTANT HOURS</i> page.</li> <li>c. User successfully browses through the lab assistant hours' table.</li> </ol> </li> <li>2. <ol style="list-style-type: none"> <li>a. User clicks on a different button on the main menu bar displayed on the home page. If guest, refer use cases 01, 02, 04. If logged in as Student, refer use cases 02, 04, 05, 06, 07, 10 and 24. If logged in as Teacher, refer use cases 02, 04, 05, 06, 07, 10, 13, 17, 20 and 24.</li> <li>b. User goes back to home page clicking on HOME menu.</li> </ol> </li> </ol>

<b>3. Exception Scenario</b>	3. -
<b>Post condition</b>	User is successfully able to look at the lab assistant schedule page.

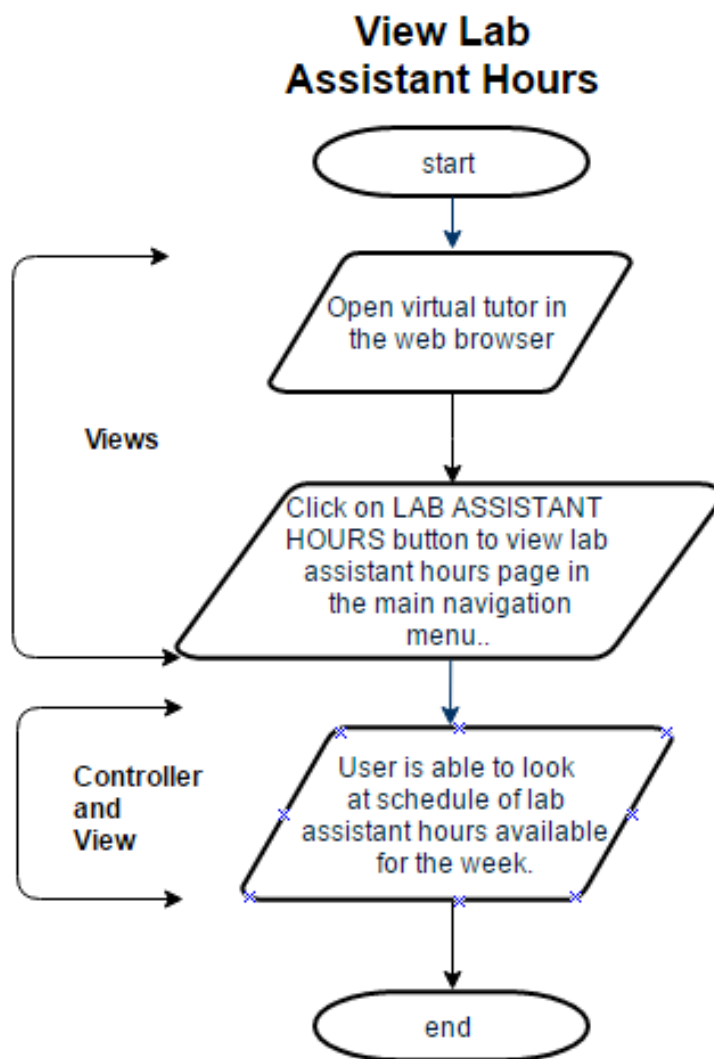


Figure 40: View lab assistant hours' full action flow diagram

#### 4.6.4 Contact Admin or Teacher

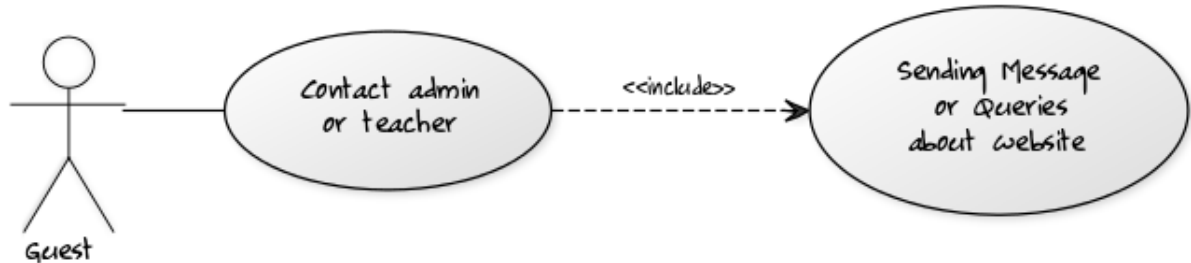


Figure 41: Contact admin or teacher page Use Case diagram

Table 20: Contact admin or teacher page use case realization table

<b>Use Case ID</b>	<b>04</b>
<b>Use Case Name</b>	<b>Contact admin or teacher</b>
<b>Brief Description</b>	This use case enables users to send messages with concerns or queries to the admin or teacher.
<b>Actors</b>	Guest, Student
<b>Pre-condition</b>	1. User has internet service available. (Optional: Use may or may not be logged in)
<b>Step by Step description</b>	<ol style="list-style-type: none"> <li>1. Two major ways to get to FAQ page. <ol style="list-style-type: none"> <li>a. User is not in the tool at startup (not in use): <ul style="list-style-type: none"> <li>• User opens a browser (Google chrome or Firefox for full usability).</li> <li>• User enters the virtual tutor website URL. Virtual tutor home page is displayed on the screen.</li> <li>• User clicks <i>CONTACT</i> button in the main navigation menu and is redirected to the contact us page.</li> </ul> </li> <li>b. User is already in the tool at startup (in use): <ul style="list-style-type: none"> <li>• User clicks <i>CONTACT</i> button in the main navigation menu and is</li> </ul> </li> </ol> </li> </ol>

	<p>redirected to the contact us page.</p> <ol style="list-style-type: none"> <li>2. A <i>contact us</i> form is displayed. User fills in first name, last name, email, and messages.</li> <li>3. User hits <i>Send</i> button below the form.</li> <li>4. User successfully sends the message. A confirmation box appears on the screen.</li> </ol>
<p><b>Scenarios:</b></p> <ol style="list-style-type: none"> <li><b>1. A regular scenario</b></li> <li><b>2. Alternative scenarios</b></li> <li><b>3. Exception Scenario</b></li> </ol>	<ol style="list-style-type: none"> <li>1. <ol style="list-style-type: none"> <li>a. User wants to send a major concern about the website to the admin or teacher.</li> <li>b. User opens the tool in the browser.</li> <li>c. User opens the <i>CONTACT</i> page.</li> <li>d. User fills in the contact us form and successfully sends the messages. A confirmation message is displayed on the screen.</li> </ol> </li> <li>2. <ol style="list-style-type: none"> <li>a. User clicks on a different button on the main menu bar displayed on the home page. If guest, refer use cases 01, 02, 03 If logged in as Student, refer use cases 02, 03, 05, 06, 07, 10 and 24.</li> <li>b. User goes back to home page clicking on HOME menu.</li> </ol> </li> <li>3. User wants to send a major concern about the website to the admin or teacher. User opens the tool in the browser. User opens the <i>CONTACT</i> page. User fills in the contact us form but message sending fails because of some invisible issue when <i>Send</i> button is clicked.</li> </ol>
<b>Post condition</b>	User successfully contacts the admin by sending a message.

### 4.6.5 Ask a Question

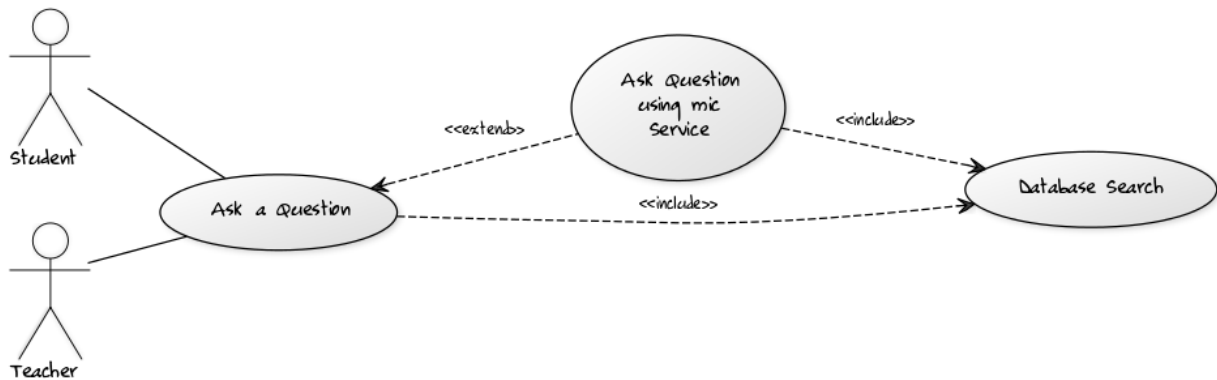



Figure 42: Ask a Question Full Use Case diagram

Table 21: Ask a Question Use Case realization table

<b>Use Case ID</b>	<b>05</b>
<b>Use Case Name</b>	<b>Ask a Question</b>
<b>Brief Description</b>	This use case enables users to ask a question to the tool, verbally using speech recognition or using a keyboard and get the answers saved in the database.
<b>Actors</b>	Teacher, Student
<b>Pre-condition</b>	<ol style="list-style-type: none"> <li>1. User have internet service available.</li> <li>2. User is logged in as student or teacher with correct credentials and is on the home page of the tool.</li> </ol>
<b>Step by Step description</b>	<ol style="list-style-type: none"> <li>1. There are two major ways to get to ASK A QUESTION page and use the service. <ol style="list-style-type: none"> <li>a. User wants to use speech recognition: <ul style="list-style-type: none"> <li>• User is on the home page of the tool.</li> <li>• User clicks on ASK A QUESTION button in the main navigation menu and is redirected to the ask a question page.</li> <li>• User clicks on mic button  that is shown on the screen which turns to</li> </ul> </li> </ol> </li> </ol>





<b>3. Exception Scenario</b>	If logged in as Teacher, refer use cases 02, 03, 06, 07, 10, 13, 17, 20 and 24. c. User goes back to home page clicking on HOME menu. 3. a. User's mic is unavailable but is trying to use speech recognition to ask the question to the system. b. User asks the question but no answer is displayed in the answer box.
<b>Post condition</b>	User successfully asks the question and answer is displayed in the answer box with useful links and a video.

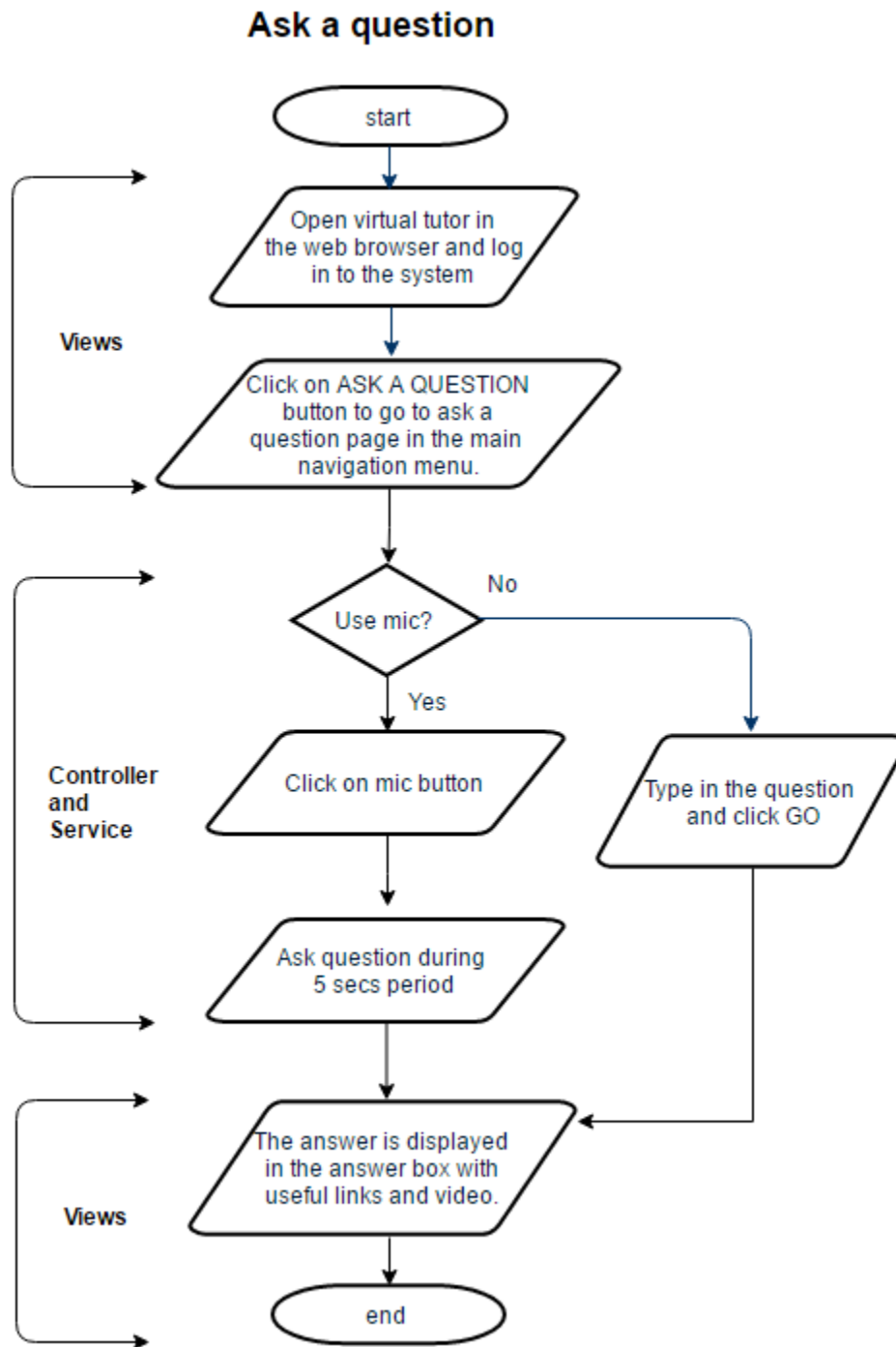


Figure 43: Ask a Question full action flow diagram

#### 4.6.6 View Chapters

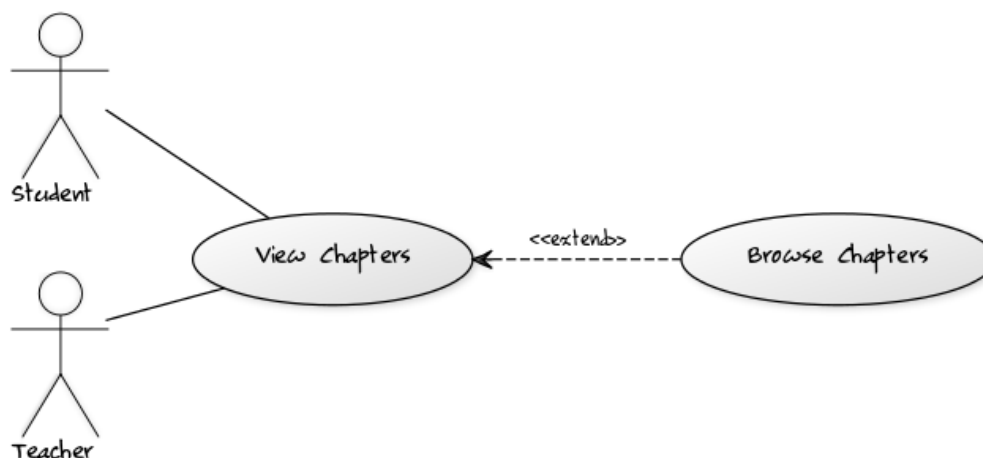


Figure 44: View Chapters Full Use Case diagram

Table 22: Chapters Use Case realization table

<b>Use Case ID</b>	<b>06</b>
<b>Use Case Name</b>	<b>View Chapters</b>
<b>Brief Description</b>	This use case enables users to view different chapters' notes and descriptions.
<b>Actors</b>	Teacher, Student
<b>Pre-condition</b>	<ol style="list-style-type: none"> <li>1. User have internet service available.</li> <li>2. User is logged in as student or teacher with correct credentials and is on the home page of the tool.</li> </ol>
<b>Step by Step description</b>	<ol style="list-style-type: none"> <li>1. User clicks on <i>STUDY MATERIALS</i> drop down menu and select <i>CHAPTERS</i> menu to go to chapters page.</li> <li>2. User browses through chapters lists (Refer to use cases 25 for search in-page functionality).</li> </ol>
<b>Scenarios:</b>	
<b>1. A regular scenario</b>	<ol style="list-style-type: none"> <li>1.             <ol style="list-style-type: none"> <li>a. User wants to browse through different</li> </ol> </li> </ol>

<p><b>2. Alternative scenarios</b></p> <p><b>3. Exception Scenario</b></p>	<p>chapters available in the tool.</p> <p>b. User logs in as a Student/Teacher with correct credentials.</p> <p>c. User is displayed with the home page of the tool.</p> <p>d. User clicks on the <i>STUDY MATERIALS</i> drop down menu and clicks on <i>CHAPTERS</i> menu on the main menu bar.</p> <p>e. User browses through different chapter's list.</p> <p>2.</p> <p>a. User clicks on a different button on the main menu bar displayed on the home page. If logged in as Student, refer use cases 02, 03, 04, 05, 07, 10 and 24. If logged in as Teacher, refer use cases 02, 03, 05, 07, 10, 13, 17, 20 and 24.</p> <p>b. User goes back to home page clicking on HOME menu.</p> <p>3. -</p>
<p><b>Post condition</b></p>	<p>User successfully browses through chapter's notes and description available in the tool.</p>

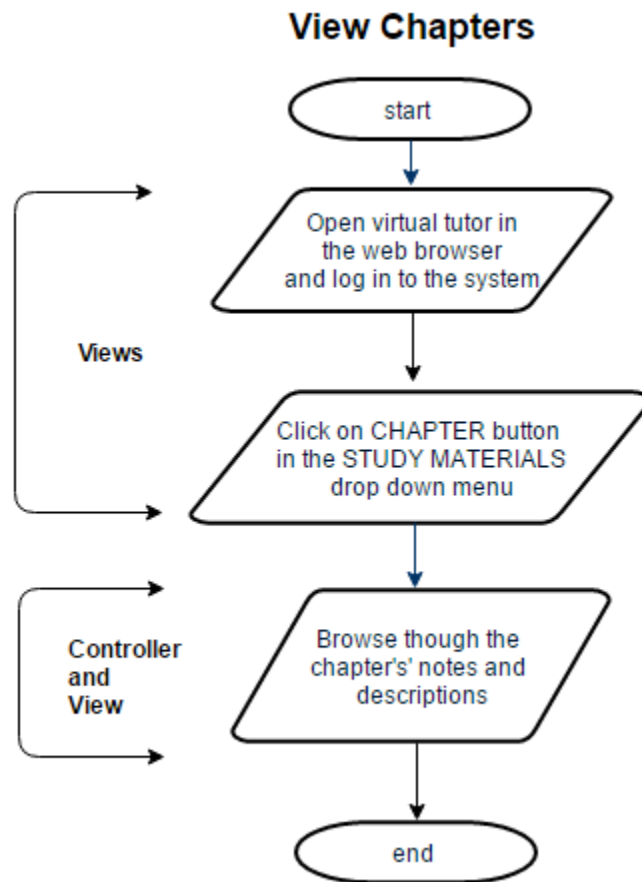


Figure 45: View Chapters full action flow diagram

#### 4.6.7 Go to Online Chat

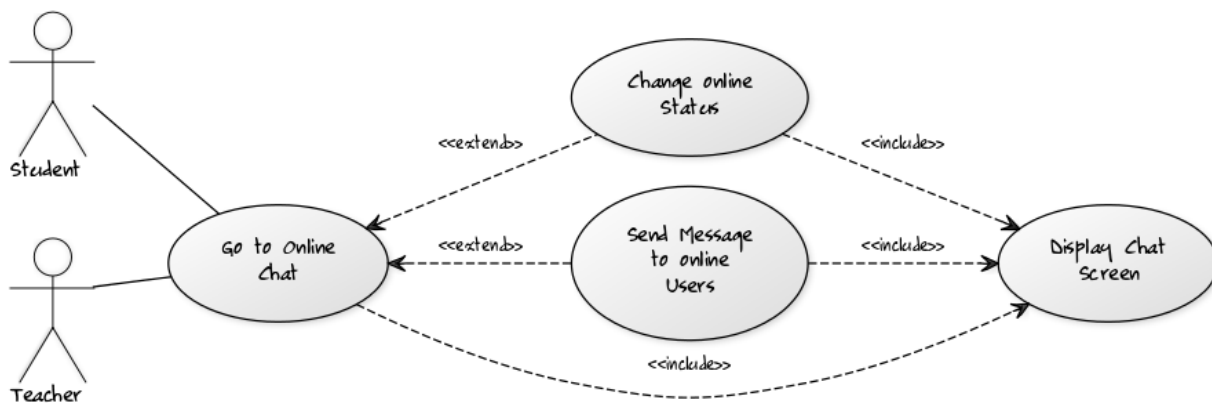


Figure 46: Go to Online Chat Full Use Case diagram

Table 23: Go to Online Chat Use Case realization table

<b>Use Case ID</b>	<b>07</b>
<b>Use Case Name</b>	<b>Go to Online Chat</b>
<b>Brief Description</b>	This use case enables users to go to online chat section where they can send messages, look at other's messages and change online status.
<b>Actors</b>	Teacher, Student
<b>Pre-condition</b>	<ol style="list-style-type: none"> <li>1. User have internet service available.</li> <li>2. User is logged in as student or teacher with correct credentials and is on the home page of the tool.</li> </ol>
<b>Step by Step description</b>	<ol style="list-style-type: none"> <li>1. User clicks on <i>ASK A TUTOR</i> drop down menu and selects <i>ONLINE CHAT</i> menu to go to online chat screen.</li> </ol>
<b>Scenarios:</b> <ol style="list-style-type: none"> <li><b>1. A regular scenario</b></li> <li><b>2. Alternative scenarios</b></li> </ol>	<ol style="list-style-type: none"> <li>1. <ol style="list-style-type: none"> <li>a. User wants to send a message to another online user in the tool.</li> <li>b. User logs in as a Student (or Teacher for other purposes) with correct credentials.</li> <li>c. User is displayed with the home page of the tool. User clicks on the <i>ASK A TUTOR</i> drop down menu and clicks on <i>ONLINE CHAT</i> menu on the main menu bar.</li> <li>d. User enters the online chat screen.</li> </ol> </li> <li>2. <ol style="list-style-type: none"> <li>a. User clicks on a different button on the main menu bar displayed on the home page. If logged in as Student, refer use cases 02, 03, 04, 05, 06, 10 and 24. If logged in as Teacher, refer use cases 02, 03, 05, 06, 10, 13, 17, 20 and 24.</li> <li>b. User goes back to home page clicking on HOME menu.</li> </ol> </li> </ol>

<b>3. Exception Scenario</b>	3. -
<b>Post condition</b>	User successfully enters online communication screen where the user can send, receive messages or change online status.

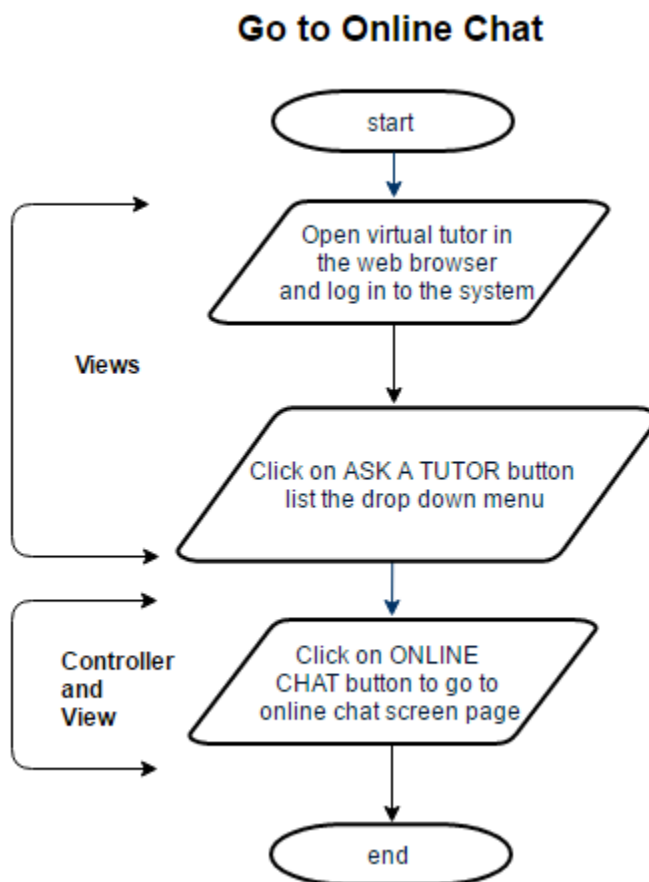


Figure 47: Go to Online Chat full action flow diagram

There are two major functions that are included in this use case (or this online chat page). Both the use cases are listed below.

#### 4.6.7.1 Send Messages to Online Users

Table 24: Send Message (in chat) Use Case realization table

<b>Use Case ID</b>	<b>08</b>
<b>Use Case Name</b>	<b>Send Messages to Online Users</b>
<b>Brief Description</b>	This use case enables users to send and receive messages to and from other online users using the tool.
<b>Actors</b>	Teacher, Student
<b>Pre-condition</b>	1. User is on the online chat page. (Refer to use case 07)
<b>Step by Step description</b>	<ol style="list-style-type: none"> <li>1. User fills in the empty text box below the ‘open room messages’ box with the message(s).</li> <li>2. User clicks on the <i>Send</i> button.</li> <li>3. Sent message is displayed in ‘open room messages’ box with the user name and message sent time stamp.</li> </ol>
<b>Scenarios:</b>  <ol style="list-style-type: none"> <li>1. <b>A regular scenario</b></li> </ol>	<ol style="list-style-type: none"> <li>1. <ol style="list-style-type: none"> <li>a. User wants to ask a question and send a message to another online user in the tool.</li> <li>b. User logs in as a Student (or Teacher for other purposes) with correct credentials.</li> <li>c. User is displayed with the home page of the tool. User clicks on the <i>ASK A TUTOR</i> drop down menu and clicks on <i>ONLINE CHAT</i> menu on the main menu bar.</li> <li>d. User enters the online chat screen.</li> <li>e. User types in message/question to other users in the text box below the online message box.</li> <li>f. User clicks ‘Send’ button to send the message</li> <li>g. The message is displayed in the online message box with user’s name and time-stamp.</li> </ol> </li> </ol>
<ol style="list-style-type: none"> <li>2. <b>Alternative scenarios</b></li> </ol>	2.



<p><b>3. Exception Scenario</b></p>	<ul style="list-style-type: none"> <li>a. User instead chooses to change the online status, refer to use case 09.</li> <li>b. User clicks on a different button on the main menu bar displayed on the home page. If logged in as Student, refer use cases 02, 03, 04, 05, 06, 10 and 24. If logged in as Teacher, refer use cases 02, 03, 05, 06, 10, 13, 17, 20 and 24.</li> <li>c. User goes back to home page clicking on HOME menu.</li> </ul> <p>3. User types in the message. Message sent failed is displayed in the status bar. The message sending process is restarted again.</p>
<p><b>Post condition</b></p>	<p>User successfully sends a message to other online users.</p>

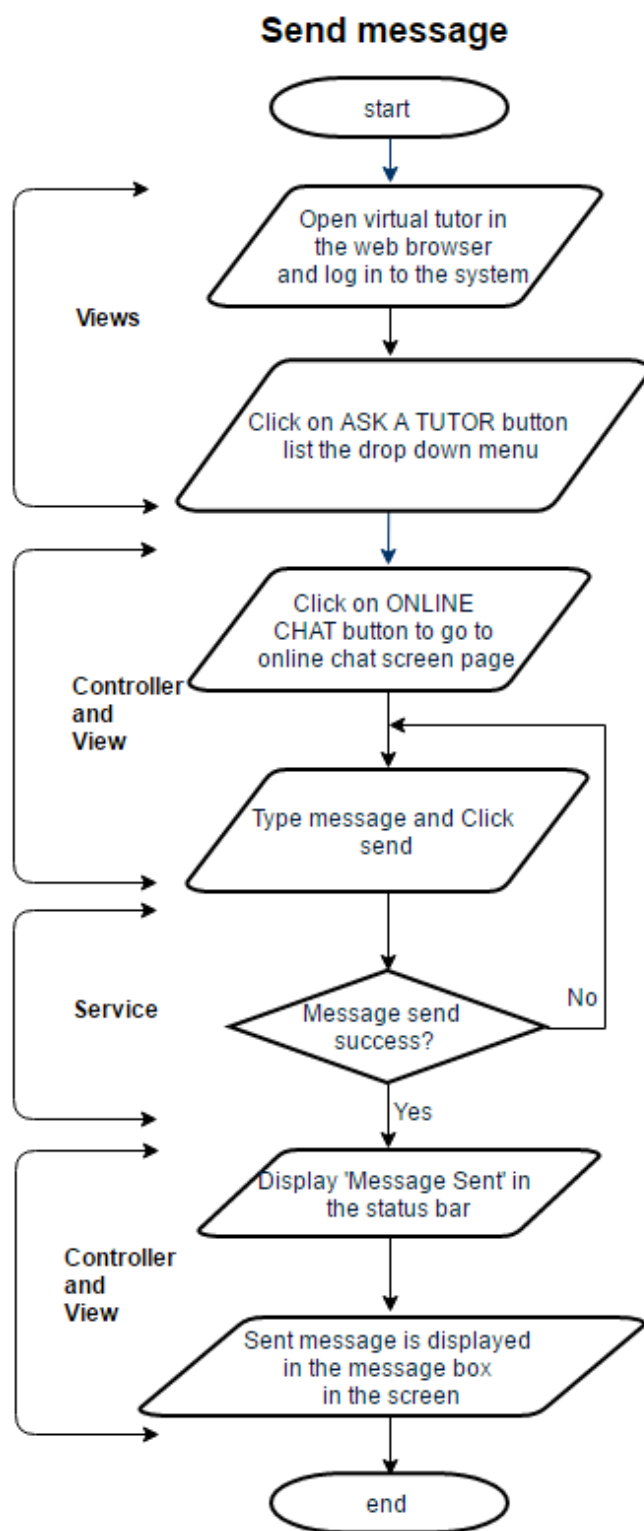


Figure 48: Send Message full action flow diagram

#### 4.6.7.2 Change Online Status

Table 25: Change Online Status Use Case realization table

<b>Use Case ID</b>	<b>09</b>
<b>Use Case Name</b>	<b>Change Online Status</b>
<b>Brief Description</b>	This use case enables users to change online status to either available, away or busy status.
<b>Actors</b>	Teacher, Student
<b>Pre-condition</b>	1. User is on the online chat page. (Refer to use case 07)
<b>Step by Step description</b>	<ol style="list-style-type: none"> <li>1. User clicks the status button in the screen.</li> <li>2. A drop-down list is opened with different statuses.</li> <li>3. User selects a status.</li> <li>4. User status is changed and visible to everyone using the tool.</li> </ol>
<b>Scenarios:</b>  <b>1. A regular scenario</b>	<ol style="list-style-type: none"> <li>1. <ol style="list-style-type: none"> <li>a. User wants to change the status from available to away.</li> <li>b. User logs in as a Student (or Teacher for other purposes) with correct credentials.</li> <li>c. User is displayed with the home page of the tool. User clicks on the <i>ASK A TUTOR</i> drop down menu and clicks on <i>ONLINE CHAT</i> menu on the main menu bar.</li> <li>d. User enters the online chat screen.</li> <li>e. User clicks the status button and chooses “away” from the drop-down list.</li> <li>f. User’s status is updated to ‘Away’ everywhere in the tool.</li> </ol> </li> </ol>
<b>2. Alternative scenarios</b>	<ol style="list-style-type: none"> <li>2. <ol style="list-style-type: none"> <li>a. User instead chooses to send messages to other online users, refer to use case 08.</li> </ol> </li> </ol>

<b>3. Exception Scenario</b>	<p>User clicks on a different button on the main menu bar displayed on the home page. If logged in as Student, refer use cases 02, 03, 04, 05, 06, 10 and 24. If logged in as Teacher, refer use cases 02, 03, 05, 06, 10, 13, 17, 20 and 24.</p> <ul style="list-style-type: none"><li>b. User goes back to home page clicking on HOME menu.</li></ul> <p>3. User wants to change the status. Status change is failed due to loss of connection to the server. User tries to change the status again.</p>
<b>Post condition</b>	User successfully changes the online status which is visible to other users.

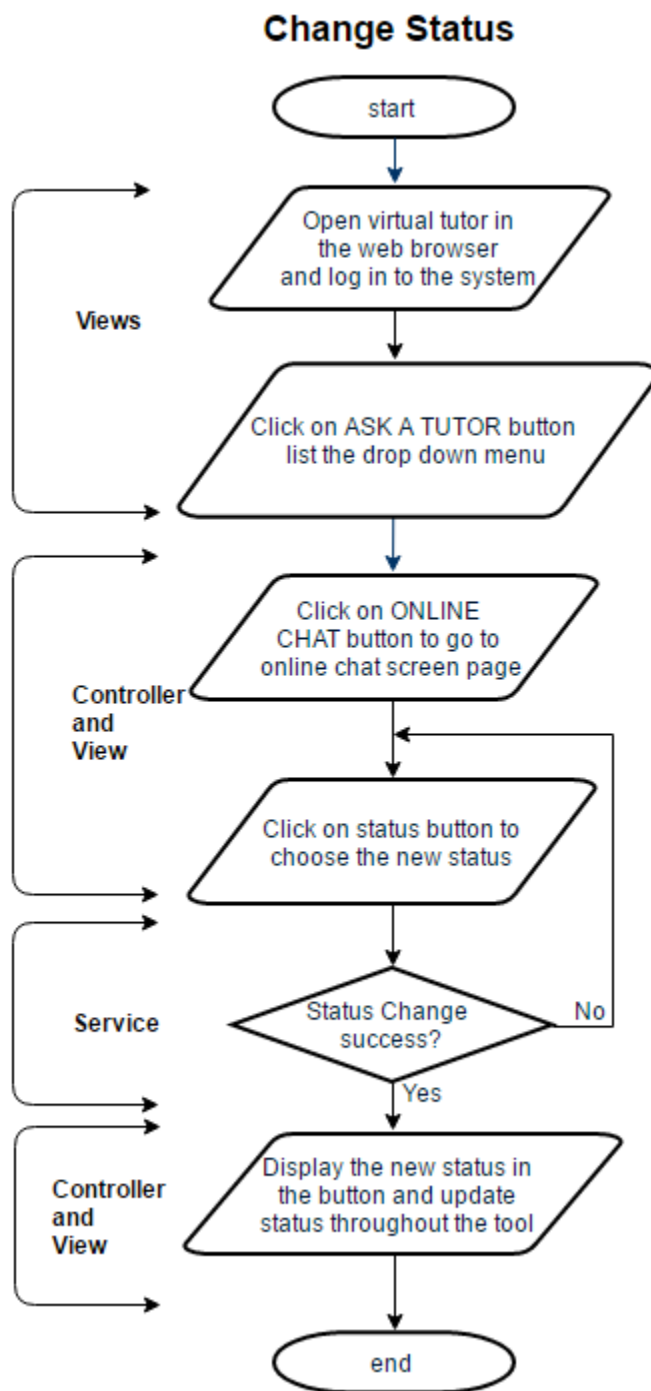


Figure 49: Change Status full action flow diagram

#### 4.6.8 Start Share Screen

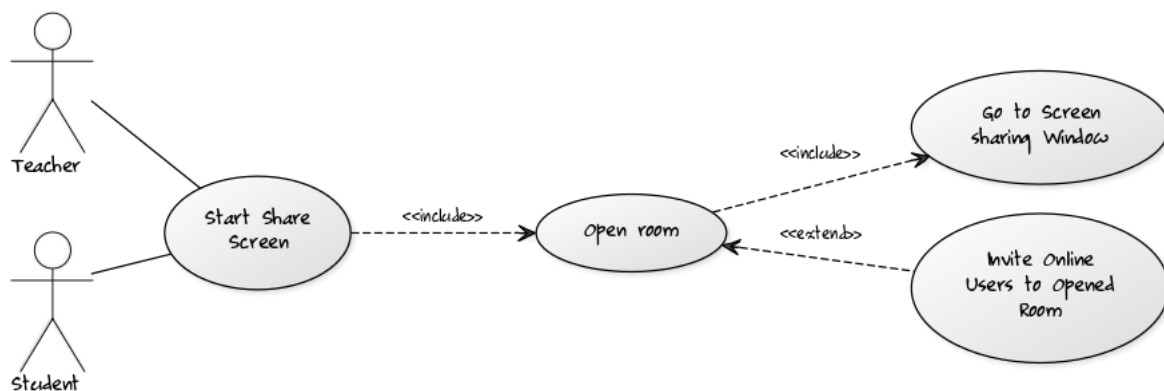


Figure 50: Share Screen Full Use Case diagram

Table 26: Start Screen Sharing Use Case realization table

<b>Use Case ID</b>	<b>10</b>
<b>Use Case Name</b>	<b>Start Share Screen</b>
<b>Brief Description</b>	This use case enables users to open a screen sharing room or join opened room with other online users of the tool.
<b>Actors</b>	Teacher, Student
<b>Pre-condition</b>	<ol style="list-style-type: none"> <li>1. User have internet service available.</li> <li>2. User is logged in as student or teacher with correct credentials.</li> </ol>
<b>Step by Step description</b>	<ol style="list-style-type: none"> <li>1. User clicks on <i>ASK A TUTOR</i> drop down menu and selects <i>SHARE SCREEN</i> menu to go to online screen sharing screen.</li> <li>2. User clicks on “Open room” button.</li> <li>3. User selects the window to be shared.</li> <li>4. User clicks on “Share” to finally start sharing the screen.</li> </ol>
<b>Scenarios:</b>	



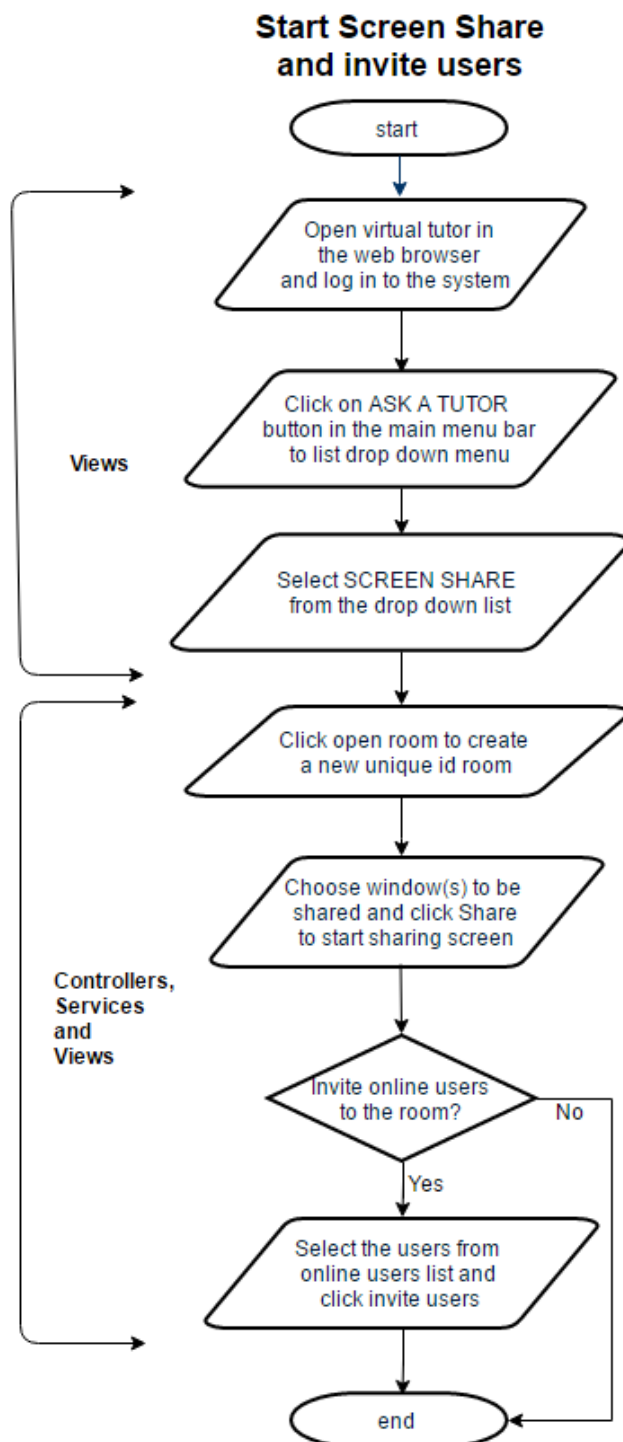


Figure 51: Start Screen Share and invite users full action flow diagram



Once the user opens a room, there is a sub use-case where the user can invite other users to the created screen sharing room. The use-case is shown in the table below.

#### 4.6.8.1 Invite Users

Table 27: Invite Users Use Case realization table

<b>Use Case ID</b>	<b>11</b>
<b>Use Case Name</b>	<b>Invite Users</b>
<b>Brief Description</b>	This use case enables a user to invite other online users to the opened screen sharing room.
<b>Actors</b>	Teacher, Student
<b>Pre-condition</b>	1. User is in Screen sharing page. (Refer to use case 10)
<b>Step by Step description</b>	<ol style="list-style-type: none"> <li>1. User selects a specific user(s) from “Online user” lists or the user clicks on “Select All” button to select all online users at once.</li> <li>2. User clicks “Send Invites” button to invite all the selected user to the room with unique to the room.</li> </ol>
<b>Scenarios:</b>  <b>1. A regular scenario</b>	<ol style="list-style-type: none"> <li>1. <ol style="list-style-type: none"> <li>a. User wants to invite the specific user to the opened room.</li> <li>b. User logs in as a Student (or Teacher for other purposes) with correct credentials.</li> <li>c. User is displayed with the home page of the tool. User clicks on the <i>ASK A TUTOR</i> drop down menu and clicks on <i>SHARE SCREEN</i> menu on the main menu bar.</li> <li>d. User enters the Screen sharing page.</li> <li>e. User opens a room with a unique id.</li> <li>f. User selects the desired user on the list of online users to invite to the room.</li> <li>g. The invitation is sent to the chosen user(s).</li> </ol> </li> </ol>



Table 28: Join Room Use Case realization table

<b>Use Case ID</b>	<b>12</b>
<b>Use Case Name</b>	<b>Join Room</b>
<b>Brief Description</b>	This use case enables users to join an opened screen sharing room.
<b>Actors</b>	Teacher, Student
<b>Pre-condition</b>	<ol style="list-style-type: none"> <li>1. User have internet service available.</li> <li>2. User is logged in as student or teacher with correct credentials.</li> <li>3. Invited room exists and is not expired.</li> <li>4. User either gets an invitation to join the room or has unique URL to the room.</li> </ol>
<b>Step by Step description</b>	<ol style="list-style-type: none"> <li>1. An invitation dialog box pops up when other users send a request/invitation to join the room.</li> <li>2. User clicks on “Join room” button.</li> <li>3. User is directed to opened room in the new browser tab.</li> </ol> <p style="text-align: center;">OR</p> <ol style="list-style-type: none"> <li>4. If the user already has a unique URL to the room, the user opens a new browser tab and go to the given URL to enter the room.</li> </ol>
<b>Scenarios:</b> <ol style="list-style-type: none"> <li><b>1. A regular scenario</b></li> <li><b>2. Alternative scenarios</b></li> </ol>	<ol style="list-style-type: none"> <li>1. <ol style="list-style-type: none"> <li>a. User wants to join a room when invitation appears in the window.</li> <li>b. User clicks on join room button to be re-directed to the opened screen sharing room.</li> </ol> <p style="text-align: center;">OR</p> <ol style="list-style-type: none"> <li>c. User enters unique URL to the new tab and goes to the room.</li> </ol> </li> <li>2. <ol style="list-style-type: none"> <li>a. User instead chooses to cancel the screen</li> </ol> </li> </ol>

<b>3. Exception Scenario</b>	sharing invitation. 3. a. User joins the room but room already expired.
<b>Post condition</b>	User successfully joins the screen sharing room after invitation or using the unique URL.

#### 4.6.10 Manage Members

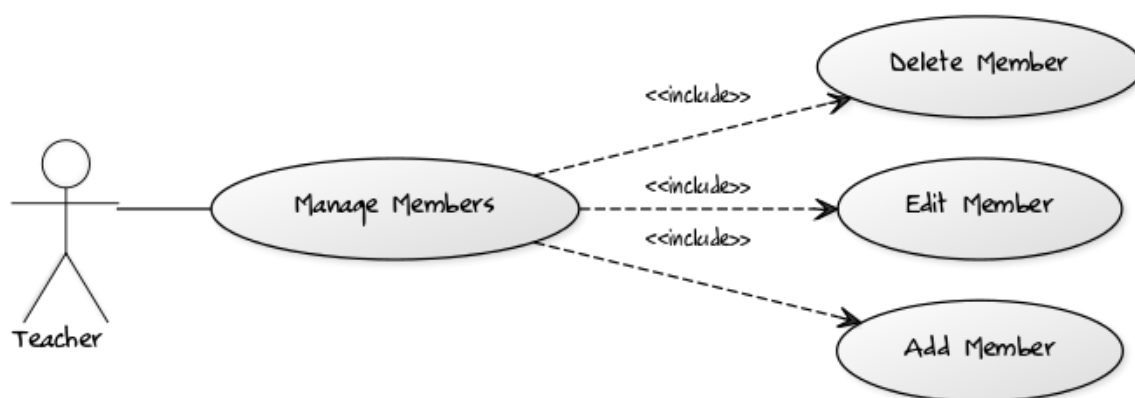


Figure 53: Manage Members Full Use Case diagram

Table 29: Manage Members Use Case realization table

<b>Use Case ID</b>	<b>13</b>
<b>Use Case Name</b>	<b>Manage Members</b>
<b>Brief Description</b>	This use case enables teachers to add and delete members and manage the information of existing members.
<b>Actors</b>	Teacher
<b>Pre-condition</b>	1. User have internet service available. 2. User is logged in as a teacher with correct credentials.



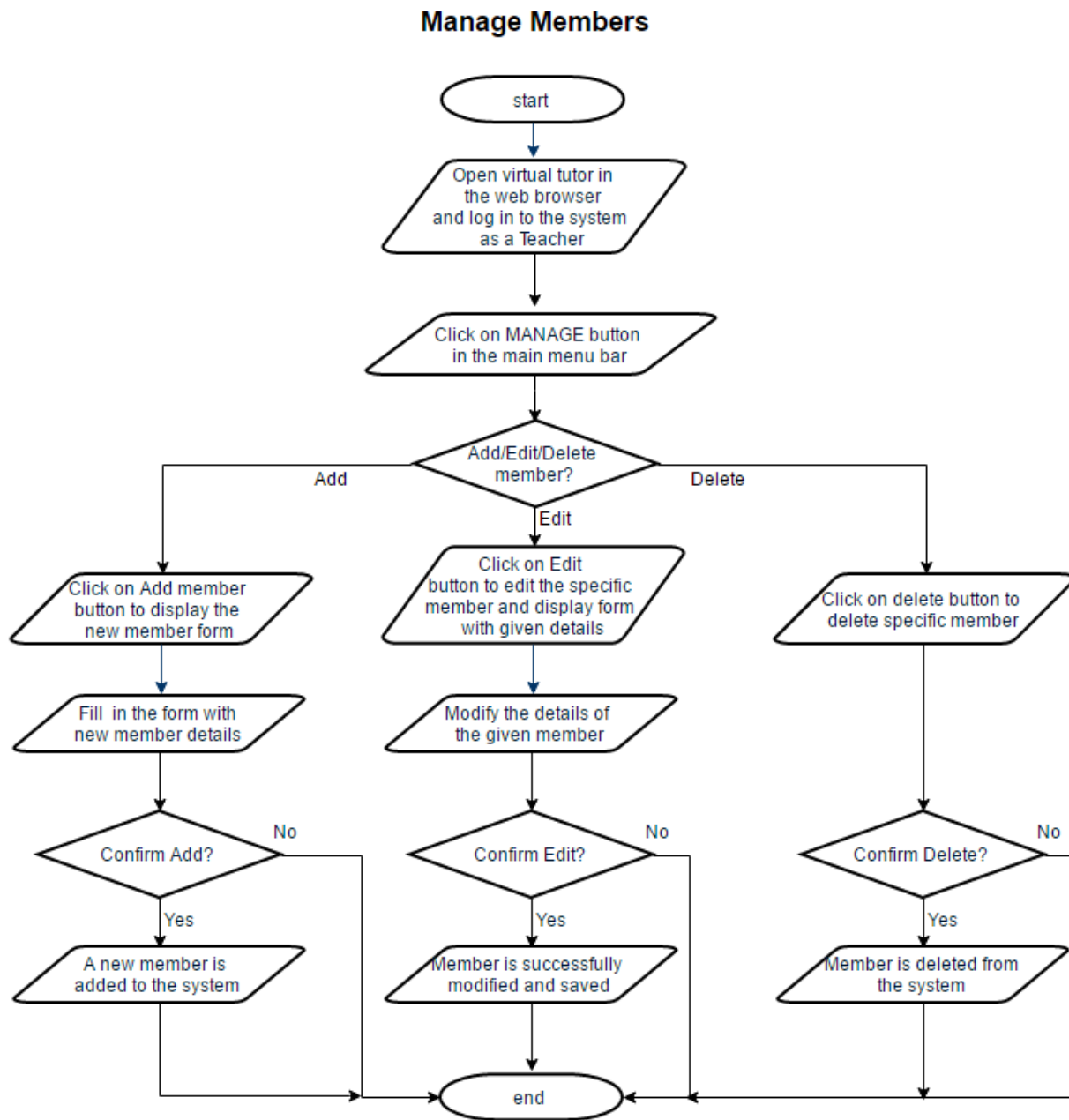


Figure 54: Manage Members full action flow diagram

Based on the requirement, this use case enables the user to perform either add, edit or delete the members. Below are the three use cases that describe the actions in details.

#### 4.6.10.1 Add Member

Table 30: Add Members Use Case realization table

<b>Use Case ID</b>	<b>14</b>
<b>Use Case Name</b>	<b>Add Members</b>
<b>Brief Description</b>	This use case enables teachers to add a new member to the system.
<b>Actors</b>	Teacher
<b>Pre-condition</b>	<ol style="list-style-type: none"> <li>1. User have internet service available.</li> <li>2. User is logged in as a teacher with correct credentials.</li> <li>3. User is in MANAGE page in the tool.</li> </ol>
<b>Step by Step description</b>	<ol style="list-style-type: none"> <li>1. User clicks on <i>Add Member</i> button in the screen that displays a new member form.</li> <li>2. User fills in the form with new user's detail information.</li> <li>3. User clicks "Confirm Add" button in the form to finally add a new member to the system.</li> </ol>
<b>Scenarios:</b> <ol style="list-style-type: none"> <li><b>1. A regular scenario</b></li> <li><b>2. Alternative scenarios</b></li> </ol>	<ol style="list-style-type: none"> <li>1. <ol style="list-style-type: none"> <li>a. User wants to add a new member to the system.</li> <li>b. User goes to the <i>MANAGE</i> page.</li> <li>c. User clicks on Add member button to display new member form.</li> <li>d. User fills in a form with the member details.</li> <li>e. User confirms the add by clicking "Confirm Add" button in the form.</li> <li>f. The new member is added to the system.</li> </ol> </li> <li>2. User clicks on "Cancel" button in the add form to cancel the whole process.</li> </ol>

<b>3. Exception Scenario</b>	3. -
<b>Post condition</b>	User is successfully able to add a new member to the system.

#### 4.6.10.2 Edit Member

Table 31: Edit Members Use Case realization table

<b>Use Case ID</b>	<b>15</b>
<b>Use Case Name</b>	<b>Edit Members</b>
<b>Brief Description</b>	This use case enables teachers to edit/modify existing member in the system.
<b>Actors</b>	Teacher
<b>Pre-condition</b>	<ol style="list-style-type: none"> <li>1. User have internet service available.</li> <li>2. User is logged in as a teacher with correct credentials.</li> <li>3. User is on the MANAGE page in the tool.</li> </ol>
<b>Step by Step description</b>	<ol style="list-style-type: none"> <li>1. User clicks on <i>Edit</i> button in the screen to modify specific member. A form with original information of the member is displayed.</li> <li>2. User fills in the form with the modified member's information.</li> <li>3. User clicks "Confirm Edit" button in the form to finally edit member in the system.</li> </ol>
<b>Scenarios:</b>	
<b>1. A regular scenario</b>	<ol style="list-style-type: none"> <li>1. <ol style="list-style-type: none"> <li>a. User wants to edit a member in the system.</li> <li>b. User goes to the <i>MANAGE</i> page.</li> </ol> </li> </ol>



<p><b>2. Alternative scenarios</b></p> <p><b>3. Exception Scenario</b></p>	<p>c. User clicks on Edit member button for the given member that displays a form with original information of the member.</p> <p>d. User fills in a form with the new details for the member.</p> <p>e. User confirms the edit by clicking “Confirm Edit” button in the form.</p> <p>f. Member is modified in the system.</p> <p>2. User clicks on “Cancel” button in the edit form to cancel the whole process.</p> <p>3. -</p>
<p><b>Post condition</b></p>	<p>User is successfully able to edit existing member in the system.</p>

#### 4.6.10.3 Delete Member

Table 32: Delete Members Use Case realization table

<p><b>Use Case ID</b></p>	<p><b>16</b></p>
<p><b>Use Case Name</b></p>	<p><b>Delete Members</b></p>
<p><b>Brief Description</b></p>	<p>This use case enables teachers to delete existing member from the system.</p>
<p><b>Actors</b></p>	<p>Teacher</p>
<p><b>Pre-condition</b></p>	<p>1. User have internet service available.</p> <p>2. User is logged in as a teacher with correct credentials.</p> <p>3. User is on the MANAGE page in the tool.</p>
<p><b>Step by Step description</b></p>	<p>1. User clicks on <i>Delete</i> button in the screen to delete a specific member. A dialog box appears to confirm the deletion of the member.</p>

	2. User clicks “Delete” button in the confirmation window to finally delete a member from the system.
<b>Scenarios:</b>  <b>1. A regular scenario</b>         <b>2. Alternative scenarios</b>  <b>3. Exception Scenario</b>	1. <ol style="list-style-type: none"> <li>a. User wants to delete a member from the system.</li> <li>b. User goes to the <i>MANAGE</i> page.</li> <li>c. User clicks on Delete button for the given member that displays a confirmation window to delete the member from the system.</li> <li>d. User confirms the deletion by clicking “Delete” button in the form.</li> <li>e. Member is deleted from the system.</li> </ol> 2. User cancels the whole process of deletion. 3. -
<b>Post condition</b>	User is successfully able to delete existing member from the system.

#### 4.6.11 Manage Messages

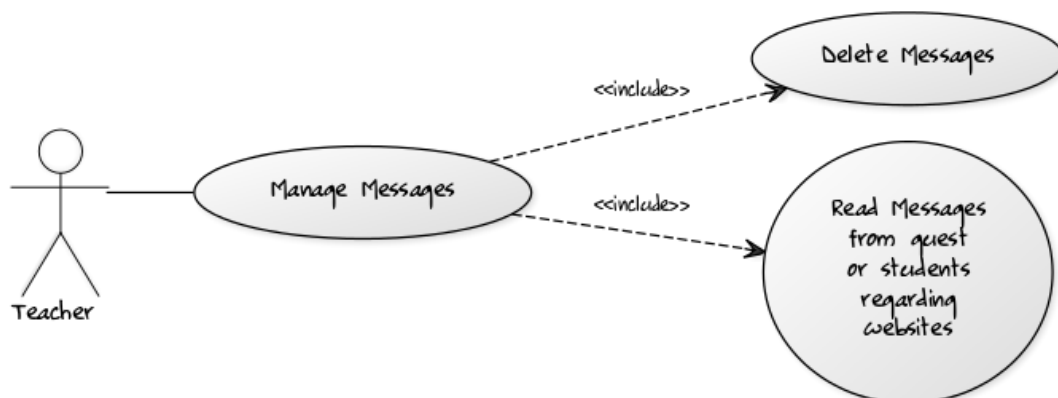


Figure 55: Manage Members Full Use Case diagram

Table 33: Manage Messages Use Case realization table

<b>Use Case ID</b>	<b>17</b>
<b>Use Case Name</b>	<b>Manage Messages</b>
<b>Brief Description</b>	This use case enables teachers to manage messages sent by guests or students using the tool. Managing includes reading and deleting the messages from the system.
<b>Actors</b>	Teacher
<b>Pre-condition</b>	<ol style="list-style-type: none"> <li>1. User have internet service available.</li> <li>2. User is logged in as a teacher with correct credentials.</li> </ol>
<b>Step by Step description</b>	<ol style="list-style-type: none"> <li>1. User clicks on <i>MESSAGES</i> menu to go to manage messages page on the main menu bar.</li> <li>2. For rest of the steps, refer to use cases 18 and 19.</li> </ol>
<b>Scenarios:</b>  <ol style="list-style-type: none"> <li><b>1. A regular scenario</b></li>            <li><b>2. Alternative scenarios</b></li> </ol>	<ol style="list-style-type: none"> <li>1. <ol style="list-style-type: none"> <li>a. User wants to go to manage messages page to read or delete messages.</li> <li>b. User logs in as a Teacher with correct credentials.</li> <li>c. User is displayed with the home page of the tool.</li> <li>d. User clicks on the <i>MESSAGES</i> menu on the main menu bar.</li> <li>e. Based on the requirement, the user performs the actions, refer to use cases 18 or 19.</li> </ol> </li>   <li>2. <ol style="list-style-type: none"> <li>a. User clicks on a different button on the main menu bar displayed on the home page. Refer use cases 02, 03, 05, 06, 07, 10, 13, 20 and 24.</li> <li>b. User goes back to home page clicking on HOME menu.</li> </ol> </li> </ol>

<p><b>3. Exception Scenario</b></p>	<p>3. -</p>
<p><b>Post condition</b></p>	<p>User is successfully able to read or delete messages upon requirements.</p>

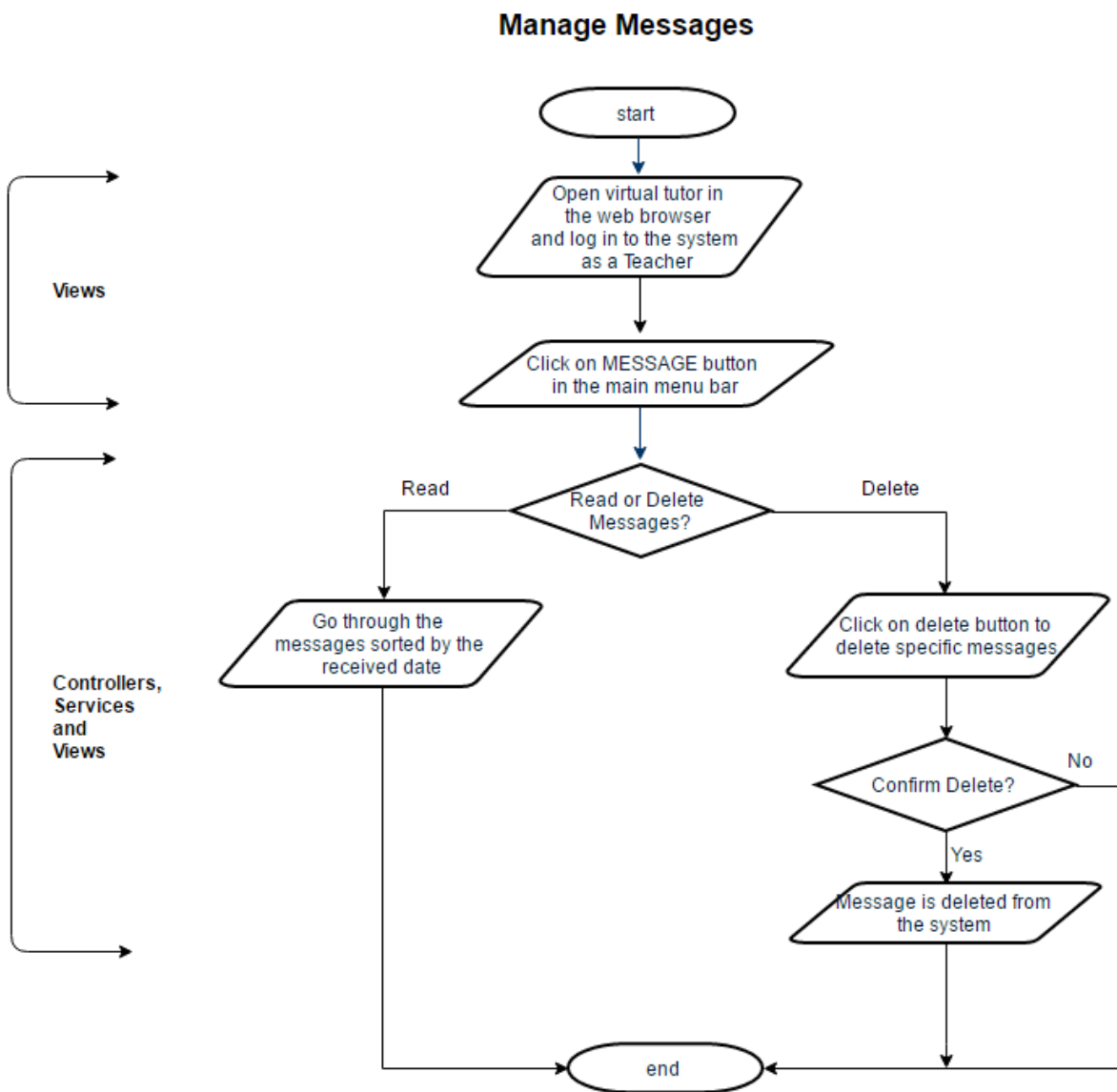


Figure 56: Manage Messages full action flow diagram

Based on the requirement, this use case enables the user to perform either read or delete the messages sent by guests or students. Below are the two use cases that describe the actions in details.

#### 4.6.11.1 Read Messages


Table 34: Read Messages Use Case realization table

<b>Use Case ID</b>	<b>18</b>
<b>Use Case Name</b>	<b>Read Messages</b>
<b>Brief Description</b>	This use case enables teachers to read messages (queries and concerns from students and guests) in the system.
<b>Actors</b>	Teacher
<b>Pre-condition</b>	<ol style="list-style-type: none"> <li>1. User have internet service available.</li> <li>2. User is logged in as a teacher with correct credentials.</li> <li>3. User is in MESSAGES page in the tool.</li> </ol>
<b>Step by Step description</b>	<ol style="list-style-type: none"> <li>1. User reads the messages in the table displayed.</li> </ol>
<b>Scenarios:</b> <ol style="list-style-type: none"> <li><b>1. A regular scenario</b></li> <li><b>2. Alternative scenarios</b></li> </ol>	<ol style="list-style-type: none"> <li>1. <ol style="list-style-type: none"> <li>a. User wants to read messages from other members in the system.</li> <li>b. User goes to the <i>MESSAGE</i> page.</li> <li>c. User reads the necessary message displayed in the table ordered by the received date.</li> </ol> </li> <li>2. <ol style="list-style-type: none"> <li>a. User clicks on a different button on the main menu bar displayed on the home page. Refer use cases 02, 03, 05, 06, 07, 10, 13, 20 and 24.</li> <li>b. User goes back to home page clicking on HOME menu.</li> </ol> </li> </ol>

<b>3. Exception Scenario</b>	3. -
<b>Post condition</b>	User is successfully able to read all necessary messages sent by other users (student and guests) in the system.

#### 4.6.11.2 Delete Messages

Table 35: Delete Messages Use Case realization table

<b>Use Case ID</b>	<b>19</b>
<b>Use Case Name</b>	<b>Delete Messages</b>
<b>Brief Description</b>	This use case enables teachers to delete messages sent by other members from the system.
<b>Actors</b>	Teacher
<b>Pre-condition</b>	<ol style="list-style-type: none"> <li>1. User have internet service available.</li> <li>2. User is logged in as a teacher with correct credentials.</li> <li>3. User is in MESSAGE page in the tool.</li> </ol>
<b>Step by Step description</b>	<ol style="list-style-type: none"> <li>1. User clicks on  button in the screen to delete a specific message. A dialog box appears to confirm the deletion of the message.</li> <li>2. User clicks “Confirm Delete” button in the confirmation window to finally delete the message from the system.</li> </ol>
<b>Scenarios:</b>	
<b>1. A regular scenario</b>	<ol style="list-style-type: none"> <li>1. <ol style="list-style-type: none"> <li>a. User wants to delete a message from the system.</li> <li>b. User goes to the <i>MESSAGE</i> page.</li> </ol> </li> </ol>

<p><b>2. Alternative scenarios</b></p> <p><b>3. Exception Scenario</b></p>	<p>c. User clicks on <b>x</b> button for the given message to be deleted and a confirmation window is displayed.</p> <p>d. User confirms the deletion by clicking “Confirm Delete” button in the form.</p> <p>e. The message is deleted from the system.</p> <p>2. User cancels the whole process of deletion by clicking on “Cancel” button in the confirmation window.</p> <p>3. -</p>
<p><b>Post condition</b></p>	<p>User is successfully able to delete existing message from the system.</p>

#### 4.6.12 Manage Lab Assistant Hours

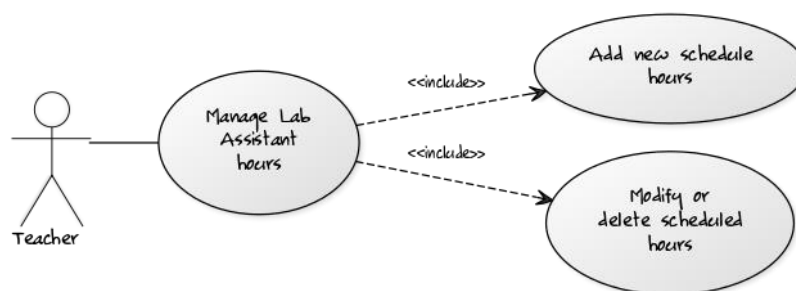


Figure 57: Manage Lab Assistant hours’ full Use Case diagram

Table 36: Manage Lab Assistant Hours Use Case realization table

<p><b>Use Case ID</b></p>	<p><b>20</b></p>
<p><b>Use Case Name</b></p>	<p><b>Manage Lab Assistant hours</b></p>
<p><b>Brief Description</b></p>	<p>This use case enables teachers to add, modify and delete lab assistant hours and manage the overall lab assistant schedule.</p>
<p><b>Actors</b></p>	<p>Teacher</p>

<b>Pre-condition</b>	<ol style="list-style-type: none"> <li>1. User have internet service available.</li> <li>2. User is logged in as a teacher with correct credentials.</li> </ol>
<b>Step by Step description</b>	<ol style="list-style-type: none"> <li>1. User clicks on <i>STUDY MATERIALS</i> to list the drop-down menu.</li> <li>2. User clicks on <i>LAB ASSISTANT HOURS</i> menu and is redirected to the lab assistant hours' page.</li> <li>3. Based on requirement steps are performed as per use cases 21 and 22.</li> </ol>
<p><b>Scenarios:</b></p> <p><b>1. A regular scenario</b></p> <p><b>2. Alternative scenarios</b></p> <p><b>3. Exception Scenario</b></p>	<ol style="list-style-type: none"> <li>1. <ol style="list-style-type: none"> <li>a. User wants to go to manage lab assistant hours' page to add new hours to the schedule table.</li> <li>b. User logs in as a Teacher with correct credentials.</li> <li>c. User is displayed with the home page of the tool.</li> <li>d. User clicks on the <i>STUDY MATERIALS</i> on the main menu to list drop down and clicks on <i>LAB ASSISTANT HOURS</i> to go to the lab assistant hours' page.</li> <li>e. User performs the actions listed on use cases 21.</li> </ol> </li> <li>2. <ol style="list-style-type: none"> <li>a. User clicks on a different button on the main menu bar displayed on the home page. Refer use cases 02, 03, 05, 06, 07, 10, 13, 17 and 24.</li> <li>b. User goes back to home page clicking on HOME menu.</li> </ol> </li> <li>3. -</li> </ol>
<b>Post condition</b>	User is successfully able to add, edit or delete hours in the lab assistant hour schedule upon requirements.



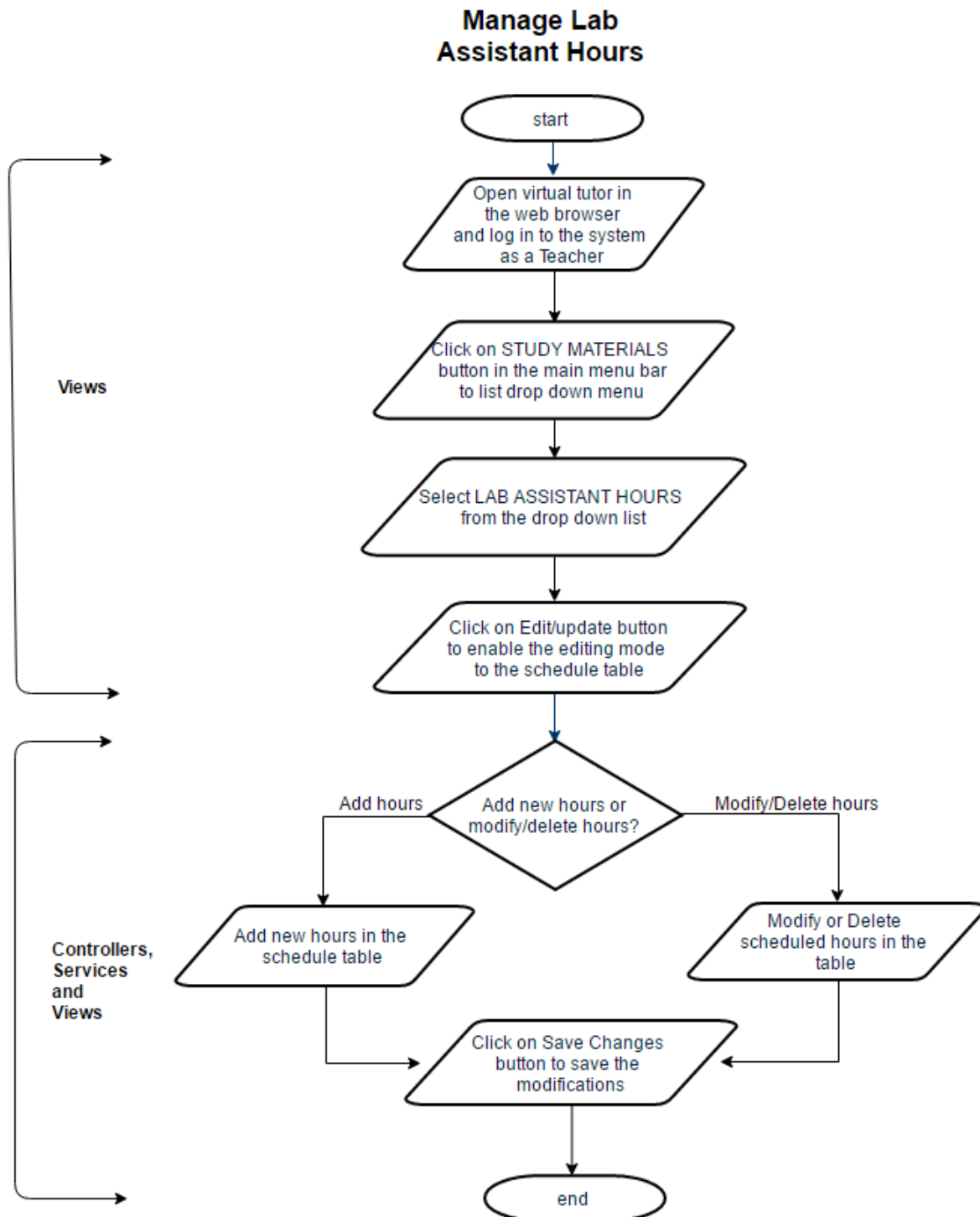


Figure 58: Manage Lab Assistant hours' full action flow diagram

Based on the requirement, this use case enables the user to perform either add, edit or delete the members. Below are the three use cases that describe the actions in details.

#### 4.6.12.1 Add New Schedule Hours

Table 37: Add new schedule Use Case realization table

<b>Use Case ID</b>	<b>21</b>
<b>Use Case Name</b>	<b>Add New Schedule Hours</b>
<b>Brief Description</b>	This use case enables teachers to add new schedule hours to the schedule table.
<b>Actors</b>	Teacher
<b>Pre-condition</b>	<ol style="list-style-type: none"> <li>1. User have internet service available.</li> <li>2. User is logged in as a teacher with correct credentials.</li> <li>3. User is in <i>LAB ASSISTANT HOURS</i> page in the tool.</li> </ol>
<b>Step by Step description</b>	<ol style="list-style-type: none"> <li>1. User clicks on <i>Edit/Update</i> button in the screen to make the schedule table editable.</li> <li>2. User fills in the new hours in the table.</li> <li>3. User clicks <i>Save Changes</i> button in the page to finally save the newly added hours to the schedule table.</li> </ol>
<b>Scenarios:</b>	
<ol style="list-style-type: none"> <li>1. A regular scenario</li> </ol>	<ol style="list-style-type: none"> <li>1.             <ol style="list-style-type: none"> <li>a. User wants to add a new hour to the system.</li> <li>b. User goes to the <i>LAB ASSISTANT HOURS</i> page.</li> <li>c. User clicks on <i>Edit/Update</i> button to make the table editable.</li> <li>d. User fills in new hours in the table.</li> <li>e. User confirms the add by clicking <i>Save Changes</i> button in the page.</li> <li>f. New hours are added to the schedule table.</li> </ol> </li> </ol>

<b>2. Alternative scenarios</b>	2. User clicks on a different button on the main menu bar. Refer use cases 02, 03, 05, 06, 07, 10, 13, 17 and 24.
<b>3. Exception Scenario</b>	3. -
<b>Post condition</b>	User is successfully able to add a new member to the system.

#### 4.6.12.2 Modify or Delete Scheduled Hours

Table 38: Modify or Delete Scheduled Hours Use Case realization table

<b>Use Case ID</b>	<b>22</b>
<b>Use Case Name</b>	<b>Modify or Delete Scheduled Hours</b>
<b>Brief Description</b>	This use case enables teachers to modify or delete existing hours in the schedule table.
<b>Actors</b>	Teacher
<b>Pre-condition</b>	<ol style="list-style-type: none"> <li>1. User have internet service available.</li> <li>2. User is logged in as a teacher with correct credentials.</li> <li>3. User is in <i>LAB ASSISTANT HOURS</i> page in the tool.</li> </ol>
<b>Step by Step description</b>	<ol style="list-style-type: none"> <li>1. User clicks on <i>Edit/Update</i> button in the screen to make the schedule table editable.</li> <li>2. User modifies or deletes the hours from the schedule table as needed.</li> <li>3. User clicks <i>Save Changes</i> button in the page to finally save the modification or deletion of hours to the schedule table.</li> </ol>
<b>Scenarios:</b>	

<p><b>1. A regular scenario</b></p> <p><b>2. Alternative scenarios</b></p> <p><b>3. Exception Scenario</b></p>	<ol style="list-style-type: none"> <li>1.             <ol style="list-style-type: none"> <li>a. User wants to delete certain lab hours from the schedule table.</li> <li>b. User goes to the <i>LAB ASSISTANT HOURS</i> page.</li> <li>c. User clicks on <i>Edit/Update</i> button to make the table editable.</li> <li>d. User deletes the required hours from the table.</li> <li>e. User confirms the deletion by clicking <i>Save Changes</i> button in the page.</li> <li>f. Desired hours are deleted from the schedule table.</li> </ol> </li> <li>2. User clicks on a different button on the main menu bar. Refer use cases 02, 03, 05, 06, 07, 10, 13, 17 and 24.</li> <li>3. -</li> </ol>
<p><b>Post condition</b></p>	<p>User is successfully able to modify or delete scheduled hours in the schedule table.</p>

#### 4.6.13 Manage Content on the Website

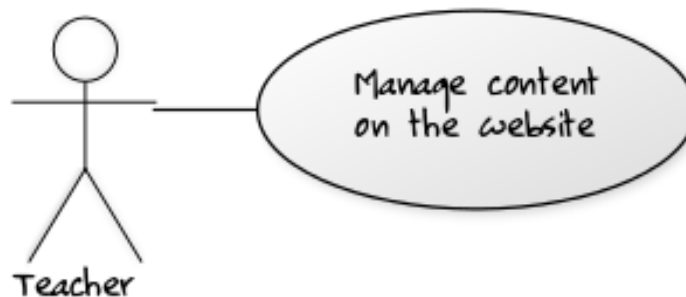


Figure 59: Manage Content on the website full Use Case diagram

Table 39: Manage Content on the website Use Case realization table

<b>Use Case ID</b>	<b>23</b>
<b>Use Case Name</b>	<b>Manage Content on the website</b>
<b>Brief Description</b>	This use case enables teachers to add, modify or delete the content on the website as needed directly from the backend.
<b>Actors</b>	Teacher
<b>Pre-condition</b>	<ol style="list-style-type: none"> <li>1. User has internet service available.</li> <li>2. User has direct access to the website database with correct credentials.</li> </ol>
<b>Step by Step description</b>	<ol style="list-style-type: none"> <li>1. User updates the content in the database.</li> <li>2. User saves the changes in the database.</li> </ol>
<b>Scenarios:</b> <ol style="list-style-type: none"> <li><b>1. A regular scenario</b></li> <li><b>2. Alternative scenarios</b></li> <li><b>3. Exception Scenario</b></li> </ol>	<ol style="list-style-type: none"> <li>1. <ol style="list-style-type: none"> <li>a. User wants to add a new set of question and answer for ask a question section of the tool.</li> <li>b. User makes the necessary changes in the database.</li> <li>c. User saves the necessary changes.</li> </ol> </li> <li>2. –</li> <li>3. User is not able to make changes because the user does not have write permission.</li> </ol>
<b>Post condition</b>	User is successfully able to add, edit or delete contents of the page directly in the backend.

#### 4.6.14 Log Out

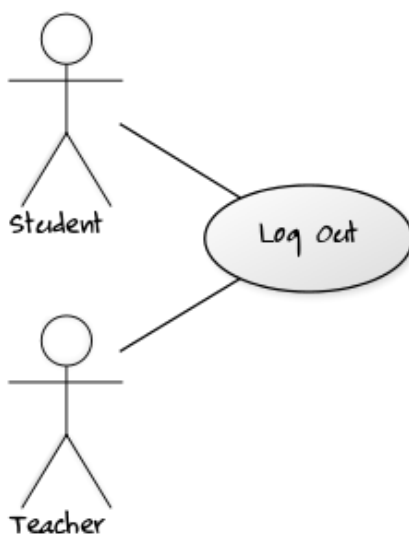


Figure 60: Log Out Full Use Case diagram

Table 40: Log Out Use Case realization table

<b>Use Case ID</b>	<b>24</b>
<b>Use Case Name</b>	<b>Log Out</b>
<b>Brief Description</b>	This use case enables users to log out of the system and end the session.
<b>Actors</b>	Teacher, Student
<b>Pre-condition</b>	<ol style="list-style-type: none"> <li>1. User have internet service available.</li> <li>2. User is logged in as Student or Teacher.</li> </ol>
<b>Step by Step description</b>	<ol style="list-style-type: none"> <li>1. User clicks on LOG OUT menu in the main navigation bar.</li> <li>2. A log out screen appears and the user is successfully logged out of the system.</li> </ol>
<b>Scenarios:</b>	
<b>1. A regular scenario</b>	1.

<p><b>2. Alternative scenarios</b></p> <p><b>3. Exception Scenario</b></p>	<p>a. User wants to terminate the current session on the website.</p> <p>b. User clicks on LOG OUT button on the main navigation bar.</p> <p>c. User successfully logs out of the system.</p> <p>2.</p> <p>a. User clicks on a different button on the main menu bar. If logged in as Student, refer use cases 2, 3, 4, 5, 6, 7 and 10. If logged in as Teacher, refer use cases 2, 3, 4, 5, 6, 7, 10, 13, 17 and 20.</p> <p>b. User goes back to home page clicking on HOME menu</p> <p>3. -</p>
<p><b>Post condition</b></p>	<p>User successfully logs out of the system.</p>

#### 4.6.15 Search Term in the Page

Following is the additional use case determined to be important for two use cases above, refer use cases 02 and 06.

Table 41: Search term in the page Use Case realization table

<p><b>Use Case ID</b></p>	<p><b>25</b></p>
<p><b>Use Case Name</b></p>	<p><b>Search term in the page</b></p>
<p><b>Brief Description</b></p>	<p>This use case enables users to search for the related content within the page.</p>
<p><b>Actors</b></p>	<p>Teacher, Student</p>
<p><b>Pre-condition</b></p>	<p>1. User have internet service available. 2. User is logged in as Student or Teacher.</p>

	3. User is either in <i>FAQ</i> or <i>Chapters</i> page. Refer use cases 02 and 06.
<b>Step by Step description</b>	<ol style="list-style-type: none"> <li>1. User types in the search text in the search field.</li> <li>2. The result is shown below the search field with the result dynamically as soon as the user starts typing in the search field.</li> </ol>
<b>Scenarios:</b> <ol style="list-style-type: none"> <li>1. <b>A regular scenario</b></li> <li>2. <b>Alternative scenarios</b></li> <li>3. <b>Exception Scenario</b></li> </ol>	<ol style="list-style-type: none"> <li>1. <ol style="list-style-type: none"> <li>a. User wants to look for all contents in the FAQ page related to the specific term.</li> <li>b. User opens the FAQ page (refer use case 02).</li> <li>c. User types in the search term in the search field.</li> <li>d. The result is displayed below the search field.</li> </ol> </li> <li>2. -</li> <li>3. The search term returns no result because the exact term is not available on the page.</li> </ol>
<b>Post condition</b>	User is able to search and list all the content on the page related to the given search term.



## **Chapter 5: SYSTEM IMPLEMENTATION**

### **5.1 Chapter Overview**

This chapter summarizes the overall implementation of the system. It includes the code implementation following the major implementation work flow process used in the system. The development environment used in making the system is to be discussed as well.

### **5.2 Design Decisions**

1. The system to be developed is a standalone web application and will be implemented using HTML, CSS, JavaScript and PHP.
2. All the interfaces are designed using Photoshop and implemented using HTML, CSS, and Bootstrap.
3. The system launches with the home screen interface which will be similar to every kind of user roles.
4. Each functionality and features are represented by a button in the interfaces, which is selectable by the mouse click.
5. The details in the fields of the forms and any text boxes can be input directly by the keyboard.
6. The data and information are saved in either text files or the database tables.
7. Each of the core functionalities of the project is divided into sub-functionalities for providing complete enhanced features.
8. Each of the major functions has their dedicated view, controller and service scripts.

### **5.3 Development Environment**

#### **5.3.1 Installation and Configuration of Tools**

Various kinds of tools were installed and configured depending upon requirements for the project as listed in section 3.5.

A LAMP web development platform was used for the overall development of the project and was set up in DigitalOcean which does the webhosting for the servers. LAMP uses the Linux

Operating System, the Apache HTTP Server, the MYSQL relational database management system and the PHP programming language.

Below is the list of tools that were installed and configured on the local computer to prepare for the development of the system.

1. Sublime Text

This tool was installed and used for writing all sets of codes required for full development of the system.

2. Google Chrome – Inspector

This tool is very easy to use and to inspect various UI property of the tool. Google Chrome browser was installed to make use of the inspector.

3. WinSCP

This was a really important tool to transfer the important script files between local and remote server.

There were also tools like Photoshop and Paint that were used extensively for making the UI design of the project.

### **5.3.2 Language Environment**

The language environment consisted of *scripting* and *markup* languages. JavaScript and PHP were the scripting languages used for overall development of the project. Similarly, HTML and CSS were the markup languages used for building the majority of the UI. Bootstrap and Angular JS web application frameworks were also used extensively for the project. Both the frameworks are JavaScript-based which are designed to help developers in making web application.

## 5.4 Code Implementations and Major Source Codes

This section includes the code implementation and major source codes used in making the system. The codes are divided into three categories: Views, Controllers, and Services. The views are all the UIs, controllers process the data in the UIs and Services does all the backend services and communications with the servers and databases. Figure 61 shows the overall high level visualization of the communication between scripts.

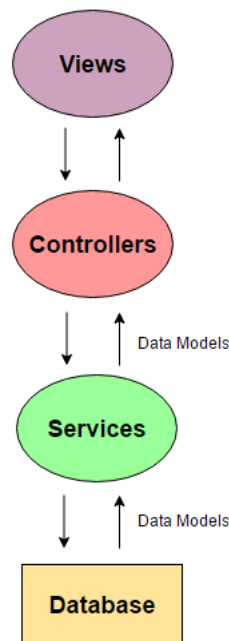


Figure 61: Communication between scripts

Views are mainly written using HTML, CSS and using the Bootstrap framework. Bootstrap is mainly used to create forms and pop-up dialog boxes. In this section, only some of the major controllers and services scripts are discussed in details. All controllers are written in JavaScript and services are written in PHP.

### 5.4.1 Login Feature Implementation

Login system is one of the major components of the project. To completely build this feature, a controller script to process the login action is written. The action is processed using a service script which is written to take the request to the server and validate the login action.

Controller and service communicate with each other at every login request. Figure 62 below shows the full steps of processing the login request.

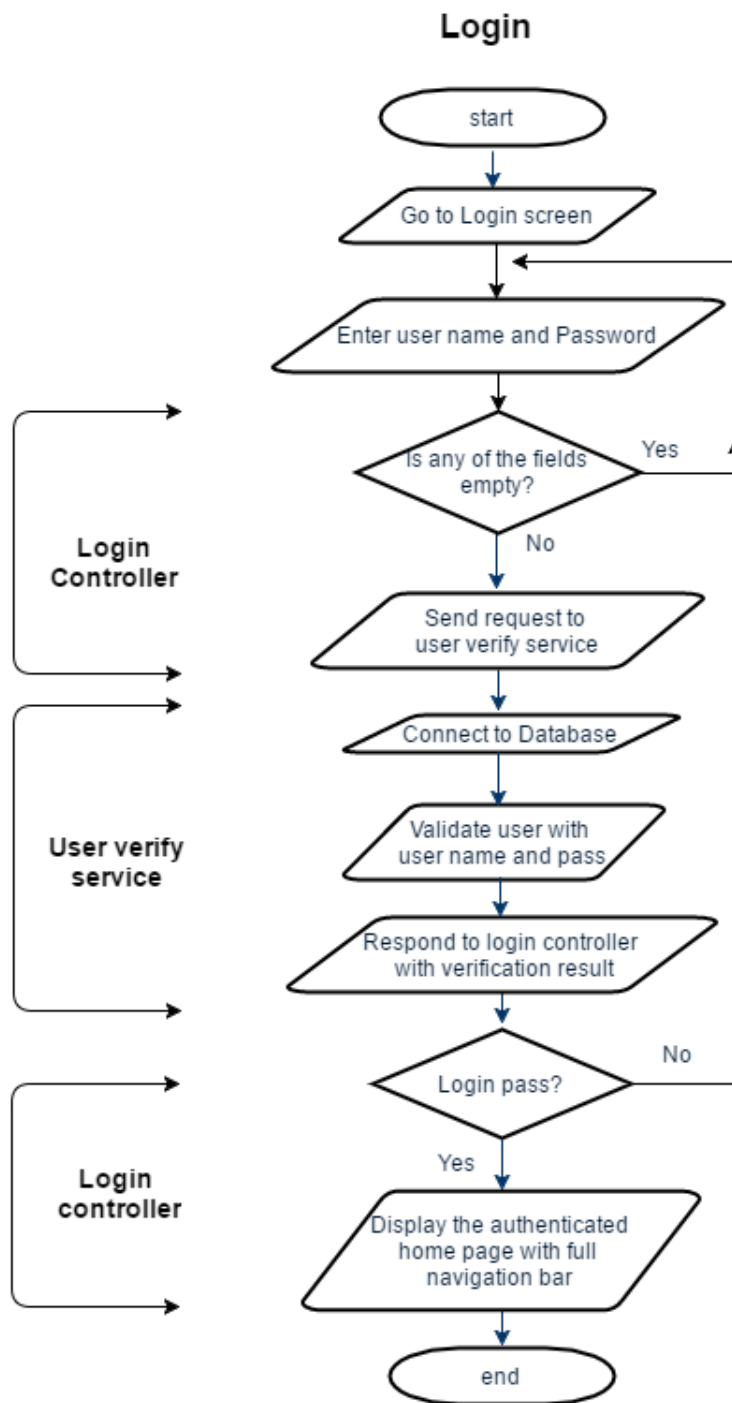


Figure 62: Login mechanism full action flow

### 5.4.1.1 Login Controller

Login Controller handles the login functionality of the system. This is written using Angular JS. Its main job is to get data from the user in the form of user name and password and pass it to *service* script to perform the authentication. Below is the snippet code of the controller.

```
//This is the angular JS app in the login page
var app = angular.module('mainApp',[]);

// Following app-controller grabs the login controller from the app above and performs the username and password null checking
// Prepares and sends data to service script for final verification
// Process the response to show the authentication result
app.controller('login_controller',function($scope, $http, $location)
{ ... });
```

See Appendix A.1.1 for full source code.

### 5.4.1.2 User Verify Service

Once the login controller collects the user's info as user name and password, they are passed to user verify service script for final authentication which is written in PHP. A connection to the database is made and user's credential is matched in the database [22].

```
// This script is responsible for verifying the user credentials
// 1. Prepare the database connection
// 2. Verify the user credentials and send response back
// 3. Close the database connection
function verifyUser {...}
```

See Appendix A.1.2 for full source code.

## 5.4.2 Ask a Tutor Page Implementation Using Speech Recognition

This is another important component of the tool. In this part of the project, the user can use the speech recognition mechanism to ask a question to the tool. In return, the tool gives the answer with important links and a video tutorial on the topic. The questions should be strictly related to C++. Figure 63 shows the complete flow of the ask a tutor page implementation.

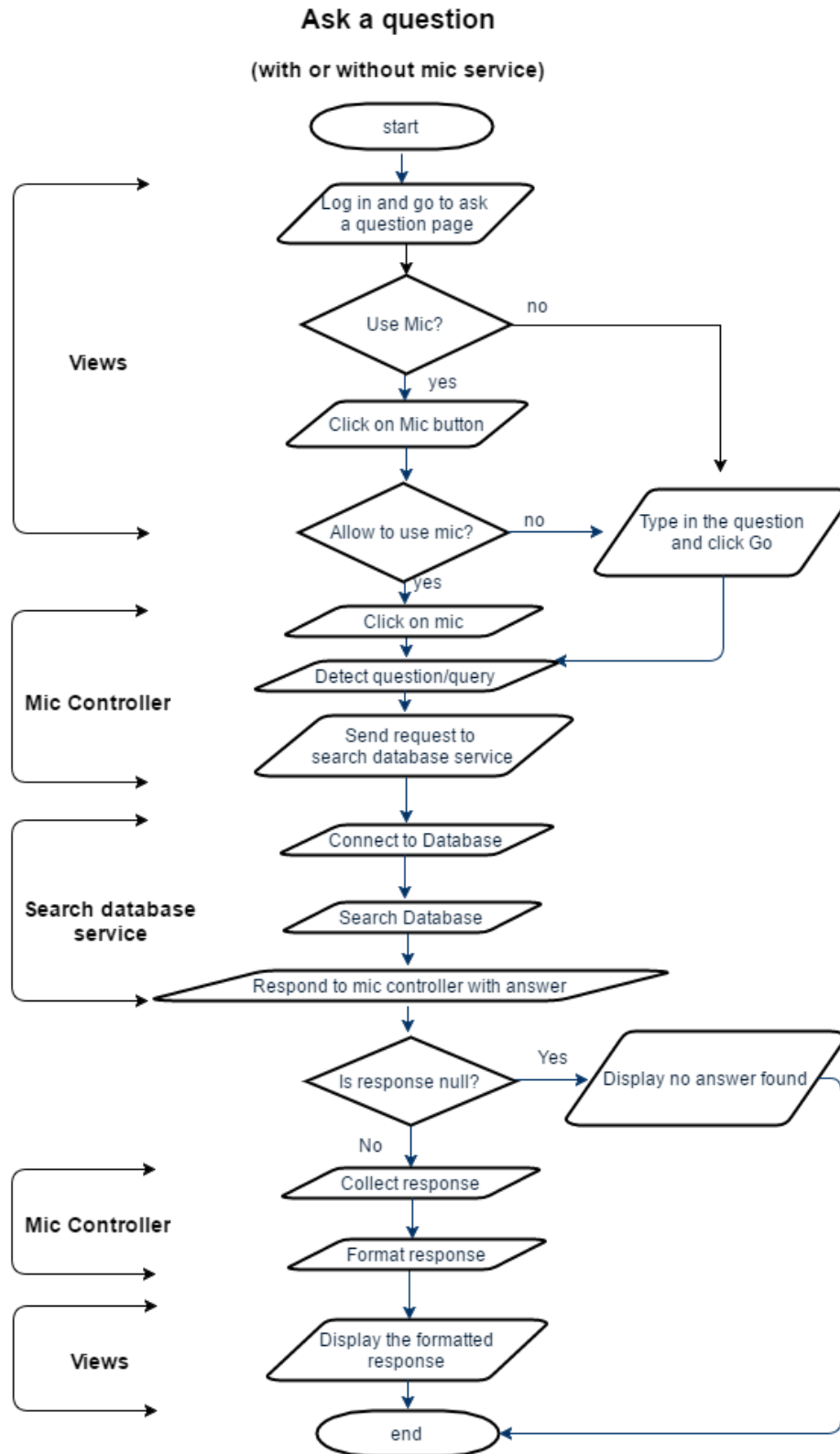


Figure 63: Ask a Tutor full action flow

### 5.4.2.1 Mic Controller

This controller script uses SpeechRecognition API which listens to the web browser. The script is supported to listen to chrome and firefox browsers. The script is enhanced by mics animation. Overall mic controller script is written in JavaScript. Once the message is recorded and collected, it is then sent to service script to make a database search and get the result. Below is the snippet of different functions in mic controller script which loads every time the page loads so that speech recognizer is ready to listen.

```
// This function does the following:
// 1. Tests browser support
// 2. Create a new recognizer
// 3. Collect the data once recognizer starts listening
function performSpeechRecognition() {...}
```

Below is the function that initiates the recording and animation of the mic.

```
function startMic() {
    recognizer.start();
    turnMicOn();
}

function turnMicOn (){
    // Mic animation is done here and is done for 5 secs
    ...
    // after 5 secs mic is turned off
    stopMic();
}

function stopMic(){
    recognizer.stop();
}
```

Once mic is turned off, database searching is done. The data collected from the user is sent to service script which does the database searching. Note that, if the user is not using the mic, the typed message is collected and sent to the service script. The function to send the final data to the service script is shown below.

```

// The function collects the final data either using mic or manual
typing by user
// Make AJAX call to search database script
// Collect the response and display the formatted result
function askQuestion() {...}

```

See Appendix A.2.1 for full source code.

#### 5.4.2.2 Search Database

Database searching script is written in PHP. It collects the data from mic controller and makes the close matching to the questions available in the database and responds with the details answers with a link to videos and useful links.

```

// This script is responsible for searching database for answer
asked by the user performing followings steps:
// 1. Prepare the database connection
// 2. Search for the answer in the database
// 3. Calculate and sort the matching percentage
// 4. Prepare and send back the list of final answers with good
matching percentage
// 3. Close the database connection
{...}

```

See Appendix A.2.2 for full source code.

#### 5.4.3 Online Chat Implementation

Online Chat offers the user to have the ability to communicate with other online users of the tool in real time. There are three parts of the online chat:

- Displaying the latest 20 messages in the chat box along with a list of all online users in the view. This is handled by Message Display Controller and Get File Data Service.



- Sending a message and saving it on the server with sent time stamp. This is handled by Chat Message Controller and Sending Message Service.
- Handling the status of the user. There are three kinds of statuses available for user: Available, Away and Busy. This is handled by Status Controller, Current Status service, and Change Status service scripts.

#### 5.4.3.1 Message Display Controller

This controller is responsible for collecting all the recent messages sent in the online communication page along with a list of online users at the moment. It is written in Angular JS. The script updates the chat box and online users list content every 1 sec. The script is assisted by a service which collects the necessary information from the server as listed in 5.4.3.2. Below is the sample code that does the necessary job.

```
//This is the angular JS app in the online chat page
var app = angular.module('mainApp', []);

// Following app-controller updates the message in the chat box every 1 sec
// Make http call to service script to get the recent online messages
// Collect, format and display the response from service script
app.controller('onlineUserController', function($scope, $http) {});
```

See Appendix A.3.1 for full source code.

#### 5.4.3.2 Get File Data Service

This service script detects the caller (controller) and sends response accordingly. The main job of this service is to open the necessary file on the server and collect the contents and send it back to the caller. The script is written in PHP. Below is the code description of the script.

```
// Based on request from the controller, this script opens file and collects data
```

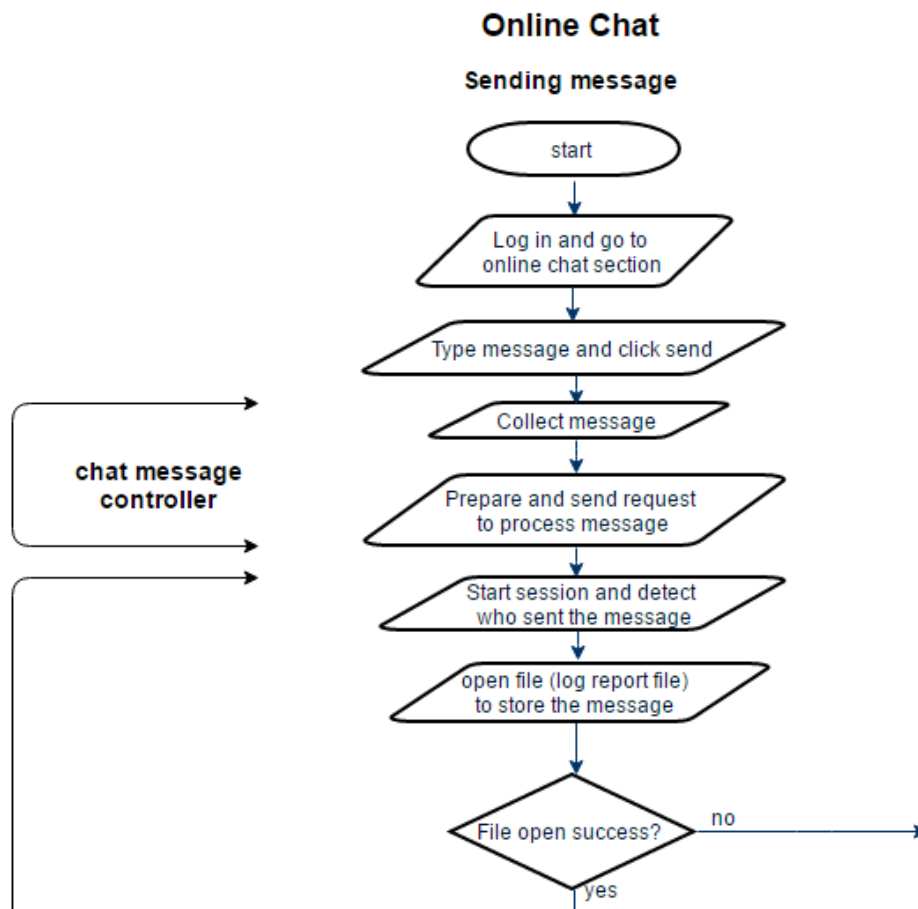
```
// Example: request may be to collect list of online users or to
collect recent online messages sent
{...}
```

```
// This function formats the file content and returns response
function array2string($data) {...}
```

See Appendix A.3.2 for full source code.

### 5.4.3.3 Chat Message Controller

This script written in JavaScript using jQuery collects the message from the chat text box responding to send action performed in the interface. Once the message is collected, a necessary ajax call is made to run the service script (section 5.4.3.4) which saves the necessary message in the dedicated file saved on the server. Figure 64 below shows the detail steps to successfully send a message using the tool.



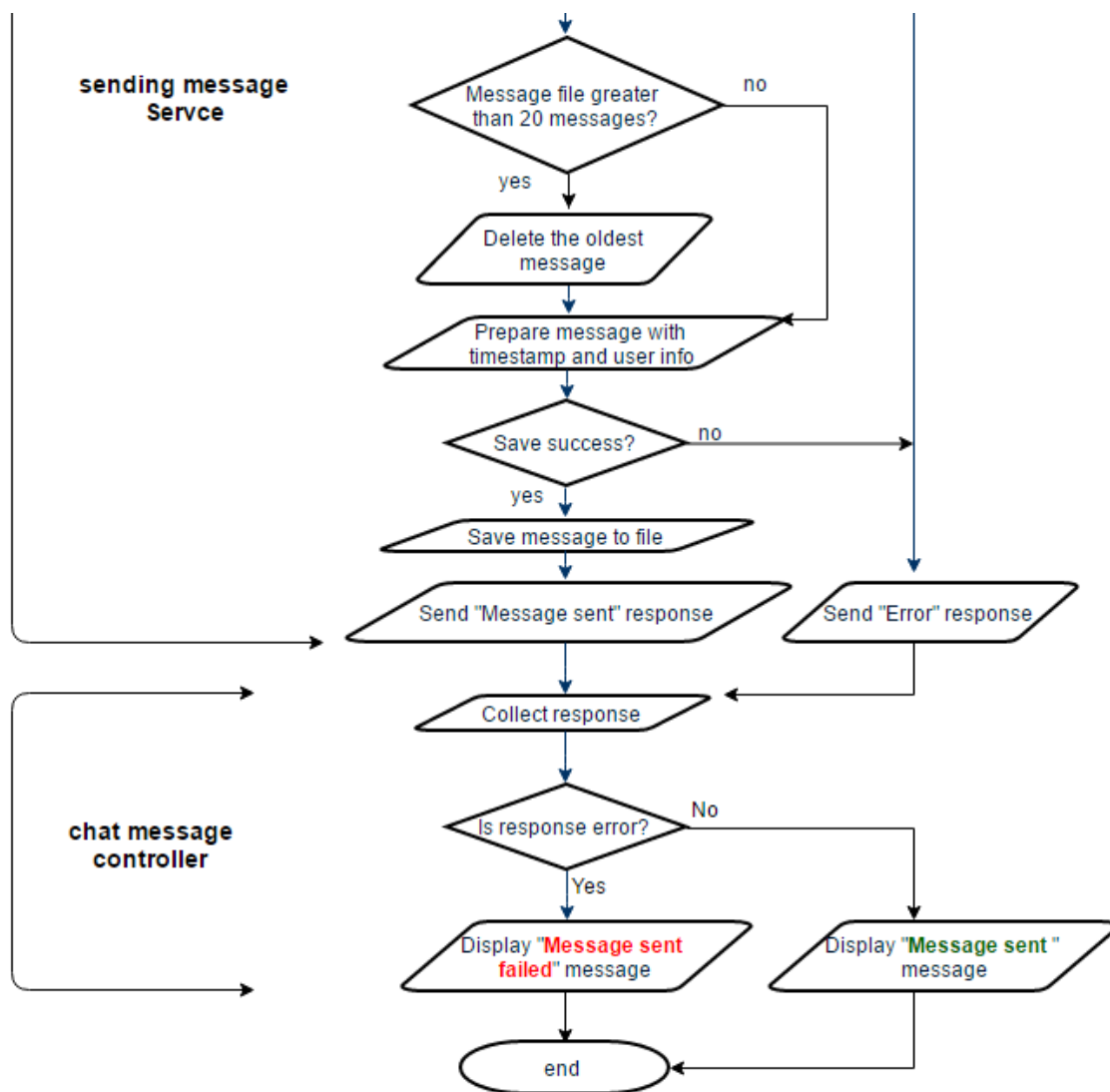


Figure 64: Sending a Message full action flow

```

// This jQuery function:
// 1. Collects the new message & sends to service script to save
// 2. Collect the response from service script to notify user
$(document).ready(function() {
    $('#messageForm').on('submit', function(e) {
        ...
    });
});

```

See Appendix A.3.3 for full source code.

### 5.4.3.4 Sending Message Service

This is helper service script in PHP to save the message sent by chat message controller above to the necessary file in the server. The saving of the message is done as shown below.

```
// This script performs following:
// 1. Collects the message sent by the user from controller
// 2. Add new message to the file with new time stamp
{...}
```

See Appendix A.3.4 for full source code.

### 5.4.4 Status Controller

This controller written in JavaScript controls the status of the user in the tool. It is responsible for reflecting the current status of the user in real time. To accomplish this task, this controller is assisted by services to reflect current status and change of status. The tool offers three kinds of statuses: Available, Away and Busy. Below is the code snippet for the status controller.

```
// This function updates the status as saved in the server
function updateCurrentStatus() {...}

// This function is responsible for changing the status:
// 1. Collect and display the new status
// 2. Make AJAX call to change the status details in the server
function changeStatus(status) {...}
```

See Appendix A.4 for full source code.

#### 5.4.4.1 Current Status and Change Status Services

Both the service scripts are written in PHP. Current Status Service script returns the current online status of the user that is saved in the session. See Appendix A.4.1 for full source code of Current Status Service script. Change status service script sets the new status of the user in the current session. Hence, the status of the user can be changed by the user as long as session exists.

### **Change Status Service**

```
// The script performs the following:  
// 1. Collect the new status to be set sent by controller  
// 2. Collect the list of online users  
// 3. Find the user in the online user list and change status  
// 4. Save the list of online users with updated status of the user  
{...}
```

See Appendix A.4.1 for full source code.

### **5.4.5 Screen Sharing Implementation**

This is the most important part of the project. This allows the user to share their screen with other users who may or may not be online in the tool. However, to be the organizer of the screen sharing, the user needs to log in to the system. There are multiple controllers and services that together offers the full functionality of the screen sharing service.

Some major things to note about this feature:

- a. A separate browser extension is needed to use the feature. Currently, google chrome and Firefox extension is available and therefore screen sharing is supported only on these browsers. Note that, the extension is not required for audiences of the screen sharing room.
- b. A unique id is created for every room session which is used to create a unique room URL. The URL can be used to join the opened room using any browser.
- c. The feature also supports inviting other online users using the tool directly to the created screen sharing room.
- d. A free signaling server is used to organize the screen sharing. Nonetheless, the session is fully secured.

The controllers and services that are needed for the screen sharing are listed and discussed in details in this section. Below is the complete high level flow diagram of the screen sharing, step by step.

## Share Screen

Process is long, following flow-chart is to show simplest work flow to start screen sharing

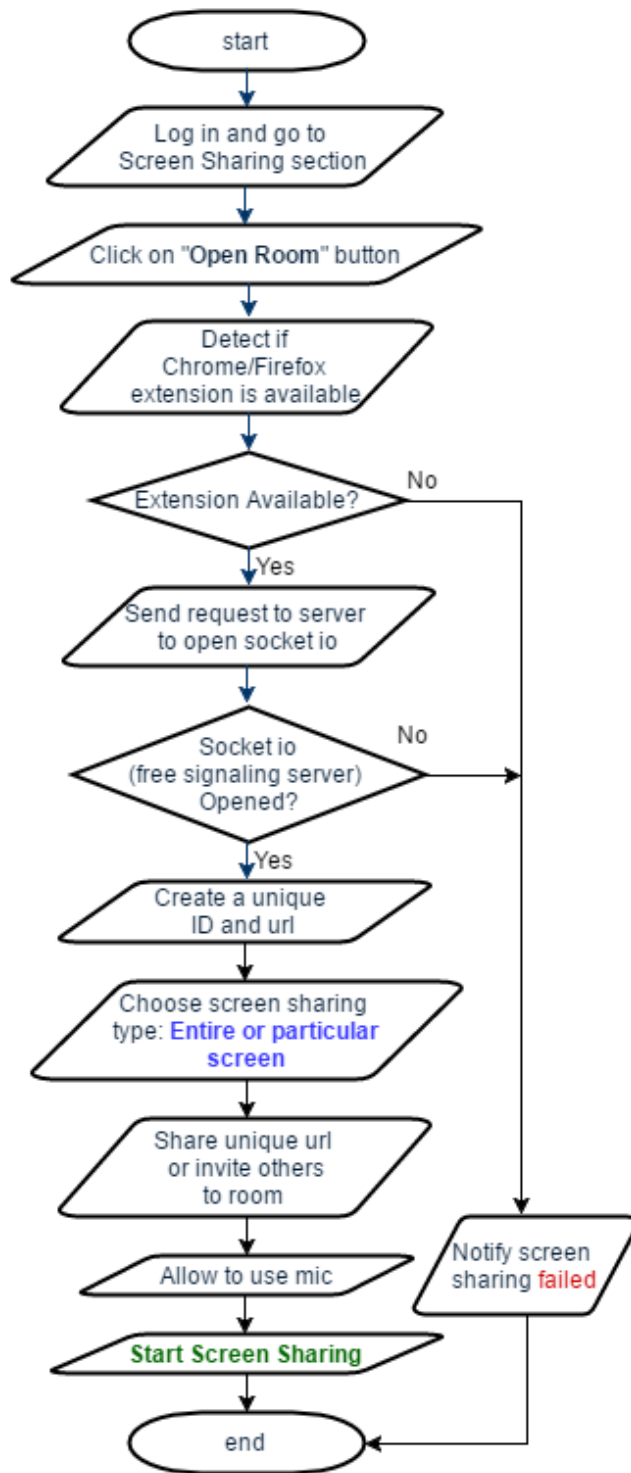


Figure 65: Screen Sharing full action flow

### 5.4.5.1 Screen Sharing Controller

Screen sharing controller handles the major part of screen sharing feature. This controller is responsible for creating an actual screen sharing room using free signaling server, creating a unique id for the room and hence creating a unique URL for each opened room. Currently, the controller is written to support audio and screen sharing. User attending the screen sharing room can see the organizer's screen as well as talk to the organizer.

The script is written using JavaScript. An open sourced RTCMulticonnection which is a WebRTC-library is used write the script to be able to start screen sharing [21]. This script is called when a button to open a screen sharing is clicked in the view. Below is the detail description of the screen sharing controller.

First of all, following three libraries are included in the project.

```
<script src="https://cdn.webrtc-
experiment.com:443/rmc3.min.js"></script>
  <script src="//cdn.webrtc-
experiment.com/getScreenId.js"></script>
  <script src="https://cdn.webrtc-
experiment.com/getMediaElement.js"></script>
```

Then, an object of RTCMulticonnection is created and properties are set.

```
var connection = new RTCMultiConnection();
// Set connection properties
// set the video and audio container in the view page
{...}
```

Now connection can be started using on-stream event and the media element can be displayed in the container.

```
// On receiving the onstream event, this function prepares the video
containers and plays the video
```

```
connection.onstream = function(event)
{...};
```

The user-id is returned from the `getScreenId.js` library which is included in the project. The function below is responsible for detecting the extension available in the browser. If the extension is not detected, an error is thrown. As a user, the screen sharing won't start on the screen.

```
// The function determines the screen constraints using the
installed extension in the browser
connection.getScreenConstraints = function(callback)
{...};
```

Again, `getScreenConstraints` is included in the `getScreenId.js` library.

Now when the streaming ends, the `onstreamended` event is called and the screen sharing view is removed.

```
// The function triggers when the stream ends and stops the
streaming
connection.onstreamended = function(event)
{...};
```

The function below is responsible for handling the room id.

```
function handleRoomID() {
    // Get the id if available in local storage else create one
    ...
}
```

Once the room id is created following function is responsible for generating unique URL to the room.

```
function showRoomURL(roomid)
{...}
```



See Appendix A.5.1 for full source code.

Once the room is created, following controllers and services enhances the feature abilities.

#### 5.4.5.2 Screen Sharing Online User List Controller and Get File Data Service

Screen sharing online user list controller and get file data service helps to list all the online users along with their statuses. Both the controller and service script are similar as described in 5.4.3.1 and 5.4.3.2.

However, screen sharing online user list controller also helps choose users from the online user list to send an invitation to the opened screen sharing room. Below is the series of outlines of the codes in JavaScript that performs the tasks of invitation which is eventually assisted by sending invitation service on the backend side.

```
// This function helps user to select users from the online list and
send invites
function sendInvites() {

    // Detect if the room has been created yet
    ...

    // If not ask user to open room first to send invitation
    // Detect all the selected users from online user list
    // create an array of invited user list with room id
    // if the checkbox is checked, user is selected for
    invitation
    // If no user is selected, report user to select at least
    one user to send the invitation
    ...

    // Send the invitation and notify
    ...
}
```

```

// This function helps select all the users at once
function selectAll() {...}

// This function helps user to clear the selection
function clearAll()
{...}

// This function finally sends invitation to other users making AJAX
calls
function sendInvitation()
{...}

```

See Appendix A.5.2 for full source code.

#### 5.4.5.3 Send Invitation Service

Once the list of selected users to send the invitation is confirmed, this service written in PHP receives the invited user list. The received list of invited users is saved to the file on the server so that *invitation detection service* (discussed in 5.4.5.5) can pick up the data about invited users along with the room id. Below are the outlines of the code to do this job.

```

// The script performs the following:
// 1. Collect the invitation sender info
// 2. Gather list of invited users from the controller
// 3. Create a pattern with sender id and name to match to the
//    gathered list of invited users
// 4. If the invitation is for self, skip otherwise save the info
// 5. Save all the invited user to the file
{...}

```

See Appendix A.5.3 for full source code.

#### 5.4.5.4 Detect Invitation to Screen Controller

This controller is responsible for detecting if there is any invitation sent. If the invitation is detected, this pops up a dialog box with the URL to the opened room with the invitation sender information. This script runs in every user's session to detect invitation. The script

makes use of the Bootstrap framework and is written in JavaScript. This can pop up in any of the pages in the tool. This controller is assisted by *detect invitation to screen service* (discussed in 5.4.5.5) which is responsible for collecting the invitation information from the server.

```
// This function checks
// 1. if there is any invitation for current user
// 2. create a bootstrap pop-up dialog box with
//     a. unique URL and
//     b. inviter details
function checkInvitation()
{...}
```

See Appendix A.5.4 for full source code.

#### 5.4.5.5 Detect Invitation to Screen Service

This service written in PHP assists in detecting the invitation communicating with the server. If invitation is detected, it collects the room id and inviter information and sends back to the controller making the request.

```
// The script performs the following:
// 1. Collect the current user id
// 2. Collect the current user name
// 3. Collect the invited users list
// 4. Detect if any invitation is made for current user
// 5. If any invitation is detected, respond back to controller with
//     a. room key and
//     b. inviter name
{...}
```

In addition, following function is included in the script.

```
// This function performs the following:
// 1. Assists in deleting the detected invitation in the
// server so that invitation is not made again and again
//
// Input: user name
```

*// Output: Deletes the detected invitation from the server*

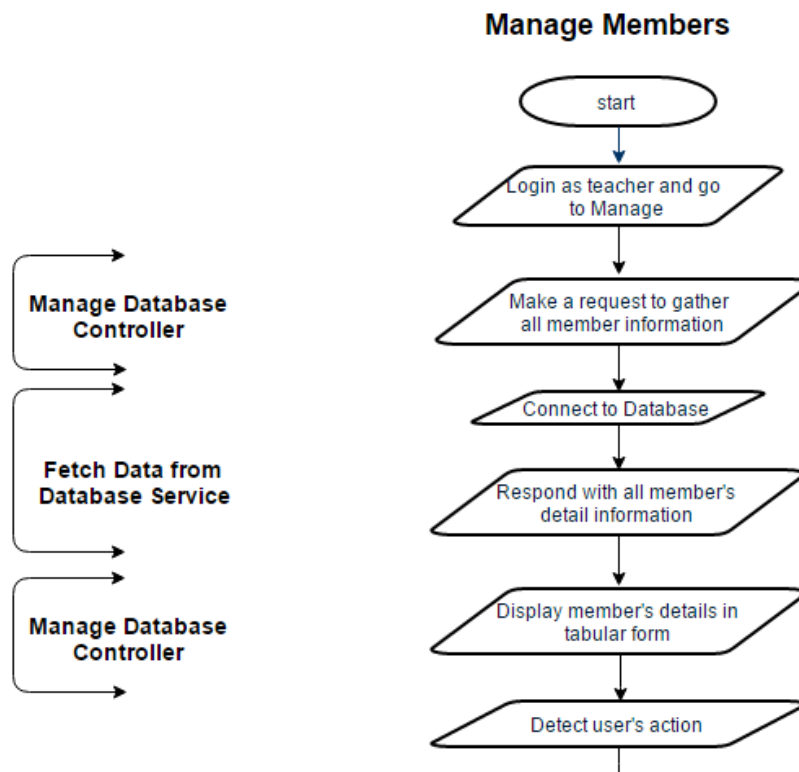
```
function deleteDetectedinInvitationList($a_user)
{...}
```

See Appendix A.5.5 for full source code.

#### 5.4.6 Manage Members Implementation

Manage Member is another important feature of this project. With this feature, the user is able to add a new member to the system. This feature also lets the user modify the existing member and even delete the user. There are two kinds of members using the tools, Teachers, and Student. This feature is only available for members who are Teachers.

Below shown is the complete flow-chart describing the steps to complete different transaction offered by this feature.



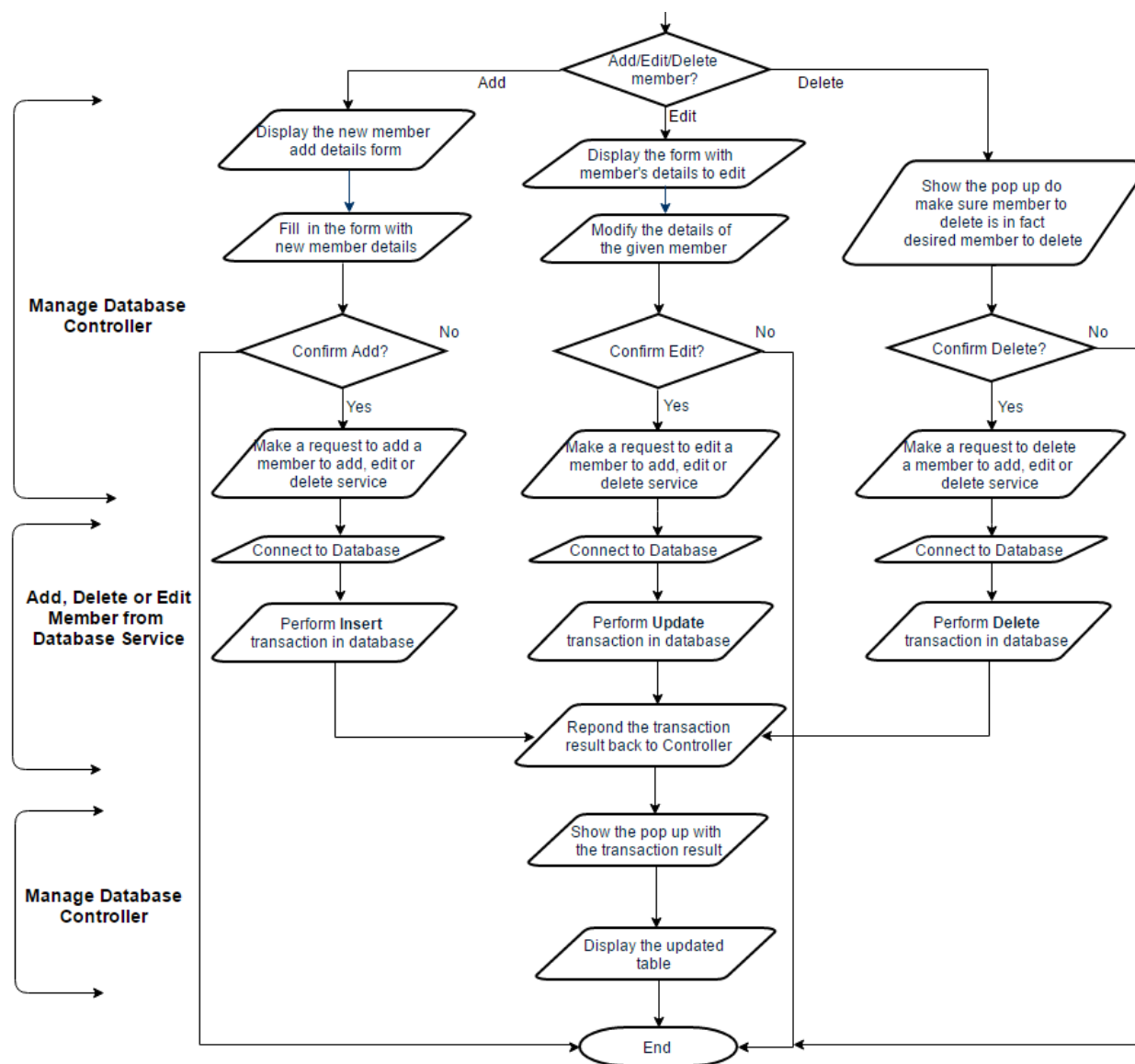


Figure 66: Members Management full action flow

The controller and service scripts that are involved in giving full functionality of this feature are discussed below.

#### 5.4.6.1 Manage Database Controller

This controller is responsible for displaying all the member's information in a tabular form. This also controls the actions of adding a new member, deleting or editing the existing user.

The controller is written using JavaScript and uses the Bootstrap framework. First, this controller makes a call to *fetch data database service* (discusses in 5.4.6.2) to gather all the member's information and displays them in a tabular format. The functions involved in this controller are mentioned below.

```
// This function makes a call to service to gather all the member information  
function fetchData()  
{...}  
  
// This function parses the response from service and displays in the table in readable format  
function updateTable(response)  
{...}
```

See Appendix A.6.1 for full source code.

This controller also controls the major actions of adding a new member to the system and deleting/editing the existing member. Below discussed are the code implementation for these three major actions.

### **Add a Member**

For adding a new member to the system, first of all, details of the new member are collected from the user and once confirmed, a call to service is made to finally save the new member in the database. In course of action, a unique password is generated automatically to the new user.

```
// This function on call displays the form to fill in new member's details  
function addThisItem()  
{...}
```

Once the form is displayed and filled in and is confirmed by the user, new member details are collected and send to the add/edit/delete service to perform INSERT transaction to the database.

```
// On add confirmation, this function sends the add request to
// service with member details to finally add to the database
// The series of steps performed in this functions are:
// 1. Collect First Name, Last Name and user role (Teacher/Student)
// 2. Generate unique pass for new user
// 3. Prepare data to send to service to add to the database
// 4. Make AJAX call to service and send the prepared data
function confirmAddThisItem()
{...}

// This function is used to create a unique password
function createID()
{...}
```

See Appendix A.6.1.1 for full source code.

### **Edit a Member**

For editing a member, details of the member are gathered and shown in the editing form. The member can edit the form with the member details as needed.

```
// Given the id of the user to be modified, this function collects
// member details and show them in the editing form
function editThisItem(id)
{...}
```

Once the member details are modified and confirmed, the modified details are collected. Then a call to add/edit/delete service is made to finally UPDATE the member's details in the database.

```
// On edit confirmation, this function sends the UPDATE request to
// service with modified member details to finally reflect the
// modification in the database
// The series of steps performed in this functions are:
```

```
// 1. Collect the new details from the editing form
// 2. Prepare data to send to service to make the Update transaction
// 3. Make AJAX call to service and send the prepared data
```

```
function confirmEditThisItem(id)
{...}
```

See Appendix A.6.1.2 for full source code.

### Delete a Member

If the user wishes to delete a member, a form is shown with the member's detail to confirm the deletion.

```
// This function displays a form with the member detail to delete
function deleteThisItem(id)
{...}
```

On confirmation, a request to add/edit/delete service is made to perform the DELETE transaction on the database.

```
// On delete confirmation, this function sends the DELETE request to
// service with member's ID
// The series of steps performed in this functions are:
// 1. Prepare data to send to service script
// 2. Perform DELETE transaction
// 3. Make AJAX call to service and send the prepared data
function confirmDeleteThisItem(id)
{...}
```

See Appendix A.6.1.3 for full source code.

### 5.4.6.2 Fetch Data Database Service

This service is dedicated for collecting all the member's details and return to the controller. The member's information is collected and returned in json format for the easy processing.



```

// The series of steps performed in this scripts are:
// 1. Prepare the database connection
// 2. Collect all members
// 3. Return the final record and Close the connection to database
{...}

```

See Appendix A.6.2 for full source code.

### 5.4.6.3 Add, Edit, or Delete Database Service

This service gets the transaction request from the controller with the member's details and based on the transaction type, it performs either INSERT, UPDATE or DELETE transaction on the database in the Members table. Below are the complete service script steps and functions written in PHP to perform the different transaction.

```

// The script performs the following steps to gather necessary
// details before starting the final transaction of add/edit/delete
// 1. Collect the sent data from the controller
// 2. Determine the transaction type and call the necessary function
//    with the collected data
{...}

```

```

// This function with the given data performs the UPDATE(edit)
// transaction performing following steps
// 1. Extract the data from the sent data package
// 2. Prepare the database connection
// 3. Perform the UPDATE transaction
// 4. Respond with the success result back to the controller
function editMember($sentData)
{...}

```

```

// This function with the given data performs the INSERT(add)
// transaction performing following steps
// 1. Extract the data from the sent data package
// 2. Prepare the database connection
// 3. Perform the INSERT transaction
// 4. Respond with the success result back to the controller
function addMember($sentData)
{...}

```

```
// This function with the given data performs the DELETE transaction  
// performing following steps  
// 1. Extract the data from the sent data package  
// 2. Prepare the database connection  
// 3. Perform the DELETE transaction  
// 4. Respond with the success result back to the controller  
function deleteMember($sentData)  
{...}
```

See Appendix A.6.3 for full source code.

#### **5.4.7 Displaying and Modifying Lab Assistant Hours' Implementation**

Displaying and managing lab assistant hours' is one of the major features of the tool. This feature enables users to look at the current lab assistant working hours. If logged in as a teacher, this lets the user modify the lab assistant's schedule upon requirement. The user is able to add new hours and modify/delete existing hours of operations. To meet all these requirements, a controller and a service script are written which are discussed in sections 5.4.7.1 and 5.4.7.2. Below is the complete flow of actions in displaying and managing the lab assistant hours' table.

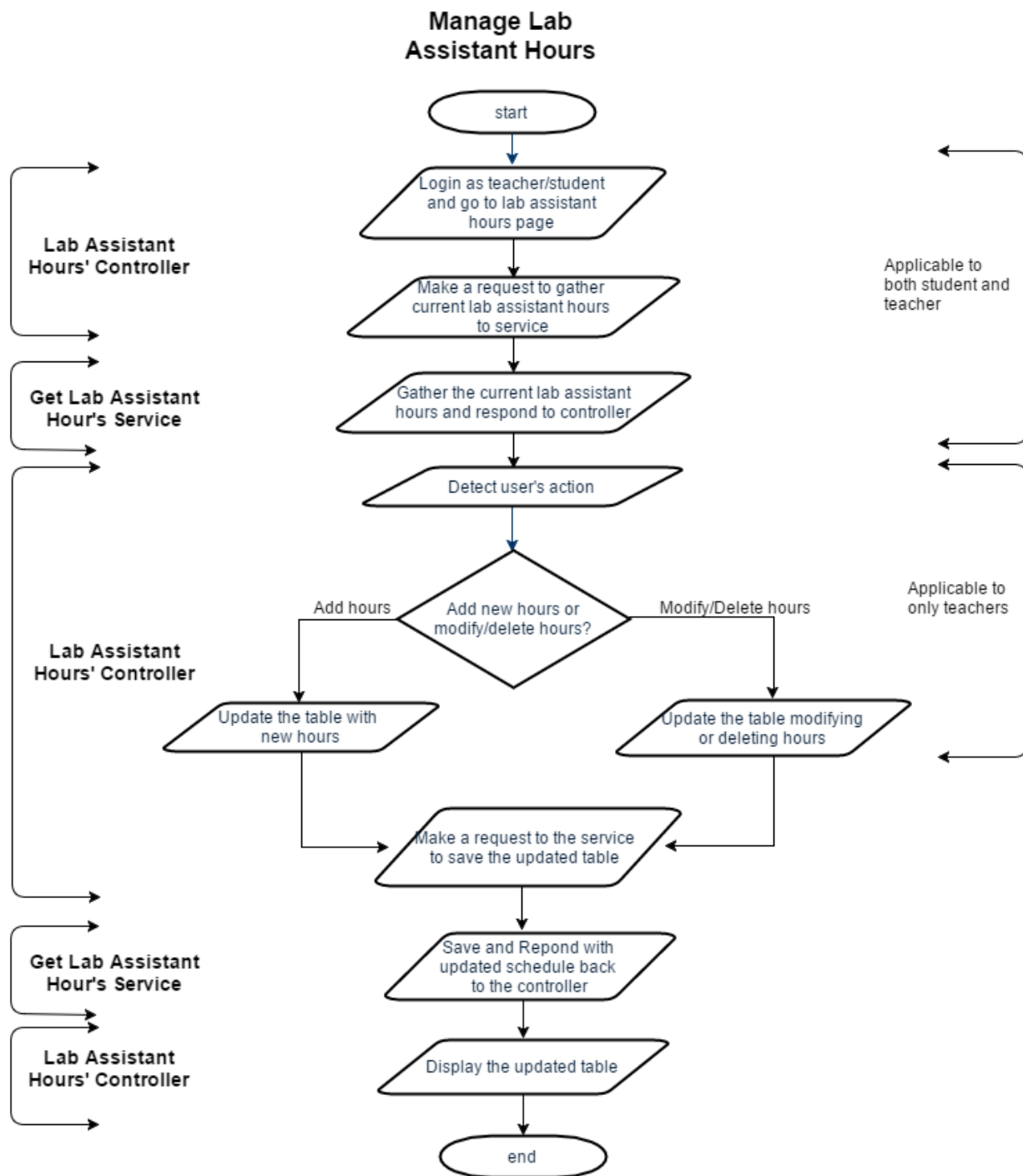


Figure 67: Lab Assistant hours' management full action flow

### 5.4.7.1 Lab Assistant Hours' Controller

This controller is responsible for displaying and managing the lab assistant hour's table. Below are the functions written in JavaScript using jQuery to perform the actions of displaying and managing the table with the help of service discussed in section 5.4.7.2.

```
// This jQuery on Load gets the current schedule, making a request
to service and display in the table
$.get("url to get lab assistant hours service", function(data){
    ...
});

// Below functions are ready to be triggered on Load
$(document).ready(function() {
    // this function on click makes the table editable
    $("#edit_update" ).click(function() {...});

    // this function on click saves the modified table making a
request to service
    $("#save" ).click(function() {...});
});
```

See Appendix A.7.1 for full source code.

### 5.4.7.2 Get Lab Assistant Hours' Service

This service assists lab assistant hours' controller (discussed in 5.4.7.1) to collect the latest lab assistant hours' schedule.

```
// Detect the type of request
// If request is POST, save the modified data sent as message
// else respond with current schedule to the controller
{...}

// this helper function converts array to string
function array2string($data) {...}
```

See Appendix A.7.2 for full source code.

#### 5.4.8 Search Controller (for FAQ and Chapters Page)

This controller is used for searching the content within the page. This controller is dedicated and used only in FAQ and Chapters page. To implement this search mechanism, a bootstrap twitter typehead with json api is used. In order to be able to use this feature, following library should be linked.

```
<link rel="stylesheet"
href="//maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.cs
s">
```

See Appendix A.8 for full source code.

#### 5.4.9 Contact Message Sending Implementation

This feature lets users send the message to the system admins (Teachers/Tutors). It is for the students or even the guests to send the concerns and queries, which will be visible to all the admins. This feature is implemented by using a controller and a service, which are discussed in details below.

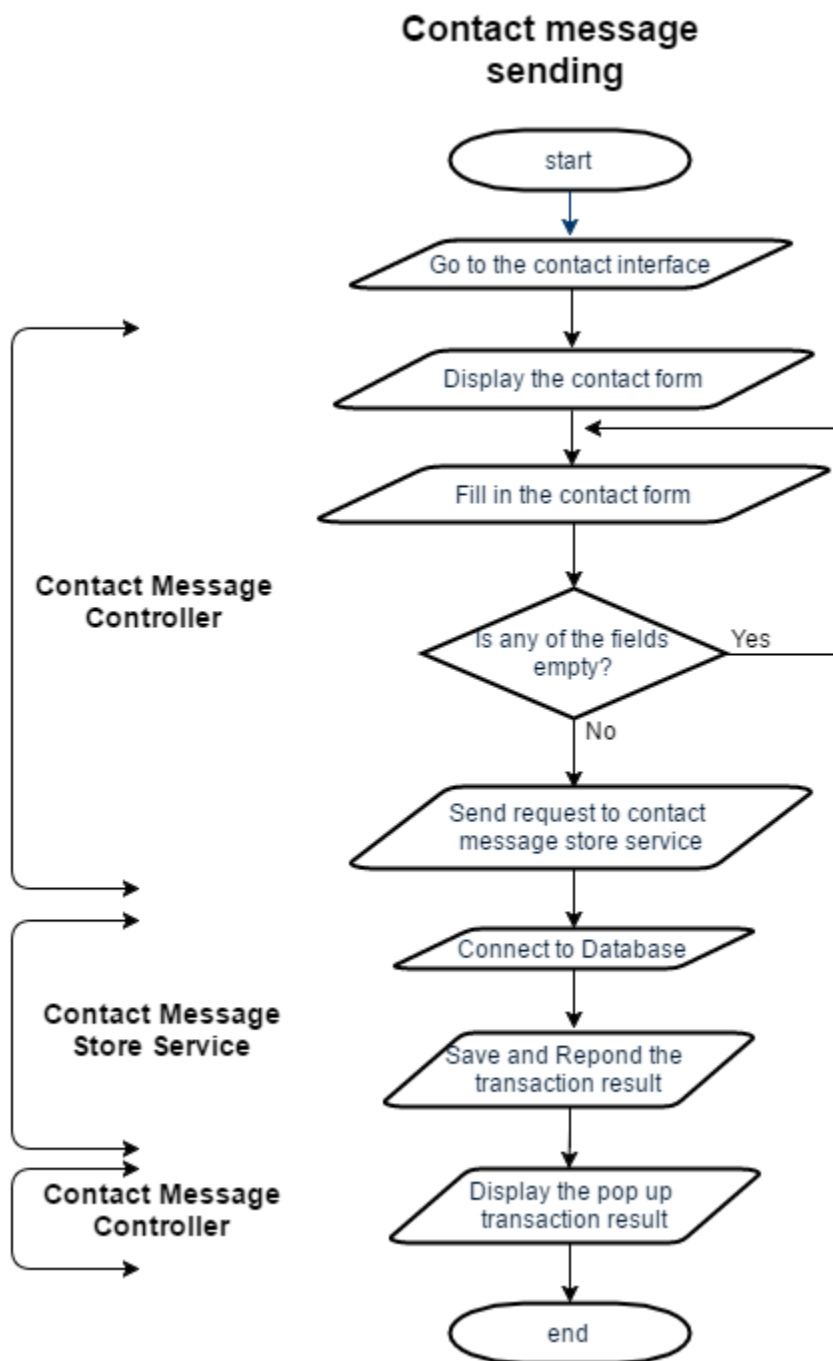


Figure 68: Contact Message sending full action flow

### 5.4.9.1 Contact Message Controller

This controller written using Angular JS is responsible for collecting the messages and sender details. Then, a request to the service is made to store the messages in the database.

```
//This is the angular JS app in the contact us page
var app = angular.module('mainApp',[]);

// Following app-controller collects the contact message along with the sender details
// Make http call to service script to save the messages in database
// Notify the success result to the user
app.controller('contact_message_controller',function($scope, $http)
{ ... });
```

See Appendix A.9.1 for full source code.

### 5.4.9.2 Contact Message Store Service

This service helps contact message controller (discussed in 5.4.9.1) in storing the sent messages to the database in the ContactMessages table. This is written in PHP.

```
// This script performs the following:
// 1. Collect the sender details and messages
// 2. Prepare the database connection
// 3. Make INSERT transaction to Messages table in the database
// 4. Respond with transaction result
{...}
```

See Appendix A.9.2 for full source code.

### 5.4.10 Contact Message Management Implementation

As a teacher, a user should be able to manage the messages send by other users. This implementation helps teachers read messages and manage them. The controller and services discussed below aids in implementing this feature.

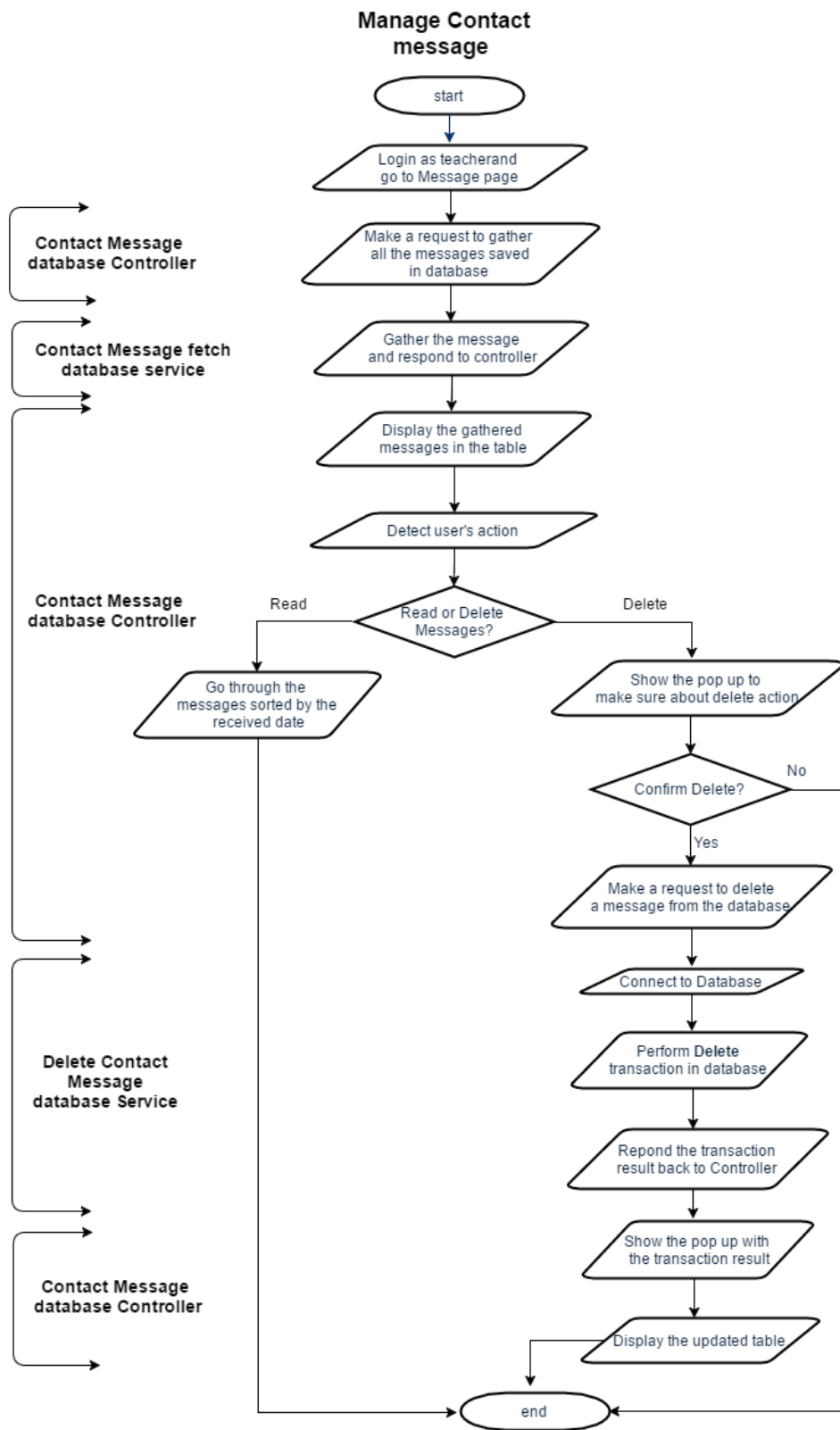


Figure 69: Manage Contact messages full action flow



### 5.4.10.1 Contact Messages Database Controller

This controller collects all the messages sent by other users (students/guests) sent as a concern and query to the teachers. It is responsible for displaying all those messages in the tabular format for easy reading and handling. This controller along with two services script which are discussed in sections 5.4.10.2 and 5.4.10.3, helps teachers read messages and manage them. The implementation of this controller in JavaScript is shown below in detail.

The fetching of data is done and shown in the table with the help of following functions in the script.

```
// This function fetches all messages from database by making a request to service
function fetchData() {...}

// This function displays each record found in the database in the table
function updateTable(response) {...}
```

When the teachers want to delete messages from the database, following functions are triggered which makes a request to services to perform the job.

```
// This function finally deletes the item with given id from the database making a request to service
function confirmDeleteThisItem(id) {...}
```

See Appendix A.10.1 for full source code.

### 5.4.10.2 Contact Messages Fetch Database Service

The main function of this script written in PHP is to help controller gather all the messages saved in the ContactMessages table from the database. Once gathered, the information is passed back to the controller to display in the table for the users. Below are the steps in implementation of this script.

```
// The series of steps performed in this scripts are:
// 1. Prepare the database connection
// 2. Collect all contact messages from the database
// 3. Return the collected records and Close the connection to
database
{...}
```

See Appendix A.10.2 for full source code.

### **5.4.10.3 Delete Contact Message Database Service**

This service is more dedicated to handling the delete transaction in the ContactMessages table in the database. This is triggered when controller sends a request for delete transaction and a transaction result is sent back to the controller. The steps in the script are shown in detail below which are written in PHP.

```
// This script with the given data performs the DELETE transaction
// performing following steps
// 1. Collect the data from the sent data package
// 2. Prepare the database connection
// 3. Perform the DELETE transaction
// 4. Respond with the success result back to the controller
{...}
```

See Appendix A.10.3 for full source code.

### **5.4.11 Resume Session Controller**

This is an important controller that helps in controlling the session time. With the help of resume session service (discussed in section 5.4.11.1), this controller helps the user to extend the session time. It shows a pop-up dialog box displaying the notification for users about the session timeout and asks them to take action. If the user chooses to extend the time, a request to service is made to extend the time. Below is the code snippet of implementation of the controller.

```
// This Function notifies user of nearing the ending session and
suggesting to add time to the session
```

```
function resumeSession() {...}
```

See Appendix A.11 for full source code.

#### **5.4.11.1 Resume Session Service**

This service is responsible to extend the time period for the session. Once the request comes from the controller, this script stamps the new timeout for the session for the user.

See Appendix A.11.1 for full source code.

#### **5.4.12 Additional Services**

There are additional services that are needed to offer additional services. These are needed to make sure about different unseen features such as time out checking of the session, logging out service and image slider service that is used on the home page of the tool.

##### **5.4.12.1 Timeout Session Check (for Student and Teacher)**

This service is more dedicated to continuously monitoring the timeout session of the user. This makes use of resume session controller discussed in section 5.4.11 to keep track of session time limit. Below discussed are time session check services applicable for student's session and teacher's session.

##### **Student Time Check:**

The session time is set as 200 minutes. After 180 minutes a pop-up dialog box is shown to the user if they want to renew the session. If the user does not renew the session, they are automatically logged out of the session and system.

See Appendix A.12.1 for full source code.

**Teacher Time Check:**

The session time check for a teacher is similar to student's time check. The only difference is when the *user type check* is made at the bottom of the script. Below shown is the script needed for teacher's time check.

```
// Same as student time check  
...
```

See Appendix A.12.1 for full source code.

**5.4.12.2 Logout Service**

This service is triggered when the user wants to log out of the system. This is responsible for killing the session and destroying all the details collected during the session. This is also activated when the time out check fails for either teacher or student. Below shown is the full description of functionality of logout service written in PHP.

```
// This script displays appropriate notification or Logs out the  
user depending on different scenarios  
// 1. Display the message if the user was logged out because of  
session timeout  
// 2. Display message if user was logged out in different window and  
Log out  
// 3. Otherwise logs out the user, delete the user's details from  
the online user list and destroys the session  
{...}
```

See Appendix A.12.2 for full source code.

**5.4.12.3 Image Slider Service**

Image slider service is used on the home page of the system. This is responsible for collecting the images and setting up a slider mechanism to display the collected slides with animation. The script is written in JavaScript using jQuery and is shown in details below.

```
// This jQuery function loads up all the images available every time  
window loads and runs the slides show  
$(function() {  
    ...  
  
    // This helper function collects the images  
    (function preloader(){  
        ...  
    })();  
  
    // This helper function starts the slideshow and continue to  
display next slides  
    function SlideShow() {  
        ...  
    }  
});
```

See Appendix A.12.3 for full source code.

## Chapter 6: TESTING

### 6.1 Chapter Overview

This chapter summarizes the overall testing performed to measure the accuracy and functionality of the system. Different kinds of testing were performed to test the system. The testing carried out were unit testing, integration testing and overall system testing.

### 6.2 Unit Testing

The major functionalities were carried out with unit testing to measure the correctness of the functions. Unit testing was done for different functions related to different individual features of the system to ensure the accuracy. Followings are the list of unit testing done for different major functionalities of the system.

#### 6.2.1 User Verification

This test case ensures the user verifications are done accurately.

Test Case Name	User Verification
Test Case Description	This test case validates the accuracy of user verification
Pre-conditions	-
Input test data	User name and password – see table below
Steps to be Executed	Enter the username and password
Expected Results	If the user's username and password are in the database, login should success otherwise it should fail with 100% success rate.
Actual Results	100% user verification success
Pass/Fail	<b>Pass</b>

User Name	Password	Expected Result	Actual Result	Test Pass/Fail
santosh	Basnet	Success	Success	<b>Pass</b>

santosh	Random	Fail	Fail	<b>Pass</b>
Ritu	Tamang	Success	Success	<b>Pass</b>
Ritu	Random	Fail	Fail	<b>Pass</b>
abc	qwerty	Success	Success	<b>Pass</b>
abc	1234	Fail	Fail	<b>Pass</b>

### 6.2.2 Speech Recognition Activation Period

This test case ensures that the user can make use of speech recognition and turn the mic on for the given period of time.

<b>Test Case Name</b>	<b>Speech Recognition Activation Period Testing</b>
Test Case Description	This test case validates the activation period of speech recognition
Pre-conditions	Microphone is enabled
Input test data	N/A
Steps to be Executed	Click on speech recognition icon
Expected Results	The mic button should turn red for 5 secs and should go back to normal
Actual Results	The mic button turned red for 5 secs
Pass/Fail	<b>Pass</b>

### 6.2.3 Speech Recognition

This test case ensures the accuracy of speech recognition.

<b>Test Case Name</b>	<b>Speech Recognition Test</b>
Test Case Description	This test case validates the accuracy of speech recognition
Pre-conditions	Microphone is enabled
Input test data	See table below

Steps to be Executed	Click the mic button and start speaking after mic activation
Expected Results	Words, phrases, sentences should be detected with 100% accuracy
Actual Results	83.33% accurate
Pass/Fail	<b>Pass</b>

#### Sample speech recognition Test result

Spoken Text	Detected Text	Result – Pass/Fail
Testing Speech Recognition	Testing Speech Recognition	<b>Pass</b>
Define Class	Define Class	<b>Pass</b>
What is JavaScript	What is JavaScript	<b>Pass</b>
How is C++ different than Java	how is C plus plus different in Java	<b>Fail</b>
Define Constructor	Define Constructor	<b>Pass</b>
What is Destructor	what is destruct	<b>Fail</b>
How to use while loop	how to use while loop	<b>Pass</b>
Difference between for and while	difference between for and while	<b>Pass</b>
How to give input in C++	how to give input in C plus plus	<b>Pass</b>
How to print output in C++	how to print output in C plus plus	<b>Pass</b>
What is operator overloading	what is operator overloading	<b>Pass</b>
Some random text	some random text	<b>Pass</b>

#### 6.2.4 Database Search for Answer

This test case ensures the accuracy of the answering capability of the tool to the asked questions.



<b>Test Case Name</b>	<b>Database search for answer</b>
Test Case Description	Once the input is made either using a mic or not, the question asked is used for searching answer in the database. This test case validates the accuracy of answer detection in the database
Pre-conditions	Question to be asked is ready
Input test data	See table below
Steps to be Executed	Type the question in the text box and click go
Expected Results	Answers related to the question should be displayed with 100% accuracy
Actual Results	91.66% accurate
Pass/Fail	<b>Pass</b>

Currently, the database holds the answer for the definition of class, functions, destructor and input in C++. So, to measure the accuracy, questions related to these topics are asked. If questions related to topics other than mentioned above is asked, answer returned is null.

Sample asked questions and generated answers test result

<b>Question typed and asked</b>	<b>Generated answer</b>	<b>Result – Pass/Fail</b>
What is class	Related to Class	<b>Pass</b>
Define Class	Related to Class	<b>Pass</b>
What is function	Related to Functions	<b>Pass</b>
How to use function	Related to Class	<b>Pass</b>
Define function	Related to Class	<b>Pass</b>
Define Destructor	Related to destructor	<b>Pass</b>
What is Destructor	Related to destructor	<b>Pass</b>
How to give input in C++	Related to input in C++	<b>Pass</b>
Cin in C++	No matching result found	<b>Fail</b>

Difference between constructor and overloaded operator	Related to constructor and overloaded operator	<b>Pass</b>
How to use functions in class	Related to functions	<b>Pass</b>
Some random question	No matching result found	<b>Pass</b>
Function class destructor	Related to destructor	-

For the last question asked in the table, the answer returned is related to destructor because matching percentage is higher for *destructor* keyword than other two.

### 6.2.5 Change Status

This test case ensures if the user can change their status accurately without any error.

Test Case Name	Change status
Test Case Description	This test case verifies if status is changed accurately upon request
Pre-conditions	-
Input test data	-
Steps to be Executed	Change the status multiple times and see if new status is saved and reflected
Expected Results	Status should change with 100% accuracy
Actual Results	100 % success
Pass/Fail	<b>Pass</b>

### 6.2.6 Send Message in Online Chat

This test case ensures if the user can send messages successfully.

Test Case Name	Send Message
Test Case Description	Verify if messages are sent successfully

Pre-conditions	-
Input test data	Any text message
Steps to be Executed	a. Type the message in the text box b. Click send button Repeat the process multiple times
Expected Results	Each message should be sent successfully and saved in the file in the server
Actual Results	100 % success
Pass/Fail	<b>Pass</b>

### 6.2.7 Receive Messages Sent in Online Chat

This test case ensures the user can receive the messages successfully.

<b>Test Case Name</b>	<b>Receive Messages</b>
Test Case Description	This test case verifies if messages sent by other users are received and displayed
Pre-conditions	Multiple users are present in the online chat
Input test data	-
Steps to be Executed	Try sending message and see if it is displayed on the screen in the chat box in another users' screen
Expected Results	Each message sent by every user should appear in the chat box
Actual Results	100 % success
Pass/Fail	<b>Pass</b>

### 6.2.8 Save Online User Details

This test case ensures if the user details are saved successfully for managing the user's session. This detail assists in online communication including status updates. This also helps in managing the screen sharing invitation.

<b>Test Case Name</b>	<b>Save online user details</b>
Test Case Description	This test case verifies if online user details are saved in the text in the server
Pre-conditions	User is using the system
Input test data	-
Steps to be Executed	<ol style="list-style-type: none"> <li>a. Test the user details in the dedicated text file in the server</li> <li>b. Repeat with different user credentials with different id, name and different status</li> </ol> Repeat the process several times.
Expected Results	<p>Following user details should be saved.</p> <ol style="list-style-type: none"> <li>a. Status</li> <li>b. ID</li> <li>c. Name</li> </ol>
Actual Results	100 % success
Pass/Fail	<b>Pass</b>

### 6.2.9 Open Screen Sharing Room

This test case ensures if the user can open a screen sharing room successfully.

<b>Test Case Name</b>	<b>Open Screen sharing room</b>
Test Case Description	This test case validates the accuracy of opening a screen sharing room upon request
Pre-conditions	<ol style="list-style-type: none"> <li>a. Microphone and use of camera is enabled in browser</li> <li>b. Required Extension is installed in the browser</li> </ol>
Input test data	-
Steps to be Executed	<ol style="list-style-type: none"> <li>a. Click the open room button</li> <li>b. Choose window to share in the pop-up dialog box</li> </ol>

	c. Click share button Repeat the process several times.
Expected Results	A new screen shared room should be created successfully in every attempt
Actual Results	100 % success
Pass/Fail	Pass

### 6.2.10 Send Invitation

This test case ensures that the user can send an invitation to other users successfully.

Test Case Name	Send invitation
Test Case Description	This test case validates the accuracy of sending a request to other users.
Pre-conditions	<ul style="list-style-type: none"> <li>a. Other online user's details are available on the server in the online user list file</li> <li>b. Online users are listed in the screen with check boxes</li> <li>c. A unique room id is provided to detect the invitation</li> </ul>
Input test data	-
Steps to be Executed	<ul style="list-style-type: none"> <li>a. Choose the online users to send the invitation from the list and check the checkboxes</li> <li>b. Click send invitation</li> </ul> Repeat the process several times.
Expected Results	The full invitation details that include inviter and invitee details along with unique id is saved in invitation text file on the server.
Actual Results	100 % success
Pass/Fail	<b>Pass</b>

### 6.2.11 Lab Assistant Hours' Update Verification

This test case ensures that the user can update the lab assistant hours successfully.

<b>Test Case Name</b>	<b>Lab assistant hours' verification</b>
Test Case Description	This test case validates if the updates made in the lab assistant hours' table is saved and reflected
Pre-conditions	User can modify the table
Input test data	
Steps to be Executed	<ol style="list-style-type: none"> <li>a. Add new hours to the schedule or modify/delete the existing hours.</li> <li>b. Hit save button</li> <li>c. Repeat the process several times.</li> </ol>
Expected Results	New updated table should be saved and reflected in every iteration
Actual Results	100 % success
Pass/Fail	<b>Pass</b>

### 6.2.12 Send Contact Message

This test case ensures that the user can send contact messages successfully using the contact form.

<b>Test Case Name</b>	<b>Send Contact message</b>
Test Case Description	This test case validates if users can send the contact messages and are saved in the database
Pre-conditions	-
Input test data	<p>Text input for following fields:</p> <ol style="list-style-type: none"> <li>a. First name</li> <li>b. Last name</li> <li>c. Email</li> </ol>

	d. Message
Steps to be Executed	a. Fill in the input details and click on send button b. Check the message in the database c. Repeat the process for multiple times
Expected Results	The message should be sent each time and saved in the database with 100% accuracy
Actual Results	100 % success
Pass/Fail	<b>Pass</b>

### 6.2.13 Add a New Member to the System

This test case ensures that the user can add a new member to the system successfully.

<b>Test Case Name</b>	<b>Add a new member</b>
Test Case Description	This test case validates if a new user can be added to the system
Pre-conditions	User can add a new member to system and new member details are provided
Input test data	Text input for following fields: a. First name b. Last name c. User type of the new member
Steps to be Executed	a. Fill in the input details and click on confirm add button to add a new member b. Repeat the process for multiple new members
Expected Results	Each member must be added to the table and hence to the database
Actual Results	100 % success
Pass/Fail	<b>Pass</b>

### 6.2.14 Edit a Member in the System

This test case ensures that the user can edit an existing member in the system successfully.

<b>Test Case Name</b>	<b>Edit a member</b>
Test Case Description	This test case validates if a member can be edited successfully in the system
Pre-conditions	User can edit a member in the system
Input test data	Text input for fields that needs to be modified which can be: <ul style="list-style-type: none"> <li>a. First name and/or</li> <li>b. Last name and/or</li> <li>c. User type of the new member</li> </ul>
Steps to be Executed	<ul style="list-style-type: none"> <li>a. Fill in the modified input details and click on confirm edit button to edit the member</li> <li>b. Repeat the process for multiple members</li> </ul>
Expected Results	Each member must be edited successfully and must be reflected in the database and system
Actual Results	100 % success
Pass/Fail	<b>Pass</b>

### 6.2.15 Delete a Member from the System

This test case ensures that the user can delete an existing member from the system successfully.

<b>Test Case Name</b>	<b>Delete a member</b>
Test Case Description	This test case validates if a member can be deleted successfully from the system
Pre-conditions	User can delete a member from the system
Input test data	-



Steps to be Executed	<ul style="list-style-type: none"> <li>a. Select member to delete from the list and confirm delete</li> <li>b. Repeat the process for multiple members</li> </ul>
Expected Results	Each member must be deleted successfully and must be reflected in the database and system
Actual Results	100 % success
Pass/Fail	<b>Pass</b>

### 6.2.16 Delete a Contact Message from the System

This test case ensures that the user can delete contact messages from the system successfully.

<b>Test Case Name</b>	<b>Delete a Contact Message</b>
Test Case Description	This test case validates if a member can delete a contact message from the system
Pre-conditions	User can delete a message from the system
Input test data	-
Steps to be Executed	<ul style="list-style-type: none"> <li>a. Select a message to delete from the list and confirm delete</li> <li>b. Repeat the process for multiple messages</li> </ul>
Expected Results	Each message must be deleted successfully and must be reflected in the database and system
Actual Results	100 % success
Pass/Fail	<b>Pass</b>

### 6.2.17 Search Content in the Page

This test case ensures that the user can do the searching of the content in the page successfully.

<b>Test Case Name</b>	<b>Search content in the page</b>
Test Case Description	This test case validates if a member can perform a search in the current page in the tool
Pre-conditions	-
Input test data	Any term to search in the page
Steps to be Executed	a. Type the search term in the search text box field b. Repeat the process with multiple searching terms
Expected Results	The search result should be displayed dynamically as soon as typing in the search field starts with the closest matching contents in the page
Actual Results	The searching result was seen as soon as typing started
Pass/Fail	<b>Pass</b>

### 6.3 Integration Testing

Integration testing was done for various major integration cases to make sure different components work together successfully. Following are different cases of integration testing done.

#### 6.3.1 ‘Searching the Database’ Integrated with ‘Speech Recognition’ Functionality

The searching of the database to look for answers to the question was integrated with speech recognition functionality and integration testing was done to ensure the functionalities work best together.

<b>Test Case Name</b>	<b>Testing of Integration of database search for answer with speech recognition</b>
Test Case Description	This test case validates if a member can use the speech recognition to look for the answer searching the database
Pre-conditions	Microphone is enabled in the browser

Input test data	Test 1: Voice input – “How to use Function?” Test 2: Voice input – “What is Class?” Test 3: Voice input – “Random question?”
Steps to be Executed	a. Click on mic button to activate the speech b. Ask the question
Expected Results	Answer related to asked question is should be displayed in the answer box. If no answer is found, no answer found should be displayed.
Actual Results	For test 1 and 2, the answers were displayed. For test 3, no answer found was displayed.  100 % success
Pass/Fail	<b>Pass</b>

### 6.3.2 ‘Send/Receive Messages’ Integrated with ‘Status Changing Mechanism’

The sending messages, receiving messages and change status were integrated together to give full service of online communication.

<b>Test Case Name</b>	<b>Testing of Integration of sending/receive messages and change status</b>
Test Case Description	This test case validates if a member can change status and send/receive messages at the same time.
Pre-conditions	-
Input test data	Test input data sets: 1. Text input – “Hello, this is US.”. Change status from Available to Away. 2. Text input – “This is Santosh.”. Change status from Away to Busy. 3. Text input – “This is the third message.”. Change status from Busy to Available.

Steps to be Executed	<ul style="list-style-type: none"> <li>a. Type the message in the message text box and click send</li> <li>b. Change the status as mentioned in the test input data sets above.</li> <li>c. Repeat the process for all the test input data sets</li> </ul>
Expected Results	Changing status and sending a message should happen successfully without any error. The messages should be displayed on the screen (which means the sent message was stored in the database successfully and hence was received by all the users).
Actual Results	All the test input data sets were used. Changing of status and message sending were executed successfully. 100 % success
Pass/Fail	<b>Pass</b>

### 6.3.3 ‘Open Screen Sharing Room’ Integrated with ‘Send Invitation’

Open screen sharing room was integrated with send invitation to give the user the ability to send an invitation to available users in the online user's list to the opened room.

<b>Test Case Name</b>	<b>Testing of Integration of open screen sharing room and send invitation</b>
Test Case Description	This test case validates if a member can send an invitation to other users to the opened screen sharing room.
Pre-conditions	<ul style="list-style-type: none"> <li>1. User can open a screen sharing room</li> <li>2. Other users are available to send the invitation</li> </ul>
Input test data	-
Steps to be Executed	<ul style="list-style-type: none"> <li>a. Open a screen sharing room</li> </ul>

	<ul style="list-style-type: none"> <li>b. Choose the users from the online user's list to send the invitation to the created room.</li> <li>c. Repeat process for multiple times</li> </ul>
Expected Results	The user should be able to open a screen sharing room and be able to send an invitation to all the chosen users from the online user's list. The invitation details should be saved in the invitation text file in the server.
Actual Results	The user was able to open the screen sharing room. The invitation details were successfully registered in the invitation text file in the server.  100 % success
Pass/Fail	<b>Pass</b>

#### 6.3.4 'User Verification' Integrated with 'Save Online User's Details'

User verification was integrated with save online user's details so that after every successful user verification, user's details is saved on the server. User's details need to be saved to process some important function such as changing user's status, sending invitation and messages, etc.

<b>Test Case Name</b>	<b>Testing of Integration of user verification and save online user's details</b>
Test Case Description	This test case validates if necessary user's details are saved in the server after each successful user verification process.
Pre-conditions	1. The user has user-name and password to start user verification process.
Input test data	Text inputs – user name and password of the user
Steps to be Executed	a. Fill in the user name and password to initiate the user verification process

	b. Repeat process for multiple times
Expected Results	Following user's necessary details are saved in the online user list text file in the server. a. ID b. Name c. Status (default is Available)
Actual Results	Each time, necessary user's details were saved in a dedicated file in the server. 100 % success
Pass/Fail	<b>Pass</b>

#### 6.4 Overall System Testing

Overall System Testing was carried out to test all the functionalities available in the system to meet the desired system requirement specifications. It was also performed to measure the accuracy of the entire system.

All the test cases are described in details below. The test cases included from URL validation to GUI testing to all the major and minor functionality of the system.

##### 6.4.1 System URL Validation

<b>Test Case ID</b>	<b>1</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Verify whether web tool URL is valid
Pre-conditions	User have the URL address to the tool
Input test data	URL address
Steps to be Executed	a. Open any web browser b. Enter web tool URL address in the address box c. Press Enter Repeat process using other browsers.

Expected Results	URL should open properly to the home page of the tool
Actual Results	URL opened as expected
Pass/Fail	<b>Pass</b>

## 6.4.2 Home page – Guest (After Launching the System in the Browser)

### 6.4.2.1 Main Menu Bar

<b>Test Case ID</b>	<b>2</b>
<b>Test Case Type</b>	<b>GUI</b>
Test Case Description	Verify whether full menu bar is available on top of the home page
Pre-conditions	User is on the home page of the tool
Input test data	-
Steps to be Executed	Check the menu bar items list
Expected Results	A list of the items should be present in the main menu bar as follows: Home, ask a question, ask a tutor, faq, lab assistant hours, login and contact
Actual Results	Main menu bar was present with given item lists.
Pass/Fail	<b>Pass</b>

### 6.4.2.2 Main Menu Test

<b>Test Case ID</b>	<b>3</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Verify all menus in the main menu bar
Pre-conditions	User is not logged in to the system
Input test data	-
Steps to be Executed	Click on each menu item

Expected Results	<ul style="list-style-type: none"> <li>a. Home menu is the current page</li> <li>b. FAQ menu should redirect to FAQ page with the list of C++ FAQs</li> <li>c. Lab Assistant Hours menu should redirect to Lab Assistant hours' page where user can see the current lab assistant hours for the week</li> <li>d. Contact menu should redirect to contact page with a form to contact system admin</li> <li>e. Every other menu should redirect to login page</li> </ul>
Actual Results	Only FAQ, Lab Assistant Hours and Contact menu redirected to their own page. Rest of the menus lead to the login page.
Pass/Fail	<b>Pass</b>

#### 6.4.2.3 Home Page Interface Look

<b>Test Case ID</b>	<b>4</b>
<b>Test Case Type</b>	<b>GUI</b>
Test Case Description	Verify the look and feel of the home page
Pre-conditions	User is on the home page of the tool
Input test data	-
Steps to be Executed	-
Expected Results	<ul style="list-style-type: none"> <li>a. The main navigation bar should be present on top of the page</li> <li>b. System should have an imager slider on top of the page</li> <li>c. A welcome message box should be present</li> <li>d. More columns should be present to display additional information</li> </ul>



Actual Results	Home page interface included everything as expected.
Pass/Fail	<b>Pass</b>

#### 6.4.2.4 Image Slider

<b>Test Case ID</b>	<b>5</b>
<b>Test Case Type</b>	<b>GUI &amp; Functions</b>
Test Case Description	Verify the look and feel of the image slider
Pre-conditions	User is on the Home page of the tool
Input test data	-
Steps to be Executed	-
Expected Results	<ul style="list-style-type: none"> <li>a. Image should change with animation every 3 secs</li> <li>b. Image sub title should change on the bottom of the page along with the image</li> </ul>
Actual Results	Images changed with subtitle as expected.
Pass/Fail	<b>Pass</b>

#### 6.4.3 Login

##### 6.4.3.1 Login Form/Interface

<b>Test Case ID</b>	<b>6</b>
<b>Test Case Type</b>	<b>GUI</b>
Test Case Description	Verify whether Login page is displayed with appropriate login form
Pre-conditions	User is on the Login page of the tool
Input test data	-
Steps to be Executed	-
Expected Results	Login form should be displayed with at least following fields:

	<ul style="list-style-type: none"> <li>a. User Name field</li> <li>b. Password Field</li> <li>c. Login button</li> </ul>
Actual Results	Login form with required fields was displayed.
Pass/Fail	<b>Pass</b>

#### 6.4.3.2 Main Menu Bar Interface and Functional Test

Refer to test 2 and 3 in sections 6.4.2.1 and 6.4.2.2. Same test cases apply.

#### 6.4.3.3 Login Test

<b>Test Case ID</b>	<b>7</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Verify if the user can login successfully
Pre-conditions	The user must be registered in the database.
Input test data	Correct User Name and Password
Steps to be Executed	<ul style="list-style-type: none"> <li>a. Enter correct user name and password in the form</li> <li>b. Click Login button</li> </ul>
Expected Results	The user must successfully login to the system and authenticated home page should be displayed. In addition, the user's name should display on the main menu bar.
Actual Results	As expected, the user could login and name appeared in the main menu bar on the authenticated home page with the full menu.
Pass/Fail	<b>Pass</b>

**Negative Login Test**

<b>Test Case ID</b>	<b>8</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Verify if the unregistered user can login to the system
Pre-conditions	
Input test data	Incorrect User Name and Password
Steps to be Executed	a. Enter incorrect user name and password in the form b. Click Login button
Expected Results	The user must not be able to login. The Proper error should be displayed and prompt to login again.
Actual Results	The user was not able to login with incorrect credentials and error was displayed. The user was prompted to enter username and password again.
Pass/Fail	<b>Pass</b>

<b>Test Case ID</b>	<b>9</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Verify if entering only correct user name with empty password field, user can login to the system
Pre-conditions	User must be registered
Input test data	Valid User Name and empty Password
Steps to be Executed	a. Enter correct user name and no password in the form b. Click Login button
Expected Results	The user must not be able to login. The Proper error should be displayed and prompt to login again.
Actual Results	The user was not able to login with only valid username and empty password field. An error was also displayed. The user was prompted to enter username

	and password again.
Pass/Fail	<b>Pass</b>

<b>Test Case ID</b>	<b>10</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Verify if entering empty user name with correct password field, user can login to the system
Pre-conditions	User's password is registered
Input test data	Empty User Name and valid Password
Steps to be Executed	<ol style="list-style-type: none"> <li>a. Enter only valid password in the password field with empty user name field in the form</li> <li>b. Click Login button</li> </ol>
Expected Results	The user must not be able to login. The Proper error should be displayed and prompt to login again.
Actual Results	The user was not able to login with only valid password and empty username field. An error was also displayed. The user was prompted to enter the username and password again.
Pass/Fail	<b>Pass</b>

<b>Test Case ID</b>	<b>11</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Verify if the user can login with empty username and empty password field
Pre-conditions	-
Input test data	-
Steps to be Executed	Click Login button
Expected Results	The user must not be able to login. The Proper error should be displayed and prompt to login again.

Actual Results	The user was not able to login. An error was displayed. The user was prompted to enter the username and password again.
Pass/Fail	<b>Pass</b>

#### 6.4.3.4 Password Field Mask Test

<b>Test Case ID</b>	<b>12</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Verify if the password is masked in the password field in the login form i.e. password must be in bullets or asterisks
Pre-conditions	-
Input test data	Any registered or unregistered password
Steps to be Executed	a. Enter some characters in the password field
Expected Results	The password field should display the characters in asterisks or bullets such that it is not visible on the screen
Actual Results	The characters in the password field were displayed in bullets.
Pass/Fail	<b>Pass</b>

#### 6.4.3.5 Password Field Case Sensitivity Test

<b>Test Case ID</b>	<b>13</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Test if login handles case sensitivity of the password
Pre-conditions	The password is registered user's password which is originally in lower case changed to upper case or vice

	versa
Input test data	Valid user name and case changed valid registered password
Steps to be Executed	a. Enter username and case changed password in respective fields in the form
Expected Results	The user should not be able to login and the appropriate error message should be displayed. It should prompt the user to log in again.
Actual Results	The user was not able to login and an error message was displayed. The user was prompt to log in again.
Pass/Fail	<b>Pass</b>

#### 6.4.3.6 Password Field Copy/Paste Safety

<b>Test Case ID</b>	<b>14</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Test if password entered in password field can be copied from the field
Pre-conditions	-
Input test data	Valid user name and password
Steps to be Executed	a. Enter username and password in respective fields in the form b. Copy and paste password field contents on another tool/screen
Expected Results	The password should not be copied and pasted.
Actual Results	The password could not be copied and pasted.
Pass/Fail	<b>Pass</b>

## 6.4.4 Contact

### 6.4.4.1 Contact Form/Interface

<b>Test Case ID</b>	<b>15</b>
<b>Test Case Type</b>	<b>GUI</b>
Test Case Description	Test whether Contact page is displayed with appropriate contact form
Pre-conditions	User is on the contact page of the tool
Input test data	-
Steps to be Executed	-
Expected Results	Contact form should be displayed with at least following fields: a. User's First Name Field b. User's Last Name Field c. Email Field d. Message Field
Actual Results	Contact form with required fields was displayed.
Pass/Fail	<b>Pass</b>

### 6.4.4.2 Main Menu Bar Interface and Functional Test

Refer to test 2 and 3 in sections 6.4.2.1 and 6.4.2.2. Same test cases apply.

### 6.4.4.3 Contact with Message Test

<b>Test Case ID</b>	<b>16</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Verify if user can contact successfully (Send query/concern as message)
Pre-conditions	User is in contact page

Input test data	Text input for First name, Last name, Email, Message
Steps to be Executed	a. Enter valid first name, last name, email, and messages b. Click Send button
Expected Results	The user must successfully send a message (contact) to the system and a modal view with message send result should appear.
Actual Results	As expected user was able to send a message.
Pass/Fail	<b>Pass</b>

### Negative Login Test

<b>Test Case ID</b>	<b>17</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Verify if entering first name, last name, email, and empty message, user can contact the system
Pre-conditions	User is in contact page
Input test data	Text input for first name, last name and email
Steps to be Executed	a. Enter first name, last name and email in the form b. Click Send button
Expected Results	User must not be able to contact the system and missing field error should be displayed
Actual Results	User was not able to contact the system with empty message field
Pass/Fail	<b>Pass</b>

<b>Test Case ID</b>	<b>18</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Verify if entering first name, last name, empty email, and message, user can contact the system



Pre-conditions	User is in contact page
Input test data	Text input for the first name, last name, and message
Steps to be Executed	a. Enter first name, last name, and message in the form b. Click Send button
Expected Results	User must not be able to contact the system and missing field error should be displayed
Actual Results	User was not able to contact the system with empty email field
Pass/Fail	<b>Pass</b>
<b>Test Case ID</b>	<b>19</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Verify if entering first name, empty last name, email, and message, user can contact the system
Pre-conditions	User is in contact page
Input test data	Text input for the first name, email, and message
Steps to be Executed	a. Enter first name, email, and message in the form b. Click Send button
Expected Results	User must not be able to contact the system and missing field error should be displayed
Actual Results	User was not able to contact the system with empty last name field
Pass/Fail	<b>Pass</b>

<b>Test Case ID</b>	<b>20</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Verify if entering empty first name, last name, email, and message, user can contact the system
Pre-conditions	User is in contact page

Input test data	Text input for the last name, email, and message
Steps to be Executed	a. Enter last name, email, and message in the form b. Click Send button
Expected Results	User must not be able to contact the system and missing field error should be displayed
Actual Results	User was not able to contact the system with empty first name field
Pass/Fail	<b>Pass</b>

## 6.4.5 Logout

### 6.4.5.1 Logout Interface

<b>Test Case ID</b>	<b>21</b>
<b>Test Case Type</b>	<b>GUI</b>
Test Case Description	Verify the look and feel of the Logout interface.
Pre-conditions	User is in the Logged-out page of the tool
Input test data	-
Steps to be Executed	-
Expected Results	Logged out page should have a message saying the user is logged out of the system.
Actual Results	Log out page displayed the message that user has been logged out.
Pass/Fail	<b>Pass</b>

### 6.4.5.2 Main Menu Bar Interface and Functional Test

Refer to test 2 and 3 in sections 6.4.2.1 and 6.4.2.2. Same test cases apply.

### 6.4.5.3 Back Button After Log Out

<b>Test Case ID</b>	<b>22</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Verify if the user can go back to logged in content after coming in the logged-out screen.
Pre-conditions	User is in the logged-out page of the tool
Input test data	-
Steps to be Executed	Click on the back button in the browser navigation ←.
Expected Results	The user should not be redirected to any of the authenticated pages. Instead, the user should be redirected to login page.
Actual Results	The user was redirected to the login page.
Pass/Fail	<b>Pass</b>

### Following test cases are applicable when logged in as a Student

#### 6.4.6 Home Page - Student

##### 6.4.6.1 Interface

Refer to test 4 in section 6.4.2.3. Same test case applies.

##### 6.4.6.2 Main Menu Bar

<b>Test Case ID</b>	<b>23</b>
<b>Test Case Type</b>	<b>GUI</b>
Test Case Description	Verify whether full menu bar is available on top of the home page for <b>student</b> as per requirement specifications for student user role
Pre-conditions	User is in the logged in home page of the tool
Input test data	-

Steps to be Executed	Check the main menu bar items list
Expected Results	A dedicated list of the items should be present in the main menu bar as shown below: Home, Ask a Question, Ask a Tutor (online chat, share screen), Study Materials (Chapters, Lab Assistant hours, FAQ), logout and Contact.
Actual Results	As expected all main menu items were present in the main menu bar.
Pass/Fail	<b>Pass</b>

#### 6.4.6.3 Main Menu Test

<b>Test Case ID</b>	<b>24</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Validate if all menus in the main menu bar including drop down buttons functions as expected
Pre-conditions	User is in the logged in home page of the tool
Input test data	-
Steps to be Executed	<ol style="list-style-type: none"> <li>a. Click on each menu item</li> <li>b. Click on drop down button to check more menu items</li> </ol>
Expected Results	<ol style="list-style-type: none"> <li>a. Each item should redirect to the respective page</li> <li>b. Each drop down button should work effectively.</li> <li>c. Log out should log out the user out of the system.</li> </ol>
Actual Results	The menu items worked as expected and redirected to each individual page.
Pass/Fail	<b>Pass</b>

#### 6.4.6.4 Verify Authenticated Home Page URL After Logging Out

<b>Test Case ID</b>	<b>25</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Verify if the user can go back to authenticated home page after logging out.
Pre-conditions	-
Input test data	-
Steps to be Executed	<ol style="list-style-type: none"> <li>a. Log in to the system.</li> <li>b. Copy the URL of the home page.</li> <li>c. Log out.</li> <li>d. Paste the copied URL into the browser.</li> </ol>
Expected Results	The URL should not be redirected to the authenticated home page content. Instead, it should redirect to the log in page.
Actual Results	The user was redirected to the login page.
Pass/Fail	<b>Pass</b>

#### 6.4.7 Ask a Question Page - Student

##### 6.4.7.1 Interface

<b>Test Case ID</b>	<b>26</b>
<b>Test Case Type</b>	<b>GUI</b>
Test Case Description	Verify the look and feel of the ask a question page
Pre-conditions	User is logged in and is in ask a question page of the tool
Input test data	-
Steps to be Executed	-
Expected Results	The user should be able to see the following fields:

	<ul style="list-style-type: none"> <li>a. Instructions set on how to use the feature</li> <li>b. A text field</li> <li>c. A button with microphone icon which should turn black on hover.</li> <li>d. 'Go' button</li> </ul>
Actual Results	All the fields were present and behaved as expected with a set of instructions to use the feature on top.
Pass/Fail	<b>Pass</b>

#### 6.4.7.2 Main Menu Bar

Refer to test 23 in section 6.4.5.2. Same test case applies.

#### 6.4.7.3 Main Menu Test

Refer to test 24 in section 6.4.5.3. Same test case applies.

#### 6.4.7.4 Speech Recognition Test

<b>Test Case ID</b>	<b>27</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Verify if the system can recognize the user's speech
Pre-conditions	<ul style="list-style-type: none"> <li>a. User is logged in and is in ask a question page of the tool</li> <li>b. Microphone is enabled in the browser</li> </ul>
Input test data	Voice input – "This is my name"
Steps to be Executed	<ul style="list-style-type: none"> <li>a. Click on mic button in the screen</li> <li>b. Speak the voice input within next 5 secs</li> </ul>
Expected Results	<p>The user should be able to see following observations:</p> <ul style="list-style-type: none"> <li>a. Microphone button should turn red and animation should start and last for 5 secs</li> </ul>

	<ul style="list-style-type: none"> <li>b. “This is my name” should be displayed in the text field on the screen at the end of 5 secs</li> <li>c. After 5 secs, mic animation should stop and go back to normal state</li> </ul>
Actual Results	Microphone animation started as soon as mic button was clicked. The animation lasted for 5 secs and the spoken text was visible in the text field.
Accuracy	Excellent
Pass/Fail	<b>Pass</b>

#### 6.4.7.5 Answer Result Box Validation

##### Using speech recognition

<b>Test Case ID</b>	<b>28</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Verify if the system can display the answer to the question asked using the speech recognition
Pre-conditions	<ul style="list-style-type: none"> <li>a. User is logged in and is in ask a question page of the tool</li> <li>b. Microphone is enabled in the browser</li> </ul>
Input test data	Voice input – “Define Class”
Steps to be Executed	<ul style="list-style-type: none"> <li>a. Click on mic button in the screen</li> <li>b. Speak “Define Class” within next 5 secs</li> </ul>
Expected Results	The user should be able to see the answer related to ‘class’ in the answer box in the screen after 5 secs.
Actual Results	The answer box was populated with the answer related to class.
Accuracy	Excellent
Pass/Fail	<b>Pass</b>

**Manually typing question (without using speech recognition)**

<b>Test Case ID</b>	<b>29</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Verify if the system can display the answer to the question asked typing in the given text field
Pre-conditions	a. User is logged in and is in ask a question page of the tool
Input test data	Text input – “Define Class”
Steps to be Executed	a. Type the question in the text box field b. Click on ‘Go’ button in the screen or press enter on the keyboard
Expected Results	The user should be able to see the answer related to ‘class’ in the answer box in the screen.
Actual Results	The answer box was populated with the answer related to class.
Accuracy	Excellent
Pass/Fail	<b>Pass</b>

**Negative test case**

<b>Test Case ID</b>	<b>30</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Verify if the system displays the answer with empty question field
Pre-conditions	a. User is logged in and is in ask a question page of the tool
Input test data	-
Steps to be Executed	a. Click on ‘Go’ button in the screen
Expected Results	The user should not be able to see any kind of text in the answer box. It should display “No matching result



	found”.
Actual Results	The answer box said, “No matching result found”.
Pass/Fail	<b>Pass</b>

<b>Test Case ID</b>	<b>31</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Verify if the system displays the answer to the question for randomly asked question.
Pre-conditions	a. User is logged in and is in ask a question page of the tool (Microphone is enabled in the browser if mic button is used)
Input test data	Text/Voice input – “What is my name”
Steps to be Executed	a. Use microphone button to ask the question <b>OR</b> a. Type the question in the text box field b. Click on ‘Go’ button in the screen or press enter on the keyboard
Expected Results	The user should not be able to see any kind of text in the answer box. It should display “No matching result found”.
Actual Results	The answer box said, “No matching result found”.
Accuracy	Excellent
Pass/Fail	<b>Pass</b>

#### 6.4.7.6 Verify Authenticated Ask a Question URL After Log Out

<b>Test Case ID</b>	<b>32</b>
<b>Test Case Type</b>	<b>Functional</b>

Test Case Description	Verify if the user can go back to authenticated ask a question page after logging out.
Pre-conditions	-
Input test data	-
Steps to be Executed	<ol style="list-style-type: none"> <li>a. Log in to the system.</li> <li>b. Navigate to Ask a Question Page.</li> <li>c. Copy the URL of the page.</li> <li>d. Log out.</li> <li>e. Paste the copied URL into the browser.</li> </ol>
Expected Results	URL should not be redirected to the authenticated Ask a Question page content. It should redirect to the log in page.
Actual Results	The user was redirected to the login page.
Pass/Fail	<b>Pass</b>

## 6.4.8 Online Chat Page - Student

### 6.4.8.1 Interface

<b>Test Case ID</b>	<b>33</b>
<b>Test Case Type</b>	<b>GUI</b>
Test Case Description	Verify the look and feel of the online chat communication page
Pre-conditions	User is logged in and is in online chat communication page of the tool
Input test data	-
Steps to be Executed	-
Expected Results	<p>The user should be able to see the following fields:</p> <ol style="list-style-type: none"> <li>a. A status button with color signal</li> <li>b. Online users list box</li> </ol>

	<ul style="list-style-type: none"> <li>c. Open room messages box</li> <li>d. New message text field</li> <li>e. Send button</li> <li>f. Message send Status box</li> <li>g. Go to screen sharing button</li> </ul>
Actual Results	All the fields were present as expected.
Pass/Fail	<b>Pass</b>

#### 6.4.8.2 Main Menu Bar

Refer to test 23 in section 6.4.5.2. Same test case applies.

#### 6.4.8.3 Main Menu Test

Refer to test 24 in section 6.4.5.3. Same test case applies.

#### 6.4.8.4 Change Status

<b>Test Case ID</b>	<b>34</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Verify whether the user can change the status
Pre-conditions	<ul style="list-style-type: none"> <li>a. User is in the online chat page of the tool</li> <li>b. User's current status is 'Available'.</li> </ul>
Input test data	-
Steps to be Executed	<ul style="list-style-type: none"> <li>a. Click on the status button which says 'Available' with the green signal by default.</li> <li>b. Choose the new status as 'Busy'</li> </ul>
Expected Results	<ul style="list-style-type: none"> <li>a. New status should be reflected in the button as 'Busy'</li> <li>b. The status should also be reflected in the online user's list with a red color signal with status in</li> </ul>

	parenthesis.
Actual Results	The status changed as expected. The status button changed to red with 'Busy as the button label text. The online user's list also got updated with new status.
Pass/Fail	<b>Pass</b>

#### 6.4.8.5 Online Users List Box

<b>Test Case ID</b>	<b>35</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Verify whether 'online users list' box gets populated with correct online users
Pre-conditions	<ul style="list-style-type: none"> <li>a. Multiple users are logged in from different machine</li> <li>b. User is in online chat page to verify the listing of users</li> </ul>
Input test data	-
Steps to be Executed	Check the online user's list box in the online chat page
Expected Results	All the logged in users must be listed in the box with their appropriate statuses
Actual Results	All different users were listed in the online user's list box with their correct statuses.
Pass/Fail	<b>Pass</b>

<b>Test Case ID</b>	<b>36</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Verify whether 'online users list' box gets updated when a member logs in and another log out.
Pre-conditions	<ul style="list-style-type: none"> <li>a. Multiple users are logged in from different machine</li> <li>b. User is in online chat page to verify the listing of</li> </ul>

	users
Input test data	-
Steps to be Executed	<ol style="list-style-type: none"> <li>a. One user with name 'Sam' logs in from one machine and another with name 'Dean' logs out</li> <li>b. Check the online user's list box in the online chat page</li> </ol>
Expected Results	The list should get updated. The list should have 'Sam' added to the list whereas 'Dean' should be removed from the list.
Actual Results	'Sam' was added to the list and 'Dean' was removed. The list was updated as expected.
Pass/Fail	<b>Pass</b>

<b>Test Case ID</b>	<b>37</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Test if other user's change of status is reflected in the online user list
Pre-conditions	<ol style="list-style-type: none"> <li>a. Multiple users are logged in from different machine</li> <li>b. User is in online chat page to verify the status of all the users in the online user's list box</li> </ol>
Input test data	-
Steps to be Executed	<ol style="list-style-type: none"> <li>a. An online user 'Sam' changes the status from 'Available' to 'Away'</li> <li>b. An online user 'Dean' changes the status from 'Available' to 'Busy'</li> <li>c. Observe the online user's list box</li> </ol>
Expected Results	<ol style="list-style-type: none"> <li>a. The status of 'Sam' should change to 'Away' and the status of 'Dean' should change to 'Busy'.</li> <li>b. The new status should be reflected for all the users</li> </ol>

	in the online user's list box.
Actual Results	The status changed for both 'Sam' and 'Dean' as expected. The status of them was reflected in the online user's list box.
Pass/Fail	<b>Pass</b>

#### 6.4.8.6 Send Message

<b>Test Case ID</b>	<b>38</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Verify whether user can send the message
Pre-conditions	a. User is in the online chat page of the tool
Input test data	Text input – "Hello World!"
Steps to be Executed	a. Type the message "Hello World!" in the message text field on the screen b. Click Send button
Expected Results	c. The message status box should say "Message Sent" in green. The message should also appear in the "Open room message" box.
Actual Results	The new message appeared in the open room messages box. The message status box was also updated with "Message Sent" in green.
Pass/Fail	<b>Pass</b>

<b>Test Case ID</b>	<b>39</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Verify if the user can send empty message
Pre-conditions	a. User is in the online chat page of the tool
Input test data	-

Steps to be Executed	a. Leave the message text field empty b. Click Send button
Expected Results	There should not be any change in the page.
Actual Results	No single change was seen.
Pass/Fail	<b>Pass</b>

#### 6.4.8.7 Open Room Messages Box

<b>Test Case ID</b>	<b>40</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Verify whether 'Open Room Messages' box gets populated with recent messages
Pre-conditions	a. User is in online chat page b. Recent 10 messages are saved in the server
Input test data	-
Steps to be Executed	Check the open room messages box in the online chat page
Expected Results	The box should be populated with the recent 10 messages
Actual Results	The box was populated with recent last 10 messages with time stamps.
Pass/Fail	<b>Pass</b>

<b>Test Case ID</b>	<b>41</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Verify whether 'Open Room Messages' box gets updated when a logged in member sends a message.
Pre-conditions	a. Multiple users are logged in from different machine b. Users are in the online chat page

	<ul style="list-style-type: none"> <li>c. A user sends message “This is Santosh”</li> <li>d. Another user Dean sends message “This is Dean” from different machine</li> </ul>
Input test data	Text input – “This is Santosh” and “This is Dean”
Steps to be Executed	<ul style="list-style-type: none"> <li>a. Type the message “This is Santosh” and click Send button</li> <li>b. Make the user Dean to type the message “This is Dean” and click send</li> <li>c. Check the Open Room Messages box in the online chat page</li> </ul>
Expected Results	The message box should get updated with two new messages from the user and Dean with correct timestamp. User’s message “This is Santosh” Should appear before Dean’s message “This is Dean”.
Actual Results	Both the messages were seen in the messages box. “This is Santosh” appeared before “This is Dean”.
Pass/Fail	<b>Pass</b>

<b>Test Case ID</b>	<b>42</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Verify whether the old messages are removed when message box limit of 20 messages is met.
Pre-conditions	<ul style="list-style-type: none"> <li>a. User is in online chat page</li> <li>b. User sends multiple messages until message box display overflows</li> </ul>
Input test data	Multiple text input
Steps to be Executed	<ul style="list-style-type: none"> <li>a. Keep typing messages and send until message display box overflows</li> <li>b. Observe the Open Room Messages box in the</li> </ul>



	online chat page
Expected Results	The oldest message should be truncated from the top and new messages should be listed with latest time stamp messages from the bottom of the list.
Actual Results	The old messages were removed from the top and new messages kept appearing from the bottom of the list
Accuracy	Excellent
Pass/Fail	<b>Pass</b>

#### 6.4.8.8 Go to Screen Sharing Button

<b>Test Case ID</b>	<b>43</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Verify whether 'Go to Screen Sharing' button redirects user to screen sharing page
Pre-conditions	User is in online chat page
Input test data	-
Steps to be Executed	Click on 'Go to Screen Sharing' button
Expected Results	The user should be redirected to the screen sharing page.
Actual Results	The user was directed to the screen sharing page.
Pass/Fail	<b>Pass</b>

#### 6.4.8.9 Verify Authenticated Online Chat URL After Log Out

<b>Test Case ID</b>	<b>44</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Verify if the user can go back to authenticated online chat page after logging out.

Pre-conditions	-
Input test data	-
Steps to be Executed	<ol style="list-style-type: none"> <li>a. Log in to the system.</li> <li>b. Navigate to Online chat Page.</li> <li>c. Copy the URL of the page.</li> <li>d. Log out.</li> <li>e. Paste the copied URL into the browser.</li> </ol>
Expected Results	URL should not be redirected to the authenticated Online chat page content. It should redirect to the log in page.
Actual Results	The user was redirected to the login page.
Pass/Fail	<b>Pass</b>

#### 6.4.9 Screen Sharing Page - Student

##### 6.4.9.1 Interface

<b>Test Case ID</b>	<b>45</b>
<b>Test Case Type</b>	<b>GUI</b>
Test Case Description	Test the look and feel of the screen sharing page
Pre-conditions	User is logged in and is in screen sharing page of the tool
Input test data	-
Steps to be Executed	-
Expected Results	<p>The user should be able to see following fields:</p> <ol style="list-style-type: none"> <li>a. A set of instructions box with two hyperlinks to screen sharing extension page</li> <li>b. Online users list box with check boxes</li> <li>c. Select All and Clear All buttons</li> <li>d. Send Invites Button</li> </ol>

	<ul style="list-style-type: none"> <li>e. Refresh online list buttons</li> <li>f. Room id text field box</li> <li>g. Open Room button</li> </ul>
Actual Results	All the fields were present as expected.
Pass/Fail	<b>Pass</b>

#### 6.4.9.2 Main Menu Bar

Refer to test 23 in section 6.4.5.2. Same test case applies.

#### 6.4.9.3 Main Menu Test

Refer to test 24 in section 6.4.5.3. Same test case applies.

#### 6.4.9.4 Online Users List Box

<b>Test Case ID</b>	<b>46</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Verify whether 'online users list' box gets populated with correct online users
Pre-conditions	<ul style="list-style-type: none"> <li>a. Multiple users are logged in from different machine</li> <li>b. User is in Screen sharing page to verify the listing of users</li> </ul>
Input test data	-
Steps to be Executed	Check the online user's list box in the screen sharing page
Expected Results	All the logged in users must be listed in the box with their appropriate statuses
Actual Results	All different users were listed in the online user's list box with their correct statuses.
Pass/Fail	<b>Pass</b>

<b>Test Case ID</b>	<b>47</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Verify whether 'online users list' box gets updated when a member logs in and another log out.
Pre-conditions	<ul style="list-style-type: none"> <li>a. Multiple users are logged in from different machine</li> <li>b. User is in screen sharing page to verify the listing of users</li> <li>c. Among multiple users, 'Sam' and 'Dean' are two users</li> </ul>
Input test data	-
Steps to be Executed	<ul style="list-style-type: none"> <li>a. User with name 'Sam' logs in from one machine and 'Dean' logs out</li> <li>b. Check the online user's list box in the screen sharing page</li> </ul>
Expected Results	The list should get updated. The list should have 'Sam' added to the list whereas 'Dean' should be removed from the list.
Actual Results	'Sam' was added to the list and 'Dean' was removed. The list was updated as expected.
Pass/Fail	<b>Pass</b>

<b>Test Case ID</b>	<b>48</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Verify if other user's change of status is reflected in the online user list
Pre-conditions	<ul style="list-style-type: none"> <li>a. Multiple users are logged in from different machine</li> <li>b. User is in screen sharing page to verify the status of all the users in the online user's list box</li> <li>c. Among multiple users, 'Sam' and 'Dean' are two</li> </ul>

	users
Input test data	-
Steps to be Executed	<ol style="list-style-type: none"> <li>a. An online user 'Sam' changes the status from 'Available' to 'Away'</li> <li>b. An online user 'Dean' changes the status from 'Available' to 'Busy'</li> <li>c. Observe the online user's list box</li> </ol>
Expected Results	<ol style="list-style-type: none"> <li>a. The status of 'Sam' should change to 'Away' and the status of 'Dean' should change to 'Busy'.</li> <li>b. The new status should be reflected for all in online users list box.</li> </ol>
Actual Results	The status changed for both 'Sam' and 'Dean' as expected. The status of them was reflected in the online user's list box.
Pass/Fail	<b>Pass</b>

#### 6.4.9.5 Checkbox Usability

<b>Test Case ID</b>	<b>49</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Verify whether the checkboxes in the online user list box functions properly
Pre-conditions	<ol style="list-style-type: none"> <li>a. Multiple users are listed in the online user's list box</li> <li>b. User is in Screen sharing page to verify the listing of users</li> </ol>
Input test data	-
Steps to be Executed	<ol style="list-style-type: none"> <li>a. Check and uncheck box clicking mouse button</li> <li>b. Repeat the process for multiple users</li> </ol>
Expected Results	The checkboxes should react to clicking normally and

	should check and uncheck appropriately
Actual Results	Checkboxes were updated normally.
Pass/Fail	<b>Pass</b>

#### 6.4.9.6 Select All Button

<b>Test Case ID</b>	<b>50</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Verify whether the clicking on Select All button checks all the checkboxes in the online user's list
Pre-conditions	a. Multiple users are listed in the online user's list box b. User is in Screen sharing page
Input test data	-
Steps to be Executed	Click on Select All button and observe the checkboxes
Expected Results	The checkboxes should all be checked with a single click on Select All button.
Actual Results	Checkboxes were all checked.
Pass/Fail	<b>Pass</b>

#### 6.4.9.7 Clear All Button

<b>Test Case ID</b>	<b>51</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Verify whether the clicking on Clear All button unchecks all the checkboxes in the online user's list
Pre-conditions	a. Multiple users are listed in the online user's list box b. User is in Screen sharing page
Input test data	-
Steps to be Executed	Click on Clear All button and observe the checkboxes

Expected Results	The checkboxes should all be unchecked with a single click on Clear All button.
Actual Results	Checkboxes were all unchecked.
Pass/Fail	<b>Pass</b>

#### 6.4.9.8 Open Room

<b>Test Case ID</b>	<b>52</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Verify if open room button prompts the user with share screen pop-up window.
Pre-conditions	a. User is in Screen sharing page
Input test data	-
Steps to be Executed	Click on Open Room button and observe
Expected Results	The share screen pop-up window should appear asking the user to choose the window to be shared with two buttons at bottom of the dialog box.
Actual Results	The share screen pop-up window appeared with application window to choose from. Also, two buttons with text label 'Share' and 'Cancel' were seen.
Pass/Fail	<b>Pass</b>

#### 6.4.9.9 Share Screen Pop – Up Window Interface

<b>Test Case ID</b>	<b>53</b>
<b>Test Case Type</b>	<b>GUI</b>
Test Case Description	Verify the look of the share screen pop-up window.
Pre-conditions	a. User is in Screen sharing page
Input test data	-

Steps to be Executed	a. Click on Open Room button and observe the pop-up window interface
Expected Results	<p>a. A pop-up window should have two tabs with title ‘Your Entire Screen’ and ‘Application window’.</p> <p>b. Each tab should present correspond screen shots of either entire screen or separate application window.</p> <p>c. The pop-up window should also have two buttons with text labels ‘Share’ and ‘Cancel’ at the bottom.</p>
Actual Results	The pop-up window had two tabs as ‘Your Entire Screen’ and ‘Application window’. Respective screen shots were also seen when tabs were changed. Two buttons at the bottom were also available.
Pass/Fail	<b>Pass</b>

#### 6.4.9.10 ‘Share’ Button in Share Screen Pop – Up Window

<b>Test Case ID</b>	<b>54</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Verify if the user can start screen sharing by clicking on the share button in the share screen pop-up window.
Pre-conditions	a. User is in Screen sharing page
Input test data	-
Steps to be Executed	<p>a. Click on Open Room button to display the pop-up window.</p> <p>b. Click on Share button</p>
Expected Results	<p>a. Screen sharing should start immediately.</p> <p>b. Unique URL for the screen sharing room should be listed on the screen.</p> <p>c. Live Shared Screen window should appear below</p>



	the unique URL listings.
Actual Results	The screen sharing started with unique URLs listed on the screen.
Pass/Fail	<b>Pass</b>

#### 6.4.9.11 ‘Cancel’ Button in Share Screen Pop – Up Window

<b>Test Case ID</b>	<b>55</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Verify if ‘Cancel’ button in share screen pop-up window exits the pop-up window.
Pre-conditions	a. User is in Screen sharing page
Input test data	-
Steps to be Executed	a. Click on Open Room button to display the screen share pop-up window. b. Click on Cancel button
Expected Results	The pop-up window should close.
Actual Results	The pop-up window closed when cancel button was clicked.
Pass/Fail	<b>Pass</b>

#### 6.4.9.12 ‘Tabs’ Usability in Share Screen Pop – Up Window

<b>Test Case ID</b>	<b>56</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Verify if tabs work appropriately as desired in the screen share pop-up window.
Pre-conditions	a. User is in Screen sharing page
Input test data	-

Steps to be Executed	<ul style="list-style-type: none"> <li>a. Click on Open Room button to display the screen share pop-up window.</li> <li>b. Play with tabs</li> <li>c. Observe the changes that happen with changing tabs</li> </ul>
Expected Results	<ul style="list-style-type: none"> <li>a. There should be two tab options.</li> <li>b. Changing tabs should work flawlessly.</li> <li>c. The screen shot of entire screen should appear when selecting 'Your Entire Screen' tab.</li> <li>d. The screen shot of separate application window screens should appear when selecting 'Application window' tab.</li> </ul>
Actual Results	<p>The pop-up window had two tabs with the title as 'Your Entire Screen' and 'Application window'. Selecting 'Your Entire Screen' displayed the screen shot image of the entire screen whereas choosing 'Application window' showed the images of each opened separate application windows.</p>
Pass/Fail	<b>Pass</b>

#### 6.4.9.13 LIVE Shared Screen Test

<b>Test Case ID</b>	<b>57</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Verify if application offers live shared screen
Pre-conditions	a. User is in Screen sharing page
Input test data	-
Steps to be Executed	a. Click on Open Room button to display the screen

	<p>share pop-up window.</p> <p>b. Choose the screen/application window to share</p> <p>c. Click Share and observe the shared screen window</p>
Expected Results	<p>a. The screen share should start as soon as share button is clicked.</p> <p>b. Full shared screen should appear in the dedicated html div box.</p> <p>c. It should reflect the live screen that is being shared.</p>
Actual Results	<p>Screen share started as soon as Share button was clicked. The shared screen/application window broadcasted live which could be seen in the dedicated html div box.</p>
Pass/Fail	<b>Pass</b>

#### 6.4.9.14 LIVE Shared Screen Interface

<b>Test Case ID</b>	<b>58</b>
<b>Test Case Type</b>	<b>GUI</b>
Test Case Description	Verify the look and feel of the shared screen window
Pre-conditions	a. User is in Screen sharing page with live shared window playing
Input test data	-
Steps to be Executed	Observe the live shared window
Expected Results	<p>a. Live window should be playing in dedicated box</p> <p>b. Live window should have a full screen (⌘) button at the right topmost part of the window</p>
Actual Results	<p>Live window was present with the full screen (⌘) button at the top right position of the window</p>
Pass/Fail	<b>Pass</b>

#### 6.4.9.15 Full Screen (⌘) Button in LIVE Shared Screen

<b>Test Case ID</b>	<b>59</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Verify if user can view shared screen in full screen view clicking on full screen (⌘) button
Pre-conditions	a. User is in Screen sharing page with live shared window playing
Input test data	-
Steps to be Executed	Click on the full screen (⌘) button
Expected Results	Live shared window should expand to full screen view
Actual Results	Clicking on full screen button expanded the shared screen view to full screen.
Pass/Fail	<b>Pass</b>

<b>Test Case ID</b>	<b>60</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Verify if user can get out of full screen view by clicking on (⌘) button
Pre-conditions	a. User is in live shared screen room page with full screen view
Input test data	-
Steps to be Executed	Click on the full screen exit (⌘) button
Expected Results	Live shared window should come back to normal view from full screen view
Actual Results	Clicking on the full screen exit (⌘) button collapsed the full screen view back to normal view.
Pass/Fail	<b>Pass</b>

#### 6.4.9.16 LIVE Shared Screen Unique URL Validation

<b>Test Case ID</b>	<b>61</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Verify if generated unique screen shared room URL redirects to a page with screen sharing session
Pre-conditions	<ul style="list-style-type: none"> <li>a. User is in Screen sharing page with live shared window playing</li> <li>b. Unique room URL is listed on the screen</li> </ul>
Input test data	-
Steps to be Executed	<ul style="list-style-type: none"> <li>a. Copy the unique room URL listed on the screen</li> <li>b. Open a new browser and paste the copied URL</li> </ul>
Expected Results	<ul style="list-style-type: none"> <li>a. The shared screen window should appear and the user should be able to see the screen sharing</li> <li>b. The unique room id should be listed on the top of the page</li> </ul>
Actual Results	A live shared screen appeared and started playing in the browser with the unique room id listed on the top of page
Accuracy	Excellent
Pass/Fail	<b>Pass</b>

#### Negative case

<b>Test Case ID</b>	<b>62</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Verify if generated unique screen shared room URL redirects to a page with screen sharing page when the actual screen sharing session has been ended by organizer

Pre-conditions	User has the unique URL to the room
Input test data	-
Steps to be Executed	<ol style="list-style-type: none"> <li>a. Copy the unique room URL listed on the screen</li> <li>b. Terminate the screen sharing session</li> <li>c. Open a browser and paste the copied URL</li> </ol>
Expected Results	The user should not able to view any kind of screen sharing session.
Actual Results	A new tab was opened with unique URL to the screen sharing room but no live screen sharing was going on. The page was blank with only unique id to the room.
Pass/Fail	<b>Pass</b>

#### 6.4.9.17 Send Invites to The Shared Screen Room

<b>Test Case ID</b>	<b>63</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Verify if user can send invitation to the selected user from the online user's list to the screen sharing room
Pre-conditions	<ol style="list-style-type: none"> <li>a. User is in Screen sharing page with live shared window playing</li> <li>b. Multiple users are online and are listed in the online user's list box in screen sharing page</li> </ol>
Input test data	-
Steps to be Executed	<ol style="list-style-type: none"> <li>a. Select the users to invite to the screen sharing room from the online user's list by checking the checkboxes</li> </ol> <p style="text-align: center;">OR</p> <p>Click 'Select All' button to select all online users at once</p>

	b. Click send invites button
Expected Results	<p>a. Invitation should be sent to all the selected users and a confirmation modal view (dialog box) should appear with confirmation of the invitation</p> <p>b. A modal view (dialog box) with invitation message should appear in each user's window who have been invited to the shared screen room.</p>
Actual Results	Once the 'send invites' button was clicked, a modal view appeared with invitation confirmation message. Similarly, all the selected users got invitation modal view appeared on the screen immediately after the screen sharing organizer clicked on send invites button.
Pass/Fail	<b>Pass</b>

### Negative cases

#### Send invites without opening a room

<b>Test Case ID</b>	<b>64</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Verify if the user can send invitation without opening a screen sharing room
Pre-conditions	<p>a. User is in Screen sharing page</p> <p>b. Multiple users are online and are listed in the online user's list box in screen sharing page</p>
Input test data	-
Steps to be Executed	<p>a. Select some members from the online user list to send invitation</p> <p>b. Click on send invites button</p>
Expected Results	a. The invitation should fail. A modal view should appear with the reason of invitation failure.

Actual Results	Invitation failed and a modal view appeared saying a screen sharing room need to be created first to send an invitation to other users.
Pass/Fail	<b>Pass</b>

#### Send invites without selecting any users

<b>Test Case ID</b>	<b>65</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Verify if user can send invitation without selecting any user from the online user's list to the screen sharing room
Pre-conditions	<ul style="list-style-type: none"> <li>a. User is in Screen sharing page and a live screen share room is already opened and ready</li> <li>b. Multiple users are online and are listed in the online user's list box in screen sharing page</li> </ul>
Input test data	-
Steps to be Executed	<ul style="list-style-type: none"> <li>a. Do not select any user from the online user list</li> <li>b. Click on send invites button</li> </ul>
Expected Results	The invitation should fail. A modal view should appear with the reason of invitation failure.
Actual Results	Invitation failed and a modal view appeared saying at least one-member must be selected for invitation.
Pass/Fail	<b>Pass</b>

#### 6.4.9.18 Invitation Modal View Interface - Screen Sharing

<b>Test Case ID</b>	<b>66</b>
<b>Test Case Type</b>	<b>GUI</b>
Test Case Description	Verify look and feel of invitation modal view



Pre-conditions	a. User has received an invitation to go to the screen sharing room
Input test data	-
Steps to be Executed	Observe the invitation modal view on the screen
Expected Results	<ul style="list-style-type: none"> <li>a. Reason for the appearance of modal view should be mentioned</li> <li>b. Go to room button should be visible to take directly to the room</li> <li>c. A close button should be present to cancel the invitation</li> </ul>
Actual Results	Modal view appeared with title “You have been invited to a screen sharing room by users”. There was a button in the middle of the modal view with the title “Go to room” and a close button was present on the right bottom section of the view.
Pass/Fail	<b>Pass</b>

#### 6.4.9.19 Invitation Modal View Validation

<b>Test Case ID</b>	<b>67</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Verify if a user receives the invitation modal view
Pre-conditions	<ul style="list-style-type: none"> <li>a. User is logged in to the system.</li> <li>b. <u>A</u> screen sharing organizer is ready to send the invitation to the user</li> </ul>
Input test data	-
Steps to be Executed	<ul style="list-style-type: none"> <li>a. A screen sharing organizer selects the user to send invitation to the room</li> <li>b. Organizer clicks on send invite button</li> </ul>

Expected Results	An invitation modal view should appear on the screen as soon as invitation is sent by the organizer
Actual Results	Modal view appeared on the screen when organizer sends the invitation to the room.
Pass/Fail	<b>Pass</b>

#### 6.4.9.20 Go to Room Button in Invitation Modal View

<b>Test Case ID</b>	<b>68</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Verify if clicking on ‘Go to Room’ button takes the user to the shared screen room
Pre-conditions	a. User has received an invitation to go to the screen sharing room
Input test data	-
Steps to be Executed	Click on “Go to Room” button in the invitation modal view on the screen
Expected Results	The user should be redirected to the screen sharing room in a new tab.
Actual Results	A new tab was opened with unique URL to the screen sharing room and live screen sharing started.
Pass/Fail	<b>Pass</b>

#### Negative case

<b>Test Case ID</b>	<b>69</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Verify if clicking on ‘Go to Room’ takes to shared screen room when the actual screen sharing session has been ended by organizer

Pre-conditions	<ul style="list-style-type: none"> <li>a. User has received an invitation to go to the screen sharing room</li> <li>b. Organizer has ended the screen sharing session</li> </ul>
Input test data	-
Steps to be Executed	Click on “Go to Room” button in the invitation modal view on the screen
Expected Results	The user should not able to view any kind of screen sharing session.
Actual Results	A new tab was opened with unique URL to the screen sharing room but no live screen sharing was going on. The page was blank with only unique id to the room.
Pass/Fail	<b>Pass</b>

#### 6.4.9.21 Close Button - Invitation Modal View

<b>Test Case ID</b>	<b>70</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Verify if clicking on cancel button cancels the invitation
Pre-conditions	<ul style="list-style-type: none"> <li>a. User has received an invitation to go to the screen sharing room</li> </ul>
Input test data	-
Steps to be Executed	Click on “Cancel” button in the invitation modal view on the screen
Expected Results	Invitation modal view should disappear with no actions being taken.
Actual Results	Invitation modal view disappeared. Nothing happened.
Pass/Fail	Pass


### 6.4.9.22 Verify Authenticated Screen Sharing Page URL After Logging Out

<b>Test Case ID</b>	<b>71</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Verify if the user can go back to the authenticated screen sharing page after logging out.
Pre-conditions	-
Input test data	-
Steps to be Executed	<ol style="list-style-type: none"> <li>a. Log in to the system.</li> <li>b. Navigate to Screen sharing Page.</li> <li>c. Copy the URL of the page.</li> <li>d. Log out.</li> <li>e. Paste the copied URL into the browser.</li> </ol>
Expected Results	The URL should not be redirected to the authenticated screen sharing page content. It should redirect to the log in page.
Actual Results	The user was redirected to the login page.
Pass/Fail	<b>Pass</b>

### 6.4.10 Chapters Page - Student

#### 6.4.10.1 Interface

<b>Test Case ID</b>	<b>72</b>
<b>Test Case Type</b>	<b>GUI</b>
Test Case Description	Test the look and feel of the Chapters page
Pre-conditions	User is logged in and is in Chapters page of the tool
Input test data	-
Steps to be Executed	-
Expected Results	The user should be able to see following fields:

	<ul style="list-style-type: none"> <li>a. A search text field</li> <li>b. A button </li> <li>c. List of chapters with the title of each chapter</li> </ul>
Actual Results	All the fields were present as expected.
Pass/Fail	<b>Pass</b>

#### 6.4.10.2 Main Menu Bar

Refer to test 23 in section 6.4.5.2. Same test case applies.

#### 6.4.10.3 Main Menu Test

Refer to test 24 in section 6.4.5.3. Same test case applies.



#### 6.4.10.4 Search

<b>Test Case ID</b>	<b>73</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Verify if user can use the search function
Pre-conditions	User is logged in and is in Chapters page of the tool
Input test data	Text input – “Expanding”
Steps to be Executed	Type “Expanding” in the search field
Expected Results	The user should be able to see the search result as soon as the user starts typing.
Actual Results	The searching result was shown dynamically as soon as user started typing in the search field.
Pass/Fail	<b>Pass</b>

<b>Test Case ID</b>	<b>74</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Verify if search works when search field is left empty

Pre-conditions	User is logged in and is in Chapters page of the tool
Input test data	-
Steps to be Executed	Click on search text field but leave it empty
Expected Results	The user should not see any change on the screen.
Actual Results	No change was observed on the screen.
Pass/Fail	<b>Pass</b>

#### 6.4.10.5 Search Clear Button

<b>Test Case ID</b>	<b>75</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Verify if user can clear and restart the search function by clicking on search clear button 
Pre-conditions	User is logged in and is in Chapters page of the tool
Input test data	Text input – “Expanding”
Steps to be Executed	a. Type “Expanding” in the search field b. Click on search clear button 
Expected Results	Search text field should be clear and searching process should restart.
Actual Results	The search text field was cleared as soon as the button was clicked and searching restarted.
Pass/Fail	<b>Pass</b>

#### 6.4.10.6 Collapsible Titles and Subtitles

<b>Test Case ID</b>	<b>76</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Verify if titles and subtitles of each chapter are collapsible

Pre-conditions	User is logged in and is in Chapters page of the tool
Input test data	-
Steps to be Executed	<ol style="list-style-type: none"> <li>a. Click on each title to expand and click again to collapse</li> <li>b. Click on each subtitle to expand and click again to collapse</li> </ol>
Expected Results	Each title and subtitle should be collapsible such that it reveals and hides the content when needed.
Actual Results	Each of the titles and subtitles was observed to be collapsible.
Pass/Fail	<b>Pass</b>

#### 6.4.10.7 Verify Authenticated Chapters Page URL After Logging Out

<b>Test Case ID</b>	<b>77</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Verify if the user can go back to authenticated Chapters page after logging out.
Pre-conditions	-
Input test data	-
Steps to be Executed	<ol style="list-style-type: none"> <li>a. Log in to the system.</li> <li>b. Navigate to Chapters Page.</li> <li>c. Copy the URL of the page.</li> <li>d. Log out.</li> <li>e. Paste the copied URL into the browser.</li> </ol>
Expected Results	The URL should not be redirected to the authenticated chapters page. It should redirect to the log in page.
Actual Results	The user was redirected to the login page.
Pass/Fail	<b>Pass</b>

## 6.4.11 Lab Assistant Hours Page - Student

### 6.4.11.1 Interface

<b>Test Case ID</b>	<b>78</b>
<b>Test Case Type</b>	<b>GUI</b>
Test Case Description	Verify the look and feel of the Lab Assistant Hours page
Pre-conditions	User is logged in and is in Lab Assistant Hours page of the tool
Input test data	-
Steps to be Executed	-
Expected Results	The user should be able to see a schedule table with tutor/lab assistant availability.
Actual Results	A schedule table with tutor/lab assistant availability was shown.
Pass/Fail	<b>Pass</b>

### 6.4.11.2 Main Menu Bar

Refer to test 23 in section 6.4.5.2. Same test case applies.

### 6.4.11.3 Main Menu Test

Refer to test 24 in section 6.4.5.3. Same test case applies.

### 6.4.11.4 Verify Authenticated Lab Assistant Hours Page URL After Logging Out


<b>Test Case ID</b>	<b>79</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Verify if the user can go back to authenticated Lab Assistant Hours page after logging out.



Pre-conditions	-
Input test data	-
Steps to be Executed	<ol style="list-style-type: none"> <li>a. Log in to the system.</li> <li>b. Navigate to Lab Assistant Hours Page.</li> <li>c. Copy the URL of the page.</li> <li>d. Log out.</li> <li>e. Paste the copied URL into the browser.</li> </ol>
Expected Results	The URL should not be redirected to the authenticated Lab Assistant Hours page content. It should redirect to the log in page.
Actual Results	The user was redirected to the login page.
Pass/Fail	<b>Pass</b>

## 6.4.12 FAQ Page - Student

### 6.4.12.1 Interface

<b>Test Case ID</b>	<b>80</b>
<b>Test Case Type</b>	<b>GUI</b>
Test Case Description	Verify the look and feel of the FAQ page
Pre-conditions	User is logged in and is in FAQ page of the tool
Input test data	-
Steps to be Executed	-
Expected Results	<p>The user should be able to see the following fields:</p> <ol style="list-style-type: none"> <li>a. Title of the page</li> <li>b. A search text field and a button </li> <li>c. List of hyperlinked questions</li> <li>d. Rest of the content with the FAQs related to C++</li> <li>e. At the end of each FAQ, a hyperlinked text “Go to top” should be present</li> </ol>

Actual Results	All the expected fields were seen.
Pass/Fail	<b>Pass</b>

#### 6.4.12.2 Main Menu Bar

Refer to test 23 in section 6.4.5.2. Same test case applies.

#### 6.4.12.3 Main Menu Test

Refer to test 24 in section 6.4.5.3. Same test case applies.

#### 6.4.12.4 Search

Refer to test 73 in section 6.4.10.4. Same test cases apply.

#### 6.4.12.5 Search Clear Button

Refer to test 75 in section 6.4.10.5. Same test case applies.

#### 6.4.12.6 Hyperlinked Question Titles

<b>Test Case ID</b>	<b>81</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Verify if hyperlinked question titles redirect to respective content in the page
Pre-conditions	User is logged in and is in FAQ page of the tool
Input test data	-
Steps to be Executed	Click on each question titles
Expected Results	Each hyperlinked title should redirect to respective question and answer content in the page.
Actual Results	The hyperlinked titles routed to the respective section in the content page as expected.
Pass/Fail	<b>Pass</b>

#### 6.4.12.7 “Go to Top” Hyperlinked Text

<b>Test Case ID</b>	<b>82</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Verify if “Go to top” hyperlinked text redirect user to top of the page
Pre-conditions	User is logged in and is in FAQ page of the tool
Input test data	-
Steps to be Executed	Click on each “Go to top” hyperlinked text at the end of each FAQs
Expected Results	Each “Go to top” hyperlinked text at the end of every FAQs should redirect the user to the top of the page
Actual Results	Each hyperlinked text redirected user to the top of the page.
Pass/Fail	<b>Pass</b>

#### 6.4.12.8 Verify Authenticated FAQ Page URL After Logging Out

<b>Test Case ID</b>	<b>83</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Verify if the user can go back to authenticated FAQ page after logging out.
Pre-conditions	-
Input test data	-
Steps to be Executed	<ol style="list-style-type: none"> <li>a. Log in to the system.</li> <li>b. Navigate to FAQ Page.</li> <li>c. Copy the URL of the page.</li> <li>d. Log out.</li> <li>e. Paste the copied URL into the browser.</li> </ol>

Expected Results	The URL should not be redirected to the authenticated FAQ page content. It should redirect to the log in page.
Actual Results	The user was redirected to the login page.
Pass/Fail	<b>Pass</b>

### 6.4.13 Contact Page - Student

Refer to following test cases since same test cases apply:

- a. Test case 15 in section 6.4.4.1
- b. Test case 23 in section 6.4.6.2
- c. Test case 24 in section 6.4.6.3
- d. Test case 16-20 in section 6.4.4.3

#### 6.4.13.1 Verify Authenticated Contact Page URL After Logging Out

<b>Test Case ID</b>	<b>84</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Verify if the user can go back to authenticated Contact page after logging out.
Pre-conditions	-
Input test data	-
Steps to be Executed	<ol style="list-style-type: none"> <li>a. Log in to the system.</li> <li>b. Navigate to Contact Page.</li> <li>c. Copy the URL of the page.</li> <li>d. Log out.</li> <li>e. Paste the copied URL into the browser.</li> </ol>
Expected Results	The URL should not redirect to the authenticated Contact page and should redirect to the login page.
Actual Results	The user was redirected to the login page.
Pass/Fail	<b>Pass</b>

Following test cases are applicable when logged in as a Teacher

#### 6.4.14 Home Page - Teacher

##### 6.4.14.1 Interface

Refer to test 4 in section 6.4.2.3. Same test case applies.

##### 6.4.14.2 Main Menu Bar

<b>Test Case ID</b>	<b>85</b>
<b>Test Case Type</b>	<b>GUI</b>
Test Case Description	Verify whether the full menu bar is available on top of the home page for <b>teacher</b> as per requirement specifications for teacher user role
Pre-conditions	User is in the authenticated home page of the tool
Input test data	-
Steps to be Executed	Check the menu bar items list
Expected Results	A dedicated list of the items should be present in the main menu bar as listed below: Home, Ask a Question, Ask a Tutor (online chat, share screen), Study Materials (Chapters, Lab Assistant hours, FAQ), Manage, logout, and Messages.
Actual Results	As expected all main menu items were present in the main menu bar.
Pass/Fail	<b>Pass</b>

##### 6.4.14.3 Main Menu Test

<b>Test Case ID</b>	<b>86</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Validate all the menus in the main menu bar

Pre-conditions	User is in the logged in home page of the tool
Input test data	-
Steps to be Executed	Click on each menu item
Expected Results	a. Each item should redirect to the respective page b. Log out should log out user out of the system.
Actual Results	The menu items worked as expected.
Pass/Fail	<b>Pass</b>

#### **6.4.14.4 Verify Logged in URL After Log Out**

Refer to test 25 in section 6.4.6.4. Same test case applies.

#### **6.4.15 Ask a Question Page - Teacher**

Refer to following test cases since same test cases apply:

- a. Test case 26 in section 6.4.7.1
- b. Test case 85 in section 6.4.14.2
- c. Test case 86 in section 6.4.14.3
- d. Test case 27 in section 6.4.7.4 - Test case 32 in section 6.4.7.6

#### **6.4.16 Online Chat Page - Teacher**

Refer to following test cases since same test cases apply:

- a. Test case 33 in section 6.4.8.1
- b. Test case 85 in section 6.4.14.2
- c. Test case 86 in section 6.4.14.3
- d. Test case 34 in section 6.4.8.4 - Test case 44 in section 6.4.8.9

#### **6.4.17 Screen Sharing Page - Teacher**

Refer to following test cases since same test cases apply:

- a. Test case 45 in section 6.4.9.1
- b. Test case 85 in section 6.4.14.2
- c. Test case 86 in section 6.4.14.3

- d. Test case 46 in section 6.4.9.4 - Test case 71 in section 6.4.9.22

#### 6.4.18 Chapters Page - Teacher

Refer to following test cases since same test cases apply:

- a. Test case 72 in section 6.4.10.1
- b. Test case 85 in section 6.4.14.2
- c. Test case 86 in section 6.4.14.3
- d. Test case 73 in section 6.4.10.4 - Test case 77 in section 6.4.10.7

#### 6.4.19 Lab Assistant Hours Page - Teacher

Refer to following test cases since same test cases apply:

- a. Test case 85 in section 6.4.14.2
- b. Test case 86 in section 6.4.14.3
- c. Test case 79 in section 6.4.11.4

##### 6.4.19.1 Interface

<b>Test Case ID</b>	<b>87</b>
<b>Test Case Type</b>	<b>GUI</b>
Test Case Description	Verify the look and feel of the Lab Assistant Hours page
Pre-conditions	User is logged in and is in Lab Assistant Hours page of the tool
Input test data	-
Steps to be Executed	-
Expected Results	<ul style="list-style-type: none"> <li>a. The user should be able to see a schedule table with tutor/lab assistant availability.</li> <li>b. User should be able to see two buttons on top of the table and one of them should be disabled</li> </ul>

Actual Results	A schedule table with tutor/lab assistant availability was shown with two buttons on the top of the page. 'Save changes' button was disabled.
Pass/Fail	<b>Pass</b>

#### 6.4.19.2 Update Table

<b>Test Case ID</b>	<b>88</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Verify if the user can update the table
Pre-conditions	User is logged in and is in Lab Assistant Hours page of the tool
Input test data	Text input – “Santosh”
Steps to be Executed	<ol style="list-style-type: none"> <li>a. Click on edit/update button</li> <li>b. Enter the input “Santosh” somewhere in the table</li> <li>c. Delete some names from the table</li> <li>d. Click on save changes button</li> </ol>
Expected Results	<ol style="list-style-type: none"> <li>a. User should be able to see the updated table with the name “Santosh” added to the table whereas other names which have been deleted should be removed</li> <li>b. Save changes button should be disabled after making changes</li> </ol>
Actual Results	The table was updated as expected and save changes button was again disabled.
Pass/Fail	<b>Pass</b>

#### 6.4.20 FAQ Page - Teacher

Refer to following test cases since same test cases apply:



- a. Test case 80 in section 6.4.12.1
- b. Test case 85 in section 6.4.14.2
- c. Test case 86 in section 6.4.14.3
- d. Test case 73 in section 6.4.10.4
- e. Test case 75 in section 6.4.10.5
- f. Test case 81 in section 6.4.12.6 - Test case 83 in section 6.4.12.8

### 6.4.21 Manage Page - Teacher

#### 6.4.21.1 Interface

<b>Test Case ID</b>	<b>89</b>
<b>Test Case Type</b>	<b>GUI</b>
Test Case Description	Verify the look and feel of the Manage page
Pre-conditions	User is logged in and is in Manage page of the tool
Input test data	-
Steps to be Executed	-
Expected Results	<ul style="list-style-type: none"> <li>a. User should be able to see a table full of rows with all the member details</li> <li>b. Each row should have two buttons with the title “Edit” and “Delete”</li> <li>c. At the bottom of the page, user should be able to another button with the title label “Add a new member”</li> </ul>
Actual Results	A table filled with member’s information was shown on the page with each row having two buttons with titles edit and delete. At the bottom of the page, a button for adding a new member was present.
Pass/Fail	<b>Pass</b>

#### 6.4.21.2 Main Menu Bar

Refer to test case 85 in section 6.4.14.2. Same test case applies.

#### 6.4.21.3 Main Menu Test

Refer to test case 86 in section 6.4.14.3. Same test case applies.

#### 6.4.21.4 Edit Member Form Interface

<b>Test Case ID</b>	<b>90</b>
<b>Test Case Type</b>	<b>GUI</b>
Test Case Description	Verify if the edit member form is displayed with appropriate form
Pre-conditions	User is logged in and is in Manage page of the tool
Input test data	-
Steps to be Executed	a. Choose one of the rows with the member's details and click on Edit button which is at the end of the row
Expected Results	A form should appear with following fields: a. First name b. Last name c. User name d. Password e. User role f. Confirm and cancel button
Actual Results	'Edit form' with required fields was displayed.
Pass/Fail	<b>Pass</b>

#### 6.4.21.5 Edit a Member

<b>Test Case ID</b>	<b>91</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Verify if user can edit the details of selected member by clicking on confirm button after updating details
Pre-conditions	<ul style="list-style-type: none"> <li>a. User is logged in and is in Manage page of the tool</li> <li>b. User has chosen a member to edit</li> </ul>
Input test data	Text input for first and last name
Steps to be Executed	<ul style="list-style-type: none"> <li>a. Click on edit button at the end of the chosen member row</li> <li>b. Update the details in the edit form</li> <li>c. Click confirm button</li> </ul>
Expected Results	The member details with updated first and last name should be displayed in the table
Actual Results	The table was updated with selected member's new details.
Pass/Fail	<b>Pass</b>

#### Negative Case

<b>Test Case ID</b>	<b>92</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Verify if user can edit the details of selected member with empty first name
Pre-conditions	<ul style="list-style-type: none"> <li>a. User is logged in and is in Manage page of the tool</li> <li>b. User has chosen a member to edit</li> </ul>
Input test data	-
Steps to be Executed	<ul style="list-style-type: none"> <li>a. Click on edit button at the end of the chosen member row</li> </ul>

	<ul style="list-style-type: none"> <li>b. Delete the first name in the edit form</li> <li>c. Click confirm button</li> </ul>
Expected Results	The user should not be able to update the member details with the empty first name. The Proper error must be shown.
Actual Results	Proper error “Please fill out this field” was shown and the user was not able to update the member details.
Pass/Fail	<b>Pass</b>

<b>Test Case ID</b>	<b>93</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Verify if user can edit the details of selected member with empty last name
Pre-conditions	<ul style="list-style-type: none"> <li>a. User is logged in and is in Manage page of the tool</li> <li>b. User has chosen a member to edit</li> </ul>
Input test data	-
Steps to be Executed	<ul style="list-style-type: none"> <li>a. Click on edit button at the end of the chosen row</li> <li>b. Delete the last name in the edit form</li> <li>c. Click confirm button</li> </ul>
Expected Results	The user should not be able to update the member details with empty last name. The Proper error must be shown.
Actual Results	Proper error “Please fill out this field” was shown and the user was not able to update the member details.
Pass/Fail	<b>Pass</b>
<b>Test Case ID</b>	<b>94</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Verify if user can edit the details of selected member with empty user name

Pre-conditions	<ul style="list-style-type: none"> <li>a. User is logged in and is in Manage page of the tool</li> <li>b. User has chosen a member to edit</li> </ul>
Input test data	-
Steps to be Executed	<ul style="list-style-type: none"> <li>a. Click on edit button at the end of the chosen member row</li> <li>b. Delete the user name in the edit form</li> <li>c. Click confirm button</li> </ul>
Expected Results	The user should not be able to update the member details with the empty user name. The Proper error must be shown.
Actual Results	Proper error “Please fill out this field” was shown and the user was not able to update the member details.
Pass/Fail	<b>Pass</b>

<b>Test Case ID</b>	<b>95</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Verify if user can edit the details of selected member with empty password
Pre-conditions	<ul style="list-style-type: none"> <li>a. User is logged in and is in Manage page of the tool</li> <li>b. User has chosen a member to edit</li> </ul>
Input test data	-
Steps to be Executed	<ul style="list-style-type: none"> <li>a. Click on edit button at the end of the chosen member row</li> <li>b. Delete the password in the edit form</li> <li>c. Click confirm button</li> </ul>
Expected Results	The user should not be able to update the member details with an empty password. The Proper error must be shown.
Actual Results	Proper error “Please fill out this field” was shown and

	the user was not able to update the member details.
Pass/Fail	<b>Pass</b>

<b>Test Case ID</b>	<b>96</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Verify if user can edit the details of selected member with empty user role
Pre-conditions	<ul style="list-style-type: none"> <li>a. User is logged in and is in Manage page of the tool</li> <li>b. User has chosen a member to edit</li> </ul>
Input test data	-
Steps to be Executed	<ul style="list-style-type: none"> <li>a. Click on edit button at the end of the chosen member row</li> <li>b. Delete the user role in the edit form</li> <li>c. Click confirm button</li> </ul>
Expected Results	The user should not be able to update the member details with an empty user role. The Proper error must be shown.
Actual Results	Proper error “Please fill out this field” was shown and the user was not able to update the member details.
Pass/Fail	<b>Pass</b>

<b>Test Case ID</b>	<b>97</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Verify if user can edit the details of selected member if cancel button is clicked in the edit form
Pre-conditions	<ul style="list-style-type: none"> <li>a. User is logged in and is in Manage page of the tool</li> <li>b. User has chosen a member to edit</li> </ul>
Input test data	-
Steps to be Executed	<ul style="list-style-type: none"> <li>a. Click on edit button at the end of the chosen</li> </ul>

	<p>member row</p> <p>b. Update the details in the edit form</p> <p>c. Click cancel button</p>
Expected Results	The member details should not be changed. The table should remain same.
Actual Results	The table was not updated.
Pass/Fail	<b>Pass</b>

#### 6.4.21.6 Add a Member Form Interface

<b>Test Case ID</b>	<b>98</b>
<b>Test Case Type</b>	<b>GUI</b>
Test Case Description	Verify if the 'add a member form' is displayed with appropriate form
Pre-conditions	User is logged in and is in Manage page of the tool
Input test data	-
Steps to be Executed	Click on Add a new member button at the bottom of the page
Expected Results	<p>A form should appear with following fields:</p> <p>a. First name</p> <p>b. Last name</p> <p>c. User role</p> <p>d. Confirm and cancel button</p>
Actual Results	'Add a member form' with required fields was displayed.
Pass/Fail	<b>Pass</b>

#### 6.4.21.7 Add a Member

<b>Test Case ID</b>	<b>99</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Verify if user can add a new member by clicking on confirm button after filling up the form
Pre-conditions	a. User is logged in and is in Manage page of the tool
Input test data	Text input for first and last name
Steps to be Executed	<ul style="list-style-type: none"> <li>a. Click on add a new member button at the bottom of the page</li> <li>b. Fill in the first and last name field with the input</li> <li>c. Choose user role type from the drop down</li> <li>d. Click confirm button</li> </ul>
Expected Results	The new member with given first name, last name and user role should be added and displayed in the table.
Actual Results	The table was updated with the new member' details.
Pass/Fail	<b>Pass</b>

#### Negative Case

<b>Test Case ID</b>	<b>100</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Verify if user can add a new member with empty first name
Pre-conditions	a. User is logged in and is in Manage page of the tool
Input test data	Text input for the last name
Steps to be Executed	<ul style="list-style-type: none"> <li>a. Click on add a new member button at the bottom of the page</li> <li>b. Fill in the last name field and leave the first name field blank</li> </ul>



	<ul style="list-style-type: none"> <li>c. Choose user role type from the drop down</li> <li>d. Click confirm button</li> </ul>
Expected Results	The user should not be able to add a new member with the empty first name. The Proper error must be shown.
Actual Results	Proper error “Please fill out this field” was shown and the user was not able to add the new member.
Pass/Fail	<b>Pass</b>

<b>Test Case ID</b>	<b>101</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Verify if user can add a new member with empty last name
Pre-conditions	a. User is logged in and is in Manage page of the tool
Input test data	Text input for the first name
Steps to be Executed	<ul style="list-style-type: none"> <li>a. Click on add a new member button at the bottom of the page</li> <li>b. Fill in the first name field and leave the last name field blank</li> <li>c. Choose user role type from the drop down</li> <li>d. Click confirm button</li> </ul>
Expected Results	The user should not be able to add a new member with empty last name. The Proper error must be shown.
Actual Results	Proper error “Please fill out this field” was shown and the user was not able to add the new member.
Pass/Fail	<b>Pass</b>

<b>Test Case ID</b>	<b>102</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Verify if user can add a new member with empty user

	role
Pre-conditions	a. User is logged in and is in Manage page of the tool
Input test data	Text input for first and last name
Steps to be Executed	<ul style="list-style-type: none"> <li>a. Click on add a new member button at the bottom of the page</li> <li>b. Fill in the first and the last name field</li> <li>c. Leave user role type field blank</li> <li>d. Click confirm button</li> </ul>
Expected Results	The user should not be able to add a new member with an empty user role. The Proper error must be shown.
Actual Results	Proper error “Please fill out this field” was shown and the user was not able to add the new member.
Pass/Fail	<b>Pass</b>

<b>Test Case ID</b>	<b>103</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Verify if user can add a new member if cancel button is clicked in the add a new member form
Pre-conditions	a. User is logged in and is in Manage page of the tool
Input test data	Text input for first and last name
Steps to be Executed	<ul style="list-style-type: none"> <li>a. Click on add a new member button at the bottom of the page</li> <li>b. Fill in the first and last name field with the input</li> <li>c. Choose user role type from the drop down</li> <li>d. Click confirm button</li> </ul>
Expected Results	The new member details should not be added to the table. The table should remain same.
Actual Results	The table was not updated.
Pass/Fail	<b>Pass</b>

#### 6.4.21.8 Delete Member Form Interface

<b>Test Case ID</b>	<b>104</b>
<b>Test Case Type</b>	<b>GUI</b>
Test Case Description	Verify if the delete member form is displayed with appropriate form
Pre-conditions	User is logged in and is in Manage page of the tool
Input test data	-
Steps to be Executed	a. Choose one of the rows with member's details and click on Delete button which is at the end of the row
Expected Results	A confirmation modal view (form) should appear that contains a Delete button to confirm the deletion. A cancel button (x) should appear at the top of the modal view.
Actual Results	Delete form with the required buttons was displayed.
Pass/Fail	<b>Pass</b>

#### 6.4.21.9 Delete a Member

<b>Test Case ID</b>	<b>105</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Verify if user can delete the selected member by clicking on Delete button in the modal view
Pre-conditions	a. User is logged in and is in Manage page of the tool b. User has chosen a member to delete
Input test data	-
Steps to be Executed	a. Click on delete button at the end of the chosen member row

	b. Click Delete button in the generated form
Expected Results	The selected member should be deleted and updated table should be displayed
Actual Results	The table was updated with the deletion of selected member.
Pass/Fail	<b>Pass</b>

### Negative Case

<b>Test Case ID</b>	<b>106</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Verify if user can delete the selected member if cancel button is clicked in the delete modal view (form)
Pre-conditions	a. User is logged in and is in Manage page of the tool
Input test data	-
Steps to be Executed	a. Click on delete button at the end of the chosen member row b. Click cancel (x) button in the generated form
Expected Results	The selected member should not be deleted from the table. The table should remain same.
Actual Results	The table was not updated.
Pass/Fail	<b>Pass</b>

### 6.4.21.10 Verify Authenticated Manage Page URL After Logging Out

<b>Test Case ID</b>	<b>107</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Verify if the user can go back to authenticated Manage page after logging out.
Pre-conditions	-

Input test data	-
Steps to be Executed	<ol style="list-style-type: none"> <li>a. Log in to the system.</li> <li>b. Navigate to Manage Page.</li> <li>c. Copy the URL of the page.</li> <li>d. Log out.</li> <li>e. Paste the copied URL into the browser.</li> </ol>
Expected Results	The URL should not redirect to the logged in Manage page content. It should redirect to the log in page.
Actual Results	The user was redirected to the login page.
Pass/Fail	<b>Pass</b>

#### 6.4.22 Message Page - Teacher

##### 6.4.22.1 Interface

<b>Test Case ID</b>	<b>108</b>
<b>Test Case Type</b>	<b>GUI</b>
Test Case Description	Verify the look and feel of the Message page
Pre-conditions	User is logged in and is in Message page of the tool
Input test data	-
Steps to be Executed	-
Expected Results	<ol style="list-style-type: none"> <li>a. User should be able to see a table full of rows with all the messages that exist in the system</li> <li>b. Each row should contain message id, first name, last name, email, messages and date message received</li> <li>c. Each row should have buttons with label “Delete”</li> </ol>
Actual Results	A table filled with messages was shown on the page with each row having a button to delete.
Pass/Fail	<b>Pass</b>

#### 6.4.22.2 Main Menu Bar

Refer to test case 85 in section 6.4.14.2. Same test case applies.

#### 6.4.22.3 Main Menu Test

Refer to test case 86 in section 6.4.14.3. Same test case applies.

#### 6.4.22.4 Delete Message Form Interface

<b>Test Case ID</b>	<b>109</b>
<b>Test Case Type</b>	<b>GUI</b>
Test Case Description	Verify if the delete member form is displayed with appropriate form
Pre-conditions	User is logged in and is in Message page of the tool
Input test data	-
Steps to be Executed	a. Choose one of the rows with a message and click on Delete button which is at the end of the row
Expected Results	A confirmation modal view (form) should appear that contains a Delete button to confirm the deletion. A cancel button (x) should appear at the top of the modal view.
Actual Results	Delete form with the required button was displayed.
Pass/Fail	<b>Pass</b>

#### 6.4.22.5 Delete a Message

<b>Test Case ID</b>	<b>110</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Verify if user can delete the selected message by clicking on Delete button in the modal view

Pre-conditions	a. User is logged in and is in Message page of the tool b. User has chosen a message to delete
Input test data	-
Steps to be Executed	a. Click on delete button at the end of the chosen message row b. Click Delete button in the generated form
Expected Results	The selected message should be deleted and updated table should be displayed
Actual Results	The table was updated with the deletion of selected message.
Pass/Fail	<b>Pass</b>

#### Negative Case

<b>Test Case ID</b>	<b>111</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Verify if user can delete the selected message if cancel button is clicked in the delete modal view (form)
Pre-conditions	a. User is logged in and is in Message page of the tool
Input test data	-
Steps to be Executed	c. Click on delete button at the end of the chosen message row d. Click cancel (x) button in the generated form
Expected Results	The selected message should not be deleted from the table. The table should remain same.
Actual Results	The table was not updated.
Pass/Fail	<b>Pass</b>

#### 6.4.22.6 Verify Authenticated Message Page URL After Logging Out

<b>Test Case ID</b>	<b>112</b>
<b>Test Case Type</b>	<b>Functional</b>
Test Case Description	Verify if the user can go back to authenticated Message page after logging out.
Pre-conditions	-
Input test data	-
Steps to be Executed	<ol style="list-style-type: none"> <li>a. Log in to the system.</li> <li>b. Navigate to Message Page.</li> <li>c. Copy the URL of the page.</li> <li>d. Log out.</li> <li>e. Paste the copied URL into the browser.</li> </ol>
Expected Results	The URL should not be redirected to the logged in Message page content. It should redirect to the log in page.
Actual Results	The user was redirected to the login page.
Pass/Fail	<b>Pass</b>



## Chapter 7: DEPLOYMENT

### 7.1 Chapter Overview

This chapter outlines the settings for the deployment of the system.

#### 7.1.1 Deployment Settings

Following is the system setup for the deployment of the system

**Domain name:** vtstcloudstate.com

**Protocol type:** HTTPS (secure)

**Linux** - Linux 14.04.4 - Ubuntu SMP

**Apache server:** Apache/2.4.7 (Ubuntu)

**PHP** 5.5.9-1, ubuntu 4.17

**MySQL** Version 14.14 Distribution 5.5.49

## Chapter 8: EVALUATION

### 8.1 Chapter Overview

This chapter outlines the evaluation of the system after the deployment. The evaluation is fully described in four main categories of accomplishments, limitation, future enhancements and maintenance followed by a conclusion section.

### 8.2 Accomplishments

A system is built that can support the current lab assistants and teachers in providing extra guidance to the students in CSCI 201 (C++). Now both students and teachers can benefit from this system in providing virtual help. With the deployment of the tool, all the major functionalities listed in section 1.5 are accomplished. A speech recognition system to find out the closest answer is built with the helpful links and videos. A screen sharing system to have the online screen sharing conference is built. An online chat section is built where a student can share the common questions with each other about C++. Anyone can volunteer to help each other. Also, available teachers/assistants can step in to help students in need. A contact section is built to help students send the queries to all teachers/assistant at once.

In addition, the management system is built to handle adding and deleting members and messages which are some of the very important aspects of managing the system. Finally, login and logout system is built to successfully allow only dedicated users to use the system.

### 8.3 Limitations

Most of the functions listed in section 1.5 are met. However, there are limitations on some of the features.

1. In online chat communication, there is only one main room available. The user cannot open a private chat room yet.
2. While screen sharing, users (audiences) can NOT hear/talk to each other. It is one-to-many kind of topology; it is NOT a mesh or many-to-many.

3. File sharing is not supported yet.
4. Ask a Tutor section works best when the answer is available in the database. Currently, there is limited information in the database which is needed to be updated with more information.
5. Currently, to use all the features offered by the system, it is recommended to use either Google chrome or Firefox. Other browsers are not yet fully supported.
6. The best experience can be achieved if the system is used in windows or mac system. It is responsive when used in smaller devices, but full functionality may not be supported.

#### **8.4 Future Enhancements**

Although the goals of the project were met, there are still many cases where enhancements can be made. All the limitations mentioned above can be considered in future enhancements.

1. First, the tool can be enhanced such that file sharing is possible, which could be very helpful for students.
2. In online chat section, a feature can be added such that user can open a separate room for online chat communication.
3. In screen sharing section, the feature can be updated such that all the audiences can also hear and talk to each other.
4. The tool is not supported by some of the available browsers. Although, most of the functions are supported by all of the browsers, organizing screen sharing may not be possible. However, becoming an audience is possible in any kinds of browsers.

5. Similarly, the tool cannot be fully utilized when using in small devices like androids and iOS devices. Even though the tool is designed to be responsive to the user's behavior and environment based on screen sizes and platform, some of the functions are still not supported. This can be considered a high priority on the list of future enhancements since the popularity of small devices is getting higher and students tend to use small devices a lot more in current days.
6. A 'Send Reply' button can be added in the 'Message' page for teacher/lab assistant such that it is easier to reply to the sender from the page itself.
7. Currently, the user is not able to change the default password. Only teacher/lab assistant can change the password. This function can be added so that user can have the password as they wish.
8. Currently, a free signaling server is used for screen sharing. In future, a dedicated server can be created to eliminate the use of free signaling server.

The above listed functions and features can be added to provide more functionality to the user. Some of them may be necessary while some of them may not be required at all. Depending on time and resources, it can be decided which features/functions are to be integrated.

### **8.5 Maintenance**

The system is all set up and ready to provide the service. However, at least one dedicated admin is needed to add all the necessary contents required to go live with users/students. Once the tool is up and running, maintenance of the tool is very necessary for either fixing faults or extending the functionality of the website (enhancements). Even though developing the tool is regarded as a major accomplishment, maintenance is however equally important to smoothly run the tool for the long term. In regard to this project, at least an admin is necessary to frequently monitor the tool. Since the tool is controllable by more than one member (teachers and lab assistants), it may be easy to update the necessary content in the

tool (especially lab assistant hours' table). However, to work on enhancements, a dedicated member is required.

## 8.6 Conclusion

The system was successfully implemented meeting all the specified requirements. A complete system offering different kinds of features was developed. The initial scope of the project was to develop a system to help answering student queries about C++. However, more features were added later, which could significantly help students in learning more. Online chat and screen sharing sections are those features that were added later.

All the major functionalities that were proposed to add are listed in section 1.5. By the end of the deployment, all those listed features and functionalities are integrated into the tool. During the time of building up the tool, several steps were carried out. The system investigations, technical investigations, relevant background analysis were carried out for major features of the tool that includes speech recognitions, online communication, screen sharing and more. The investigations and background analysis helped a lot in determining the major requirements of the systems, the possibility of integrating similar systems, and also helped in defining the necessary approach in making up the tool. All the functional and non-functional requirements of the project were determined along with resource requirements.

Once the investigations and background analysis were completed, system design was performed following Model-View-Controller architecture. To help the controllers control different features, services were introduced that could communicate with the server to perform the necessary tasks on the server. UML modeling techniques were used to document the design using user case diagrams. The use case diagrams were then used to perform interface design, extracting all the necessary controllers and services that were needed to complete the project. At the end, use case realizations were done for each of the use cases with extended and refined scenarios along with the help of detailed flow chart diagrams. The development of the detailed use case realizations was very helpful in developing the codes and eventually in code implementations. HTML, CSS, PHP and JavaScript along with

Angular JS, Bootstrap framework were used in developing the overall system. Different libraries and API were used in developing major features of the system such as Speech API, WebRTC-library, and jQuery.

Finally, it was possible to develop a tool that could potentially benefit both students and teachers to help struggling students in C++. The tool enables the user to perform all the actions that were described in the project scope successfully.

## REFERENCES

- [1]. Treehouse Island, Inc. 2016. *About Treehouse*. [Online]  
Available at: <https://teamtreehouse.com/>
- [2]. Udemy. 2016. *Learning about Udemy* [Online]  
Available at: <https://about.udemy.com/>
- [3]. Stockley D.; Rossner V., " The Virtual-TA: Moving Beyond the Traditional Teaching Assistant," in *N.A.WEB 96 - The Second International North America World Wide Web Conference*  
Available at: <http://www.uvm.edu/~hag/naweb96/zstockley.html>
- [4]. Google Inc. n.d. *Web Speech API Specification* [Online]  
Available at: <https://dvcs.w3.org/hg/speech-api/raw-file/tip/speechapi.html>
- [5]. Google Inc. n.d. *Web Speech API Demonstration* [Online]  
Available at: <https://www.google.com/intl/en/chrome/demos/speech.html>
- [6]. Microsoft Cognitive Services. n.d. *Microsoft Speech Recognition Documentation* [Online]  
Available at: <https://www.microsoft.com/cognitive-services/en-us/speech-api/documentation/overview>
- [7]. Microsoft Cognitive Services. n.d. *Microsoft Speech recognition demo* [Online]  
Available at: <https://www.microsoft.com/cognitive-services/en-us/speech-api>
- [8]. Show My PC. 2016. n.d. *About Screen Sharing API* [Online]  
Available at: <https://showmypc.com/faq/screen-sharing-api.html>

- [9]. Screen Leap API. 2016. *Introduction to Screen Leap* [Online]  
Available at: <http://www.screenleap.com/api>
- [10]. Wikipedia. 2016. *Google-Hangouts - Wikipedia*, the free encyclopedia [online]  
Available at: [https://en.wikipedia.org/wiki/Google\\_Hangouts](https://en.wikipedia.org/wiki/Google_Hangouts)
- [11]. HipChat. 2016. *About HipChat* [online]  
Available at: <https://www.hipchat.com/>
- [12]. Anon. n.d. *Dragon - Continuous vs. Discrete Speech Recognition* [online]  
Available at: <http://www.synapseadaptive.com/naturallyspeaking/define.html>
- [13]. George Orno. 2014. *The HTML 5 Speech Recognition API* [online]  
Available at: <https://shapedshed.com/html5-speech-recognition-api/>
- [14]. Sam Dutton. 2014. *Getting Started with WebRTC* [Online]  
Available at: <https://www.html5rocks.com/en/tutorials/webrtc/basics/>
- [15]. Muaz Khan. 2016. *Getting Started Guide for RTCMultiConnection* [Online]  
Available at: <https://github.com/muaz-khan/RTCMultiConnection/blob/master/docs/getting-started.md>
- [16]. Muaz Khan. 2016. *WebRTC Experiments and Demos* [Online]  
Available at: <https://github.com/muaz-khan/WebRTC-Experiment>
- [17]. Ahmad Murey. *Simple Text Chat Box Tutorial*. [Online]  
Available at: <http://www.dreamincode.net/forums/topic/44808-simple-text-chat-box/>
- [18]. Peter Cowburn. PHP Documentation Group. 2016. *PHP manual*. [Online]  
Available at: <http://php.net/manual/en/index.php>



- [19]. W3Schools. 2016. *How to open, read, and close file in php*. [Online]  
Available at: [https://www.w3schools.com/php/php\\_file\\_open.asp](https://www.w3schools.com/php/php_file_open.asp)
- [20]. Google Chrome Developer. 2016. *MVC Architecture*. [Online]  
Available at: [https://developer.chrome.com/apps/app\\_frameworks](https://developer.chrome.com/apps/app_frameworks)
- [21]. Muaz Khan. 2016. *WebRTC Experiments and Demos* [Online]  
Available at: <https://github.com/muaz-khan/WebRTC-Experiment>
- [22]. W3Schools. 2016. *PHP MySQL Database*. [Online]  
Available at: [https://www.w3schools.com/php/php\\_mysql\\_intro.asp](https://www.w3schools.com/php/php_mysql_intro.asp)

## Appendix A – SOURCE CODES

### A.1. Login Feature Implementation

#### A.1.1. Login Controller (Angular JS)

```

//This is the angular JS app in the Login page
var app = angular.module('mainApp',[]);

// This grabs the login controller from the app above
app.controller('login_controller',function($scope, $http, $location)
{
    $scope.submit = function() {
        var uname    = $scope.username;
        var password = $scope.password;

        ...
        // Grab user name and pass and do null checkings
        ...

        // Preparing data to send to service script for final
verification
        var formData = {
            username : uname,
            password : password
        };

        // Making call to service script for verification
        $http({
            method: 'POST',
            url: 'url to user verify service script',
            data: $.param(formData),
            headers: {'Content-Type': 'application/x-www-form-
urlencoded'}
        }).then(function(response) {
            ...
            // Process response to show the result of authentication
            ...
        });
    }
});

```

### A.1.2. User Verify Service (PHP)

```

// Connect to the database
$name = $_POST['username'];
$username = $_POST['username'];
$password = $_POST['password'];

// Prepare the database connection
$db_host = 'localhost';
$db_user = 'root';
$db_pass = 'admin';
$db_name = 'VirtualTutor';

$connection = mysqli_connect($db_host,$db_user,$db_pass,$db_name);

if (mysqli_connect_errno()) {
    echo "Failed to connect to MySQL: " . mysqli_connect_error();
}

// Check if user's info exist in the database
$sql = "select * from Members where user_login = '$username' and
BINARY user_pass = '$password'";

$result = $connection->query($sql);
$row = $result->fetch_assoc();
$id = $row["ID"];

// if user exist and ID is found
if ($id) {
    ...
    // new session is registered for users
    // users info is registered in online users lists file for later
use
    ...

    // Return the user type to the controller on sucess:
teacher/student
    echo strtolower($type);
}
else {
    // Return failure result
    echo "Invalid username or password";
}

```

```
// Database connection close
$connection->close();
```

## A.2. Ask a Tutor Page Implementation Using Speech Recognition

### A.2.1. Mic Controller (JavaScript)

```
// Voice recognition script
// Testing browser support
window.SpeechRecognition = window.SpeechRecognition || // Chrome
                           window.webkitSpeechRecognition || //
Firefox
                           null;
if (window.SpeechRecognition === null) {

document.getElementsByClassName('microphone').setAttribute('disabled
', 'disabled');
}
else {
    // Creating a new recognizer
    var recognizer = new window.SpeechRecognition();
    var recordedMessage =
document.getElementById('readQuestionArea');

    // Collect the data once recognizer starts listening
    recognizer.onresult = function(event) {
        recordedMessage.value = '';

        for (var i = event.resultIndex; i < event.results.length; i++) {
            if (event.results[i].isFinal) {
                recordedMessage.value = event.results[i][0].transcript;
            } else {
                recordedMessage.value += event.results[i][0].transcript;
            }
        }
    };
};
```

Below is the function that initiates the recording and animation of the mic.

```
function startMic() {
    recognizer.start();
    turnMicOn();
}
```

```

}

function turnMicOn (){

    // Mic animation is done here and is done for 5 secs
    ...

    // after 5 secs mic is turned off
    stopMic();

}

function stopMic(){
    recognizer.stop();
}

```

Once mic is turned off, database searching is done. The code snippet to send the final data to the service script is shown below.

```

function askQuestion() {

    // The final data either using mic or manual typing is collected
    here
    var data = document.getElementById('readQuestionArea').value;

    // Making ajax call to search database script
    $.ajax({
        type: 'post',
        url: 'url to search database script',
        data: ({info:data}),
        success: function( response ) {
            // Collect the response and display the formatted result
        }
    });
}

```

### A.2.2. Search Database (PHP)

```

// Prepare the database connection
$db_host = 'localhost';

```

```

$db_user = 'root';
$db_pass = 'admin';
$db_name = 'VirtualTutor';

$connection = mysqli_connect($db_host,$db_user,$db_pass,$db_name);

if (mysqli_connect_errno()) {
    echo "Failed to connect to MySQL: " . mysqli_connect_error();
}

$sql = "select ID,Question from QandA";
$result = $connection->query($sql);

// calculate matching percentage
$result_array = [];
$index = 0;
while($row = $result->fetch_assoc()) {
    similar_text($data, $row['Question'], $percent);
    $result_array[$row['ID']] = $percent;
}

// sorting the matching percentage in array maintaining index
association
arsort($result_array);

// preparing the answer results to send back based on matching
percentage. Collect all answers above 40% matching
$result_keys = array_keys($result_array);
$result_values = array_values($result_array);

if (sizeof($result_keys) > 0) {
    $counter = 0;
    $index = 0;
    $outp = "";
    for ($i = 0; $i < sizeof($result_array); $i++) {
        if ($result_values[$i] > 40) {
            $sql = "select Answer,Sample_code,video_link,useful_link
from QandA where ID =". $result_keys[$i];
            $result = $connection->query($sql);
            $row = $result->fetch_assoc();

            if ($outp != "") {$outp .= "::::";}
            $outp .= $row['Answer'] . "::::" . $row['Sample_code'] .
"::::" . $row['video_link'] . "::::" . $row['useful_link'];

```

```

        $counter ++;
    }
    if ($counter > 4) $i = sizeof($result_array);
}
echo ($outp);
}
else {
    echo "No matching results found.";
}

// database connection close
$conn->close();

```

### A.3. Online Chat Implementation

#### A.3.1. Message Display Controller (JavaScript)

```

var app = angular.module('myApp', []);

app.controller('onlineUserController', function($scope, $http) {
    // Update message in the chat box every 1 sec
    setInterval(function(){
        $scope.update();
    }, 1000)

    $scope.update = function() {
        // http call to get the recent online messages with user
        info
        $http({
            method: 'POST',
            url: 'url to file opening and returning service',
            // param to identify caller to service
            data: $.param({infotype:"messages"}),
            headers: {'Content-Type': 'application/x-www-form-
            urlencoded'}}

        ).then(function(response) {
            // Collect and format the response to display in the
            view
            var html = "Formatted response";
            $("#chatBox").html(html);
        });

        // http call to get list of online users with their statuses

```

```

$http({
  method: 'POST',
  url: 'url to file opening and returning service',
  data: $.param({infotype:"users"}),
  headers: {'Content-Type': 'application/x-www-form-
urlencoded'}}

}).then(function(response) {
  // Collect and format the response to display the online
users
  var html = "Formatted online users";
  $("#usersOnLine").html(html);
});
}
});

```

### A.3.2. Get File Data Service (PHP)

```

// Look for infotype param sent by caller
$data = $_POST['infotype'];

// Based on $data, open file, collect contents
// example: request may be to collect list of online users
// or to collect recent online messages sent
if ($data == "online_users") {
  // open file containing list of online users
} else if ($data == "recent_messages") {
  // open file containing recent messages
}

echo array2string($fileContent);

// This function formats the file content and returns in usable
format
function array2string($data){
  $log_a = "";
  foreach ($data as $key => $value) {
    if(is_array($value)) $log_a .= array2string($value) .
"\n";
    else $log_a .= $value . "\n";
  }
  return $log_a;
}

```



### A.3.3. Chat Message Controller (jQuery)

```

$(document).ready(function() {
    $('#messageForm').on('submit', function(e) {
        e.preventDefault();

        var message = document.getElementById("message").value;
        if (!message) {return false;}
        $.ajax({
            type: 'post',
            url: 'url to necessary service script to communicate with
backend',
            data: ({message:message}),
            success: function( response ) {
                // this response determines if saving of message was
sucess
                // notify user based on response result
            }
        });
    });
});

```

### A.3.4. Send Message Service (PHP)

```

// Collect the user information and message sent
session_start();
$user = $_SESSION['name'];
$id   = $_SESSION['ID'];
$message = $_POST['message'];

// Collect the content of file
$chat_file=file("message containing file",FILE_IGNORE_NEW_LINES);

// Add new message to the array containing old messages with time
stamp
$chat_file[]=date("Y-m-d H:i:s")."<!--> ".$id." ".$user." :
".$message;

// See if message count is less than 20, slice oldest message if
otherwise
if (count($chat_file)>20)$chat_file=array_slice($chat_file,1);

// Save the new list of messages to the file

```

```

$file_save = fopen("message containing file", "w+");
flock($file_save, LOCK_EX);
for($line = 0; $line < count($chat_file); $line++){
    fputs($file_save, $chat_file[$line]. "\n");
};
flock($file_save, LOCK_UN);
fclose($file_save);
echo "Message Sent";

```

#### A.4. Status Controller

```

//Get the current status responsible for changing the status
window.onload = updateCurrentStatus;
setInterval(updateCurrentStatus, 4000);

// This function updates the current status as saved in the server
function updateCurrentStatus() {
    $.get('url to current status service', function(result) {
        changeStatus(result);
    });
}

// This function is responsible for changing the status
function changeStatus(status) {
    var mainStatus = status;
    // Collect the new status which is triggered from the view
    // Display the new status with images
    ...
    $(".dropdown").html(html);

    // AJAX call to change the status and save the new status in the
server
    $.ajax({
        type: 'post',
        url: 'url to change status service',
        data: ({status:mainStatus}),
        success: function( data ) {
        }
    });
}

```

### A.4.1. Current Status and Change Status Services (PHP)

#### Current Status

```
session_start();
echo $_SESSION['status'];
```

#### Change Status

Change status service script sets the new status of the user to the current session. Hence, the status of the user can be changed by the user as long as session exists.

```
// Collect the new status to be set sent by controller
session_start();
$status = $_POST['status'];
$id = $_SESSION['ID'];
$name = $_SESSION['Name'];

$newInfo = $status . ' ' . $_SESSION['ID'] . ' ' . $_SESSION['name'];
$oldInfo = $_SESSION['status'] . ' ' . $_SESSION['ID'] . ' ' .
$_SESSION['name'];

// Collect the list of online users
$online_users = file("url to file with online user
list", FILE_IGNORE_NEW_LINES);
$pattern = '/' . $id . '\s' . $name . '/';
$matchFound = false;

// Find the user in the online user list and change status
for ($i=0; $i<count($online_users);$i++) {
    if (preg_match($pattern, $online_users[$i])) {
        $online_users[$i] = $newInfo;
        $_SESSION['status'] = $status;
        break;
    }
}

// Save the list of online users with updated status of the user
$myfile = fopen("url to file with online user list", "a")
    or die("Unable to open file!");
$file_save=fopen("url to file with online user list", "w+");
flock($file_save, LOCK_EX);
for($line = 0;$line<count($online_users);$line++){
    fputs($file_save,$online_users[$line]."\n");
}
```

```
};
flock($file_save, LOCK_UN);
fclose($file_save);
```

## A.5. Screen Sharing Implementation

### A.5.1. Screen Sharing Controller (JavaScript)

An object of RTCMulticonnection is created and properties are set.

```
var connection = new RTCMultiConnection();
connection.socketURL = 'signaling url here';
connection.socketMessageEvent = 'message for debugging';
connection.session = {
  audio: 'two-way', // audio is two-way, rest is one way
  screen: true,
  oneway: true
};

connection.sdpConstraints.mandatory = {
  OfferToReceiveAudio: true,
  OfferToReceiveVideo: true
};

// set the video and audio container in the view page
connection.videosContainer = document.getElementById('video-
container');
connection.audiosContainer = document.getElementById('audio-
container');
```

Now connection can be started using on-stream event and the media element can be displayed in the container.

```
connection.onstream = function(event) {
  var width = connection.videosContainer.clientWidth;
  var mediaElement = getMediaElement(event.mediaElement, {
    title: event.userid,
    width: width,
  });
});

event.stream.isScreen ? {
```

```

        connection.videosContainer.appendChild(mediaElement);
    } : {
        connection.audiosContainer.appendChild(mediaElement);
    }

    setTimeout(function() {
        mediaElement.media.play();
    }, 5000);

    mediaElement.id = event.streamid;
};

```

The user-id is returned from the `getScreenId.js` library which is included in the project. The function below is responsible for detecting the extension available in the browser. If the extension is not detected, an error is thrown. As a user, the screen sharing won't start on the screen.

```

connection.getScreenConstraints = function(callback) {
    getScreenConstraints(function(error, screen_constraints) {
        if (!error) {
            screen_constraints =
connection.modifyScreenConstraints(screen_constraints);
            callback(error, screen_constraints);
            return;
        }
        throw error;
    });
};

```

Again, `getScreenConstraints` is included in the `getScreenId.js` library.

Now when the streaming ends, the *onstreamended* event is called and the screen sharing view is removed.

```

connection.onstreamended = function(event) {
    var mediaElement = document.getElementById(event.streamid);
    if(mediaElement) {
        mediaElement.parentNode.removeChild(mediaElement);
    }
}

```

```
};
```

The function below is responsible for handling the room id.

```
var roomid = '';
// Get the id if available in local storage else create one
if (localStorage.getItem(connection.socketMessageEvent)) {
    roomid = localStorage.getItem(connection.socketMessageEvent);
} else {
    roomid = connection.token();
}
document.getElementById('room-id').value = roomid;

var hashString = location.hash.replace('#', '');
var roomid = params.roomid;
if(!roomid && hashString.length) {
    roomid = hashString;
}

(function() {
    var params = {},
        r = /(?:[&=]+)=?(?:[^\&]*)/g;

    function d(s) {
        return decodeURIComponent(s.replace(/\+/g, ' '));
    }
    var match, search = window.location.search;
    while (match = r.exec(search.substring(1)))
        params[d(match[1])] = d(match[2]);
    window.params = params;
})();
```

Once the room id is created following function is responsible for generating unique URL to the room. For example, below is an example to show the unique URL using the vtstcloudstate domain name.

```
function showRoomURL(roomid) {
    var roomHashURL = '#' + roomid;
    var roomQueryStringURL = '?roomid=' + roomid;
```

```

var html = '<h2>Unique URL for your room:</h2><br>';

html += '<i>Hash URL: <font
color="blue">https://www.vtstcloudstate.com/Views/screen.php' +
roomHashURL + '</font></i>';
html += '<br>';
html += '<i>QueryString URL: <font
color="blue">https://www.vtstcloudstate.com/Views/screen.php' +
roomQueryStringURL + '</font></i>';

var roomURLsDiv = document.getElementById('room-urls');
roomURLsDiv.innerHTML = html;

roomURLsDiv.style.display = 'block';
}

```

### A.5.2. Screen Sharing Online User List Controller (JavaScript)

#### Invitation source code part

```

// This function helps user to select users from the online list and
// send invites
var invitation_successful = false;
var invited_users = [];
function sendInvites() {

    // Detect if the room has been created yet
    if ($('#videos-container').is(':empty')) {
        // Ask user to open room first to send invitation
        return;
    }

    var atLeastOneSelected = false, index = 0;
    for (var i = 0; i < online_user_array.length-1; i++) {
        // Detect all the selected users from online user list
        // create an array of invited user list with room id
        // if the checkbox is checked, user is selected for
invitation
        if ($('#'+id).prop('checked')) {
            invited_users[index] = online_user_array[i] + ' ' +
$("#room-id").val();
            atLeastOneSelected = true;
            index++;
        }
    }
}

```

```

    }

    if (!atLeastOneSelected) {
        // Report user to select at least one user to send the
invitation
        return;
    }

    sendInvitation();

    // Display the invitation successfull message here
    $(' .modal-title').html('<font color="green">Invitation
sent.</font>');
    $(' .modal-body').html('Invitation Successful!');
    atLeastOneSelected = false;
    clearAll();
}

// This function helps select all the users at once
function selectAll() {
    if (!online_user_array) {return;}
    for (var i = 0; i < online_user_array.length-1; i++) {
        var id = 'checkbox' + i;
        $(' "#"+ id ).prop( "checked", true );
    }
}

// This function helps user to clear the selection
function clearAll() {
    if (!online_user_array) {return;}
    for (var i = 0; i < online_user_array.length-1; i++) {
        var id = 'checkbox' + i;
        $(' "#"+ id ).prop( "checked", false );
    }
}

// This function finally sends invitation to other users making AJAX
calls
function sendInvitation() {
    var jsonString = JSON.stringify(invited_users);
    $.ajax({
        type: 'post',
        url: 'url to send invitation service',
        data: ({message:jsonString}),
    }

```



```

        success: function( data ) {}
    });
}

```

### A.5.3. Send Invitation Service (PHP)

```

// Collect the invitation sender info
session_start();
$user = $_SESSION['name'];
$id = $_SESSION['ID'];

// gather list of invited users from the controller
$data = json_decode(stripslashes($_POST['message']));

// create a pattern with sender id and name to match to the gathered
// list of invited users
// id is needed because it is unique to every individual user
$sender = '/' . $id . '\s' . $user . '/';

$myfile = fopen("url to invitation.txt", "w") or die("Unable to open
file!");
$invited_users = [];
foreach($data as $aData){
    if (isset($aData)){
        // if the invitation is for self skip else save the info
        if (preg_match($sender, $aData)) {
            continue;
        }
        $invited_users[] = $aData . ' ' . $user;
    }
}

$file_save=fopen("url to invitation.txt", "w+");
flock($file_save, LOCK_EX);
for($line=0;$line<count($invited_users);$line++){

    // Save all the invited user to the file
    fputs($file_save,$invited_users[$line]."\n");
};
flock($file_save, LOCK_UN);
fclose($file_save);

```

#### A.5.4. Detect Invitation to Screen Controller (JavaScript)

```

window.onload = checkInvitation;
setInterval(checkInvitation,3000);

// This function checks if there is any invitation for current user
function checkInvitation() {
    $.get('url to detect invitation to screen service',
function(result) {

        // result contains the unique room id and inviter
        if (result && result != 0) {
            var keyAndInviter = result.split(" ");
            var url = 'https://www.vtstcloudstate.com/screen.php#' +
keyAndInviter[0];

            // Create a bootstrap pop-up dialog box with url and
inviter information
            $('.modal-title').html('<font color="Green">You have
been invited to a screen sharing room by</font> <b>' +
keyAndInviter[1] + '</b> .');

            var html = 'Please use url below to go to the room or
click on the go to link button below';
            html += '<br>'+ url;
            html += '<br><br><div id="browse_app"><a class="btn btn-
large btn-info center-block" href="' + url + '" target="_blank">GoTo
Room</a></div>';

            $('.modal-body').html(html);
            $('.center-block').css("width", "40%");

            $("#myModal").modal();
            $('#myModal').css("width", "100%");
        }
    });
}

```

#### A.5.5. Detect Invitation to Screen Service (PHP)

```

session_start();

// if either id or name is not set, return

```

```

if (!isset($_SESSION['ID']) || !isset($_SESSION['name'])) { echo 0;
return;}

// Collect the current user id and name
$id = $_SESSION['ID'];
$name = $_SESSION['name'];

// collect the invited users list
$invitation_users=file("url to
invitation.txt",FILE_IGNORE_NEW_LINES);

// Create a pattern with id and name
// id is needed because it is unique to every individual user
$user_pattern = '/' . $id . '\s' . $name . '/';
$key = 0;
$name = '';

// detecting if any invitation is made for current user
for ($i=0; $i<count($invitation_users);$i++) {
    if (preg_match($user_pattern, $invitation_users[$i])) {
        $singleArray = explode(" ", $invitation_users[$i]);
        deleteDetectedinInvitationList($invitation_users[$i]);
        $key = $singleArray[2];
        $name = $singleArray[3];
        break;
    }
}

// if any room key is found, respond back to controller with room
// key and inviter name
if ($key) {
    echo $key . ' ' . $name;
}

// This function assists in deleting the detected invitation so that
// invitation is not made again and again
function deleteDetectedinInvitationList($a_user){
    $user = $a_user;
    $filename = "url to invitation.txt";
    $data = file($filename);

    $share_screen_users = array();

    //Storing all the users back that do not match with this users

```

```

foreach($data as $a_user) {
    if(trim($a_user) != $user) {
        $share_screen_users[] = $a_user;
    }
}

// put back all the users that have not been invited yet
$fp = fopen($filename, "w+");
flock($fp, LOCK_EX);
foreach($share_screen_users as $a_user) {
    fwrite($fp, $a_user);
}
flock($fp, LOCK_UN);
fclose($fp);
}

```

## A.6. Manage Members Implementation

### A.6.1. Manage Database Controller (JavaScript)

```

// Display members information on window Load
window.onload = fetchData;

// This function makes a call to service to gather all the member
information
function fetchData() {
    $.get('url to fetchdata database service', function(data) {
        if (data) {
            updateTable(data);
        }
    });
}

// This function parses the response from service and displays in
the table in readable format
function updateTable(response) {
    var data = JSON.parse(response);
    var html = "";

    for(var i = 0;i<data.records.length;i++) {
        // Data contains list of members with their full details in
particular format
        // Parse them and display in the table

```

```

        html += ...
    }
    $("#members").html(html);
}

```

#### A.6.1.1. Add a Member (JavaScript)

*// This function on call displays the form to fill in new member's details*

```

function addThisItem() {

    // Display the form to gather new member's details using Bootstrap
    $(".modal-title").html("Add New Member");
    var html = '<form role="form"><div class="form-group"><label for="fn">First Name:</label><input class="form-control input-sm" id="fn" type="text" required>';
    html += '</div><div class="form-group"><label for="ln">Last Name:</label><input class="form-control input-sm" id="ln" type="text" required></div>';
    html += '<div class="form-group"><label for="type">User Role</label><select class="form-control " id="type" required><option></option><option>Teacher</option><option>Student</option></select></div>';
    html += '<div id="confirmButton"><button type="submit" class="btn btn-success btn-responsive center-block" onclick="confirmAddThisItem()">Confirm Add</button></div></form>';

    $(".modal-body").html(html);
    $('#myModal1').modal();
}

```

*// On add confirmation, this function sends the add request to service with member details to finally add to the database*

```

function confirmAddThisItem() {
    // Collect First Name, Last Name and user role as Teacher/Student
    var firstname = document.getElementById('fn').value;
    var lastname = document.getElementById('ln').value;
    var type = document.getElementById('type').value;
    var username = firstname;

```

*// Return is one of the item is missing*

```

if (!firstname || !lastname || !type) return;

// Generate unique pass for new user
var password = createID();

// Prepare data to send to service to add to the database
var data = {
    'transaction_type': 'add',
    'fname': firstname,
    'lname': lastname,
    'uname': username,
    'pword': password,
    'type': type
};

// Make AJAX call to service and send the prepared data
$.ajax({
    type: 'post',
    url: 'url to add/edit/delete database services',
    data: {fullData:data},
    success: function( response ) {
        // Transaction is success if response is 1
        if (response == 1){
            // Refill the table with the new member
            fetchData();
            // Display necessary success notification
        } else {
            alert("Something went wrong. try again later.");
        }
    }
});
}

// This function is used to create a unique password
function createID() {
    var id = "";
    var possible =
    "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789";
    for( var i=0; i < 6; i++ )
        id += possible.charAt(Math.floor(Math.random() *
possible.length));
    return id;
}

```

### A.6.1.2. Edit a Member (JavaScript)

```

// Given the id of the user to be modified, collect member details
// and show them in the editing form
function editThisItem(id) {
    // Collect details of member to be edited and show in the form
    // using the table with full member details
    ...

    // make the form and display
    ...

    // display member's details in the editing form
    ...
}

// On edit confirmation, this function sends the UPDATE request to
// service with modified member details to finally reflect the
// modification in the database
function confirmEditThisItem(id) {

    // Collect the new details from the editing form
    ...

    // return if one of the information is missing
    if (!firstname || !lastname || !username || !password || !type)
return;

    // prepare data to send to the service to make the Update
    transaction
    var data = {
        'transaction_type':'edit',
        'idNum':id,
        'fname':firstname,
        'lname':lastname,
        'uname':username,
        'pword':password,
        'type':type
    };

    // make AJAX call with the prepared data
$.ajax({
    type: 'post',
    url: 'url to add/edit/delete database service',

```

```

data: {fullData:data},
success: function( response ) {
    // Transaction is success if response is 1
    if (response == 1){
        // Refill the table with the member's modified details
        fetchData();
        // Display necessary success notification
    } else {
        alert("Something went wrong. try again later.");
    }
}
});
}

```

### A.6.1.3. Delete a Member (JavaScript)

```

// This function displays a form with the member detail to delete
function deleteThisItem(id) {
    var firstname =
document.getElementById('firstname'+id).innerHTML;
    var lastname = document.getElementById('lastname'+id).innerHTML;
    $(".modal-title").html("<font color='red'>Delete
Member</font>");
    $(".modal-body").html('Are you sure you want to delete this
member? <br><br>' + firstname + ' ' + lastname + '</b>');
    $(".modal-footer").html('<button type="button" class="btn btn-
danger btn-responsive center-block"
onclick="confirmDeleteThisItem('+id+')">Delete</button>');

    // display the form
    $('#myModal1').modal();
}

// On delete confirmation, this function sends the DELETE request to
service with member's ID
function confirmDeleteThisItem(id) {
    // prepare the data to send to service
    var data = {
        'transaction_type':'delete',
        'idNum':id
    };

    // make AJAX call with collected data to perform DELETE transaction

```



```

$.ajax({
  type: 'post',
  url: '../Services/add_edit_delete_database_services.php',
  data: {fullData:data},
  success: function( response ) {
    // Transaction is success if response is 1
    if (response == 1){
      // Refill the table with left member's details
      fetchData();
    } else {
      alert("Something went wrong. try again later.");
    }
  }
});
}

```

#### A.6.2. Fetch Data Database Service (PHP)

```

// Prepare the database connection
$db_host = 'localhost'; $db_user = 'root'; $db_pass = 'admin';
$db_name = 'VirtualTutor';

$connection = mysqli_connect($db_host,$db_user,$db_pass,$db_name);

if (mysqli_connect_errno()) {
  echo "Failed to connect to MySQL: " . mysqli_connect_error();
}

// Collect all members
$sql = "select * from Members";
$result = $connection->query($sql);
$outp = "";
while($rs = $result->fetch_array(MYSQLI_ASSOC)) {
  if ($outp != "") {$outp .= ",";}
  $outp .= '{"ID":"' . $rs["ID"] . ',';
  $outp .= '"Firstname":"' . $rs["user_fname"] . ',';
  $outp .= '"Lastname":"' . $rs["user_lname"] . ',';
  $outp .= '"User":"' . $rs["user_login"] . ',';
  $outp .= '"Pass":"' . $rs["user_pass"] . ',';
  $outp .= '"UserType":"' . $rs["user_type"] . '"}';
}
$outp = '{"records":[' . $outp . ']}';

```

```
// return the final record and Close the connection to database
echo($outp);
$connexion->close();
```

### A.6.3. Add, Edit, or Delete Database Service (PHP)

```
// Collect the send data from the controller
$sentData = $_POST['fullData'];
$transaction_type = $sentData['transaction_type'];

// Determine the transaction type and call the necessary function
if ($transaction_type == 'edit') { editMember($sentData);}
else if ($transaction_type == 'add'){ addMember($sentData);}
else { deleteMember($sentData);}

// This function with the given data performs the UPDATE transaction
function editMember($sentData){
    // Extract the data from the sent data package
    $id = $sentData['idNum'];
    $firstname = $sentData['fname'];
    $lastname = $sentData['lname'];
    $username = $sentData['uname'];
    $password = $sentData['pword'];
    $type = $sentData['type'];

    // Prepare the database connection
    $db_host = 'localhost'; $db_user = 'root'; $db_pass = 'admin';
    $db_name = 'VirtualTutor';

    $connection =
mysqli_connect($db_host,$db_user,$db_pass,$db_name);

    if (mysqli_connect_errno()) {
        echo "Failed to connect to MySQL: " . mysqli_connect_error();
    }

    $sql = "UPDATE Members SET user_fname = '$firstname', user_lname
= '$lastname', user_login='$username', user_pass='$password',
user_type='$type' WHERE ID = '$id'";
    $result = $connection->query($sql);
    $connection->close();

    // Respond with the success result back to the controller
```

```
    echo($result);
}

function addMember($sentData){
    // Extract the data from the sent data package
    $firstname = $sentData['fname'];
    $lastname = $sentData['lname'];
    $username = $sentData['uname'];
    $password = $sentData['pword'];
    $type = $sentData['type'];

    // Prepare and make the database connection as in edit member
    function
    ...
    $sql = "INSERT INTO Members (user_fname, user_lname,
user_login,user_pass,user_type) VALUES ('$firstname', '$lastname',
'$username', '$password', '$type')";

    $result = $connection->query($sql);
    $connection->close();

    // Respond with the success result back to the controller
    echo($result);
}

function deleteMember($sentData){
    // Extract the data from the sent data package
    $id = $sentData['idNum'];

    // Prepare and make the database connection as in edit member
    function
    ...
    $sql = "DELETE FROM Members WHERE ID = '$id'";
    $result = $connection->query($sql);
    $connection->close();

    // Respond with the success result back to the controller
    echo($result);
}
```

## A.7. Display and Modifying Lab Assistant Hours' Implementation

### A.7.1. Lab Assistant Hours' Controller (jQuery)

```

// This jQuery on Load gets the current schedule, making a request
to service
$.get("url to get lab assistant hours service", function(data){
    // Display the table with data in html element with id as content
    $("#content").html(data);
});

// Below functions are ready to be triggered on Load
$(document).ready(function() {
    // this function on click makes the table editable
    $("#edit_update" ).click(function() {
        $("#content").attr("contenteditable", "true");
        $("#info_edit").html('<i><font color="red">Table is now
editable ...</font></i>');
        $('#save').removeAttr('disabled');
    });

    // this function on click saves the modified table making a
request to service
    $("#save" ).click(function() {
        var message = $("#content").html();

        // make a AJAX post request to save the table
        $.ajax({
            type: 'post',
            url: 'url to get lab assistant hours service',
            data: ({message:message}),
            success: function( data ) {
                // update the table attributes once save is done
                $("#content").attr("contenteditable", "false");
                $("#info_edit").html('<br>');
                $('#save').attr('disabled', 'disabled');
            }
        });
    });
});

```

### A.7.2. Get Lab Assistant Hours' Service (PHP)

```
// Detect the request and if request is POST, save the modified data
sent as message else respond with current schedule to the controller
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $message = $_POST['message'];
    file_put_contents('url to file with schedule', $message);
    echo "Schedule Updated";
} else {
    $lab_hours_schedule_file=file("url to file with
schedule",FILE_IGNORE_NEW_LINES);
    echo array2string($lab_hours_schedule_file);
}

// this function converts array to string
function array2string($data){
    $log_a = "";
    foreach ($data as $key => $value) {
        if(is_array($value))    $log_a .= array2string($value) . "\n";
        else                    $log_a .= $value . "\n";
    }
    return $log_a;
}
```

### A.8. Search Controller

In order to be able to use this feature, following library should be linked.

```
<link rel="stylesheet"
href="//maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.cs
s">
```

Below is the code that implements the search mechanism saved in the table.

```
<div>
    <div class="row">
        <div class="col-md-3">
            <form action="#" method="get">
                <div class="input-group">
                    <input class="form-control" id="system-search"
name="q" placeholder="Search for" required>
```

```

        <span class="input-group-btn">
            <button type="submit" class="btn btn-
default"><i class="glyphicon glyphicon-search"></i></button>
        </span>
    </div>
</form>
</div>
<div class="col-md-12">
    <table class="table table-list-search">
        <tbody>
            <!--List of datas in the table-->
            <tr><td>Data1</td></tr>
            <tr><td>Data1</td></tr>
            <tr><td>Data1</td></tr>
        </tbody>
    </table>
</div>
</div>
</div>

```

## A.9. Contact Message Sending Implementation

### A.9.1. Contact Message Controller (Angular JS)

```

// App that is registered in the view
var app = angular.module('mainApp', []);
app.controller('contact_message_controller', function($scope, $http)
{
    // this is called once submit button is clicked
    $scope.submit = function() {
        // collect the data from the scope
        var name    = $scope.fname;
        var lname   = $scope.lname;
        var email    = $scope.email;
        var msg     = $scope.message;
        var missing = false;

        // return if either the name, lname, email or message is missing
        if (missing) { // display missing error and return
            ...
            return;
        }

        // prepare the form data to send to the service
    }
}

```

```

var formData = {
    firstname : name,
    lastname  : lname,
    email     : email,
    message   : msg
};

// make http POST call to service with form data to save the
messages in database
$http({
    method: 'POST',
    url: 'url to contact message store service',
    data: $.param(formData),
    headers: {'Content-Type': 'application/x-www-form-
urlencoded'}}
).then(function(response) {
    if (response.data) {
        ... // display the pop up with success notification
    }
    else {
        ... // display the pop up with failed notification
    }
});
});

```

### A.9.2. Contact Message Store Service (PHP)

```

// Collect the sender details and messages
$firstname = $_POST['firstname']; $lastname = $_POST['lastname'];
$email = $_POST['email']; $message = $_POST['message'];

// Prepare the database connection
$db_host = 'localhost'; $db_user = 'root'; $db_pass = 'admin';
$db_name = 'VirtualTutor';

$connection = mysqli_connect($db_host,$db_user,$db_pass,$db_name);

if (mysqli_connect_errno()) {
    echo "Failed to connect to MySQL: " . mysqli_connect_error();
}

// make INSERT transaction to Messages table in the database

```

```

$sql = "INSERT INTO ContactMessages (FirstName, LastName,
Email,Messages) VALUES ('$firstname', '$lastname',
'$email','$message')";
$result = $connection->query($sql);
$connection->close();

// respond with transaction result
echo($result);

```

## A.10. Contact Message Management Implementation

### A.10.1. Contact Message Database Controller (JavaScript)

First, the fetching of data is done and shown in the table.

```

//Get all the messages and display them on load
window.onload = fetchData;

// This function fetches all messages from database by making a
request to service
function fetchData() {
    $.get('url to contact messages fetch database service',
function(data) {
    // if data is available, display them in table
    if (data) {
        updateTable(data);
    }
    });
}

function updateTable(response) {
    var data = JSON.parse(response);
    var html = "";
    for(var i = 0;i<data.records.length;i++) {
        // display each records found in the database in the table
        // tagging each row with id of the messages to id of the row
        html += ...
        ...
    }
    // display collected message in html element with id
all_messages
$("#all_messages").html(html);
}

```



When the teachers want to delete messages from the database, following functions are triggered which makes a request to services to perform the job.

```
// This function finally deletes the item with given id
// from the database making a request to service
function confirmDeleteThisItem(id) {
    // prepare data to send while making post request
    var data = {
        'MessageID':id
    };

    // Making ajax post call to service with id of the message to be
    // deleted
    $.ajax({
        type: 'post',
        url:
        '../Services/delete__contact_message_database_services.php',
        data: {data:data},
        success: function( response ) {
            if (response == 1){
                // display the pop up with sucess notification
                ...
                // refill the table with updated data
                fetchData();
            } else {
                alert("Something went wrong. try again later.");
            }
        }
    });
}
```

#### A.10.2. Contact Message Fetch Database Service (PHP)

```
// Prepare the database connection
$db_host = 'localhost'; $db_user = 'root'; $db_pass = 'admin';
$db_name = 'VirtualTutor';

$connection = mysqli_connect($db_host,$db_user,$db_pass,$db_name);

if (mysqli_connect_errno()) {
    echo "Failed to connect to MySQL: " . mysqli_connect_error();
}
```

```

}

// main query
$sql = "select * from ContactMessages";

// execute the query
$result = $connection->query($sql);
$outp = "";

// collect the output in json format for easy handling
while($rs = $result->fetch_array(MYSQLI_ASSOC)) {
    if ($outp != "") {$outp .= ",";}
    $outp .= '{"MessageID":"' . $rs["MessageID"] . ',';
    $outp .= '"FirstName":"' . $rs["FirstName"] . ',';
    $outp .= '"LastName":"' . $rs["LastName"] . ',';
    $outp .= '"Email":"' . $rs["Email"] . ',';
    $outp .= '"Messages":"' . $rs["Messages"] . ',';
    $outp .= '"Datetime":"' . $rs["Datetime"] . '"}';
}
$outp = '{"records":[' . $outp . ']}';
$connection->close();

// repond to the controller with the collected information
echo($outp);

```

### A.10.3. Delete Contact Message Database Service (PHP)

```

// Collect data sent by controller
$sentData = $_POST['data'];
$messageID = $sentData['MessageID'];

// Prepare the database connection
$db_host = 'localhost'; $db_user = 'root'; $db_pass = 'admin';
$db_name = 'VirtualTutor';

$connection = mysqli_connect($db_host,$db_user,$db_pass,$db_name);

if (mysqli_connect_errno()) {
    echo "Failed to connect to MySQL: " . mysqli_connect_error();
}

// main query
$sql = "DELETE FROM ContactMessages WHERE MessageID = '$messageID'";

```

```

// execute the query
$result = $connection->query($sql);
$connection->close();

// respond with the transaction result
echo($result);

```

### A.11. Resume Session Controller (JavaScript)

```

// Function to notify user of nearing the ending session and
// suggesting to add time to the session
function resumeSession() {
    var r = confirm("Session is about to expire, Click 'OK' to stay
logged in or else you will get logged out in 20 minutes.");
    if ( r == true) {
        // Calling resume session service to increase the time
        $.ajax ({
            type: "GET",
            url: "url to resume session service",
            success: function( result ) {
                alert(result);
            }
        });
    }
}

```

#### A.11.1. Resume Session Service (PHP)

```

session_start();
$_SESSION['last_logged_in_time'] = time();
echo "extended" ;
$_SESSION['extended'] = 1;

```

### A.12. Additional Services

#### A.12.1. Timeout Session Check (PHP)

##### Student Time Check

```

session_start();

```

```

// Warn the user with pop-up notifying about the session time about
to end
if ($_SESSION['last_logged_in_time'] < time() - 180 * 60) {
    echo "<script type='text/javascript'
src='//code.jquery.com/jquery-1.7.1.min.js'></script>
    <script src='url to resume session
controller'></script><script>resumeSession();</script>";
}

// Log out if session time limit is reached
if ($_SESSION['last_logged_in_time'] < time() - 200 * 60) {
    header('Location: url to logout service');
}

// check if the user is listed in the online users list
$online_users=file("url to onlineUsers.txt",FILE_IGNORE_NEW_LINES);
$pattern = '/' . $_SESSION['ID'] . '\s' . $_SESSION['name'] . '/';
$matchFound = false;
for ($i=0; $i<count($online_users);$i++) {
    if (preg_match($pattern, $online_users[$i])) {
        $singleArray = explode(" ", $online_users[$i]);
        $_SESSION['status'] = $singleArray[0];
        $matchFound = true;
        break;
    }
}

if (!$matchFound) {
    // if match not found, user logged out from another system
    // code is 10, re-route user to log out service
    $_SESSION['connect'] = 10;
    header('Location: url to logout service');
}

// if user uses authenticated url without logging in, immediately log
out and route to login screen
if (!$_SESSION['connect'] || $_SESSION['type'] != "student") {
    header('Location: url to login view');}
$name = $_SESSION['name'];

```

### Teacher Time Check

```

// Same as student time check except for the last part

```

...

```
// if it happens that user comes to page who is not logged in and or
// is not teacher,
// immediately log out and route to login screen
if (!$SESSION['connect'] || $SESSION['type'] != "teacher") {
header('Location: url to login view');}
$name = $SESSION['name'];
```

### A.12.2. Logout Service (PHP)

```
session_start();
// Display message if user was logged out because session timeout
if (isset($SESSION['timeout'])) {
    if ($SESSION['timeout'] < time() - 75 * 60) {
        echo "<script>alert('Session ended. Logging out.');";
    }
}
```

```
// Display message if user logged out in different window & Log out
if ($SESSION['connect'] == 10){
    $message = "You logged out in different system. Hence, you
automatically logged out from here.";
    echo "<script
type='text/javascript'>alert('$message');

```

```
$online_users = array();
```

```
// Storing all the users that did not log out yet
foreach($data as $a_user) {
    if(trim($a_user) != $user) {
        $online_users[] = $a_user;
    }
}
```

```
// put back all the online users who are still online
$fp = fopen($filename, "w+");
```

```

flock($fp, LOCK_EX);
foreach($online_users as $a_user) {
    fwrite($fp, $a_user);
}
flock($fp, LOCK_UN);
fclose($fp);

$_SESSION['connect'] = 0;
$_SESSION=array();

// Unset the session parameters and destroy the session
unset($_SESSION);
session_destroy();

```

### A.12.3. Image Slider Service (jQuery)

```

$(function() {
    // collect the slides details
    var slides = $(".slideshow>li");
    var slideCount = 0;
    var totalSlides = slides.length;
    var slideCache = [];
    var information_images = [// an array of subtitles strings];

    // This function loads up all the images with class slideshow
    and with tag li
    (function preloader(){
        if (slideCount < totalSlides) {
            //Load images
            slideCache[slideCount] = new Image();
            slideCache[slideCount].src =
slides.eq(slideCount).find('img').attr('src');
            slideCache[slideCount].onload = function(){
                slideCount++; preloader();
            }
        }
        else {
            //run the slideshow
            slideCount = 0;
            SlideShow();
        }
    }());

```

```

    // This function starts the slideshow and continue to display
    next slides
    function SlideShow() {
        slides.eq(slideCount).fadeIn(1000).delay(2000).fadeOut(1000,
function(){
    var info_image =
document.getElementById("information_bar");
    var i = 0;
    if (slideCount+1 >= totalSlides) {
        i = 0;
    }
    else {
        i = slideCount + 1;
    }
    info_image.innerHTML = information_images[i];
    slideCount < totalSlides - 1 ? slideCount++ : slideCount
= 0;
    SlideShow();
    })
}
});

```