

3-2018

Learning Management System (LMS) Using C#, ASP.Net and SQL SERVER

Dipti Arora

St. Cloud State University, ardi1101@stcloudstate.edu

Follow this and additional works at: https://repository.stcloudstate.edu/csit_etds



Part of the [Computer Sciences Commons](#)

Recommended Citation

Arora, Dipti, "Learning Management System (LMS) Using C#, ASP.Net and SQL SERVER" (2018). *Culminating Projects in Computer Science and Information Technology*. 25.

https://repository.stcloudstate.edu/csit_etds/25

This Starred Paper 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.

Learning Management System (LMS) Using C#, ASP.Net and SQL SERVER

by

Dipti Arora

A Starred Paper

Submitted to the Graduate Faculty of

St. Cloud State University

in Partial Fulfillment of the Requirement

for the Degree

Master of Science

in Computer Science

March 2018

Starred Paper Committee:
Ezzat Kirmani, Chairperson
Jie Hu Meichsner
Jim Q. Chen

Abstract

A learning management system (LMS) is a software application for delivery and management of educational courses. It is a client-server type solution, typically web-based, used to handle student registration, delivery of course content, administration of tests and assignments, and related record keeping. An LMS is particularly useful for fully online courses or training programs.

The objective of this paper is two-fold. Firstly, we provide an overview of learning management systems, their architecture, and other features. This summary enables us to identify the software requirements specifications for a new LMS. The second objective of this paper is to describe the creation of our own LMS. This new LMS, named LearningMadeEasy4U (LME4U), implements the requirements identified.

The main features of LME4U include file uploading, messaging, course registration, course creation, quiz generation, and downloading capability. Various types of files can be uploaded or downloaded, making it convenient to submit assignments and upload lessons. LME4U's messaging system enables timely communication between instructor and learner. The system generates quizzes and also has an assessment engine. Login functions are activated for students and instructors. LME4U is capable of serving a wide range of students. It is customizable, and a number of applications can be integrated with it.

Table of Contents

	Page
List of Tables	5
List of Figures	6
Section	
1. Introduction	7
1.1 Basic Types of LMS	9
2. Features of Learning Management Systems	12
3. Project Overview	15
3.1 Project Description	15
3.2 Advantages of Learning Management Systems	15
3.3 Disadvantages of LMS	16
3.4 Current Issues with LMS	18
3.5 LMS Trends	19
3.6 Features of Other LMS that Inspired Development of LME4U	20
3.7 Security Features of Other LMS Systems	26
4. Design of LME4U	28
4.1 Features of the Newly Created LMS System	28
4.2 Design Details	31
4.3 Screenshots	39
5. Performance Evaluation of the Learning Management System	48
6. Conclusion	50

	4
References	51
Appendix	53

List of Tables

Table	Page
1. ‘Comparison of LMS Systems’–Moodle, Talent and LME4U	30
2. ‘Time to Learn Example’–Measurement of Learning Time	48

List of Figures

Figure	Page
1. Course Copying	20
2. Example of Course Archiving	21
3. Example of Activation and Deactivation	21
4. Example of Incorporating Multimedia Elements	22
5. Example of Glossary Addition to LMS	22
6. Example of Search Feature in LMS	23
7. Example of Learning Path in LMS	24
8. Example of Mobile LMS Support	24
9. Example of Social Media Integrated to LMS	25
10. Example of Making LMS Data Printable	26

1. Introduction

Advancements in information and communication technology have paved way for educational learning opportunities over the internet. As internet connection speeds keep improving and more and more digital content becomes available, education is becoming accessible anywhere anytime. Better interaction and knowledge acquisition techniques have resulted from the open education provided via internet platforms. Learning management systems developed over the years to deliver courseware, have made e-learning popular. Organizations have also been actively involved in the use of learning management systems to deliver online training to their employees and customers. Traditional classrooms have been transformed due to technological and market forces, which have seen virtual learning gaining a strong foothold in many parts of the world.

A learning management system (LMS) supports learning through a mixture of formal, social and experiential education. Currently, most LMS provide extra functionality, which includes social knowledge components that enable the users to consult peers, raise queries and contribute to content. The LMS market is predicted to grow at a rate of 16% from 2016 to 2020. The two main users of learning management systems are learners and administrators. Learners are the people who receive training through the platform, which enables them to access various learning materials such as course catalogs, course information, and evaluation. On the other hand, administrators are the people who manage the learning management system by creating various courses and putting up tools that enable learners to track their progress. Learning management systems are used in centralizing, deploying and measuring different types of training.

In larger organizations, learning management systems provide remote and unsynchronized learning-based environment on a client-server architecture using a web service. LMS stores information about the student's learning curves by keeping student records and learning data. This functionality enables the system to generate user performance. Most companies prefer to use LMS because it provides training to keep all the courses and learner's data during the sessions. LMS plays a key role in enabling the access and management of content by both users and administrators.

The rapid development of LMSs has highlighted the issue of proper regulation of implementation for the system. For this purpose, the "Shareable Content Object Reference Model (SCORM)" has been developed to provide an outline, define the requirements, and issue an effective implementation plan for the LMS. However, changing the architecture of existing systems to become compatible with SCORM is not an easy task.

Some LMS were developed before the inception of SCORM, and their upgrading can be challenging. Freeware and open-source LMSs may be an option as they provide rich features that are comparable with high values patented LMS [1]. Nonetheless, it is not an easy decision to discard the previous products or upgrade them to the SCORM standard since changes in LMS support can disrupt the course development process creating confusion for both teachers and students. Further, the cost factor cannot be ignored. Issues related to funding may prevent successful implementation of a new learning management system. All the expenses related to acquisition and running of an LMS must be taken into account, including subscription, hosting, and upgrading of the system from time to time [1].

The Learning Management systems business has grown to be a 10-billion-dollar industry. Powerful software, which has enabled the management of complex databases, has been

combined with great digital frameworks so that curriculums, training platforms, and evaluation tools are managed effectively. This feature has enabled most organizations to expand and achieve their learning goals. Self-learning has also been enhanced hence enabling people to multi-task since acquiring education has become a very flexible process due to the availability of these systems.

The main objective of this paper is to create a new LMS named LearningMadeEasy4u (LME4u), which satisfies the requirements of an LMS. To develop the requirements for LME4u, we need to provide an overview of existing LMSs.

1.1 Basic Types of LMS

Cost, stability and security are key considerations in choosing a LMS. Based on market use, the following are the three categories of LMS:

- Proprietary LMS
- Standards-based LMS
- Open architecture LMS

1.1.1 Proprietary LMS

Proprietary LMS can be defined as systems, which are developed in-house then sold for a fee for use by institutes. Currently, these kinds of LMS are at the end of their existence. The systems were developed by considering the computer-based training (CBT) systems to provide information. Before the concept of e-learning and SCORM based systems, proprietary LMS was used widely in the market. Clients who have purchased these kinds of systems say that the investment versus the cost of maintenance is in the ratio of 1:3, which implies that for each dollar of cost for purchasing such a system, extra three dollars are required for its maintenance [2]. This fact proves to be a major disadvantage of Proprietary LMS. Other disadvantages of a Proprietary

LMS are the additional costs incurred in distributing it to the students and the difficulty of customizing the system to fit user needs.

Another disadvantage of propriety LMS is its somewhat restrictive nature. In other words, it does not give users sufficient chance to test the new features of the proprietary LMS. In addition, source code is also not readily available for clients; hence customizing the learning portal becomes an impossible task. Proprietary LMS were however advantageous since they are reliable and enables the creation of training modules and supported learning processes in an efficient manner. In addition, they offered accountability for the way the system functioned.

1.1.2 Standards-Based LMS

Standards-Based LMS functions based on the SCORM standard. This sort of learning management system is rapidly developing. The biggest names in the field of proprietary LMS are now shifting toward standards-based learning management system to remain an integral part of the market. This adaptability of standards has paved the way for the incorporation of other components and better interoperability level for LMS. Moreover, it has changed the e-learning contents as well since most organizations today use SCORM. However, the issue is that SCORM is continually changing due to innovative progress and new outcomes [1]. One of the examples of these kinds of LMS is Docebo, which is based on SCORM and XAPI standards. The importance of SCORM is based in the fact that successful learning is reliant on the ability of an LMS to read and run various reports on course content. Learning management systems cannot achieve anything if they do not have quality content and a platform to distribute the same. Therefore, SCORN is an interface that enables this objective to be achieved as it integrates the learning platforms and course content hence enabling them to work together.

1.1.3 Open Architecture LMS

Proprietary LMS maintains a closed kind of approach. In other words, it does not offer an effective way to connect with people neither does it enable the use of sharing content or social interaction. Numerous developments of the LMS frameworks have guaranteed that their framework is “really open”. However, as we start working with those LMS frequently, they turn out not to be really open. Open here means that these kinds of LMS would promote open interaction, and open connection. New versions of LMS are mostly built on the foundation of openness. [3] Majority of such LMS have a free basic package and can be used limitlessly. Therefore, adding advanced features to the system would mean that money has to be paid by the organization. The advantage of such a system is that it is customizable and easy to grasp its use. The users of such a system are able to access source code, hence personalize their own learning portal, add whatever features they want and fix any bugs. Open systems are mainly built by collaborative communities hence leading to better systems due to the combined knowledge of different people. However, the disadvantage is that open systems have many costs, which are not obvious to the user at first purchase. Some of these costs are for hosting, system maintenance, system back up, storage and technical support.

2. Features of Learning Management Systems

There are certain features in learning management systems that ensure the highest level of learnability and usability for students and administrators. The following attributes can be defined as the main features of any LMS:

Scalability—this refers to the ability of the computing processes to be used in a wide range of capabilities. An effective LMS should be able to manage a large amount of data and any number of users at a given time. LMS should be scalable to handle the request for new users as well.

Cost Efficient—Cost efficiency is defined as the aggregate estimation of the rate of profitability of the LMS. It also reduces the cost of learning through managing course content, delivering teaching material, taking quizzes, assignments, and assessment.

Ease of Use—A good LMS system should enable the user to have a good user experience when using it. The user should be able to learn how to operate the system in an easily. The interface of the LMS should be efficient to enhance user experience.

Integration—A good LMS should have data sharing capabilities so that students are able to gain the most out of it. In an organization, the LMS should be linkable with systems such as talent management, management of workers, human resource information system, and customer relationship management system and compliance platforms. In a university setting, the same should be integrated with registration and admission systems.

Content Management—A good learning management system should have a content management capability. The system should be able to support the latest standards of online learning and link the learners to web content that is hosted on other systems. The system should

also enable learning to be organized into an effective path and materials for learning to be shared between different courses.

Mobile Learning Support—Learning management systems should be supported by a mobile learning system. Students should be able to access content through a tablet or mobile phone. The LMS should therefore be designed in such a way that it has adjustable screens that can fit into different devices. Mobile learning will enhance interaction and users will be able to have better social engagement by participating more in discussion boards.

Blended Learning—LMS should be able to support both online and offline learning. Learners should be able to register themselves or be registered by the administration. Offline learning should be trackable to a learning path.

Testing and Assessment—The LMS should allow for flexible testing and assessment by allowing for pre-tests and post-tests. Test banks should be available on the system and multiple questions should have an ability to be randomized. The assessment portfolios completed by students should be stored in the system for as long as necessary.

Reporting and Tracking—Reports enable learner progress to be tracked. The system should, therefore, be able to provide learning path reports, exam reports and in the case of organizations, e-commerce reports.

Security—The LMS system must be secure so that hackers are not able to access and breach any data. The data must be protected, and authorizations carefully handled. Data integrity has to be maintained and personal data of students is protected by having secure sign-on procedures.

Customization—A good LMS system enables branding to be done by the organization. This helps to communicate well to students who want to enroll in the systems.

E-Commerce—The LMS should have functionalities that support payment gateways that can be used by those enrolling. E-mail notifications should be enabled, and the system should be able to track and report these payments.

Manageability—This is the capacity to keep up to a predictable level of learning the framework and to strengthen it. It requires meeting the ever-increasing requests of students, to meet developing infrastructural needs, and to address the constraints of financing, workforce preparation, and boost and set staff for current and future challenges [4].

Maintenance and Support—Providing proper support and maintenance can help the universities to be more receptive to the development of advanced features for LMS. Maintenance through quick fixes and updating in “day in” and “day out” support for the workforce, staff and students are two contributing elements, which guarantee the success of any LMS.

3. Project Overview

3.1 Project Description

This project starts with a review of existing web-based learning management systems, their abilities, related standards, best-known attributes, and development in accordance with the Shareable Content Object Reference Model (SCORM). After reviewing the features, we selected those that we will incorporate in our learning management system. The project includes the core phases of software development life cycle starting from requirements gathering and going until implementation. The final product is our LME4U which is a learning management system.

3.2 Advantages of Learning Management Systems

- Rapid development has opened the horizon with a worldview in support of web-based learning since it provides remarkable information to people who intend to learn [5].
With the arrival of modern technologies, the internet has offered a platform for online learning and educational source for remote students.
- It has been proved that internet learning has numerous advantages that have upgraded the knowledge of people. One of the surveys has discovered the various potential benefits in electronic and web-based learning using the internet. LMS gives learners an opportunity to learn differently, provide long lasting learning, convenience, flexibility, and many such advantages. Web-based teaching also helps to upgrade the nature of teaching and learning. It gives the learners a superior learning background compared to ordinary routine programs. These online learning systems have uplifted learning ability and showed a significant difference from those who were studying traditionally [6].

- Schools are utilizing LMS as a learning tool to support information exchange between the teachers and learners. LMS provides more insight on learning to the instructor and student as it is able to monitor learning progress. These learning systems have led to tremendous gains as one can plan himself or herself, in terms of the amount of learning they are able to do within a certain period. Educational material contained in the LMS can also be adjusted frequently to represent changes in a curriculum.
- The learning management system provides more opportunities for students and educators to interact effectively as compared to classroom-based teaching method.
- Dropping maintenance charges due to reduced content duplication and very less functional faults and downtime
- Capitalize on productivity via integration of content transformation, lessening unpredictability, and expenses of examination by provision of online exams
- Leveraging existing resources by including defined strategies and methodology

3.3 Disadvantages of LMS

- Certain important components of the material to be learned do not lend themselves to online platforms. Some jobs for example require physical skills that students cannot gain online. In professions like nursing, for example, physical presence of a student is required as the nature of the job is purely physical. Interpersonal communication cannot be acquired through online materials hence affecting the abilities of some students.
- Another drawback is that LMS can give students tunnel vision, meaning that they may not consider any other learning opportunities outside the learning management

system. If they are constantly training at home for example, they may not realize that by interacting and speaking to other students, they may learn more through interaction and asking questions.

- Informal learning is also difficult to incorporate through an LMS. The trainers may not also be able to use different teaching styles. The personality of some good trainers may also not come out through online learning, as they are not directly interacting with the students.
- The pace of advancement of students may also be greatly affected through online learning. The learners may lag behind in the learning path as students may choose not to read the required material on time. Some students may also read course material at a very quick pace while others may prefer to do it slowly hence it becomes a task for the lecturer to synchronize both of these teams without having offline activity. This makes the work of the lecturer quite difficult.
- Teachers have to take in extra work especially in the primary stages when the LMS is launched. The teacher may have to handle a physical classroom while at the same time having online students, yet they have to make sure the learning paths are at a similar stage. Teachers may therefore feel overwhelmed before they find a strategy that works for them to enable effective learning outcomes for all of their students.
- Plagiarism is another problem with LMS. Students are tempted to look up everything in the internet hence affecting quality of work. Fair assessment will be affected.

3.4 Current Issues with LMS

Learning management systems are sometimes used in inadequate ways. Members of management and teachers are using LMS for purposes of adding content on the web without applying some academic criteria. This particular technique is not recommended due to reasons such as unplanned implementation of LMS. Likewise, the teacher's academic aptitude might be a problem.

According to some analysts the LMS is not very flexible in allowing the use of constructive learning procedures. Some LMS may repeat the normal face-to-face guidelines. The Staff who offers education online, wants to utilize specialized options that are more like conventional methods of instructing [7]. There is a consensus among some part of the academic community towards constructivist learning methods. A portion of the fundamental beliefs of constructivist learning have been documented below:

- Taking part in self-motivated education
- Learning can be multi-dimensional
- Taking part in the right training along with genuine interest

It has been observed that in some cases the web-based learning tools utilizes constructive and problem-solving based technique to deal with learning in a right way [8].

3.5 LMS Trends

LMS has been evolving over the years, and one of the current trends that have been adopted is training through storytelling. This practice is done through the use of technology, and it has helped many institutions to engage with and retain their students. Videos and online games have enabled this to take effect, hence capturing the required attention of students and enhancing

their learning experience. Another trend that has started taking effect is micro learning whose aim is to reduce the amount of information that students take in, as some of it may not be necessary. LMS are slowly beginning to adopt micro learning by breaking down modules into smaller pieces. This has been made better through use of videos or activities. Due to this, students can remember a lot of information hence reducing mental burnout. Artificial Intelligence is on the rise and enables many data inputs to be accepted hence helping students to improve their learning skills. Combining it with LMS enhances learning experiences as ingenious ways of delivering content are discovered. Virtual reality has made LMS very attractive to students. The current versions make users view, listen and interact using keyboards and controllers. Students can simulate real situations in safe and controlled environments. This leads to better training for students. Social media has also taken the LMS scene by storm. Current systems are integrating with social media profiles of students hence leading to better interaction and engagement. Through this, tasks like group work have become easy, and students can form online study groups. There is a greater focus on enhancing user and administrator experience. More and more LMS are focusing on efficiency, speed, relevance, and usability. Many organizations want to satisfy the needs of a rapidly changing student population. They are therefore borrowing from the best features of the most popular systems. Use of analytics is on the rise. There is a great shift from marketing courseware so that consumption of content is increased, to using analytical data and intelligence that shows the needs of the learner hence creating better content that directly targets a learner. Better learning initiatives are being developed as learner needs are understood in a better manner.

3.6 Features of Other LMS that Inspired Development of LME4U

The focus of an LMS is to ensure efficient delivery of content to students, keep track of their learning paths and performance while ensuring effective content management. Content management maximizes one's investment hence allowing effective reuse and content delivery. The management of content will ensure that the LMS files are stored and organized well and there is version-controlled access to data. A number of LMS therefore have the following features that lead to effective content management [9].

Course Copying—Various LMS include the option of backing up all courses and using back-ups to create other copies. Individual parts of a module should also have the ability of being exported or imported into other courses. One should also be able to export quizzes into other courses [10].

The screenshot shows a two-step process for copying course materials. Step 1, 'Select Copy Type', has a dropdown menu set to 'Copy Course Materials into an Existing Course'. Step 2, 'Select Copy Options', includes a 'Destination Course ID' field with the value '201310_MATH_120_01' and a 'Browse...' button. Below this is a 'Select Course Materials' section with 'Select All' and 'Unselect All' buttons. A list of materials with checkboxes is shown: Content Areas, Syllabus, Online Homework, Glossary, Grade Center Columns and Settings, and Group Settings. At the bottom right are 'Cancel' and 'Submit' buttons. Red arrows and numbers 1 through 4 point to the 'Browse...' button, the 'Select All' button, the 'Syllabus' checkbox, and the 'Submit' button, respectively. Red text annotations include '2: (Choose course in pop-up window)' and '3: Select what to copy (or Select All)'.

Figure 1: Course Copying [11]

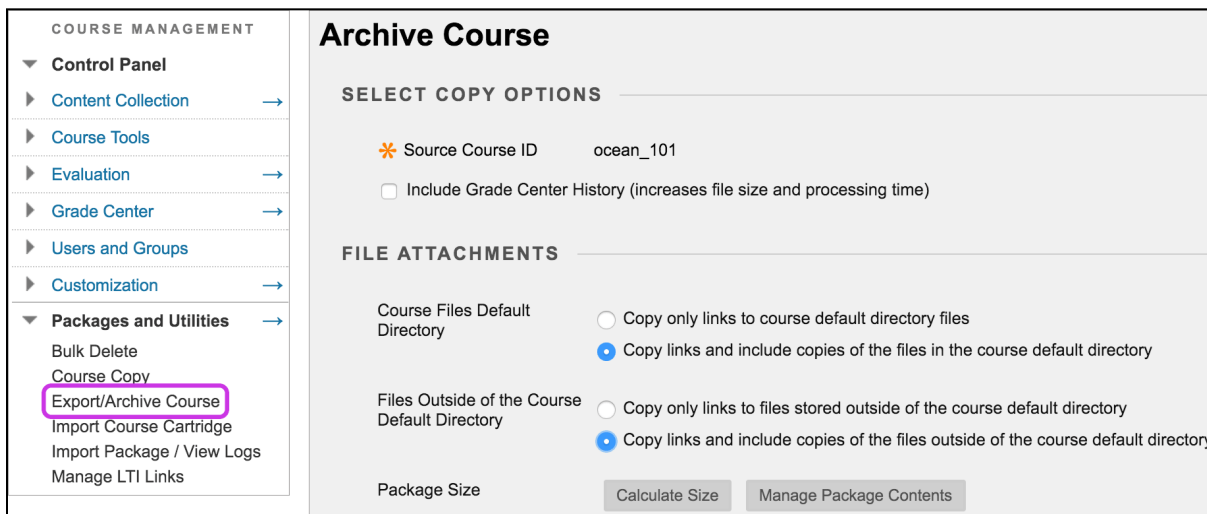


Figure 2: Example of Course Archiving [12]

Archiving and account deactivation—Some LMS have features, which allow archiving and deactivation of any courses, which are outdated so that learners are unable to view them. Administrators are able to view and edit them as necessary or reactivate them.

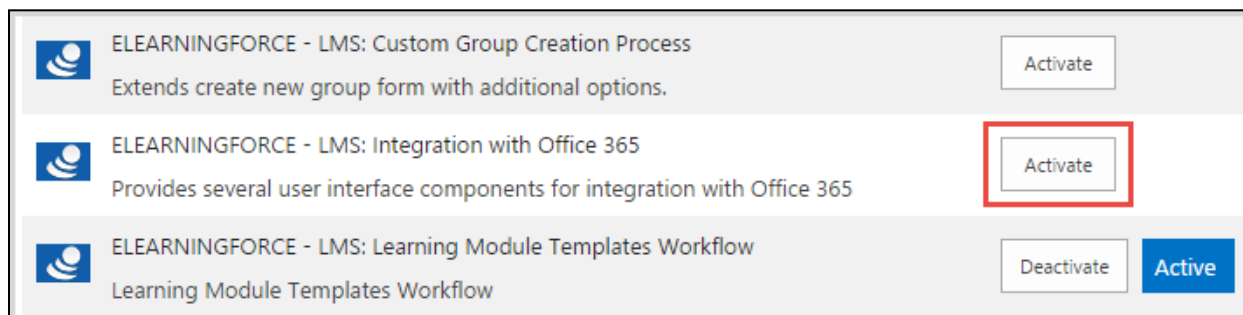


Figure 3: Example of Activation and Deactivation [13]

Incorporating Multimedia Elements—A number of LMS are able to embed multimedia elements such as videos, audios or any other file formats. They also have an inbuilt video and audio player to support audios and videos.

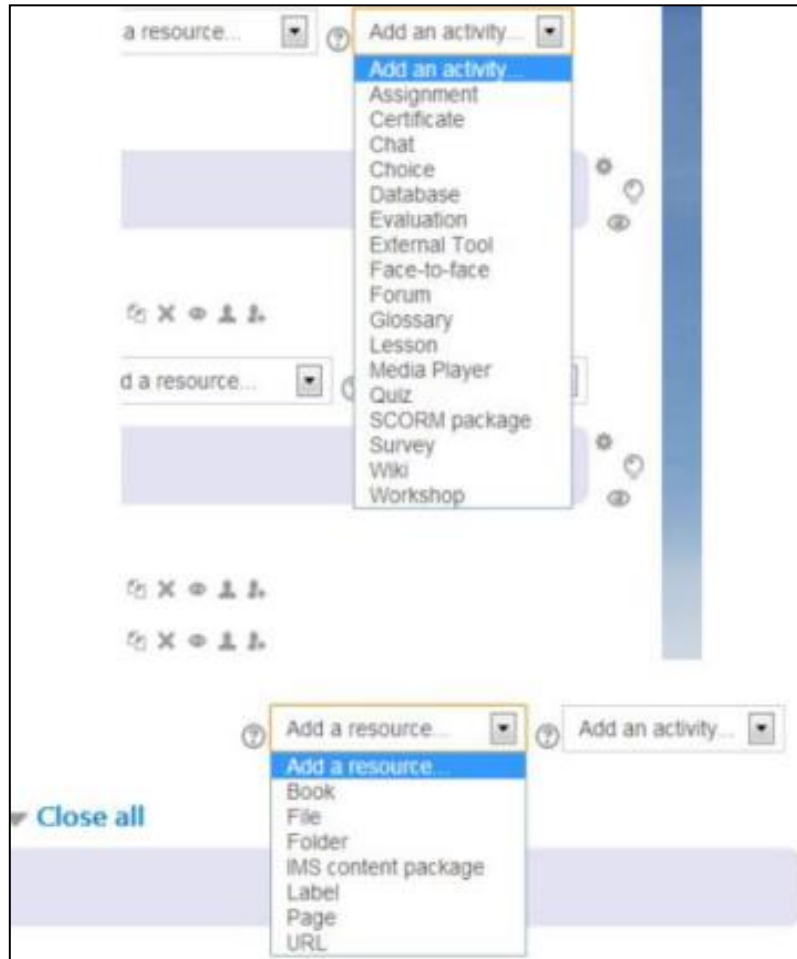


Figure 4: Example of Incorporating Multimedia Elements [14]

Glossary—A number of LMS have a glossary that assists users to understand any terms that are technical. This glossary can be linked to different content.



Figure 5: Example of Glossary Addition to LMS [15]

Search Feature–LMS are enabled with search systems so that students can find learning objects that they are looking for. Field users can be searched using username, email ID or user profile.

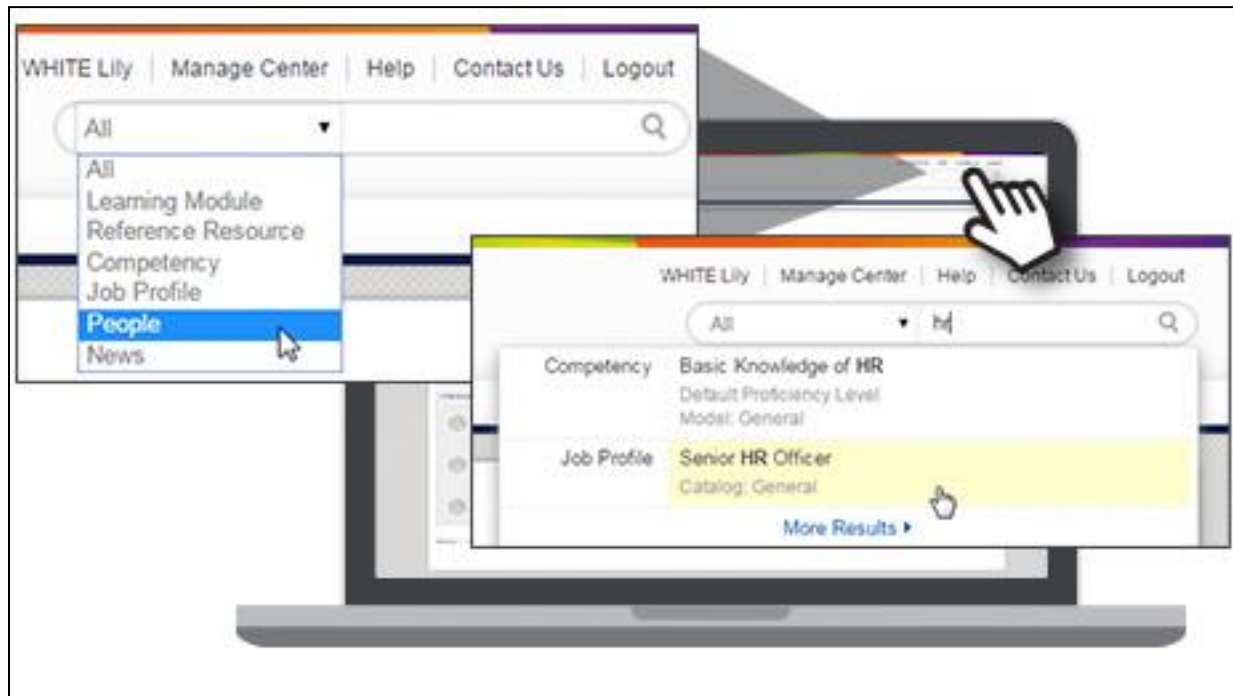


Figure 6: Example of Search Feature in LMS [16]

Learning paths have been customized –Various LMS contain learning paths which can be customized by adding content like videos , courses, evaluation courses and any other material related to learning.



Figure 7: Example of Learning Path in LMS [17]

Mobile device support—Some LMS deliver content to mobile devices. Layout changes should be possible.

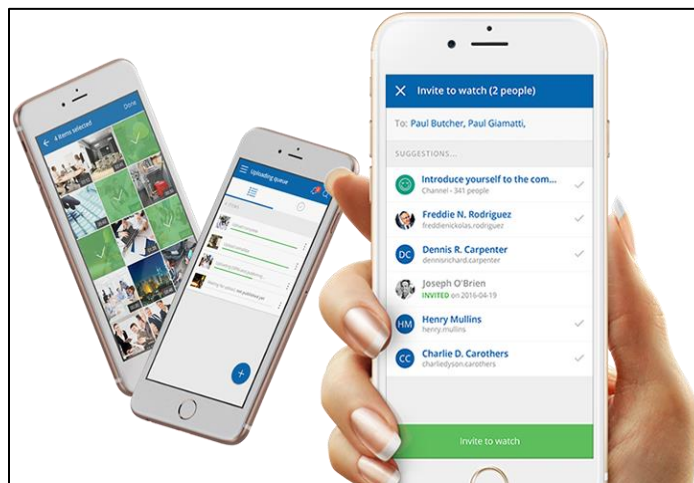


Figure 8: Example of Mobile LMS Support [18]

Social Elements Integration—Various LMS integrate social elements through social media platforms like Facebook, linked in and twitter. This allows for effective collaboration in learning.

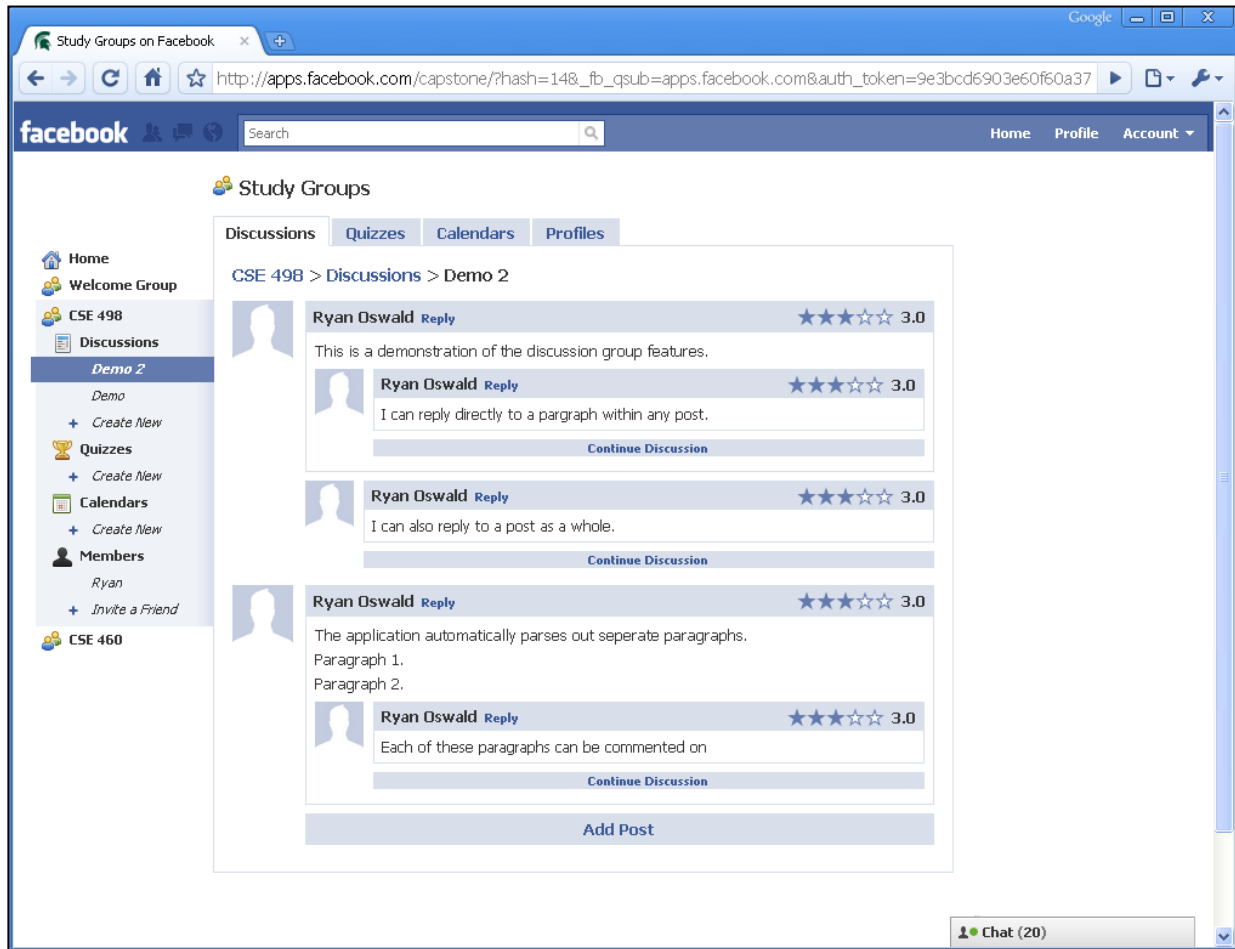


Figure 9: Example of Social Media Integrated to LMS [19]

Learning Content should be Printable—An LMS should allow for download and printing of copies of learning modules and tasks.

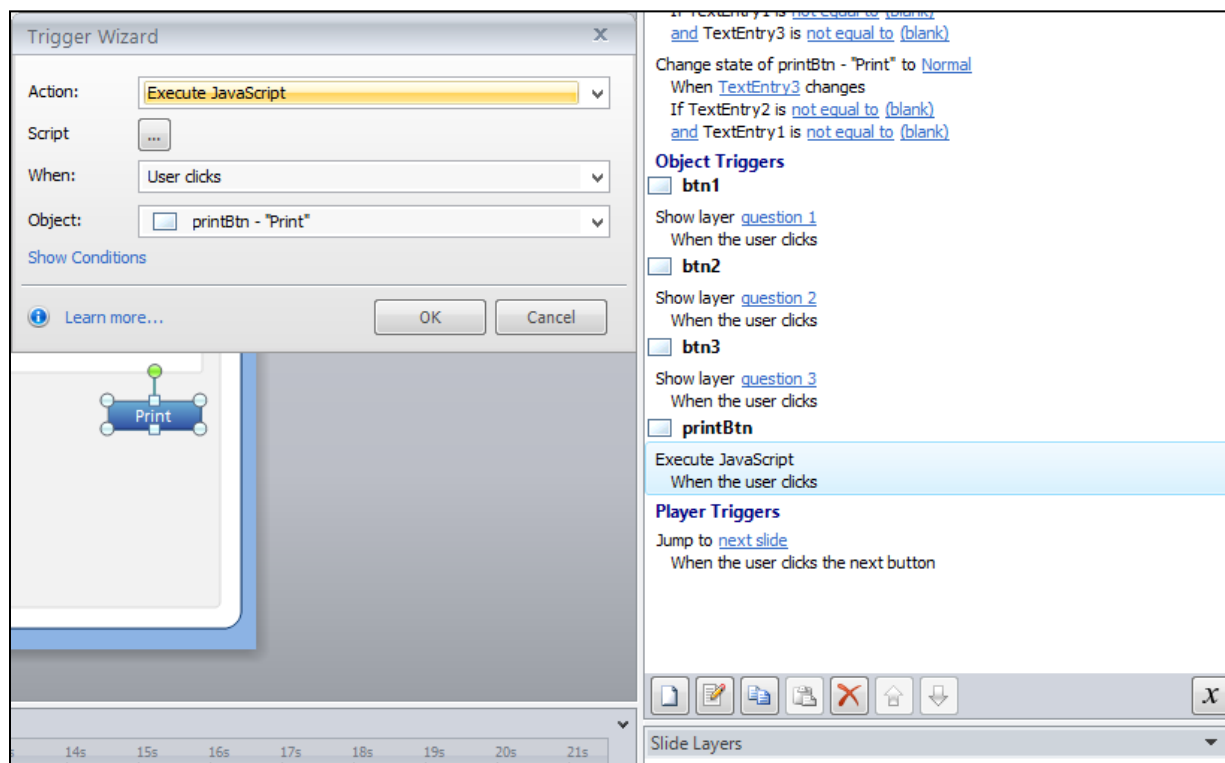


Figure 10: Example of Making LMS Data Printable [20]

3.7 Security Features of Other LMS Systems

As LME4U takes effect, various improvements can be made with time. The security features of the system should be enhanced to ensure that information is secure. Methods of securing the system should, therefore, be analyzed so that the risk of hacking is reduced. Users, for example, should be encouraged to use complex passwords as per IT security policies.

Forums can be added to this system. Through this, there will be interactive learning as students taking one similar course can interact in one forum. Discussions make learning more exciting and lead to better learning outcomes. The other improvement, which can be made, is the integration of LME4U with social media for better learning experiences. This will assist

instructors to reach students from anywhere in a more comfortable way, and students will also be able to interact with each other and learn from each other [21].

Notifications, messages, and alerts should also be incorporated into the system.

Whenever a student registers for a course, takes a quiz or has anything upcoming, the system should be able to notify them through notifications on email or mobile phone. This will ensure that learning paths are followed as required and quizzes.

LME4U should support blended learning. The system should, therefore, recommend offline knowledge, which should have a learning path. The offline education should be combined with e learning so that they follow the same learning path. There should be support to document and track offline and external certifications.

Video Conferencing should be included in the system. This will give instructors an easy time when they want to demonstrate some activities or provide clear examples. It will also help to engage the students in a better manner.

4. Design of LME4U

4.1 Features of the Newly Created LMS System

This section lists some of the features of the newly created LME4U system.

4.1.1 File Upload/Download

In the Instructor module, the course facilitator can upload file using Upload File () action method in Instructor Controller. The student module allows the student to download files using Downloads File () and Download () action method. In the future LME4U will have generic upload limits as 600MB. Normal documents like ppt,docx and pdf will have a limit of 200MB.

4.1.2 Messaging

The messaging feature in LME4u allows instructors and students to write messages to each other. This is found in the action method in StudentController and InstructorController = Messages().While sending a message, one has to specify the recipient.

4.1.3 Quiz Generation

The quiz module enables an instructor to generate the quiz against a course using “Quiz Generator” page and there can only be one quiz against one course. The instructor must select the questions from the question bank of that course. The instructor action method is InstructorController = Quiz Generator(), SelectQuestions(). Students are able to take quizzes for courses in which they have enrolled, and results are shown at the end of each quiz. The action method is Action method in StudentController = Quiz(), StartQuiz().

4.1.4 Course Registration

The instructor module allows facilitators to view all courses from the database by clicking on the ‘courses list’, located at the navigation bar. The action method for this feature is

InstructorController = ShowAllCourses(). Courses are manually assigned to the instructors. The instructor is able to view assigned courses by clicking on the “My Courses” in left side bar. The action method is the Action method in InstructorController = ShowMyCourses(). An instructor is able to add new questions with options inside the course and can show the questions of a course. Students are also able to view all courses from the database by clicking on the “Courses List” in navigation bar. All the courses taken by the student can also be viewed by clicking “My Courses” in left side bar. The Student can also take courses the he has not taken already, as a drop-down list is provided on the page to select a course.

4.1.5 Login Information

The system enables both students and facilitators to log into the system using provided passwords and usernames. Each user has a unique username and password, which can be changed later according to the user’s preference. However, a user is only able to sign in to the LME4U upon successful registration.

4.1.6 Database Tables

The databases consist of instructor and student course assignment modules. The tables also contain course information modules needed by students as well as databank of questions that are used for assessment. The database tables also hold users’ information such as name, registration number, passwords and course studied.

Table 1: ‘Comparison of LMS Systems’–Moodle, Talent and LME4U [4]

Features	Moodle	Talent LMS	LME4U
Score	System performance is at 9.1	System performance is at 9.8	System Performance cannot be measured since there are other factors involved.
Client Satisfaction	Client Satisfaction is at 98 %	Client Satisfaction is at 99%	Client Satisfaction cannot be measured since there are other factors involved.
General Description	The learning management system is free and has capabilities of enabling educators to create private websites with vibrant courses that can be accessed at any time and from any location	This learning management system won an award for best LMS in 2016. It is easy and based on cloud. It can create flexible systems for employees, partners, students and customers	The learning management system is free and aims to enhance communication between instructors and students to lead to better learning experiences
Software Details	It is a no-cost system. It allows trainers to customize their own online private hub that is filled with vibrant courses for education at any location and any time. It provides a complete set of tools, which are learner-centric and a collaborative learning environment, which enhance teaching and learning.	This is a system, which is fully customizable to one’s own needs and has simple analytics on the e-learning environment. Has strong support for SCORM & TinCan, selling courses, videoconferences, gamification and user profiles which can be extended. It is built with a responsive design in mind.	The system has full support for SCORM and is customizable.
Business Support	SMEs Large Corporation Freelancers	SMEs Large Corporation Freelancers	Institutional Learning Freelancers
Pricing Info	Moodle is free with no licensing fees and is offered as an open source module under General Public License. This assists business in carrying out projects, while benefiting from cost savings, flexibility and all the other advantages of using Moodle.	Talent LMS gives a free version and eight paid editions, which are competitively priced depending on a business size and need. The free version supports a maximum of 5 users and 10 courses. The LMS provides standard and unlimited plans where they can be billed either monthly or annually.	The new LMS system is free and has no licensing fees.
Available Integration	Moodle can be integrated with a variety of programs to meet the needs of a website.	Talent LMS can be integrated with the following applications: <ul style="list-style-type: none"> • Zapier • Zopim • Mail Chimp • Sales Force 	The New LMS can be integrated with many programs.

Available Devices	Moodle is available on Windows and Android	Talent LMS is available on Windows, Android, Web-based, Mac and iPhone/iPad	The New LMS is Web-based.
Languages Supported	USA, Canada	USA, UK, Canada, China, Germany, India, Japan	Supports USA
Features	<ul style="list-style-type: none"> • All-inclusive Calendar • Course creation in bulk • Easy backup • Tools and activities allowing for collaboration • Easy File Management • Site design and layout is customizable • Detailed Logs and Reporting • External resources can be embedded • User roles and permissions can be managed • Multilingual • Multimedia can be integrated • Progress can be tracked • Notification and alert enabled • Outcomes and rubrics • Assessment of self and peer • Dashboard is personalized • Security updates are regular • Authentication and mass enrollment enabled and secure • Management of plug-in and add-ons is simple • Text editor that is intuitive • Open standards supported 	<ul style="list-style-type: none"> • Courses can be authored • Branding enabled • Custom fields • Has an exam engine • Reporting by administrators • Course catalog can be incorporated • Customized reporting • Data can be imported/exported • User roles are defined • Transcripts are displayed • System allows grading • Individualized plans • Mobile access enabled • Management of Registration • Exchange of files • Testing • Training Metrics • Report of ILT • Homepage customized • Course marketplace • Supports e-commerce • Actions are automated • Notification enabled • Web-conferencing • Multi-structure enabled • Management of Certification 	<ul style="list-style-type: none"> • Easy File Management • Assessment Engine • Course Creation • Layout can be customized • Messaging Function • Course registration • Login functions • Download and Upload of files
Main Competitors	<ul style="list-style-type: none"> • Absorb • Accord • Geenio • SkyPrep • eSembler 	<ul style="list-style-type: none"> • Blackboard • Engrade • Accord • School Dude • Brain Cert 	<ul style="list-style-type: none"> • School Dude • Accord • Sakai • Angel • Homegrown

4.2 Design Details

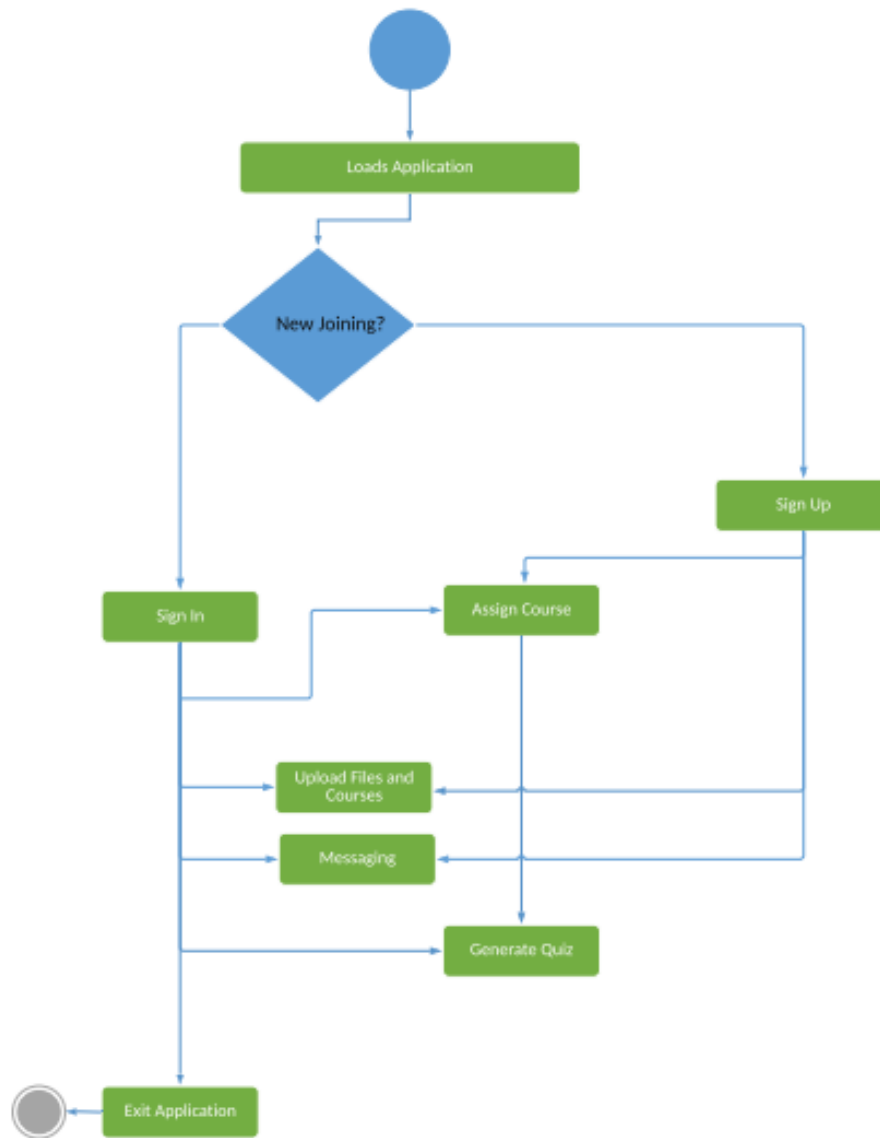
ASP.Net was used to develop LME4U. It provided services, which enabled creation and execution of various web applications for LME4U using server-side technology. The web form controls were used in building the user interface. SQL server is used in database creation for

LME4U. The database stores the structure of the data in form of tables and has other helper objects. The SQL server has various functions that enhance performance of LME4U.

The following shows the use cases for LME4U.

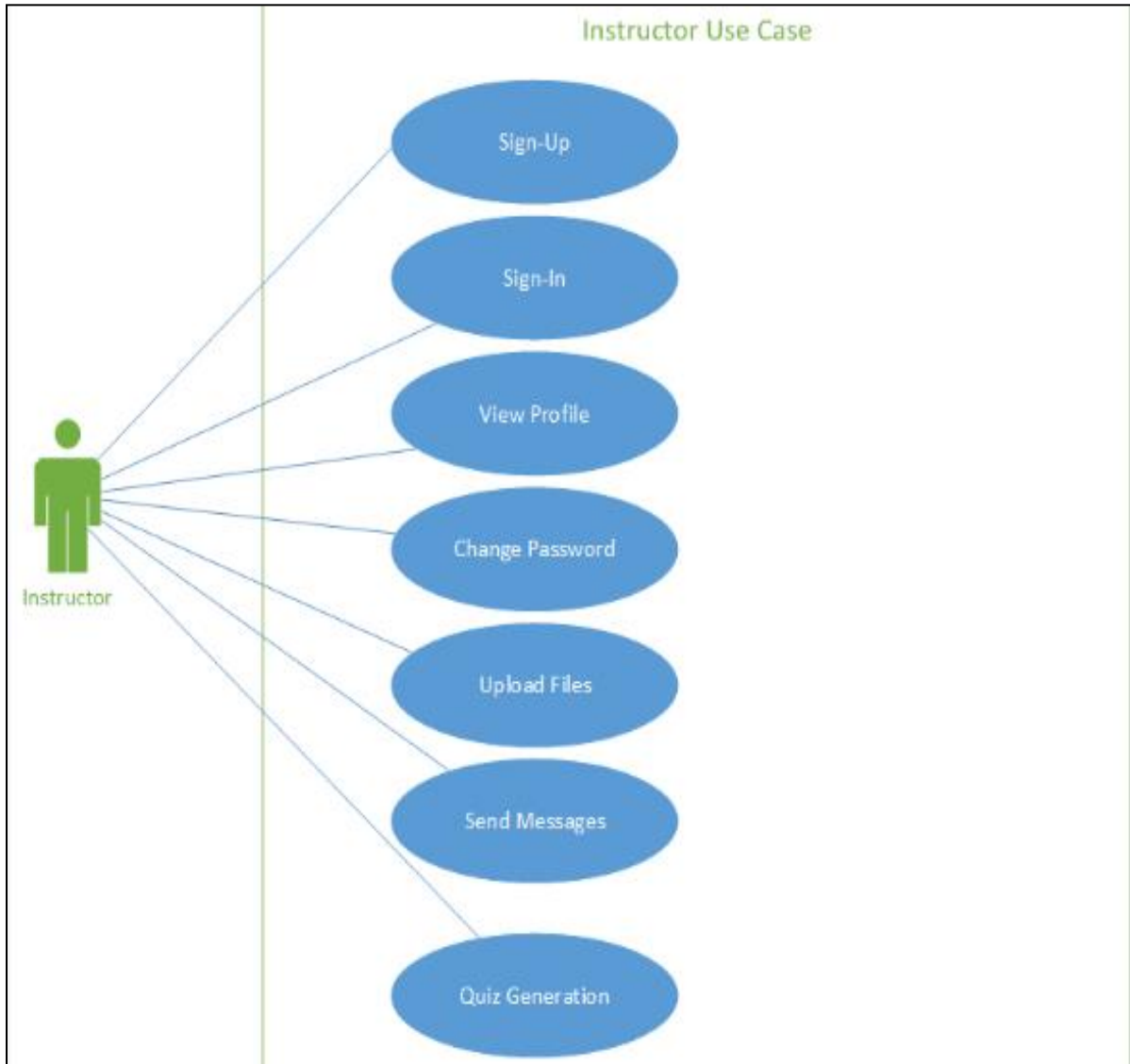
4.2.1 Instructor activity

The instructor activity diagram shows the process that will occur once an instructor logs in into the system. If the instructor is new, he or she will be directed to the sign-up portal. Once this occurs they will need a course that can be assigned to them. Only after assigning the course, the instructor will also be able to access quizzes for those courses. The other modules like Messaging, Upload/Download files, logout can be accessed without assigning a course.



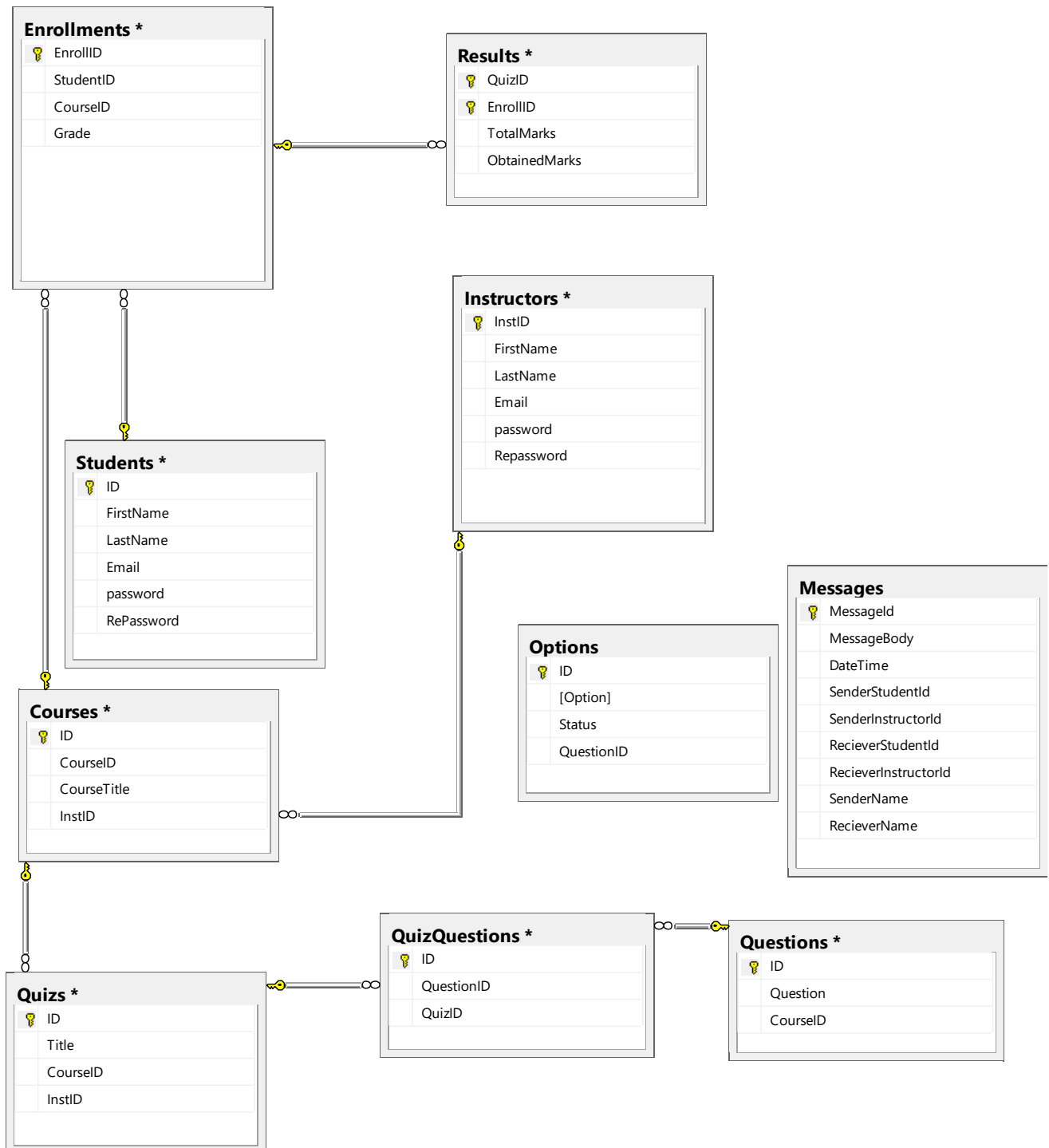
4.2.2 Instructor Use Case

The instructor use case is shown below, and it shows the various options available in the instructor module.



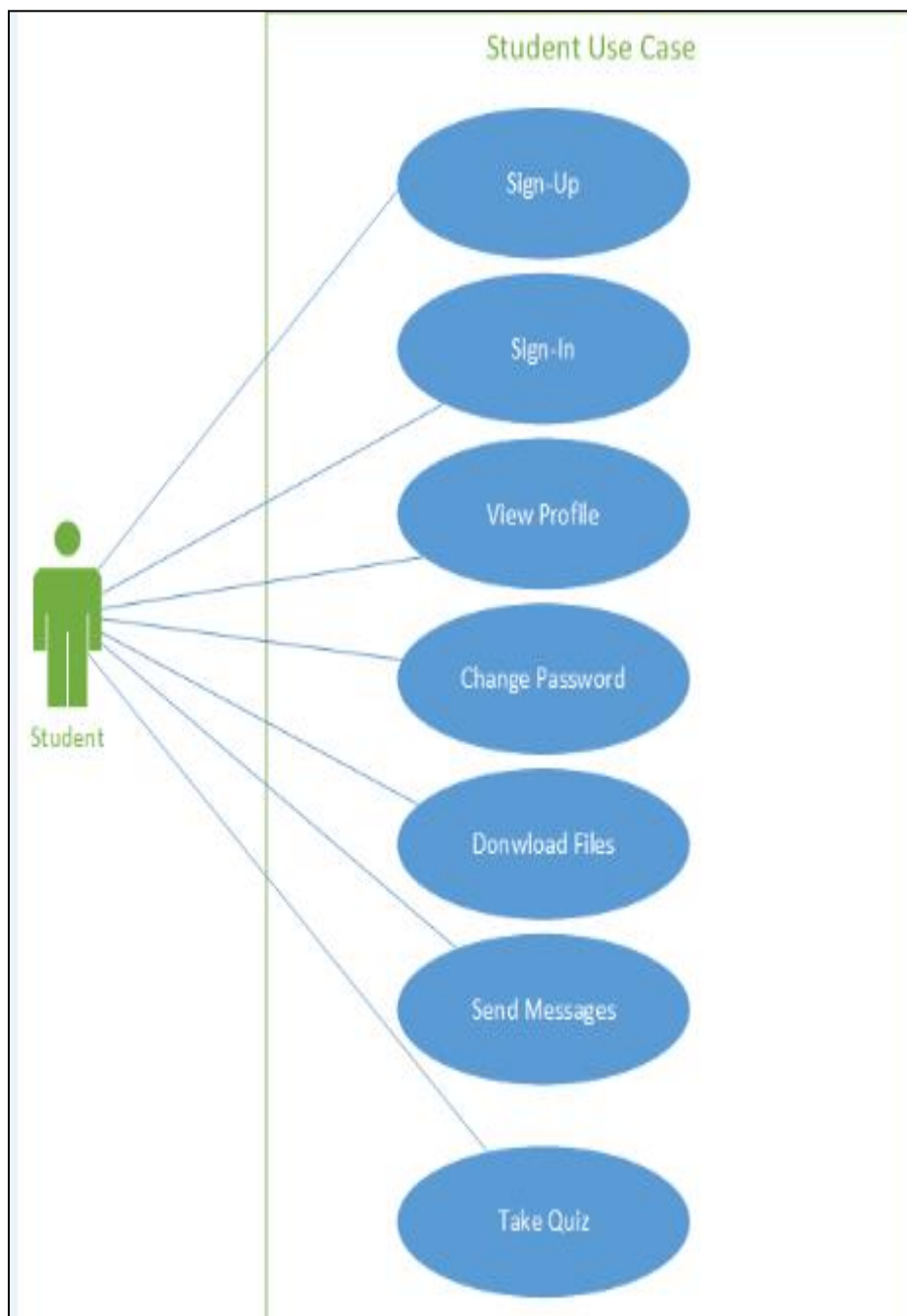
4.2.3 LMS ERD

This diagram represents the tables in LME4U.

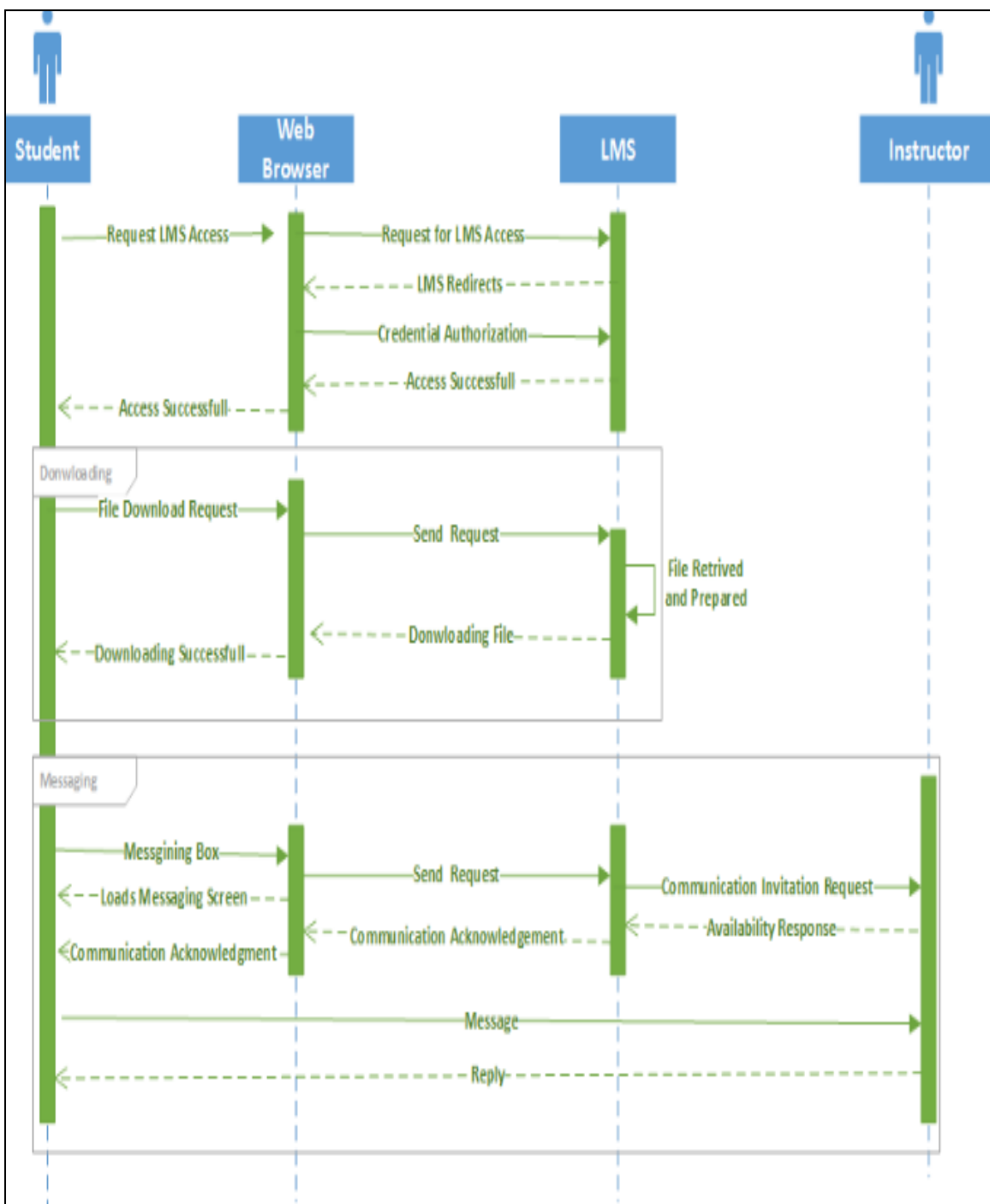


4.2.4 Student Use Case

The student use case contains the options showed in the diagram.

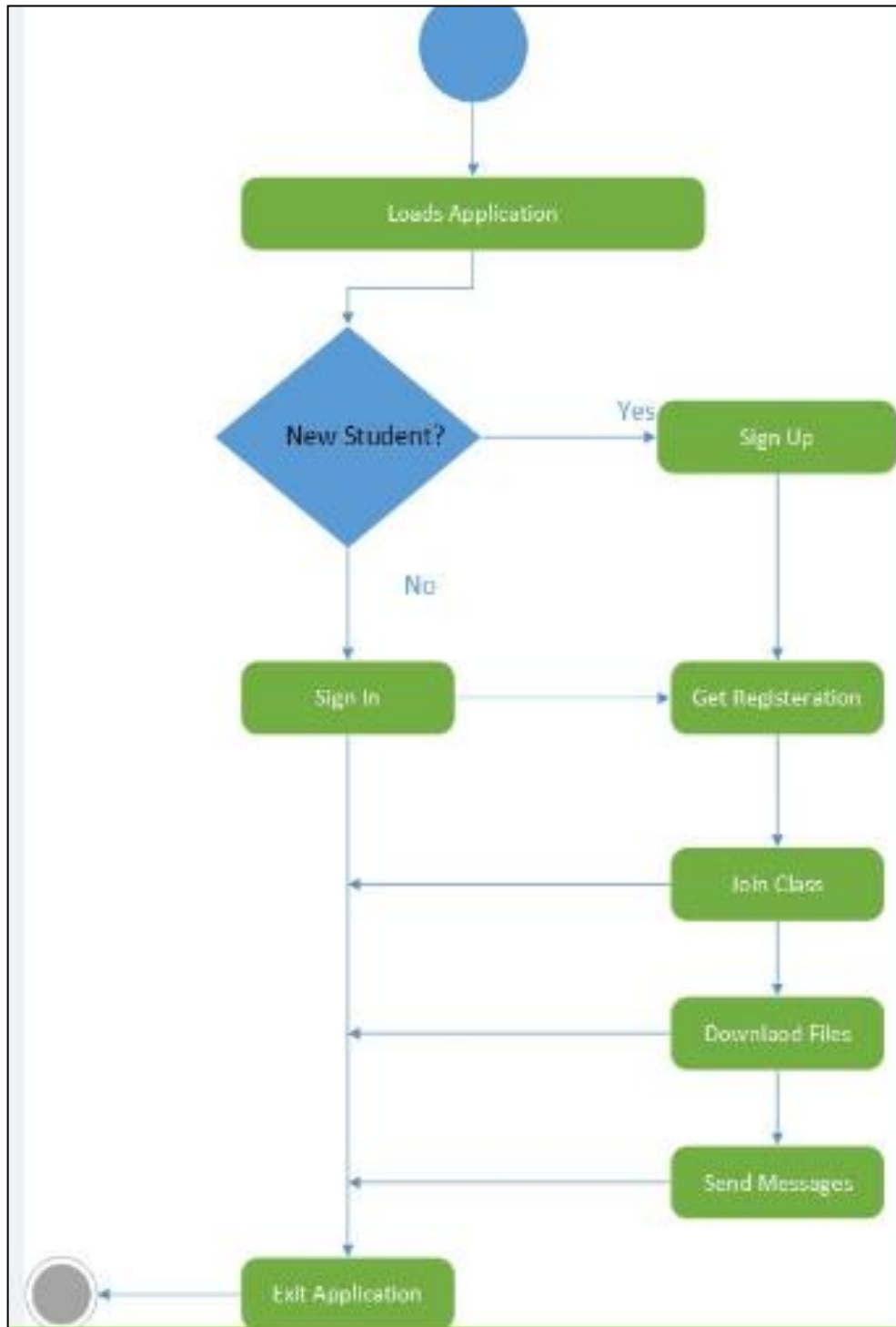


4.2.5 Sequence Diagram



4.2.6 Student Activity Diagram

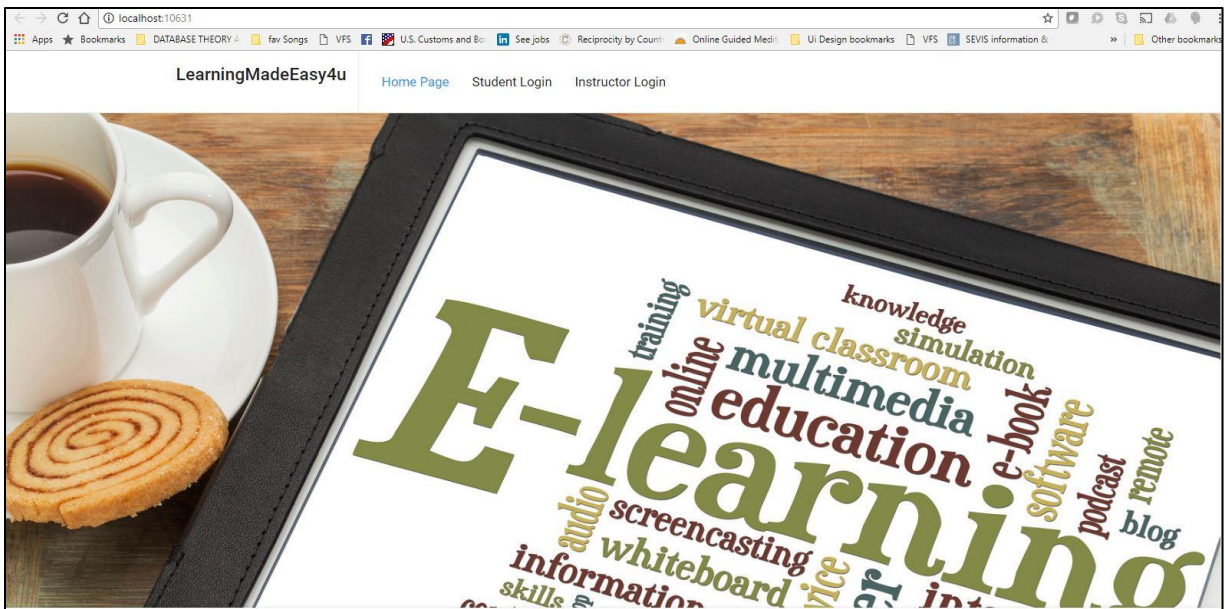
The student activity sequence is shown below.



4.3 Screenshots

Following are the screenshots from the LME4U(LearningMadeEasy4u) showing the different functionalities the application offers.

LMS Home Page–This is the homepage of the LMS – LearningMadeEasy4




Create Account–Account can be created as a student/Instructor.

Login as Student–Login using Email and password.

App By DiptiArora © Copyright 2017

Sign In



aroradipti2016@gmail.com

[LOGIN](#)

[Forgot password?](#)
[Create Account](#)

LMS Dashboard–The Dashboard gives an overall view of the courses and quizzes taken.

LearningMadeEasy4u
Courses ▾ Student ▾
dipti arora ▾

Profileimage
dipti arora

[Dashboard](#)

[Courses](#)

[Account](#)

[Take Quiz](#)

[Quiz Result](#)

[Messages](#)

[Download Files](#)

[Logout](#)

Overview

Today's date is Sun Sep 24 2017 17:43:26 GMT-0500 (Central Daylight Time)

Courses

Your recent courses

DataStructures	<div style="width: 50%; height: 10px; background-color: #76b82a;"></div>
----------------	--

[VIEW ALL](#)

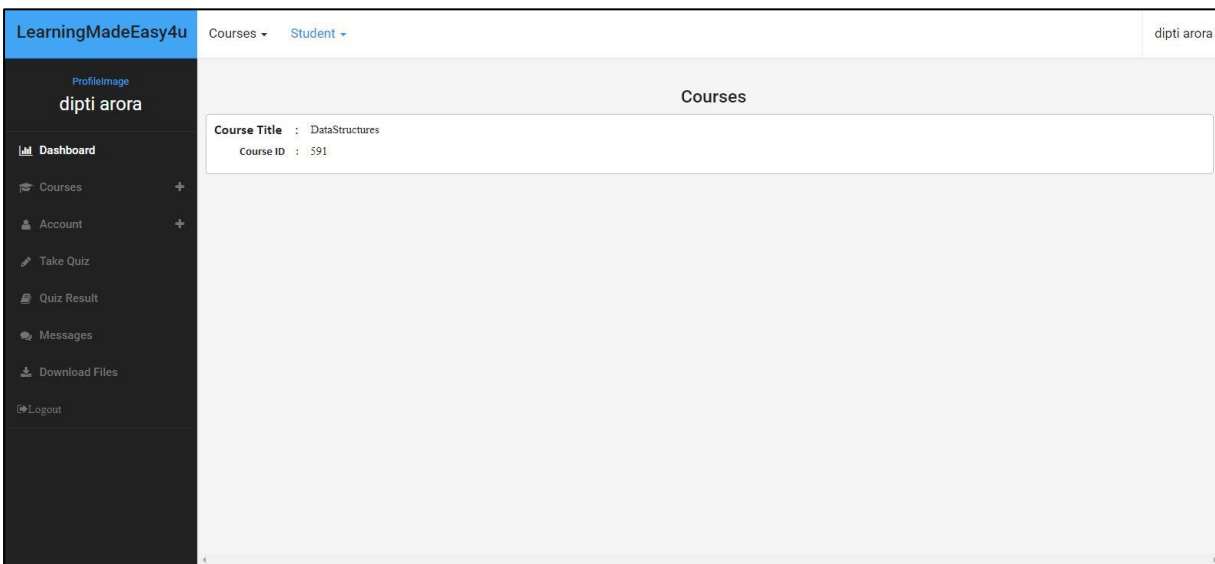
Quizzes

Your recent performance

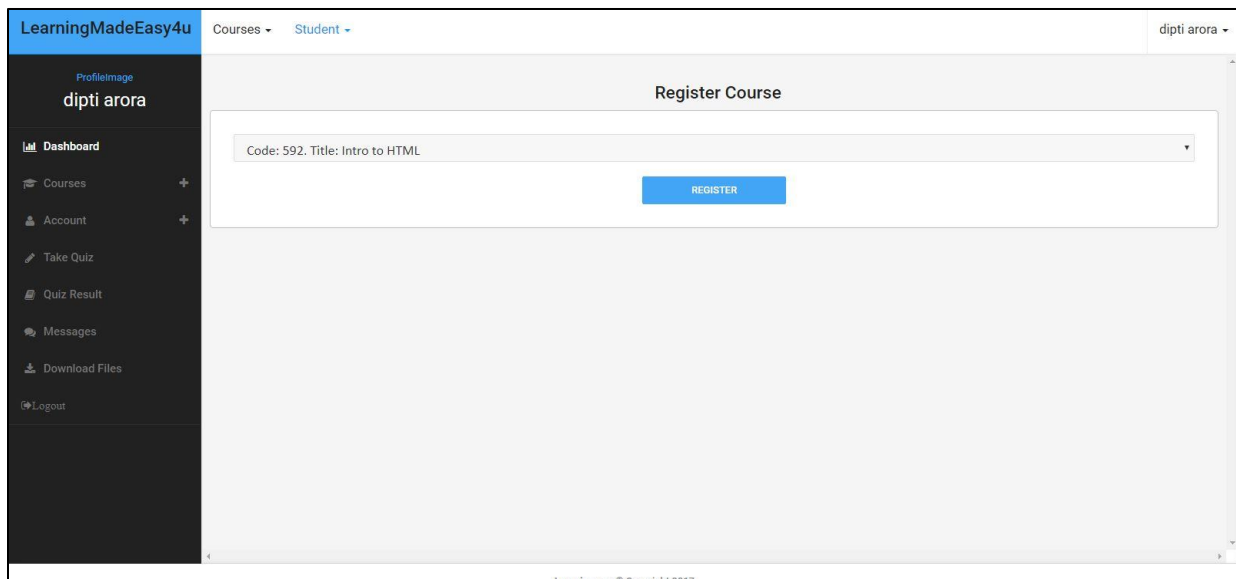
Quiz ID: 1 Course Code: 591	2/2
--------------------------------	-----

[GO TO RESULTS](#)

My Courses–Only the courses the student registered for will show here.



Register Course—A student can register for a course by selecting one of the courses from the dropdown list and clicking on Register.



Student Quiz–A quiz can be assigned by instructor and a student can take a quiz for that course.

The screenshot shows the LearningMadeEasy4u student dashboard. The user is logged in as 'dipti arora'. The dashboard includes a sidebar with navigation options: Dashboard, Courses, Account, Take Quiz, Quiz Result, Messages, Download Files, and Logout. The main content area displays a table of courses with the following data:

Course Code	Quiz Title	Actions
591	introduction to data structures	START QUIZ

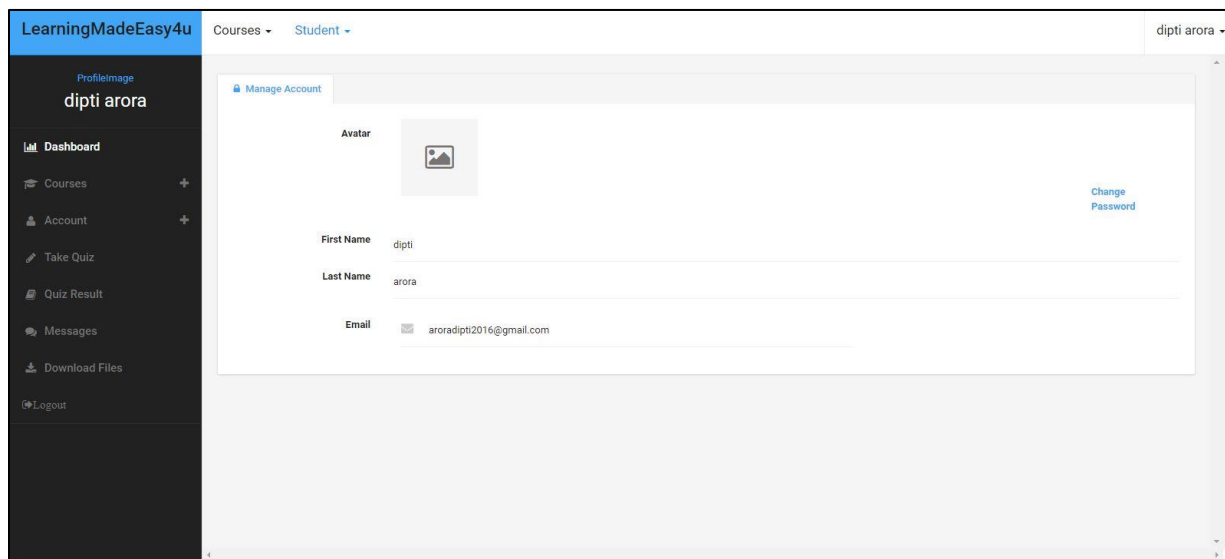
A green button labeled 'BACK TO MY COURSES' is located in the top right corner of the main content area.

Messaging Functionality–Messages can be sent to instructor and students. Messages can also be received from students and instructor.

The screenshot shows the LearningMadeEasy4u messaging interface. The user is logged in as 'dipti arora'. The interface includes a search bar with 'nancy blumer' entered, a 'SEND' button, and a 'Write message...' input field. The message history shows two messages from 'nancy blumer' sent 18 hours ago:

- hello how are you
- Please take quiz

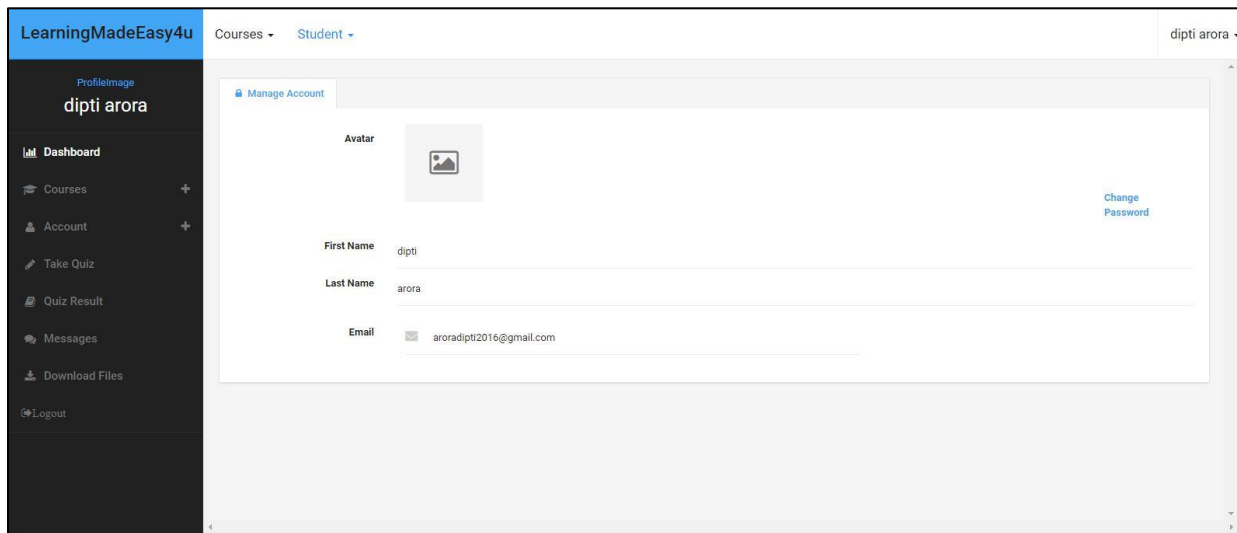
View Profile Student–The profile shows the information about the student.



The screenshot shows the 'LearningMadeEasy4u' interface. On the left is a dark sidebar with the user's name 'dipti arora' and a 'ProfileImage' placeholder. Below the name are navigation links: Dashboard, Courses, Account, Take Quiz, Quiz Result, Messages, Download Files, and Logout. The main content area is titled 'Manage Account' and displays the following information:

- Avatar:** A placeholder image with a camera icon.
- Change Password:** A blue link on the right side.
- First Name:** dipti
- Last Name:** arora
- Email:** aroradipti2016@gmail.com

Change Password–Password can be changed by coming to this screen.



This screenshot is identical to the one above, showing the 'Manage Account' page. The 'Change Password' link is highlighted in blue, indicating it is the active or selected option.

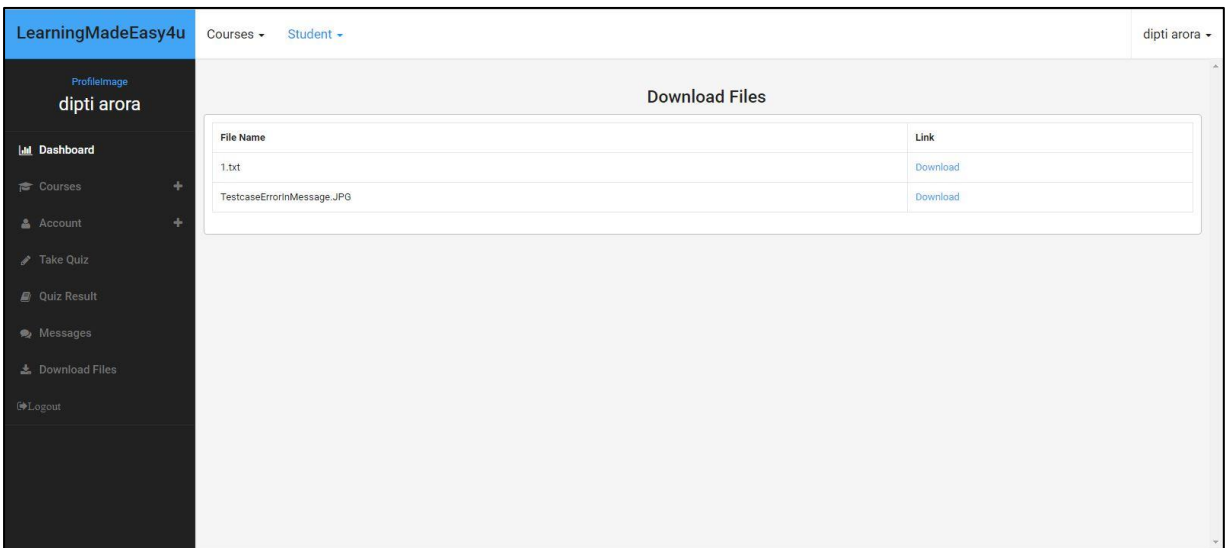
Current Password does not Match–Password needs to match your current password in order to change the password.

The screenshot shows the LearningMadeEasy4u user interface. The top navigation bar includes "Courses" and "Student" dropdown menus, and the user's name "dipti arora" is displayed on the right. The left sidebar contains a "ProfileImage" section with the name "dipti arora" and a "Dashboard" menu with options: Dashboard, Courses, Account, Take Quiz, Quiz Result, Messages, Download Files, and Logout. The main content area features a red error banner at the top that reads "Wrong password!". Below the banner is a form with a single text input field labeled "Enter Current Password" and a blue "SUBMIT" button.

Enter New Password and Confirm Password–New password can be entered here.

The screenshot shows the LearningMadeEasy4u user interface. The top navigation bar includes "Courses" and "Student" dropdown menus, and the user's name "dipti arora" is displayed on the right. The left sidebar contains a "ProfileImage" section with the name "dipti arora" and a "Dashboard" menu with options: Dashboard, Courses, Account, Take Quiz, Quiz Result, Messages, Download Files, and Logout. The main content area features a form with two text input fields: "New Password" and "Confirm Password", both containing masked characters (dots). A blue "SAVE" button is positioned below the "Confirm Password" field.

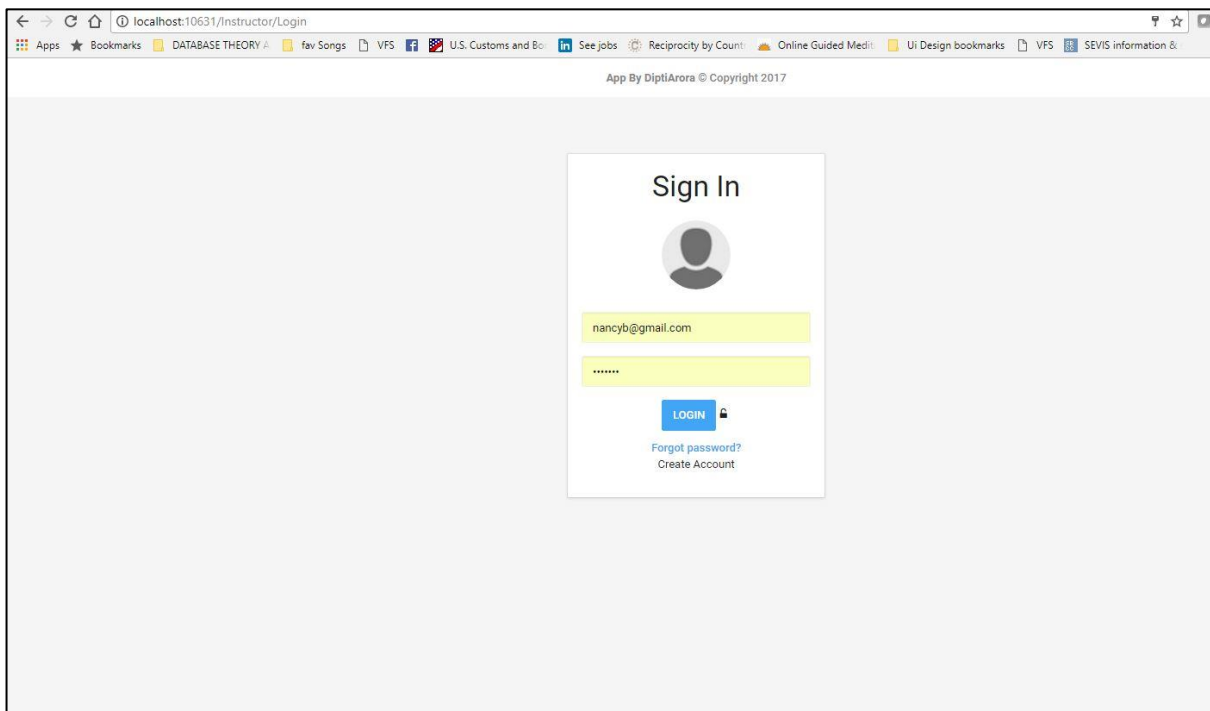
Download Course Files–Students can download all course files related to courses.



The screenshot shows the LearningMadeEasy4u student dashboard. The user is logged in as 'dipti arora'. The dashboard includes a sidebar with navigation options: Dashboard, Courses, Account, Take Quiz, Quiz Result, Messages, Download Files, and Logout. The main content area is titled 'Download Files' and contains a table with the following data:

File Name	Link
1.txt	Download
TestcaseErrorInMessage.JPG	Download

Login as Instructor–Login page for Instructors.



The screenshot shows the Instructor Login page. The browser address bar indicates the URL is localhost:10631/Instructor/Login. The page header reads 'App By DiptiArora © Copyright 2017'. The main content area features a 'Sign In' form with a user icon, a text input field containing 'nancyb@gmail.com', a password input field with masked characters, a blue 'LOGIN' button, and links for 'Forgot password?' and 'Create Account'.

Instructor View Courses.

LearningMadeEasy4u Courses ▾ Instructor ▾ nancy blumer

people
nancy blumer

Dashboard
Courses +
Account +
Quiz +
Messages
Upload File
Logout

Courses

Course Title : DataStructures
Course ID : 591

ADD QUESTIONS SHOW QUESTIONS

Instructor-Add questions to Quiz Generator.

LearningMadeEasy4u Courses ▾ Instructor ▾ nancy blumer

people
nancy blumer

Dashboard
Courses +
Account +
Quiz +
Messages
Upload File
Logout

Add Question

	Question	<input type="text"/>
<input checked="" type="radio"/>	Option 1:	<input type="text"/>
<input type="radio"/>	Option 2:	<input type="text"/>
<input type="radio"/>	Option 3:	<input type="text"/>
<input type="radio"/>	Option 4:	<input type="text"/>

ADD BACK TO QUESTIONS BACK TO MY COURSES

Courses Assigned to Instructor.

LearningMadeEasy4u Courses ▾ Instructor ▾ nancy blumer

people
nancy blumer

Dashboard
Courses +
Account +
Quiz +
Messages
Upload File
Logout

Courses

Course Title : DataStructures
Course ID : 591

Course Title : Intro to HTML
Course ID : 592

Course Title : Enrollment Continuation
Course ID : 691

Instructor-Upload Files–This is used by instructors to upload course files.

The screenshot shows the 'Upload Files' page in the LearningMadeEasy4u system. The interface includes a top navigation bar with 'LearningMadeEasy4u', 'Courses', and 'Instructor' menus, and a user profile 'nancy blumer'. A left sidebar contains navigation options: Dashboard, Courses, Account, Quiz, Messages, Upload File, and Logout. The main content area is titled 'Upload Files' and features an 'upload file:' section with a 'Choose File' button (indicating 'No file chosen') and a 'Submit' button.

Instructor Quiz Generator–Quiz will be generated based on the questions entered.

The screenshot displays the 'Quiz Generator' page. It features the same top navigation and sidebar as the previous interface. The main content area is titled 'Quiz Generator' and includes a red 'Note' stating: 'Only those courses will be shown in drop down list that have question bank !'. Below the note are two input fields: 'Title:' and 'Course:'. The 'Course:' dropdown menu is open, showing 'Code: 591, Title: DataStructures'. A blue 'GENERATE' button is positioned below the input fields.

Instructor–Message.

The screenshot shows the 'Message' page. The top navigation and sidebar are consistent with the previous screenshots. The main content area displays a message input field containing 'dipti arora', a 'SEND' button, and a 'Write message...' placeholder. Below the input field, two message bubbles are visible, both from 'Me' and dated '18 hours ago'. The first bubble contains the text 'hello how are you', and the second bubble contains 'Please take quiz'.

5. Performance Evaluation of the Learning Management System

Evaluation of a learning management system is important so that its performance can be determined. LME4U can be evaluated through these means:-

User Acceptance—the user acceptance test provides an efficient way to determine the usability of the system and evaluate whether the system meets all user requirements. This test can be achieved by explaining and giving surveys or questionnaires to learners to fill in through random sampling in the learner population.

Usability Measures—The usability measures are determined by considering some key features of the system.

1. *Time to learn*—The time to learn can be assessed to evaluate if it is user friendly by analyzing the modules such as shown in Table 2. These are rough estimates which are based off on a single user as a student and Instructor. It is assumed that the instructor can understand a LMS better since they are more exposed to it and have prior experience. The time to learn for an instructor will be less as compared to a student. However, as more and more users grow, these estimates can definitely change as the time to learn for every user could be different.

Table 2: ‘Time to Learn Example’—Measurement of Learning Time

LME4U Module	Student	Instructor
Sign up	180-200 seconds	140-200 seconds
Download File	120 -200 seconds	60-200 seconds

Messages	1200-2000 Seconds	600-1200 Seconds
View Results	600-1200 Seconds	Functionality Not applicable
Upload File	Functionality Not applicable	60-600 seconds

2. Error Rate by users–This will be represented by the number of mistakes that users do when they login to LME4U for the first time an example is loading of file types, which are not supported by the system or putting in wrong login details.
3. Retention Rate–LME4U could be capable of generating reports of student retention rate. It will show the student pass and fail rates. It will also give data on which courses students are more likely to fail or pass. This will assist in future resource planning.
4. User Satisfaction–The satisfaction of users can be measured using online questionnaires, which will be analyzed to determine the user level of satisfaction with the system. The questionnaire will have questions on LME4U general layout and design, LMS functionality, Ease of use, learnability, system usefulness and overall satisfaction.

5. Conclusion

Our overview of learning management systems helped us to identify the software requirements specifications for a new LMS. These requirements include developing a back-end system using ASP.Net and SQL server. ASP.Net has numerous advantages as it enhances performance through early binding, just-in-time functions, caching and native optimization. It also makes it easier for simple tasks like client authentication and submission of forms. We implemented the identified requirements in a new LMS—the LearningMadeEasy4u (LME4u). We described LME4u, its design, and features, in detail. Our LME4u has several advantages. The first advantage is that there is a course module, which enables the instructor and student to view all courses including the ones that have not been taken. There is also a quiz module hence the instructor can generate quizzes and students can see their quiz results. The messaging module enhances communication between the student and instructor while the file upload and download modules enable various file types to be accessed through LME4U.

Finally, we identified the main areas in which LME4u could be improved. Some of these areas included enhancing the security features of LME4U, adding forums, integration with social media and enabling notifications or message alerts. Other improvement suggestions were the inclusion of blended learning, gamification and video conferencing.

References

- [1] S. A. Ahmad, U. B. Chinade, A. M. Gambaki, S. Ibrahim, and N. A. Ala, "The need for MOODLE as a learning management system in Nigerian universities: Digesting University Utara Malaysia Learning Zone as a case study," *Academic Research International*, vol. 2, no. 3, p. 444, 2012.
- [2] M. Gosper, J. McKenzie, J. Pizzica, J. Malfroy, and K. Ashford-Rowe, "Student use of technologies for learning—what has changed since 2010," *ASCILITE*, p. 298, 2014.
- [3] F. Sandnes, H. Jian, S. Hagen, and O. Talberg, "Student evaluation of the learning management system frontier from an HCI perspective," in *International Conference on Engineering Education–ICEE*, 2007.
- [4] K. Fertalj, H. Jerkovic, and N. Hlupic, "Comparison of e-learning management systems," *WSEAS Transactions on Advances in Engineering Education*, vol. 3, no. 9, p. 795, 2006.
- [5] S. L. Wong, A. R. Bakar, A. F. M. Ayub, N. A. Sapari, P. Moses, and M. N. M. Khambari, "Malaysian students' internet use: Some research evidence," in *Workshop Proceedings of the 20th International Conference on Computers in Education*, 2012, pp. 98-104.
- [6] K. Sprague, F. Grijpink, J. Manyika, L. Moodley, B. Chappuis, K. Pattabiraman, and J. Bughin, "Offline and falling behind: Barriers to internet adoption," *McKinsey & Company, Tech. Rep.*, 2014.
- [7] D. Chappell, *Understanding. NET: A tutorial and analysis*. Pearson Education, 2002.
- [8] C. Vrasidas, "Issues of pedagogy and design in e-learning systems," in *Proceedings of the 2004 ACM Symposium on Applied Computing*, pp. 911-915. ACM.
- [9] V. Karakoidas, D. Mitropoulos, P. Louridas, and D. Spinellis, "A type-safe embedding of SQL into Java using the extensible compiler framework J%," *Computer Languages, Systems & Structures*, vol. 41, pp. 1-20, 2015.
- [10] C. Sailer, P. Kiefer, and M. Raubal, "An integrated learning management system for location-based mobile learning," International Association for Development of the Information Society, 2015.
- [11] "Copying courses," *Biola University*, <https://sites.google.com/a/biola.edu/blackboard-news/instructions/copying-courses>.
- [12] "Export and archive courses," *Blackboard help*, https://help.blackboard.com/Learn/Instructor/Course_Content/Reuse_Content/Export_and_Archive_Courses.

- [13] “LMS365 On-Premises 4.8 Administration Guide,” *Elearning Force*, <https://helpcenter.elearningforce.com/hc/en-us/articles/115002434989-LMS365-On-Premises-4-8-Administration-Guide>, 2017.
- [14] “How to embed an iframe in PowerPoint presentation,” *iSpring*, <https://www.ispringsolutions.com/blog/how-to-embed-an-iframe-in-a-powerpoint-presentation/>, 2015.
- [15] P. Long and J. Mott, “The N²GDLE Vision: The ‘Next’ Next Generation Digital Learning Environment,” 2017.
- [16] “The new role of the Learning Management System (LMS),” *NetDimensions*, <http://www.netdimensions.com/talent-management-suite/learning-management-system-lms.php>.
- [17] “Developmental math—an open program: Personalized learning paths,” *Connected PD*, <https://connectedpd.wordpress.com/2012/10/27/163934985/>, 2012.
- [18] “Learn on the go with Docebo Mobile Learning app,” *Docebo*, <https://www.docebo.com/mobile-learning-lms-app-elearning-platform/>.
- [19] “Top ten learning management systems for your company,” *Learning Management Finance Online*, <https://learning-management.financesonline.com/top-10-learning-management-software-solutions-for-your-company>, 2017.
- [20] *How Do I Register and Create Accounts?* Cambridge University Press. <https://www.cambridgelms.org/main/p/ru/frequentlyaskedquestions>.
- [21] F. J. García-Peñalvo and M. Alier Forment, “Learning management system: Evolving from silos to structures,” 2014.

Appendix

Source Code

Model : Based on the MVC. This contains all the source code related to Model.

```
/// Model for courses
```

Course.cs // This is the source code for different courses.

```
using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.Linq;

using System.Web;

namespace E_Learning_Managment_System.Models

{

    public class Course

    {

        [Key]

        public int ID { get; set; }

        public string CourseID { get; set; }

        public string CourseTitle { get; set; }

    }

}
```

```
}  
}
```

Dbconnect.cs

```
/// This is used for database creation
```

```
using System;  
using System.Collections.Generic;  
using System.Data.Entity;  
using System.Linq;  
using System.Web;  
  
namespace E_Learning_Managment_System.Models  
{  
    public class DbConnect : DbContext  
    {  
        public DbConnect() : base("DbConnect")  
        {  
  
        }  
    }  
  
    /// All tables that are going to be included in the database
```

```
public DbSet<Student> Student { get; set; }

public DbSet<Instructor> Instructor { get; set; }

public DbSet<Course> Course { get; set; }

public DbSet<InstructorCourseAssignment> InstructorCourseAssignment { get; set; }

public DbSet<StudentCourseAssignment> StudentCourseAssignment { get; set; }

public DbSet<Quiz> Quiz { get; set; }

public DbSet<Questions> Questions { get; set; }

public DbSet<Options> Options { get; set; }

public DbSet<Result> Result { get; set; }

public DbSet<Messages> Messages { get; set; }

public DbSet<QuizQuestions> QuizQuestions { get; set; }

}

}
```

Instructor.cs

```
/// Model for instructor
```

```
using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.Linq;

using System.Web;
```



```
namespace E_Learning_Managment_System.Models
{
    public class Instructor
    {
        [Key]
        public int ID { get; set; }
        public string FirstName { get; set; }
        public string LastName { get; set; }
        public string Email { get; set; }
        public string password { get; set; }
        public string Repassword { get; set; }
    }
}
```

Instructorcourseassignment.cs

/// Model for storing the courses assigned to all of the instructors

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Web;
```

```
namespace E_Learning_Managment_System.Models
{

    public class InstructorCourseAssignment
    {

        [Key]
        public int ID { get; set; }

        [Required(ErrorMessage = "Instructor ID is Required")]
        public int InstructorID { get; set; }

        [Required(ErrorMessage = "Course ID is Required")]
        public string CourseID { get; set; }
    }
}
```

Messages.cs

```
/// Model for exchanging messages in the chat
```

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
```

```
using System.Web;

namespace E_Learning_Managment_System.Models
{

    public class Messages
    {
        [Key]
        public int MessageId { get; set; }
        public string MessageBody { get; set; }
        public DateTime { get; set; }
        public int? SenderStudentId { get; set; }
        public int? SenderInstructorId { get; set; }
        public int? RecieverStudentId { get; set; }
        public int? RecieverInstructorId { get; set; }
        public string SenderName { get; set; }
        public string RecieverName { get; set; }
    }
}
```

Options.cs

```
/// Model for storing different options in the questions of courses
```

```
using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.Linq;

using System.Web;

namespace E_Learning_Managment_System.Models
{
    public class Options
    {
        [Key]
        public int ID { get; set; }

        [Required(ErrorMessage = "Option is Required !")]
        public string Option { get; set; }

        [Required(ErrorMessage = "Status is Required !")]
        public string Status { get; set; }

        [Required(ErrorMessage = "Question ID is Required !")]
        public int QuestionID { get; set; }
    }
}
```

Questions.cs

/// This is a Model for the questions for different courses

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.Linq;

using System.Web;

namespace E_Learning_Managment_System.Models

{

public class Questions

{

[Key]

public int ID { get; set; }

[Required(ErrorMessage = "Question is Required !")]

public string Question { get; set; }

[Required(ErrorMessage = "Course ID is Required !")]

public string CourseID { get; set; }

```
}  
}
```

Quiz.cs

/// Model for quiz which shows the title of the quiz and that the quiz is of which course

```
using System;  
using System.Collections.Generic;  
using System.ComponentModel.DataAnnotations;  
using System.Linq;  
using System.Web;  
  
namespace E_Learning_Managment_System.Models  
{  
    public class Quiz  
    {  
        [Key]  
        public int ID { get; set; }  
  
        [Required(ErrorMessage = "Title is Required !")]  
        public string Title { get; set; }  
    }  
}
```

```
[Required(ErrorMessage = "Course ID is Required !")]  
publicstring CourseID { get; set; }  
}  
}
```

QuizQuestions.cs

/// This model is used to store the different questions of every quiz

```
using System;  
using System.Collections.Generic;  
using System.ComponentModel.DataAnnotations;  
using System.Linq;  
using System.Web;  
  
namespace E_Learning_Managment_System.Models  
{  
publicclassQuizQuestions  
{  
[Key]  
publicint ID { get; set; }  
  
[Required(ErrorMessage = "Question ID is Required !")]
```

```
public int QuestionID { get; set; }

    [Required(ErrorMessage = "Quiz ID is Required !")]
public int QuizID { get; set; }
}
}
```

Result.cs

/// This is the model for the result generated of the student

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Web;

namespace E_Learning_Managment_System.Models
{
    public class Result
    {
        [Key]
        public int ID { get; set; }
    }
}
```



```
[Required(ErrorMessage = "Quiz ID is Required !")]
public int QuizID { get; set; }

[Required(ErrorMessage = "Student ID is Required !")]
public int StudentID { get; set; }

[Required(ErrorMessage = "Instructor ID is Required !")]
public int InstructorID { get; set; }

[Required(ErrorMessage = "Course ID is Required !")]
public string CourseID { get; set; }

[Required(ErrorMessage = "Total Marks are Required !")]
public int TotalMarks { get; set; }

[Required(ErrorMessage = "Obtained Marks are Required !")]
public int ObtainedMarks { get; set; }

}

}
```

Student.cs

```
/// This is a model for students
```

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Web;

namespace E_Learning_Managment_System.Models
{
    public class Student
    {
        [Key]
        public int ID { get; set; }
        public string FirstName { get; set; }
        public string LastName { get; set; }
        public string Email { get; set; }
        public string password { get; set; }
        public string RePassword { get; set; }
    }
}
```

studentcourseassignment.cs

/// This model is used to store student's courses in which they are registered

```
using System;
```

```
using System.Collections.Generic;
```

```
using System.ComponentModel.DataAnnotations;
```

```
using System.Linq;
```

```
using System.Web;
```

```
namespace E_Learning_Managment_System.Models
```

```
{
```

```
public class StudentCourseAssignment
```

```
{
```

```
    [Key]
```

```
    public int ID { get; set; }
```

```
        [Required(ErrorMessage = "Instructor ID is Required")]
```

```
    public int StudentID { get; set; }
```

```
        [Required(ErrorMessage = "Course ID is Required")]
```

```
    public string CourseID { get; set; }
```

```
    }
```

```
}
```

View

Databaseoperations.cs

```
    /// the function that adds student's record in the database during signup  
using E_Learning_Managment_System.Models;  
using E_Learning_Managment_System.Models.ViewModel;  
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Web;  
  
namespace E_Learning_Managment_System.ViewModel  
{  
    public class DatabaseOperations  
    {  
        /// <param name="std"></param>  
        public void StudentSignUp(Student std)  
        {  
            DbConnect db = new DbConnect();  
            db.Student.Add(std);  
            db.SaveChanges();  
        }  
    }  
}
```

```
/// <param name="inst"></param>
```

```
public void InstructorSignUp(Instructor inst)
```

```
{
```

```
    DbConnect db = new DbConnect();
```

```
    db.Instructor.Add(inst);
```

```
    db.SaveChanges();
```

```
}
```

```
/// the function that gets the student's record from database when his/her profile is opened
```

```
public Student SearchAndReturnStudent(int p)
```

```
{
```

```
    using (var db = new DbConnect())
```

```
    {
```

```
        return db.Student.Find(p);
```

```
    }
```

```
}
```

```
/// this function gets the instructor's record from the database when his/her profile in opened
```

```
public Instructor SearchAndReturnInstructor(int p)
```

```
{
```

```
    using (var db = new DbConnect())
```

```
    {
```

```
        return db.Instructor.Find(p);
```

```

    }
}

/// function used to send name suggestions in messaging module

public List<AutoCompleteViewModel> Autocomplete(string prefix, int userId, string
senderType)
{
    using (var db = new DbConnect())
    {
        if(senderType == "student")
        {
            var Student = (from student in db.Student
                where student.FirstName.StartsWith(prefix)
                && student.ID != userId
                select new AutoCompleteViewModel
                {
                    recieverType = "student",
                    senderType = "student",
                    val = student.ID,
                    label = student.FirstName + " " + student.LastName
                }).ToList();

            var Teacher = (from teacher in db.Instructor

```

```
where teacher.FirstName.StartsWith(prefix)

select new AutoCompleteViewModel
{
    recieverType = "instructor",
    senderType = "student",
    val = teacher.ID,
    label = teacher.FirstName + " " + teacher.LastName
}).ToList();

return Student.Concat(Teacher).ToList();
}

else if(senderType == "instructor")
{
    var Student = (from student in db.Student
        where student.FirstName.StartsWith(prefix)

        select new AutoCompleteViewModel
        {
            recieverType = "student",
            senderType = "instructor",
            val = student.ID,
            label = student.FirstName + " " + student.LastName
```

```

    }).ToList();

    var Teacher = (from teacher in db.Instructor
        where teacher.FirstName.StartsWith(prefix)
        && teacher.ID != userId
        select new AutoCompleteViewModel
        {
            recieverType = "instructor",
            senderType = "instructor",
            val = teacher.ID,
            label = teacher.FirstName + " " + teacher.LastName
        }).ToList();

    return Student.Concat(Teacher).ToList();
}

return null;
}
}

/// save new message in database
public void PostMessage(Messages message)
{
    using (var db = new DbConnect())

```



```

    {
        if (message.MessageId == default(int))
        {
            db.Messages.Add(message);
            db.SaveChanges();
        }
    }
}

/// get all messages with a single person
/// <param name="message"></param>
/// <returns></returns>
public List<Messages> GetMessages(Messages message)
{
    using (var db = new DbConnect())
    {
        if(message.SenderStudentId != null && message.RecieverStudentId != null)
        {
            var a = db.Messages.Where(t => t.SenderStudentId ==
message.SenderStudentId && t.RecieverStudentId ==
message.RecieverStudentId).ToList();

```

```

        var b = db.Messages.Where(t => t.SenderStudentId ==
message.RecieverStudentId && t.RecieverStudentId ==
message.SenderStudentId).ToList();

        var c = a.Concat(b).OrderByDescending(t => t.DateTime).ToList();

        return c;
    }

    else if (message.SenderStudentId != null && message.RecieverInstructorId !=
null)
    {
        var a = db.Messages.Where(t => t.SenderStudentId ==
message.SenderStudentId && t.RecieverInstructorId ==
message.RecieverInstructorId).ToList();

        var b = db.Messages.Where(t => t.SenderInstructorId ==
message.RecieverInstructorId && t.RecieverStudentId ==
message.SenderStudentId).ToList();

        var c = a.Concat(b).OrderByDescending(t => t.DateTime).ToList();

        return c;
    }

    else if (message.SenderInstructorId != null && message.RecieverInstructorId
!= null)
    {

```

```

        var a = db.Messages.Where(t => t.SenderInstructorId ==
message.SenderInstructorId && t.RecieverInstructorId ==
message.RecieverInstructorId).ToList();

        var b = db.Messages.Where(t => t.SenderInstructorId ==
message.RecieverInstructorId && t.RecieverInstructorId ==
message.SenderInstructorId).ToList();

        var c = a.Concat(b).OrderByDescending(t => t.DateTime).ToList();

        return c;
    }

    else if (message.SenderInstructorId != null && message.RecieverStudentId !=
null)
    {
        var a = db.Messages.Where(t => t.SenderInstructorId ==
message.SenderInstructorId && t.RecieverStudentId ==
message.RecieverStudentId).ToList();

        var b = db.Messages.Where(t => t.SenderStudentId ==
message.RecieverStudentId && t.RecieverInstructorId ==
message.SenderInstructorId).ToList();

        var c = a.Concat(b).OrderByDescending(t => t.DateTime).ToList();

        return c;
    }

    return null;
}

```

```

    }

    /// get all latest messages in the chat
    public List<Messages> GetAllMessages(Messages message)
    {
        using (var db = new DbConnect())
        {
            if (message.SenderStudentId != null && message.RecieverStudentId != null)
            {
                var myId = message.SenderStudentId;

                var meSending = db.Messages.Where(x => x.SenderStudentId ==
myId).ToList();

                var sendingMsgToStudent = meSending.Where(x => x.RecieverStudentId !=
null).GroupBy(x => x.RecieverStudentId).Select(x => x.Last());

                var sendingMsgToInstructor = meSending.Where(x =>
x.RecieverInstructorId != null).GroupBy(x => x.RecieverInstructorId).Select(x =>
x.Last());

                var meReceiving = db.Messages.Where(x => x.RecieverStudentId ==
myId).ToList();

                var receivingMsgFromStudent = meReceiving.Where(x =>
x.SenderStudentId != null).GroupBy(x => x.SenderStudentId).Select(x => x.Last());

```

```

var recievingMsgFromInstructor = meRecieving.Where(x =>
x.SenderInstructorId != null).GroupBy(x => x.SenderInstructorId).Select(x => x.Last());

```

```

var chatWithStudent =
sendingMsgToStudent.Concat(recievingMsgFromStudent).OrderBy(t =>
t.DateTime).ToList();

```

```

var chatWithInstructor =
sendingMsgToInstructor.Concat(recievingMsgFromInstructor).OrderBy(t =>
t.DateTime).ToList();

```

```
List<Messages> deleteRows = new List<Messages>();
```

```
foreach(var s in chatWithStudent)
```

```
{
```

```
if(myId != s.RecieverStudentId)
```

```
{
```

```
var p = s.RecieverStudentId;
```

```

var twoRows = chatWithStudent.Where(x => x.SenderStudentId == p ||
x.RecieverStudentId == p);

```

```
int i = 1;
```

```
var firstRow = new Messages();
```

```
var secondRow = new Messages();  
foreach (var t in twoRows)  
{  
    if (i == 1)  
    {  
        firstRow = t;  
    }  
    else if (i == 2)  
    {  
        secondRow = t;  
    }  
    i++;  
}  
if (firstRow != null && secondRow != null)  
{  
    if (firstRow.DateTime < secondRow.DateTime)  
    {  
        deleteRows.Add(firstRow);  
    }  
    else if (firstRow.DateTime > secondRow.DateTime)  
    {  
        deleteRows.Add(secondRow);  
    }  
}
```

```

    }
}
}

foreach (var s in chatWithInstructor)
{
    var p = 0;
    if (s.SenderInstructorId != null)
    {
        p = s.SenderInstructorId ?? default(int);
    }
    else if(s.RecieverInstructorId != null)
    {
        p = s.RecieverInstructorId ?? default(int);
    }

    var twoRows = chatWithInstructor.Where(x => x.SenderInstructorId == p
|| x.RecieverInstructorId == p);

    int i = 1;

    var firstRow = new Messages();
    var secondRow = new Messages();
    foreach (var t in twoRows)
    {

```

```
if (i == 1)
{
    firstRow = t;
}
else if (i == 2)
{
    secondRow = t;
}
i++;
}
if (firstRow != null && secondRow != null)
{
    if (firstRow.DateTime > secondRow.DateTime)
    {
        deleteRows.Add(secondRow);
    }
    else if (firstRow.DateTime < secondRow.DateTime)
    {
        deleteRows.Add(firstRow);
    }
}
}
```



```

        var allLatestMessages =
chatWithStudent.Concat(chatWithInstructor).ToList();

        foreach (var d in deleteRows)
        {
            try
            {
                var del = allLatestMessages.First(x => x.MessageId == d.MessageId);
                allLatestMessages.Remove(del);
                db.SaveChanges();
            }
            catch{ }
        }

        return allLatestMessages.OrderByDescending(t => t.DateTime).ToList();
    }

    else if (message.SenderInstructorId != null && message.RecieverInstructorId
!= null)
    {
        var myId = message.SenderInstructorId;

        var meSending = db.Messages.Where(x => x.SenderInstructorId ==
myId).ToList();

```

```
var sendingMsgToInstructor = meSending.Where(x =>
x.RecieverInstructorId != null).GroupBy(x => x.RecieverInstructorId).Select(x =>
x.Last());

var sendingMsgToStudent = meSending.Where(x => x.RecieverStudentId !=
null).GroupBy(x => x.RecieverStudentId).Select(x => x.Last());

var meRecieving = db.Messages.Where(x => x.RecieverInstructorId ==
myId).ToList();

var recievingMsgFromInstructor = meRecieving.Where(x =>
x.SenderInstructorId != null).GroupBy(x => x.SenderInstructorId).Select(x => x.Last());

var recievingMsgFromStudent = meRecieving.Where(x =>
x.SenderStudentId != null).GroupBy(x => x.SenderStudentId).Select(x => x.Last());

var chatWithInstructor =
sendingMsgToInstructor.Concat(recievingMsgFromInstructor).OrderBy(t =>
t.DateTime).ToList();

var chatWithStudent =
sendingMsgToStudent.Concat(recievingMsgFromStudent).OrderBy(t =>
t.DateTime).ToList();

List<Messages> deleteRows = new List<Messages>();

foreach (var s in chatWithInstructor)
```

```
{  
    if (myId != s.RecieverInstructorId)  
    {  
        var p = s.RecieverInstructorId;  
  
        var twoRows = chatWithInstructor.Where(x => x.SenderInstructorId ==  
p || x.RecieverInstructorId == p);  
  
        int i = 1;  
  
        var firstRow = new Messages();  
        var secondRow = new Messages();  
  
        foreach (var t in twoRows)  
        {  
            if (i == 1)  
            {  
                firstRow = t;  
            }  
            else if (i == 2)  
            {  
                secondRow = t;  
            }  
            i++;  
        }  
    }  
}
```

```
if (firstRow != null && secondRow != null)
{
    if (firstRow.DateTime < secondRow.DateTime)
    {
        deleteRows.Add(firstRow);
    }
    else if (firstRow.DateTime > secondRow.DateTime)
    {
        deleteRows.Add(secondRow);
    }
}
}

foreach (var s in chatWithStudent)
{
    var p = 0;
    if (s.SenderStudentId != null)
    {
        p = s.SenderStudentId ?? default(int);
    }
    else if (s.RecieverStudentId != null)
    {
```

```
p = s.RecieverStudentId ?? default(int);  
}  
  
var twoRows = chatWithStudent.Where(x => x.SenderStudentId == p ||  
x.RecieverStudentId == p);  
  
int i = 1;  
var firstRow = new Messages();  
var secondRow = new Messages();  
foreach (var t in twoRows)  
{  
    if (i == 1)  
    {  
        firstRow = t;  
    }  
    else if (i == 2)  
    {  
        secondRow = t;  
    }  
    i++;  
}  
if (firstRow != null && secondRow != null)  
{
```

```
        if (firstRow.DateTime > secondRow.DateTime)
        {
            deleteRows.Add(secondRow);
        }
        else if (firstRow.DateTime < secondRow.DateTime)
        {
            deleteRows.Add(firstRow);
        }
    }
}

var allLatestMessages =
chatWithInstructor.Concat(chatWithStudent).ToList();

foreach (var d in deleteRows)
{
    try
    {
        var del = allLatestMessages.First(x => x.MessageId == d.MessageId);
        allLatestMessages.Remove(del);
        db.SaveChanges();
    }
    catch { }
```



```
<link href=" ~/Design/css/vendor/all.css" rel="stylesheet" />
<link href="~/Design/css/app/app.css" rel="stylesheet" />
</head>

<body>

<!-- Fixed navbar -->
<div class="navbar navbar-default navbar-fixed-top navbar-size-large navbar-size-xlarge
paper-shadow" data-z="0" data-animated role="navigation">
<div class="container">
<div class="navbar-header">
<button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-
target="#main-nav">
<span class="sr-only">Toggle navigation</span>
<span class="icon-bar"></span>
<span class="icon-bar"></span>
<span class="icon-bar"></span>
</button>
<div class="navbar-brand navbar-brand-logo">

<h3>LearningMadeEasy4u</h3>

</div>
```



```
</div>

<!-- Collect the nav links, forms, and other content for toggling -->
<div class="collapse navbar-collapse" id="main-nav">
  <ul class="nav navbar-nav navbar-nav-margin-left">
    <li class="dropdown active">
      @Html.ActionLink("Home Page", "Index")
    </li>
    <li class="dropdown">
      @Html.ActionLink("Student Login", "Login", "Student")
    </li>
    <li class="dropdown">
      @Html.ActionLink("Instructor Login", "Login", "Instructor")
    </li>
  </ul>
</div>

<!-- /.navbar-collapse -->

</div>

</div>

<!-- body container -->

<!-- image container -->
```

```
<div class="parallax cover overlay cover-image-full home">

<div class="parallax-layer overlay overlay-full overlay-bg-white bg-transparent" data-
speed="8" data-opacity="true">
</div>
</div>

<div class="container">
<div class="page-section-heading">
<h2 class="text-display-1">Features</h2>
<p class="lead text-muted">Learn about all of the features we offer.</p>
</div>
<div class="row" data-toggle="gridalicious">

<div class="media">
<div class="media-left padding-none">
<div class="bg-green-300 text-white">
<div class="panel-body">
<i class="fa fa-film fa-2x fa-fw"></i>
</div>
</div>
</div>
</div>
```

```

<div class="media-body">
<div class="panel panel-default">
<div class="panel-body">
<div class="text-headline">Watch Best Courses</div>
<p>We have the best tutors with advance level of skills. Hurry and get them more are
waiting.</p>
</div>
</div>
</div>
</div>

<div class="media">
<div class="media-left padding-none">
<div class="bg-purple-300 text-white">
<div class="panel-body">
<i class="fa fa-life-bouy fa-2x fa-fw"></i>
</div>
</div>
</div>
</div>

<div class="media-body">
<div class="panel panel-default">
<div class="panel-body">
<div class="text-headline">Extra Support</div>

```

<p>Tutors with extra skills will be available for students to message them privately.</p>

</div>

</div>

</div>

</div>

<div class="media">

<div class="media-left padding-none">

<div class="bg-orange-400 text-white">

<div class="panel-body">

<i class="fa fa-user fa-2x fa-fw"></i>

</div>

</div>

</div>

<div class="media-body">

<div class="panel panel-default">

<div class="panel-body">

<div class="text-headline">Learn from Top Tutors</div>

<p>This website has some top level tutors who are expert in their respective fields.</p>

</div>

</div>

</div>

</div>

```
<div class="media">
<div class="media-left padding-none">
<div class="bg-cyan-400 text-white">
<div class="panel-body">
<i class="fa fa-code fa-2x fa-fw"></i>
</div>
</div>
</div>
</div>
<div class="media-body">
<div class="panel panel-default">
<div class="panel-body">
<div class="text-headline">Lesson Source Files</div>
<p>Every course will contain their respective source files, Such as document files, pdf
file and power point slides.</p>
</div>
</div>
</div>
</div>
</div>
</div>
<div class="media">
<div class="media-left padding-none">
<div class="bg-pink-400 text-white">
```

```

<div class="panel-body">
  <i class="fa fa-print fa-2x fa-fw"></i>
</div>
</div>
</div>
</div>
<div class="media-body">
  <div class="panel panel-default">
    <div class="panel-body">
      <div class="text-headline">Printed Diploma</div>
      <p>After the completion of course you have to give a final test, which will be taken in
order to test your skills and in the end if you pass it you will be given a certificate.</p>
    </div>
  </div>
</div>
</div>
</div>
</div>
<div class="media">
  <div class="media-left padding-none">
    <div class="bg-red-400 text-white">
      <div class="panel-body">
        <i class="fa fa-tasks fa-2x fa-fw"></i>
      </div>
    </div>
  </div>
</div>

```

```

</div>

<div class="media-body">

<div class="panel panel-default">

<div class="panel-body">

<div class="text-headline">New Lessons Every Day</div>

<p>Each and every day out tutors are uploading new and latest tutorials.</p>

</div>

</div>

</div>

</div>

</div>

</div>

</div>

<br />

<div class="parallax cover overlay height-300 margin-none">



<div class="parallax-layer overlay overlay-full overlay-bg-white bg-transparent" data-

opacity="true" data-speed="8">

<div class="v-center">

<div class="page-section">

<h1 class="text-display-2 overlay-bg-white margin-v-0-15 inline-block">Learn from

Home</h1>

```

```
<br />

<p class="lead text-overlay overlay-bg-white-strong inline-block">Just like others did the
same to improve their skills</p>

</div>

</div>

</div>

</div>

<!-- //body container -->

<!-- Footer -->

<section class="footer-section">
<div class="container">
<div class="row">
<div class="col-sm-6 col-md-3">
<h4 class="text-headline text-light">Corporate</h4>
<ul class="list-unstyled">
<li><a href="#">About the company</a></li>
<li><a href="#">Company offices</a></li>
<li><a href="#">Partners</a></li>
<li><a href="#">Terms of use</a></li>
<li><a href="#">Privacy</a></li>
<li><a href="#">Contact us</a></li>
```



```
</ul>

</div>

<div class="col-sm-6 col-md-3">

<h4 class="text-headline text-light">Explore</h4>

<ul class="list-unstyled">

<li><a href="">Courses</a></li>

<li><a href="">Tutors</a></li>

<li><a href="">Pricing</a></li>

<li><a href="">Become Tutor</a></li>

<li><a href="">Sign Up</a></li>

</ul>

</div>

<div class="col-xs-12 col-md-6">

<h4 class="text-headline text-light">Newsletter</h4>

<div class="form-group">

<div class="input-group">

<input type="text" class="form-control" placeholder="Enter email here...">

<span class="input-group-btn">

<button class="btn btn-grey-800" type="button">Subscribe</button>

</span>

</div>

</div>
```

```
<br />
```

```
<p>
```

```
<a href="#" class="btn btn-indigo-500 btn-circle"><i class="fa fa-facebook"></i></a>
```

```
<a href="#" class="btn btn-pink-500 btn-circle"><i class="fa fa-dribbble"></i></a>
```

```
<a href="#" class="btn btn-blue-500 btn-circle"><i class="fa fa-twitter"></i></a>
```

```
<a href="#" class="btn btn-danger btn-circle"><i class="fa fa-google-plus"></i></a>
```

```
</p>
```

```
<p class="text-subhead">
```

```
&copy; 2017 Learning App by diptiarora.
```

```
</p>
```

```
</div>
```

```
</div>
```

```
</div>
```

```
</section>
```

```
<footer class="footer">
```

```
<strong>App By DiptiArora</strong>&copy; Copyright 2017
```

```
</footer>
```

```
<!-- // Footer -->
```

```
<!-- Inline Script for colors and config objects; used by various external scripts; -->
```

```
<script>
```

```
var colors = {  
  "danger-color": "#e74c3c",  
  "success-color": "#81b53e",  
  "warning-color": "#f0ad4e",  
  "inverse-color": "#2c3e50",  
  "info-color": "#2d7cb5",  
  "default-color": "#6e7882",  
  "default-light-color": "#cfd9db",  
  "purple-color": "#9D8AC7",  
  "mustard-color": "#d4d171",  
  "lightred-color": "#e15258",  
  "body-bg": "#f6f6f6"  
};  
  
var config = {  
  theme: "html",  
  skins: {  
    "default": {  
      "primary-color": "#42a5f5"  
    }  
  }  
};  
  
</script>  
  
<script src="~/Design/js/app/app.js"></script>
```

```
<script src="~/Design/js/vendor/all.js"></script>
```

```
</body>
```

```
</html>
```

Addquestions.cshtml

```
//adding questions on the HTML page for LME4U
```

```
@{
```

```
    ViewBag.Title = "LearningMadeEasy4u";
```

```
    Layout = "~/Views/Shared/InstructorLayout.cshtml";
```

```
}
```

```
<divclass="st-pusher well"style="overflow:scroll;"id="content">
```

```
<br><h3style="margin-left:10px; text-align:center;">Add Question</h3>
```

```
<divstyle="border: 1pxsolidsilver; margin:10px10px0px10px; padding: 30px; border-  
radius: 4px; background-color:white;">
```

```
<br>
```

```
<!-- Form -->
```

```
    @using (Html.BeginForm())
```

```
    {
```

```
<divclass="container"style="width:80%;">
```

```
<tableclass="table table-striped table-bordered">
```

```
<!-- Question field -->

<tr>

<td></td>

<td>@Html.Label("Question")</td>

<td>@Html.TextArea("Question", new { @class = "form-control" })</td>

</tr>

<!-- Option1 field -->

<tr>

<td><input type="radio" name="Check" checked="checked" value="1" /></td>

<td>@Html.Label("Option 1:")</td>

<td>@Html.TextArea("Option1", new { @class = "form-control" })</td>

</tr>

<!-- Option2 field -->

<tr>

<td><input type="radio" name="Check" value="2" /></td>

<td>@Html.Label("Option 2:")</td>

<td>@Html.TextArea("Option2", new { @class = "form-control" })</td>

</tr>

<!-- Option3 field -->

<tr>

<td><input type="radio" name="Check" value="3" /></td>

<td>@Html.Label("Option 3:")</td>

<td>@Html.TextArea("Option3", new { @class = "form-control" })</td>
```

```

</tr>

<!-- Option4 field -->

<tr>

<td><input type="radio" name="Check" value="4"></td>

<td>@Html.Label("Option 4:")</td>

<td>@Html.TextArea("Option4", new { @class = "form-control" })</td>

</tr>

<!-- Buttons Container -->

<tr>

<td></td><td></td>

<td>

<input type="text" name="CourseID" value="@ViewBag.CourseID" hidden />

<input type="submit" style="width:150px;" value="Add" class="btn btn-primary" />

<button type="button" class="btn btn-

success" onclick="location.href=@Url.Action("ShowQuestions", "Instructor", new {

CourseID = ViewBag.CourseID})"><span class="glyphicon glyphicon-

backward"></span> Back To Questions</button>

<button type="button" class="btn btn-

success" onclick="location.href=@Url.Action("ShowMyCourses", "Instructor",

null)"><span class="glyphicon glyphicon-backward"></span> Back To My

Courses</button>

</td>

</tr>

```

```
</table>
```

```
</div>
```

```
    }
```

```
</div>
```

```
</div>
```

```
<script>
```

```
$(function () {
```

```
    $("form").validate({
```

```
        rules: {
```

```
            Question: "required",
```

```
            Option1: "required",
```

```
            Option2: "required",
```

```
            Option3: "required",
```

```
            Option4: "required",
```

```
        },
```

```
        messages: {
```

```
            Question: "Field cannot be empty !",
```

```
            Option1: "Field cannot be empty !",
```

```
            Option2: "Field cannot be empty !",
```

```

        Option3: "Field cannot be empty !",
        Option4: "Field cannot be empty !",
    },
    submit: function (form) {
        form.submit();
    }
});
});
</script>

```

change.cshtml

```

@{
    ViewBag.Title = "LearningMadeEasy4u";
    Layout = "~/Views/Shared/InstructorLayout.cshtml";
}

<divclass="st-pusher well"style="overflow:scroll;"id="content">
<br><h3style="margin-left:10px; text-align:center;">Add Question</h3>
<divstyle="border: 1pxsolidsilver; margin:10px10px0px10px; padding: 30px; border-
radius: 4px; background-color:white;">
<br>

```



```

<!-- Form -->

    @using (Html.BeginForm())

    {

<divclass="container"style="width:80%;">

<tableclass="table table-striped table-bordered">

<!-- Question field -->

<tr>

<td></td>

<td>@Html.Label("Question")</td>

<td>@Html.TextArea("Question", new { @class = "form-control" })</td>

</tr>

<!-- Option1 field -->

<tr>

<td><inputtype="radio"name="Check"checkedvalue="1"></td>

<td>@Html.Label("Option 1:")</td>

<td>@Html.TextArea("Option1", new { @class = "form-control" })</td>

</tr>

<!-- Option2 field -->

<tr>

<td><inputtype="radio"name="Check"value="2"></td>

<td>@Html.Label("Option 2:")</td>

<td>@Html.TextArea("Option2", new { @class = "form-control" })</td>

</tr>

```

```

<!-- Option3 field -->

<tr>

<td><input type="radio" name="Check" value="3"></td>

<td>@Html.Label("Option 3:")</td>

<td>@Html.TextArea("Option3", new { @class = "form-control" })</td>

</tr>

<!-- Option4 field -->

<tr>

<td><input type="radio" name="Check" value="4"></td>

<td>@Html.Label("Option 4:")</td>

<td>@Html.TextArea("Option4", new { @class = "form-control" })</td>

</tr>

<!-- Buttons Container -->

<tr>

<td></td><td></td>

<td>

<input type="text" name="CourseID" value="@ViewBag.CourseID" hidden/>

<input type="submit" style="width:150px;" value="Add" class="btn btn-primary"/>

<button type="button" class="btn btn-

success" onclick="location.href='@Url.Action("ShowQuestions", "Instructor", new {

CourseID = ViewBag.CourseID})'"><span class="glyphicon glyphicon-

backward"></span> Back To Questions</button>

```

```

<button type="button" class="btn btn-
success" onclick="location.href=@Url.Action("ShowMyCourses", "Instructor",
null)""><span class="glyphicon glyphicon-backward"></span> Back To My
Courses</button>

</td>

</tr>

</table>

</div>
    }
</div>
</div>

<script>
$(function () {
    $("form").validate({
        rules: {
            Question: "required",
            Option1: "required",
            Option2: "required",
            Option3: "required",
            Option4: "required",

```

```
    },  
    messages: {  
        Question: "Field cannot be empty !",  
        Option1: "Field cannot be empty !",  
        Option2: "Field cannot be empty !",  
        Option3: "Field cannot be empty !",  
        Option4: "Field cannot be empty !",  
    },  
    submit: function (form) {  
        form.submit();  
    }  
});  
});  
</script>
```

changepassword.cshtml

```
//used to change the password for LME4U  
  
@model E_Learning_Managment_System.Models.Instructor  
  
@{  
    ViewBag.Title = "LearningMadeEasy4u";  
    Layout = "~/Views/Shared/InstructorLayout.cshtml";  
}
```

```

}

<divclass="st-pusher well"style="overflow:scroll;"id="content">

<!-- Checks that if password is wrong or not -->

    @if (ViewBag.wrongpassword != null)

        {

<!-- In this container the wrong password error message will be shown -->

<divclass="alert alert-danger alert-dismissible fade in">

<a href="#" class="close" data-dismiss="alert" aria-label="close">&times;</a>

<strong>@ViewBag.wrongpassword</strong>

        @{

            ViewBag.wrongpassword = null;

        }

</div>

        }

<divstyle="border: 1pxsolidsilver; margin:10px10px0px10px; padding: 30px; border-
radius: 4px; background-color:white;">

<!-- Form -->

<formaction="/Instructor/ChangePassword"method="post">

<divclass="panel-body">

<divclass="form-group">

<!-- current password will be entered here -->

<divclass="form-control-material">

```

```
<label>Enter Current Password</label><br><br>
<input type="password" name="password" class="form-control"/>
</div>
</div><br><br>
<!-- submit button -->
<input type="submit" class="btn btn-primary"/>
</div>
</form>
</div>
</div>
<script>
//Form validation function
$(function () {
    $("form").validate({
        rules: {
            password: "required"
        },
        messages: {
            password: "Field cannot be empty !",
        },
        submit: function (form) {
            form.submit();
        }
    });
});
```

```

    });

});
</script>

```

DownloadFile.cshtml

// Downloading a file that was previously uploaded by an instructor for LME4U

```
@model List<string>
```

```
@{
```

```
    ViewBag.Title = "LearningMadeEasy4u";
```

```
    Layout = "~/Views/Shared/InstructorLayout.cshtml";
```

```
}
```

```
<divclass="st-pusher well"style="overflow:scroll;"id="content">
```

```
<br><h3style="margin-left:10px; text-align:center;">Download Files</h3>
```

```
<divstyle="border: 1pxsolidsilver; margin:10px10px0px10px; padding: 10px; border-
radius: 4px; background-color:white;">
```

```
<tablestyle="width:100%;"class="table table-bordered">
```

```
<tr>
```

```
<th>File Name</th>
```

```
<th>Link</th>
```

```
</tr>
```

```

        @for (var i = 0; i <= Model.Count - 1; i++)
        {
<tr>
<!-- File Name -->
<td>
                @Model[i].ToString()
</td>
<!-- File Download Link -->
<td>
                @Html.ActionLink("Download", "Download", new { FileName =
                @Model[i].ToString() })
</td>
</tr>
        }
</table>
</div>
</div>

```

InstructorIndex.cshtml

```
<!-- Homepage after login for instructor for LME4U-->
```

```

@{
    ViewBag.Title = "LearningMadeEasy4u";

```



```

    Layout = "~/Views/Shared/InstructorLayout.cshtml";
}

<!-- Homepage after login -->

<divclass="st-pusher" id="content">

<!-- sidebar effects INSIDE of st-pusher: -->

<!-- st-effect-3, st-effect-6, st-effect-7, st-effect-8, st-effect-14 -->

<!-- this is the wrapper for the content -->

<divclass="st-content">

<!-- extra div for emulating position:fixed of the menu -->

<divclass="st-content-inner padding-none">

<divclass="container-fluid">

<divclass="page-section">

<h1 class="text-display-1">Dashboard</h1>

</div>

<divclass="row" data-toggle="isotope">

<divclass="item col-xs-12 col-lg-6">

<!-- Courses container -->

<divclass="panel panel-default paper-shadow" data-z="0.5">

<divclass="panel-heading">

<h4 class="text-headline margin-none">My Courses</h4>

<p class="text-subhead text-light">Your recent courses</p>

</div>

<ulclass="list-group">

```

```
        @if (ViewBag.mycourses.Count > 0)
        {
            var i = 1;
            foreach (var c in ViewBag.mycourses)
            {
                if (i <= 3)
                {
                    <li class="list-group-item media v-middle">
                        <div class="media-body">
                            <a href="#" class="text-subhead list-group-link">@c.CourseTitle</a>
                        </div>
                        <div class="media-right">
                            <div class="progress progress-mini width-100 margin-none">
                                <div class="progress-bar progress-bar-green-300" role="progressbar" aria-
                                    valuenow="45" aria-valuemin="0" aria-valuemax="100" style="width: 45%;">
                                </div>
                            </div>
                        </div>
                    </li>
                    i++;
                }
            }
        }
        else
        {
```

```
break;
        }

    }
}

else
    {
<liclass="list-group-item media v-middle">
<divclass="media-body">
<a href="#" class="text-subhead list-group-link">No courses assigned yet!</a>
</div>
</li>
    }

</ul>

<divclass="panel-footer text-right">
<a href="/Instructor/ShowMyCourses" class="btn btn-primary paper-shadow
relative" data-z="0" data-hover-z="1" data-animated>View all</a>
</div>
</div>
</div>
</div>
</div>
```

```
</div>  
  
<!-- /st-content-inner -->  
  
</div>  
  
<!-- /st-content -->  
  
</div>
```

Login.cshtml

```
<!-- login page for LME4U-->  
  
@model E_Learning_Managment_System.Models.ViewModel.LoginViewModel  
  
<head>  
  
<title>Home Page</title>  
  
<linktype="text/css"href="~/Design/css/vendor/all.css"rel="stylesheet">  
<linktype="text/css"href="~/Design/css/app/app.css"rel="stylesheet"/>  
  
</head>  
  
<bodyclass="login">  
  
<divid="content">  
  
<divclass="container-fluid">  
  
<divclass="lock-container">  
  
<divclass="panel panel-default text-center paper-shadow" data-z="0.5">  
  
<h1class="text-display-1 text-center margin-bottom-none">Sign In</h1>  
  
<imgsrc="~/Views/Home/images/login.png"class="img-circle width-80">
```

```

        @if (ViewBag.Message != null)
        {
<!-- login error message will be shown here -->
<divclass="alert alert-danger alert-dismissible fade in"style="text-align:center;">
<a href="#"class="close" data-dismiss="alert"aria-label="close">&times;</a>
        @ViewBag.Message
</div>
        }
<!-- Form -->
        @using (Html.BeginForm()) {
<divclass="panel-body">
<!-- Email will be entered here -->
<divclass="form-group">
            @Html.TextBoxFor(m => m.Email, null, new { @class = "form-
control", id = "username", placeholder = "Enter Email" })
            @Html.ValidationMessageFor(m => m.Email, "", new { @style =
"color:red" })
</div>
<!-- Paswword will be entered here -->
<divclass="form-group">
            @Html.PasswordFor(m => m.password, new { @class = "form-
control", id = "password", placeholder = "Enter Password" })

```

```

        @Html.ValidationMessageFor(m => m.password, "", new { @style =
"color:red" })
    </div>

    <!-- Submit Button -->

    <input type="submit" value="Login" class="btn btn-primary"/><i class="fa fa-fw fa-
unlock-alt"></i>

    <a href="#" class="forgot-password">Forgot password?</a>

    <!-- Signup link -->

        @Html.ActionLink("Create Account", "SignUp", "Instructor", null, new {
@class = "link-text-color" })
    </div>

    }

</div>

</div>

</div>

</div>

<!-- Footer -->

<footer class="footer">

    <strong>App By Dipti Arora</strong> &copy; Copyright 2017

</footer>

<!-- // Footer -->

<!-- Inline Script for colors and config objects; used by various external scripts; -->

<script>

```

```
var colors = {  
  "danger-color": "#e74c3c",  
  "success-color": "#81b53e",  
  "warning-color": "#f0ad4e",  
  "inverse-color": "#2c3e50",  
  "info-color": "#2d7cb5",  
  "default-color": "#6e7882",  
  "default-light-color": "#cfd9db",  
  "purple-color": "#9D8AC7",  
  "mustard-color": "#d4d171",  
  "lightred-color": "#e15258",  
  "body-bg": "#f6f6f6"  
  };  
  
var config = {  
  theme: "html",  
  skins: {  
    "default": {  
      "primary-color": "#42a5f5"  
    }  
  }  
};  
  
</script>  
  
<scriptsrc="~/Design/js/vendor/all.js"></script>
```

```
<scriptsrc=~\Design\js\app\app.js"></script>  
</body>
```

Messages.cshtml

```
// To exchange messages with students and instructors for LME4U
```

```
@{  
    ViewBag.Title = "LearningMadeEasy4u";  
    Layout = "~/Views/Shared/InstructorLayout.cshtml";  
}
```

```
<style>
```

```
.disabled {  
    pointer-events: none;  
    cursor: default;  
    opacity: 0.5;  
}  
  
#historyclick:hover {  
    -o-box-shadow: 2px 2px 19px #CCC;  
    -webkit-box-shadow: 2px 2px 19px #CCC;  
    -moz-box-shadow: 2px 2px 19px #CCC;  
}
```

```
#historyclick:active {  
    -o-box-shadow: 2px 2px 19px #FFCC00;
```



```

    -webkit-box-shadow: 2px 2px 19px #FFCC00;

    -moz-box-shadow: 2px 2px 19px #FFCC00;

}

</style>

<!-- Input field with the name of user who is logged in but the field is hidden -->
<input id="sessionvalue" value="@Session["Name"].ToString()" hidden/>

<!-- Input field with the id of the user who is logged in but the field is hidden -->
<input id="sessionid" value="@Session["InstructorID"].ToString()" hidden />

<div class="st-pusher well" style="overflow:scroll;" id="content">

<div class="st-content">

<!-- extra div for emulating position:fixed of the menu -->

<div class="st-content-inner padding-none">

<div class="container-fluid">

<div class="page-section third">

<div class="media messages-container media-clearfix-xs-min media-grid">

<div class="media-left">

<!-- Input field for entering the name of recipient to the the message -->

<input style="width:250px;" class="form-control" id="searchTerm" name="searchTerm"
placeholder="Write name..." type="text" />

</div>

<div class="media-body">

<div class="form-group">

```

```
<div class="input-group">
  <div class="input-group-btn disabled" id="SendButton">
    <!-- Button for sending the message -->
    <a class="btn btn-primary" onclick="SendMessage()">
      <i class="fa fa-envelope"></i> Send
    </a>
  </div>
  <!-- /btn-group -->
  <div id="MessageBody" class="disabled">
    <!-- Input field for entering the message -->
    <input type="text" id="WriteMessage" class="form-control share-text"
      placeholder="Write message..." />
  </div>
  </div>
  <!-- Container in which all of the messages will be shown -->
  <div id="refresh" style="float:right;">
    <a class="btn btn-success" onclick="GetMessages()">
      <i class="fa fa-refresh"></i> Refresh Messages
    </a>
  </div><br><br>
  <!-- /input-group -->
</div>
<!-- Messages Container -->
```

```
<div id="MessageContent">
```

```
</div>
```

```
</div>
```

```
</div>
```

```
</div>
```

```
</div>
```

```
</div>
```

```
<!-- /st-content-inner -->
```

```
</div>
```

```
</div>
```

```
<script type="text/javascript">
```

```
var id = 0;
```

```
var senderType = "instructor";
```

```
var recieverType;
```

```
var name = null;
```

```
var check = "not selected";
```

```
//whenever the page opens this function is called everytime which then further calls
```

```
GetMessages() in it self
```

```
$(function () {
```

```

$(document).ready(function () {
    GetMessages();
});
});

```

//Function for getting the name suggestion when writing the recipient name in write name input field

```

$(function () {
    $("#searchTerm").autocomplete({
        source: function (request, response) {
            $.ajax({
                url: '/Home/AutocompleteSuggestions/',
                data: "{ 'prefix': '" + request.term + "', 'senderType': '" + senderType + "'",
                dataType: "json",
                type: "POST",
                contentType: "application/json; charset=utf-8",
                success: function (data) {
                    $("#SendButton").addClass("disabled");
                    $("#MessageBody").addClass("disabled");
                    $("#MessageContent").html("");
                    check = "not selected";
                    GetMessages();
                    response($.map(data, function (item) {

```

```
        return item;
    )))
},
error: function (response) {
},
failure: function (response) {
}
});
},
select: function (e, i) {
    id = i.item.val;
    recieverType = i.item.recieverType;
    senderType = i.item.senderType;
    name = i.item.label;
    check = "selected";
    $("#SendButton").removeClass("disabled");
    $("#MessageBody").removeClass("disabled");
    GetMessages();
},
minLength: 1
});
});
```

```
//Function for sending message

function SendMessage() {

    if ($("#WriteMessage").val() != "")

    {

        var message = ($("#WriteMessage").val());

        var Message = { MessageBody: message, Id: id, name: name, senderType:

senderType, recieverType: recieverType };

        ($("#WriteMessage").val("));

        $.post("/Home/SendMessage", Message, function (data) {

            GetMessages();

        });

    }

    else

    {

        alert("Message cannot be null !");

    }

}

//Funtion for getting all of the messages to be shown in messages container

function GetMessages() {

    if (check == "not selected")

    {
```

```

var sessionId = $("#sessionId").val();

var Message = { Id: sessionId, senderType: senderType };

$.post("/Home/GetAllMessages", Message, function (data) {

    var htmlBody = "";

    var sessionValue = $("#sessionvalue").val();

    $.each(data, function (index, value) {

        htmlBody += "<div id='historyclick' class='panel panel-default paper-
shadow'>" +

            "<input type='text' id='id' value='" + value.id + " hidden />" +

            "<input type='text' id='name' value='" + value.name + " hidden />" +

            "<input type='text' id='recieverType' value='" + value.recieverType + "
hidden />" +

            "<div class='panel-body'><div class='media v-middle'><div class='media-
body message'>" +

                "<h4 class='text-subhead margin-none'><a href='#'>" + value.name +

                "</a></h4>" +

                "<p class='text-caption text-light'><i class='fa fa-clock-o'></i> " +

                value.dateTime + "</p>" +

                "</div></div><p>" + value.messageBody + "</p></div></div>";

```

```

    });

    $("#MessageContent").html(htmlBody);

    });
}
else if (check == "selected")
{
    var Message = { Id: id, name: name, senderType: senderType, recieverType:
recieverType };

    $.post("/Home/GetMessages", Message, function (data) {

        var htmlBody = "";

        var sessionValue = $("#sessionvalue").val();

        $.each(data, function (index, value) {

            if (value.name == sessionValue) {

                htmlBody += "<div class='panel panel-default paper-shadow' style='text-
align:right;background-color:#B5E9FD;' data-z='0.5' data-hover-z='1' data-animated>" +

                "<div class='panel-body'><div class='media v-middle'><div
class='media-body message'>" +

                "<h4 class='text-subhead margin-none'><a href='#'>Me</a></h4>" +

```



```

        "<p class='text-caption text-light'><i class='fa fa-clock-o'></i> " +
value.dateTime + "</p>" +
        "</div></div><p>" + value.messageBody + "</p></div></div>";
    }
    else {
        htmlBody += "<div class='panel panel-default paper-shadow'
style='background-color:#FCFDB5;' data-z='0.5' data-hover-z='1' data-animated>" +
        "<div class='panel-body'><div class='media v-middle'><div
class='media-body message'>" +
        "<h4 class='text-subhead margin-none'><a href='#'>" + value.name +
"</a></h4>" +
        "<p class='text-caption text-light'><i class='fa fa-clock-o'></i> " +
value.dateTime + "</p>" +
        "</div></div><p>" + value.messageBody + "</p></div></div>";
    }
});

$("#MessageContent").html(htmlBody);
});
}
}

```

//Whenever you click on the message in all message's chat history then this function is called

```

$(body).on('click', '#historyclick', function () {
    var recieverId = $(this).children("#id").val();
    var recieverName = $(this).children("#name").val();
    var type = $(this).children("#recieverType").val();
    var chk = $(this).children("#check").val();

    $("#searchTerm").val(recieverName);

    id = recieverId;
    recieverType = type;
    name = recieverName;
    check = "selected";
    var Message = { Id: id, name: name, senderType: senderType, recieverType:
recieverType };

    $("#SendButton").removeClass("disabled");
    $("#MessageBody").removeClass("disabled");
    $("#MessageContent").html("");
    GetMessages();
});
</script>

```

Profile.cshtml

```
// View the profile of a student or instructor for LME4U

@model E_Learning_Managment_System.Models.Instructor

@{
    ViewBag.Title = "LearningMadeEasy4u";

    Layout = "~/Views/Shared/InstructorLayout.cshtml";
}

<divclass="st-pusher well"style="overflow:scroll;"id="content">

    @if (ViewBag.password != null)
    {
        <!-- Wrong password error message will be shown here -->

        <divclass="alert alert-success alert-dismissible fade in">

        <ahref="#"class="close"data-dismiss="alert"aria-label="close">&times;</a>

        <strong>@ViewBag.password</strong>

        @{
            ViewBag.password = null;
        }

    }

</div>

}

<!-- sidebar effects INSIDE of st-pusher: -->

<!-- st-effect-3, st-effect-6, st-effect-7, st-effect-8, st-effect-14 -->
```

```

<!-- this is the wrapper for the content -->

<divclass="st-content">

<!-- extra div for emulating position:fixed of the menu -->

<divclass="st-content-inner padding-none">

<divclass="container-fluid">

<divclass="page-section third">

<!-- Tabbable Widget -->

<divclass="tabbable paper-shadow relative" data-z="0.5">

<!-- Tabs -->

<ulclass="nav nav-tabs">

<liclass="active"><a href="app-student-profile.html"><i class="fa fa-fw fa-
lock"></i><spanclass="hidden-sm hidden-xs">Manage Account</span></a></li>

</ul>

<!-- // END Tabs -->

<!-- Panes -->

<divclass="tab-content">

<divid="account" class="tab-pane active">

<formclass="form-horizontal">

<divclass="form-group">

<divstyle="float:right;">

<br><br><br><br><labelfor="changepassword" class="col-md-2 control-
label"><a href="/Instructor/ChangePassword">Change Password</a></label>

</div>

```

```

<labelfor="inputEmail3"class="col-sm-2 control-label">Avatar</label>

<divclass="col-md-6">

<divclass="media v-middle">

<divclass="media-left">

<divclass="icon-block width-100 bg-grey-100">

<iclass="fa fa-photo text-light"></i>

</div>

</div>

</div>

</div>

</div>

</div>

<!-- First name of the user will be shown here -->

<divclass="form-group">

<labelfor="inputEmail3"class="col-md-2 control-label">First Name</label>

<divclass="col-md-10">

<divclass="row">

<divclass="form-control-material">

                                @Html.DisplayFor(m => m.FirstName, null, new {

@class = "form-control", id = "exampleInputFirstName" })

</div>

</div>

</div>

</div>

<labelfor="inputEmail3"class="col-md-2 control-label">Last Name</label>

```

```

<divclass="col-md-10">

<divclass="row">

<divclass="form-control-material">

                                @Html.DisplayFor(m => m.LastName, null, new {

@class = "form-control", id = "exampleInputLastName" })

</div>

</div>

</div>

</div>

<!-- Email of the user will be shown here -->

<divclass="form-group">

<labelfor="inputEmail3"class="col-md-2 control-label">Email</label>

<divclass="col-md-6">

<divclass="form-control-material">

<divclass="input-group">

<spanclass="input-group-addon"><i>fa fa-envelope</i></span>

                                @Html.DisplayFor(m => m.Email, null, new { @class

= "form-control", id = "inputEmail3" })

</div>

</div>

</div>

</div>

</div>

</form>

```

```
</div>

</div>

<!-- // END Panes -->

</div>

<!-- // END Tabbable Widget -->

</div>

</div>

</div>

</div>

<!-- /st-content-inner -->

</div>

<!-- /st-content -->

</div>
```

QuizGenerator.cshtml

```
@{
    ViewBag.Title = "LearningMadeEasy4u";
    Layout = "~/Views/Shared/InstructorLayout.cshtml";
}
```

```
<divclass="st-pusher well"style="overflow:scroll;"id="content">
<br><h4style="color:red;margin-left:10px;">Note:</h4>
```

```
<pstyle="color:red;margin-left:10px;">Only those courses will be shown in drop down
list that have question bank !</p>
```

```
<h3style="margin-left:10px; text-align:center;">Quiz Generator</h3>
```

```
<divstyle="border: 1pxsolidsilver; margin:10px10px0px10px; padding: 30px; border-
radius: 4px; background-color:white;">
```

```
<formaction="/Instructor/SelectQuestions">
```

```
<!-- Title of the Quiz will be entered here -->
```

```
<labelstyle="font-family:Calibri;font-size:15px;">Title:</label>
```

```
<inputtype="text"name="Title"class="form-control"style="font-family:Calibri;font-
size:15px;"/><br>
```

```
<!-- Course for the quiz will be selected from this dropdownlist -->
```

```
<labelstyle="font-family:Calibri;font-size:15px;">Course:</label>
```

```
<selectname="CourseID"class="form-control"style="font-family:Calibri;font-
size:15px;">
```

```
<optiondisabledselected>--Select a Course--</option>
```

```
    @foreach (var course in Model)
```

```
    {
```

```
<optionvalue="@course.CourseID">Code: @course.CourseID, Title:
```

```
@course.CourseTitle</option>
```

```
    }
```

```
</select><br>
```



```
<!-- Submit Button -->

<center><input type="submit" value="Generate" style="width:15%;" class="btn btn-
primary"></center>

</form>

</div>

</div>

<script>

//Function for the form validation

$(function () {

    $("form").validate({

        rules: {

            Title: "required",

            CourseID: "required",

        },

        messages: {

            Title: "Field cannot be empty !",

            CourseID: "Field cannot be empty !",

        },

        submit: function (form) {

            form.submit();
```

```

    }
  });
});
</script>

```

Showallcourses.cshtml

```

@{
    ViewBag.Title = "LearningMadeEasy4u";
    Layout = "~/Views/Shared/InstructorLayout.cshtml";
}

<divclass="st-pusher well" style="overflow:scroll;" id="content">

<br><h3style="margin-left:10px; text-align:center;">Courses</h3>

<!-- If there are courses in the database then they will be shown here -->
    @if (ViewBag.Courses.Count > 0)
    {
        foreach (var c in ViewBag.Courses)
        {
            <divstyle="border: 1pxsolidsilver; margin:10px10px0px10px; padding: 10px; border-
            radius: 4px; background-color:white;">

```



```

<table>
<tr>
<td><pstyle="font-family:Calibri; font-size:18px"><b>No Courses</b></p></td>
</tr>
</table>
</div>
    }
</div>

```

Showquizzes.cshtml

```

@{
    ViewBag.Title = "LearningMadeEasy4u";
    Layout = "~/Views/Shared/InstructorLayout.cshtml";
}

<divclass="st-pusher well"style="overflow:scroll;"id="content">
    @if (ViewBag.QuizAlert != null)
    {
<!-- whenever a new quiz is made then its success message will be shown here -->
<divclass="alert alert-success alert-dismissible fade in">

```



```
</div>  
  
    }  
  
</div>
```

showquizquestion.cshtml

```
<!-- All questions that were selected to be given in the quiz will be shown here for  
LME4U-->
```

```
@{  
    ViewBag.Title = "LearningMadeEasy4u";  
    Layout = "~/Views/Shared/InstructorLayout.cshtml";  
}
```

```
<divclass="st-pusher well"style="overflow:scroll;"id="content">
```

```
<br>
```

```
<divstyle="border: 1pxsolidsilver; margin:10px10px0px10px; padding: 10px; border-  
radius: 4px; background-color:white;">
```

```
<tableclass="table table-bordered table-striped"style="table-layout:fixed">
```

```
<tr>
```

```

<thstyle="width:318px; text-align:center;">
<h3>Questions</h3>
</th>
</tr>
<!-- All questions that were selected to be given in the quiz will be shown here -->
    @foreach (var q in Model)
    {
<tr>
<tdstyle="padding-left:10px;">@q.Question</td>
</tr>
    }
</table>
<buttontype="button"class="btn btn-
success"onclick="location.href='@Url.Action("ShowQuizes", "Instructor",
null)'"><spanclass="glyphicon glyphicon-backward"></span> Back To Show Existing
Quizes</button>
</div>
</div>

```

Signup.cshtml

```

<!--Used to sign up a Student or instructor for LME4U->

```

```

@model

```

```

E_Learning_Managment_System.Models.ViewModel.InstructorSignUpViewModel

```



```
<!DOCTYPEhtml>

<htmlclass="hide-sidebar ls-bottom-footer"lang="en">

<head>

<metacharset="utf-8">

<metahttp-equiv="X-UA-Compatible"content="IE=edge">

<metaname="viewport"content="width=device-width, initial-scale=1">

<metaname="description"content="">

<metaname="author"content="">

<title>LearningMadeEasy4u</title>

<linkhref="~/Design/css/vendor/all.css"rel="stylesheet"/>

<linkhref="~/Design/css/app/app.css"rel="stylesheet"/>

</head>

<bodyclass="login">

<divid="content">

<divclass="container-fluid">

<divclass="lock-container">

<divclass="panel panel-default text-center paper-shadow"data-z="0.5">

<h1class="text-display-1">Create account</h1>

<divclass="panel-body">

<!-- Signup form -->

    @using (Html.BeginForm())
```

```
        {  
  
        <!-- FirstName -->  
  
        <divclass="form-group">  
  
                @Html.TextBoxFor(m => m.FirstName, null, new { @class = "form-  
control", id = "firstname", placeholder = "First Name" })  
  
                @Html.ValidationMessageFor(m => m.FirstName, "", new { @style  
= "color:red" })  
  
        </div>  
  
        <!-- LastName -->  
  
        <divclass="form-group">  
  
                @Html.TextBoxFor(m => m.LastName, null, new { @class = "form-  
control", id = "lastname", placeholder = "Last Name" })  
  
                @Html.ValidationMessageFor(m => m.LastName, "", new { @style  
= "color:red" })  
  
        </div>  
  
        <!-- Email -->  
  
        <divclass="form-group">
```

```
        @Html.TextBoxFor(m => m.Email, null, new { @class = "form-
control", id = "email", placeholder = "Email" })

        @Html.ValidationMessageFor(m => m.Email, "", new { @style =
"color:red" })

</div>

<!-- Password -->

<divclass="form-group">

        @Html.PasswordFor(m => m.password, new { @class = "form-
control", id = "password", placeholder = "Password" })

        @Html.ValidationMessageFor(m => m.password, "", new {
@style = "color:red" })

</div>

<!-- Repeat Password -->

<divclass="form-group">

        @Html.PasswordFor(m => m.Repassword, new { @class = "form-
control", id = "passwordConfirmation", placeholder = "Password Confirmation" })

        @Html.ValidationMessageFor(m => m.Repassword, "", new {
@style = "color:red" })
```

```
</div>

<divclass="form-group text-center">

<divclass="checkbox">

<inputtype="checkbox" id="agree" required="required"/>

<labelfor="agree">* I Agree with <a href="#">Terms & Conditions!</a></label>

</div>

</div>

<!-- Submit button -->

<divclass="text-center">

<inputtype="submit" value="Create Account" class="btn btn-primary"/>

</div>

    }

<!-- //Signup -->

</div>

</div>

</div>

</div>

</div>

<!-- Footer -->

<footerclass="footer">

<strong>App By DiptiArora</strong>&copy; Copyright 2017

</footer>

<!-- // Footer -->
```

```
<!-- Inline Script for colors and config objects; used by various external scripts; -->
```

```
<script>
```

```
var colors = {
```

```
  "danger-color": "#e74c3c",
```

```
  "success-color": "#81b53e",
```

```
  "warning-color": "#f0ad4e",
```

```
  "inverse-color": "#2c3e50",
```

```
  "info-color": "#2d7cb5",
```

```
  "default-color": "#6e7882",
```

```
  "default-light-color": "#cfd9db",
```

```
  "purple-color": "#9D8AC7",
```

```
  "mustard-color": "#d4d171",
```

```
  "lightred-color": "#e15258",
```

```
  "body-bg": "#f6f6f6"
```

```
};
```

```
var config = {
```

```
  theme: "html",
```

```
  skins: {
```

```
    "default": {
```

```
      "primary-color": "#42a5f5"
```

```
    }
```

```
  }
```

```
};
```

```

</script>

<scriptsrc="~/Scripts/jquery-3.1.1.min.js"></script>

<scriptsrc="~/Scripts/jquery.validate.min.js"></script>

<scriptsrc="~/Scripts/jquery.validate.unobtrusive.min.js"></script>

<scriptsrc="~/Design/js/vendor/all.js"></script>

<scriptsrc="~/Design/js/app/app.js"></script>

</body>

</html>

```

UploadFile.cshtml

```

@{

    ViewBag.Title = "LearningMadeEasy4u";

    Layout = "~/Views/Shared/InstructorLayout.cshtml";

}

<divclass="st-pusher well"style="overflow:scroll;"id="content">

    @if (ViewBag.FileAlert != null)

    {

<!-- Success message for adding file will be shown here -->

<divclass="alert alert-success alert-dismissible fade in">

<a href="#" class="close" data-dismiss="alert" aria-label="close">&times;</a>

<strong>@ViewBag.FileAlert</strong>

    @{}

```

```

        ViewBag.FileAlert = null;
    }
</div>
}
<br><h3style="margin-left:10px; text-align:center;">Upload Files</h3>
<divstyle="border: 1pxsolidsilver; margin:10px10px0px10px; padding: 10px; border-
radius: 4px; background-color:white;">
<table>
<tr>
<td>
<!-- Form -->
        @using (Html.BeginForm("UploadFile", "Instructor", FormMethod.Post, new
{ enctype = "multipart/form-data" }))
        {
<!-- Browse file -->
<label>upload file:</label><br>
<inputtype="file"name="file"/><br>
<!-- Submit Button -->
<inputtype="submit"/>
        }
</td>
</tr>
</table>

```

```
</div>
```

```
</div>
```

Student-Takecourse.cshtml

```
<!--student registers a course in LME4U-->
```

```
@{
```

```
    ViewBag.Title = "LearningMadeEasy4u";
```

```
    Layout = "~/Views/Shared/StudentLayout.cshtml";
```

```
}
```

```
<divclass="st-pusher well"style="overflow:scroll;"id="content">
```

```
<br><h3style="margin-left:10px; text-align:center;">Register Course</h3>
```

```
<divstyle="border: 1pxsolidsilver; margin:10px10px0px10px; padding: 30px; border-  
radius: 4px; background-color:white;">
```

```
<!-- Form -->
```

```
<formaction="/Student/TakeCourse"method="post">
```

```
<!-- DropDownList of courses -->
```

```
<selectname="CourseID"class="form-control"style="font-family:Calibri;font-  
size:18px;">
```

```
    @if (Model.Count <= 0)
```

```
    {
```

```
<optiondisabledselected>--No courses to show--</option>
```

```
    }
```



```
else

    {

<optiondisabledselected>--Select a Course--</option>

foreach (var course in Model)

    {

<optionvalue="@course.CourseID">Code: @course.CourseID, Title:

@course.CourseTitle</option>

    }

    }

</select><br>

<!-- Submit button -->

<center><inputtype="submit"value="Register"style="width:15%;"class="btn btn-

primary"></center>

</form>

</div>

</div>

<script>

//Form validation function

$(function () {

    $("form").validate({

        rules: {

            CourseID: "required"
```

```

    },
    messages: {
        CourseID: "You have to select a course",
    },
    submit: function (form) {
        form.submit();
    }
});
});
</script>

```

Signup.cshtml

```
<!--Sign up a student or instructor for LME4U -->
```

```
@model
```

```
E_Learning_Managment_System.Models.ViewModel.StudentSignUpViewModel
```

```
<!DOCTYPEhtml>
```

```
<htmlclass="hide-sidebar ls-bottom-footer"lang="en">
```

```
<head>
```

```
<metacharset="utf-8">
```

```
<metahttp-equiv="X-UA-Compatible"content="IE=edge">
```

```
<metaname="viewport"content="width=device-width, initial-scale=1">
```

```
<metaname="description"content="">
```

```
<metaname="author"content="">
<title>LearningMadeEasy4u</title>
<scriptsrc="~/Scripts/jquery-3.1.1.min.js"></script>
<linkhref="~/Design/css/vendor/all.css"rel="stylesheet"/>
<linkhref="~/Design/css/app/app.css"rel="stylesheet"/>
</head>
<bodyclass="login">
<divid="content">
<divclass="container-fluid">
<divclass="lock-container">
<divclass="panel panel-default text-center paper-shadow" data-z="0.5">
<h1class="text-display-1">Create account</h1>
<divclass="panel-body">

<!-- Signup Form -->
        @using (Html.BeginForm())
        {
<!-- FirstName -->
<divclass="form-group">

                @Html.TextBoxFor(m => m.FirstName, null, new { @class =
"form-control", id = "firstname", placeholder = "First Name" })
```

```
        @Html.ValidationMessageFor(m => m.FirstName, "", new {
@style = "color:red" })

</div>

<!-- LastName -->

<divclass="form-group">

        @Html.TextBoxFor(m => m.LastName, null, new { @class =
"form-control", id = "lastname", placeholder = "Last Name" })

        @Html.ValidationMessageFor(m => m.LastName, "", new {
@style = "color:red" })

</div>

<!-- Email -->

<divclass="form-group">

        @Html.TextBoxFor(m => m.Email, null, new { @class =
"form-control", id = "email", placeholder = "Email" })

        @Html.ValidationMessageFor(m => m.Email, "", new { @style
= "color:red" })

</div>

<!-- Password -->
```

```

<divclass="form-group">

    @Html.PasswordFor(m => m.password, new { @class = "form-
control", id = "password", placeholder = "Password" })

    @Html.ValidationMessageFor(m => m.password, "", new {
@style = "color:red" })

</div>

<!-- Repeat Password -->

<divclass="form-group">

    @Html.PasswordFor(m => m.RePassword, new { @class =
"form-control", id = "passwordConfirmation", placeholder = "Password Confirmation" })

    @Html.ValidationMessageFor(m => m.RePassword, "", new {
@style = "color:red" })

</div>

<divclass="form-group text-center">

<divclass="checkbox">

<inputtype="checkbox" id="agree" required="required"/>

<labelfor="agree">* I Agree with <a href="#">Terms & Conditions!</a></label>

</div>

</div>

```

```
<!-- Submit button -->

<divclass="text-center">

<inputtype="submit"value="Create Account"class="btn btn-primary"/>

</div>

        }

<!-- //Signup -->

</div>

</div>

</div>

</div>

</div>

<!-- Footer -->

<footerclass="footer">

<strong>App By DiptiArora</strong>&copy; Copyright 2017

</footer>

<!-- // Footer -->

<!-- Inline Script for colors and config objects; used by various external scripts; -->

<script>

var colors = {

"danger-color": "#e74c3c",

"success-color": "#81b53e",

"warning-color": "#f0ad4e",
```

```
"inverse-color": "#2c3e50",
"info-color": "#2d7cb5",
"default-color": "#6e7882",
"default-light-color": "#cfd9db",
"purple-color": "#9D8AC7",
"mustard-color": "#d4d171",
"lightred-color": "#e15258",
"body-bg": "#f6f6f6"

};

var config = {
    theme: "html",
    skins: {
"default": {
"primary-color": "#42a5f5"
    }
    }
};

</script>

<scriptsrc="~/Design/js/vendor/all.js"></script>
<scriptsrc="~/Design/js/app/app.js"></script>
<scriptsrc="~/Scripts/jquery.validate.min.js"></script>
<scriptsrc="~/Scripts/jquery.validate.unobtrusive.min.js"></script>
```

```
</body>
```

```
</html>
```

QuizResult.cshtml

```
<!--Used to calculate the result of the quiz for LME4U-->
```

```
@{
```

```
    ViewBag.Title = "LearningMadeEasy4u";
```

```
    Layout = "~/Views/Shared/StudentLayout.cshtml";
```

```
}
```

```
<divclass="st-pusher well"style="overflow:scroll;"id="content">
```

```
<br><h3style="text-align:center;">Results</h3>
```

```
<!-- Quizes Result -->
```

```
<divstyle="border: 1pxsolidsilver; margin:10px10px0px10px; padding: 10px; border-
radius: 4px; background-color:white;">
```

```
<tablestyle="width:100%;"class="table table-bordered">
```

```
<tr>
```

```
<th>Quiz ID</th>
```

```
<th>Student ID</th>
```

```
<th>Instructor ID</th>
```

```
<th>Course Code</th>
```



```
<th>Total Marks</th>
<th>Obtained Marks</th>
</tr>
  @foreach(var r in Model)
  {
<tr>
<td>
      @r.QuizID
</td>
<td>
      @r.StudentID
</td>
<td>
      @r.InstructorID
</td>
<td>
      @r.CourseID
</td>
<td>
      @r.TotalMarks
</td>
<td>
      @r.ObtainedMarks
```

```

</td>
</tr>
    }
</table>
</div>
</div>

```

Quiz.cshtml

```

@{
    ViewBag.Title = "LearningMadeEasy4u";
    Layout = "~/Views/Shared/StudentLayout.cshtml";
}

<divclass="st-pusher well"style="overflow:scroll;"id="content">
<!-- When the quiz is finished then its result will be displayed here -->
    @if(ViewBag.finish == "yes")
    {
<br><buttontype="button"style="float:right; margin-right:15px;"class="btn btn-
success"onclick="location.href='@Url.Action("StartQuiz", "Student",
null)'"><spanclass="glyphicon glyphicon-backward"></span> Back to Start
Quiz</button><br><br>
<divstyle="border: 1pxsolidsilver; margin:10px10px0px10px; padding: 10px; border-
radius: 4px; background-color:white;">

```

```

<tableclass="table table-bordered">
<tr>
<th><h3style="font-family:Calibri; text-align:center; font-size:18px"><b>Obtained
Marks</b></h3></th>
<th><h3style="font-family:Calibri; text-align:center; font-size:18px"><b>Total
Marks</b></h3></th>
</tr>
<tr>
<td><pstyle="font-family:Calibri; text-align:center; font-
size:15px"><b>@ ViewBag.ObtainedMarks</b></p></td>
<td><pstyle="font-family:Calibri; text-align:center; font-
size:15px"><b>@ ViewBag.TotalMarks</b></p></td>
</tr>
</table>
</div>
}
elseif (ViewBag.Question == null)
{
<!-- this will show a message for telling that there are no quiz is scheduled against this
course -->
<br><buttontype="button"style="float:right; margin-right:15px;"class="btn btn-
success"onclick="location.href='@Url.Action("StartQuiz", "Student",

```

```

null)""><spanclass="glyphicon glyphicon-backward"></span> Back to Start

Quiz</button><br><br>

<divstyle="border: 1pxsolidsilver; margin:10px10px0px10px; padding: 10px; border-
radius: 4px; background-color:white;">

<table>

<tr>

<td><pstyle="font-family:Calibri; font-size:18px"><b>No quiz scheduled against this
course !</b></p></td>

</tr>

</table>

</div>

}

elseif (ViewBag.Question != null)

{

<br><h3style="margin-left:10px; text-align:center;">Quiz</h3>

<divstyle="border: 1pxsolidsilver; margin:10px10px0px10px; padding: 10px; border-
radius: 4px; background-color:white;">

<formaction="/Student/Quiz"method="post">

<inputtype="text" name="QuizID" value="@ViewBag.QuizID"hidden/>

<tableclass="table table-borderless">

<tr>

```

```

<tdstyle="width:10px; overflow:hidden;">
</td>
<!-- Quiz's question will be displayed here -->
<tdstyle="overflow:hidden;text-overflow:ellipsis;">
<h3><b>Question:</b></h3><br>
<center><pstyle="font-family:Calibri;font-
size:18px;">@ViewBag.Question.Question</p></center><inputtype="text"value="@Vie
wBag.Question.ID"name="QuestionID"hidden/>
</td>
</tr>
<!-- Question's options will be displayed here -->
        @foreach (var o in ViewBag.Options)
        {
<tr>
<tdstyle="text-align:right;">
<inputtype="radio"value="@o.ID"name="SelectedOption"/>
</td>
<td>
<pstyle="font-family:Calibri;font-size:15px;">@o.Option</p>
</td>
</tr>
        }
</tr>

```

```
<td></td>

<td>

<center><button type="submit" class="btn btn-success">Next Question
<span class="glyphicon glyphicon-forward"></span></button></center>

</td>

</tr>

</table>

</form>

</div>

}

</div>
```

DownloadFile.cshtml

```
@model List<string>
```

```
@{
```

```
    ViewBag.Title = "LearningMadeEasy4u";
```

```
    Layout = "~/Views/Shared/StudentLayout.cshtml";
```

```
}
```

```

<divclass="st-pusher well"style="overflow:scroll;"id="content">
<br><h3style="margin-left:10px; text-align:center;">Download Files</h3>
<divstyle="border: 1pxsolidsilver; margin:10px10px0px10px; padding: 10px; border-
radius: 4px; background-color:white;">
<tablestyle="width:100%;"class="table table-bordered">
<tr>
<th>File Name</th>
<th>Link</th>
</tr>
    @for (var i = 0; i <= Model.Count - 1; i++)
    {
<tr>
<!-- File Name -->
<td>
        @Model[i].ToString()
</td>
<!-- File Download Link -->
<td>
        @Html.ActionLink("Download", "Download", new { FileName =
        @Model[i].ToString() })
</td>
</tr>
    }

```

```
</table>
```

```
</div>
```

```
</div>
```

Controllers

HomeController.cs

```
/// home page before login for LME4U

using E_Learning_Managment_System.Models;

using E_Learning_Managment_System.Models.ViewModel;

using E_Learning_Managment_System.ViewModel;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

namespace E_Learning_Managment_System.Controllers

{

    public class HomeController : Controller

    {

        DatabaseOperations db = new DatabaseOperations();
```



```
public ActionResult Index()
{
    return View();
}

/// function used to give name suggestions in messaging
[HttpPost]
public JsonResult AutocompleteSuggestions(string prefix, string senderType)
{
    if(Session["StudentID"] != null)
    {
        var userId = Convert.ToInt32(Session["StudentID"]);
        var suggestions = db.Autocomplete(prefix, userId, senderType);
        return Json(suggestions);
    }
    else if(Session["InstructorID"] != null)
    {
        var userId = Convert.ToInt32(Session["InstructorID"]);
        var suggestions = db.Autocomplete(prefix, userId, senderType);
        return Json(suggestions);
    }
    return Json(null);
}
```

```
/// function used to send the message

[HttpPost]

public void SendMessage(MessageViewModel model)

{

    // Save messages

    var message = MaptoMessage(model);

    db.PostMessage(message);

}

/// <summary>

/// function used to get old messages from the chat with a specific person

/// </summary>

/// <param name="model"></param>

/// <returns></returns>

[HttpPost]

public JsonResult GetMessages(MessageViewModel model)

{

    var message = MaptoMessage(model);

    var messages = db.GetMessages(message);

    List<MessageViewModel> messagesList = new List<MessageViewModel>();

    if (messages != null)
```

```
{
    foreach (var item in messages)
    {
        MessageViewModel messageModel = new MessageViewModel();
        messageModel.messageBody = item.MessageBody;
        messageModel.dateTime = DateTimeConverter(item.DateTime);
        messageModel.name = item.SenderName;
        messagesList.Add(messageModel);
    }
}

return Json(messagesList);
}

/// <summary>
/// function to get latest message from all users
/// </summary>
/// <param name="model"></param>
/// <returns></returns>
[HttpPost]
public JsonResult GetAllMessages(MessageViewModel model)
{
    var message = MaptoAllMessages(model);
```

```
var messages = db.GetAllMessages(message);

List<MessageViewModel> messagesList = new List<MessageViewModel>();

if(messages.Count > 0)
{
    foreach (var item in messages)
    {
        var id = 0;

        var name = "";

        var userType = "";

        if(model.senderType == "student")
        {
            if (model.id == item.SenderStudentId)
            {
                if (item.RecieverStudentId != null)
                {
                    id = item.RecieverStudentId ?? default(int);

                    name = item.RecieverName;

                    userType = "student";
                }

                else if(item.RecieverInstructorId != null)
                {
                    id = item.RecieverInstructorId ?? default(int);

                    name = item.RecieverName;
                }
            }
        }
    }
}
```

```
        userType = "instructor";
    }
}
else if(model.id == item.RecieverStudentId )
{
    if (item.SenderStudentId != null)
    {
        id = item.SenderStudentId ?? default(int);
        name = item.SenderName;
        userType = "student";
    }
    else if (item.SenderInstructorId != null)
    {
        id = item.SenderInstructorId ?? default(int);
        name = item.SenderName;
        userType = "instructor";
    }
}
else if (model.senderType == "instructor")
{
    if (model.id == item.SenderInstructorId)
    {
```

```
if (item.RecieverInstructorId != null)
{
    id = item.RecieverInstructorId ?? default(int);
    name = item.RecieverName;
    userType = "instructor";
}
else if (item.RecieverStudentId != null)
{
    id = item.RecieverStudentId ?? default(int);
    name = item.RecieverName;
    userType = "student";
}
}
else if (model.id == item.RecieverInstructorId)
{
    if (item.SenderInstructorId != null)
    {
        id = item.SenderInstructorId ?? default(int);
        name = item.SenderName;
        userType = "instructor";
    }
    else if (item.SenderStudentId != null)
    {
```

```
        id = item.SenderStudentId ?? default(int);  
        name = item.SenderName;  
        userType = "student";  
    }  
}  
  
MessageViewModel messageModel = new MessageViewModel();  
messageModel.messageBody = item.MessageBody;  
messageModel.dateTime = DateTimeConverter(item.DateTime);  
messageModel.id = id;  
messageModel.name = name;  
messageModel.recieverType = userType;  
messageModel.senderType = model.senderType;  
messagesList.Add(messageModel);  
}  
}  
  
return Json(messagesList);  
}
```

/// function for mapping model for student or for instructor on all messages page

```
private Messages MaptoAllMessages(MessageViewModel model)  
{
```

```
Messages message = new Messages();  
if(model.senderType == "student")  
{  
    message.SenderStudentId = model.id;  
    message.RecieverStudentId = model.id;  
}  
else if(model.senderType == "instructor")  
{  
    message.SenderInstructorId = model.id;  
    message.RecieverInstructorId = model.id;  
}  
return message;  
}
```

```
/// <summary>
```

```
/// function for mapping model for student or for instructor for messaging with a  
specific user
```

```
private Messages MaptoMessage(MessageViewModel model)  
{  
    Messages message = new Messages();  
    message.DateTime = DateTime.UtcNow;  
    message.MessageBody = model.messageBody;  
    message.MessageId = model.messageId;
```



```
if (model.recieverType == "student")
{
    message.RecieverStudentId = model.id;
}
else if (model.recieverType == "instructor")
{
    message.RecieverInstructorId = model.id;
}

if (model.senderType == "student")
{
    message.SenderStudentId = Convert.ToInt32(Session["StudentID"]);
}
else if (model.senderType == "instructor")
{
    message.SenderInstructorId = Convert.ToInt32(Session["InstructorID"]);
}

message.SenderName = Session["Name"].ToString();
message.RecieverName = model.name;
return message;
}
}
```

Instructorcontroller.cs

```
/// homepage of instructor's module for LME4U

using E_Learning_Managment_System.Models;
using E_Learning_Managment_System.Models.ViewModel;
using E_Learning_Managment_System.ViewModel;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace E_Learning_Managment_System.Controllers
{
    public class InstructorController : Controller
    {
        DatabaseOperations obj = new DatabaseOperations();
        DbConnect db = new DbConnect();
    }
}
```

```
public ActionResult Index()
{
    if (Session["InstructorID"] != null)
    {
        List<Course> courseList = new List<Course>();

        var instructorId = (int)Session["InstructorID"];

        var courses = db.InstructorCourseAssignment.Where(x => x.InstructorID ==
instructorId).ToList();

        foreach(var c in courses)
        {
            var course = db.Course.First(x => x.CourseID == c.CourseID);

            courseList.Add(course);
        }

        ViewBag.mycourses = courseList;

        return View();
    }
    else
    {
        return RedirectToAction("Login", "Instructor");
    }
}
```

```
public ActionResult Profile()
{
    if (Session["InstructorID"] != null)
    {
        Instructor instructor =
obj.SearchAndReturnInstructor(Convert.ToInt32(Session["InstructorID"]));
        ViewBag.password = TempData["Password"];
        return View(instructor);
    }
    else
    {
        return RedirectToAction("Login", "Instructor");
    }
}

/// Function to upload files
public ActionResult UploadFile()
{
    if (Session["InstructorID"] != null)
    {
        return View();
    }
    else
```

```
{  
    return RedirectToAction("Login", "Instructor");  
}  
}
```

[HttpPost]

```
public ActionResult UploadFile(HttpPostedFileBase file)  
{  
    if (file != null && file.ContentLength > 0)  
    {  
        var fileName = Path.GetFileName(file.FileName);  
        var path = Path.Combine(Server.MapPath("~/App_Data/Files"), fileName);  
        file.SaveAs(path);  
    }  
    ViewBag.FileAlert = "Your file has been added successfully !";  
    return View();  
}
```

```
/// <summary>
```

```
/// login page of instructor's module
```

```
/// </summary>
```

```
/// <returns></returns>
```

```
public ActionResult Login()
```

```
{  
    if (Session["InstructorID"] != null)  
    {  
        return RedirectToAction("Index", "Instructor");  
    }  
    else  
    {  
        return View();  
    }  
}
```

[HttpPost]

```
public ActionResult Login(LoginViewModel inst)  
{  
    Session.Remove("StudentID");  
    if (ModelState.IsValid)  
    {  
        try  
        {  
            var ins = db.Instructor.First(m => m.Email == inst.Email && m.password  
== inst.password);  
            Session["InstructorID"] = ins.ID;
```

```
        Session["Name"] = ins.FirstName + " " + ins.LastName;

        Session["IsStudent"] = false;

        return RedirectToAction("Index");
    }

    catch
    {
        ViewBag.Message = "Invalid Email or Password !";

        return View();
    }
}

return View();
}
```

/// Function for downloading the files

```
public ActionResult DownloadsFile()
{
    if (Session["InstructorID"] != null)
    {
        var dir = new System.IO.DirectoryInfo(Server.MapPath("~/App_Data/Files"));

        System.IO.FileInfo[] fileNames = dir.GetFiles("*.");

        List<string> items = new List<string>();
    }
}
```

```
        foreach (var file in fileNames)
        {
            items.Add(file.Name);
        }

        return View(items);
    }
    else
    {
        return RedirectToAction("Login", "Instructor");
    }
}

/// the function that actually downloads file
public ActionResult Download(string FileName)
{
    return new FilePathResult("~/App_Data/Files/" + FileName,
System.Net.Mime.MediaTypeNames.Application.Octet)
    {
        FileDownloadName = FileName
    };
}
```



```
/// signup function for instructor's module

public ActionResult SignUp()
{
    if (Session["InstructorID"] != null)
    {
        return RedirectToAction("Login", "Instructor");
    }
    else
    {
        return View();
    }
}

[HttpPost]
public ActionResult SignUp(InstructorSignUpViewModel inst)
{
    if (ModelState.IsValid)
    {
        Instructor i = new Instructor();
        i.FirstName = inst.FirstName;
        i.LastName = inst.LastName;
        i.Email = inst.Email;
        i.password = inst.password;
```

```
        i.Repassword = inst.Repassword;

        obj.InstructorSignUp(i);

        return RedirectToAction("Login", "Instructor");
    }

    //ViewBag.Message = "Something went wrong either you are already registered,
or you have entered wrong values";

    return View();
}

/// this function checks that if email already exists or not during signup
[HttpPost]
public JsonResult CheckExistingEmail(string Email)
{
    return Json(!db.Instructor.Any(a => a.Email.Equals(Email)));
}

/// function used to logout instructor from his account
public ActionResult Logout()
{
    Session.Abandon();

    return RedirectToAction("Login", "Instructor");
}
```

```
/// Function to exchange messages in the chat

public ActionResult Messages()
{
    if (Session["InstructorID"] != null)
    {
        return View();
    }
    else
    {
        return RedirectToAction("Login", "Instructor");
    }
}

/// show all the courses from database

public ActionResult ShowAllCourses()
{
    if (Session["InstructorID"] != null)
    {
        List<Course> courseList = new List<Course>();

        var allCourses = db.Course.ToList();

        foreach (var course in allCourses)
        {
```

```
        courseList.Add(course);
    }

    ViewBag.Courses = courseList;
    return View();
}

else
{
    return RedirectToAction("Login", "Instructor");
}
}

/// <summary>
/// shows the courses that are assigned to the instructor
/// </summary>
/// <returns></returns>
public ActionResult ShowMyCourses()
{
    if (Session["InstructorID"] != null)
    {
        List<Course> courseList = new List<Course>();
        var instructorId = (int)Session["InstructorID"];
        if (Session["InstructorID"] != null)
```

```
{  
    var myCourses = db.InstructorCourseAssignment.Where(x => x.InstructorID  
== instructorId).ToList();  
  
    foreach (var course in myCourses)  
    {  
        var searchedCourse = db.Course.First(x => x.CourseID ==  
course.CourseID);  
        courseList.Add(searchedCourse);  
    }  
}  
  
ViewBag.Courses = courseList;  
  
return View();  
}  
else  
{  
    return RedirectToAction("Login", "Instructor");  
}  
}
```

/// shows the list of quizzes made by instructor

```
public ActionResult ShowQuizes()
{
    if (Session["InstructorID"] != null)
    {
        List<Quiz> quizzesList = new List<Quiz>();

        int instructorID = (int)Session["InstructorID"];
        var instructorCourses = db.InstructorCourseAssignment.Where(x =>
x.InstructorID == instructorID).ToList();

        if (instructorCourses != null)
        {
            foreach (var course in instructorCourses)
            {
                try
                {
                    var quizzes = db.Quiz.Where(x => x.CourseID == course.CourseID);
                    foreach (var quiz in quizzes)
                    {
                        quizzesList.Add(quiz);
                    }
                }
            }
        }
        catch { }
```

```
    }  
  }  
  ViewBag.QuizAlert = TempData["quizalert"];  
  return View(quizesList);  
}  
else  
{  
  return RedirectToAction("Login", "Instructor");  
}  
}  
  
/// shows the questions of a specific quiz  
public ActionResult ShowQuizQuestions(int QuizID)  
{  
  if (Session["InstructorID"] != null)  
  {  
    List<Questions> questionsList = new List<Questions>();  
    var questions = db.QuizQuestions.Where(x => x.QuizID == QuizID).ToList();  
  
    foreach (var q in questions)  
    {  
      var a = db.Questions.First(x => x.ID == q.QuestionID);  
      questionsList.Add(a);  
    }  
  }  
}
```

```
    }

    return View(questionsList);
}
else
{
    return RedirectToAction("Login", "Instructor");
}
}

/// this function is used to generate quiz

public ActionResult QuizGenerator()
{
    if (Session["InstructorID"] != null)
    {
        List<Course> courseList = new List<Course>();

        int instructorID = (int)Session["InstructorID"];

        var instructorCourses = db.InstructorCourseAssignment.Where(x =>
x.InstructorID == instructorID).ToList();

        if (instructorCourses != null)
```



```
{
    foreach (var course in instructorCourses)
    {
        try
        {
            var q = db.Questions.First(x => x.CourseID == course.CourseID);
            if (q != null)
            {
                var c = db.Course.First(x => x.CourseID == course.CourseID);
                courseList.Add(c);
            }
        }
        catch { }
    }
}

return View(courseList);
}
else
{
    return RedirectToAction("Login", "Instructor");
}
}
```

/// this function is used to select the questions to be given in the quiz

```
public ActionResult SelectQuestions(string CourseID, string Title)
{
    if (Session["InstructorID"] != null)
    {
        var questions = db.Questions.Where(x => x.CourseID == CourseID).ToList();
        ViewBag.CourseID = CourseID;
        ViewBag.CourseTitle = Title;

        return View(questions);
    }
    else
    {
        return RedirectToAction("Login", "Instructor");
    }
}
```

[HttpPost]

```
public ActionResult SelectQuestions(int[] QuestionIDs, string CourseID, string
Title)
{
    Quiz quiz = new Quiz();
```

```
quiz.CourseID = CourseID;

quiz.Title = Title;

db.Quiz.Add(quiz);

db.SaveChanges();

foreach (var qID in QuestionIDs)
{
    QuizQuestions quizQuestion = new QuizQuestions();

    quizQuestion.QuestionID = qID;
    quizQuestion.QuizID = quiz.ID;

    db.QuizQuestions.Add(quizQuestion);

    db.SaveChanges();
}

TempData["quizalert"] = "Quiz has been created successfully !";

return RedirectToAction("ShowQuizes", "Instructor");
}

/// using this function, the instructor can add question into the subjects he is assigned
public ActionResult AddQuestions(string CourseID)
```

```
{  
    if (Session["InstructorID"] != null)  
    {  
        ViewBag.CourseID = CourseID;  
        return View();  
    }  
    else  
    {  
        return RedirectToAction("Login", "Instructor");  
    }  
}  
  
[HttpPost]  
public ActionResult AddQuestions(FormCollection fc)  
{  
    Questions question = new Questions();  
    Options option = new Options();  
  
    int check = Int32.Parse(fc["Check"]);  
  
    question.CourseID = fc["CourseID"];  
    question.Question = fc["Question"];
```

```
db.Questions.Add(question);  
  
db.SaveChanges();  
  
for (int i=1; i<=4; i++)  
{  
    option.Option = fc["Option"+i];  
    option.QuestionID = question.ID;  
    if(i == check)  
    {  
        option.Status = "correct";  
    }  
    else  
    {  
        option.Status = "wrong";  
    }  
  
    db.Options.Add(option);  
    db.SaveChanges();  
}  
  
ViewBag.CourseID = fc["CourseID"];  
  
return View();
```

```
}

/// Show the course's questions

public ActionResult ShowQuestions(string CourseID)
{
    if (Session["InstructorID"] != null)
    {
        var questions = db.Questions.Where(x => x.CourseID == CourseID).ToList();

        return View(questions);
    }
    else
    {
        return RedirectToAction("Login", "Instructor");
    }
}

/// page for confirming the old password

public ActionResult ChangePassword()
{
    if (Session["InstructorID"] != null)
    {
        return View();
    }
}
```

```
    }  
    else  
    {  
        return RedirectToAction("Login", "Instructor");  
    }  
}  
  
[HttpPost]  
public ActionResult ChangePassword(string password)  
{  
    int instructorID = (int)Session["InstructorID"];  
    if (Session["InstructorID"] != null)  
    {  
        var ins = db.Instructor.First(x => x.ID == instructorID);  
        if (ins.password == password)  
        {  
            Session["PasswordCheck"] = "yes";  
            return RedirectToAction("Change", "Instructor");  
        }  
    }  
    else  
    {  
        Session["PasswordCheck"] = "no";  
        ViewBag.wrongpassword = "Wrong password !";  
    }  
}
```

```
        return View();
    }
}

return View();
}

/// page to change the password
public ActionResult Change()
{
    if (Session["InstructorID"] != null)
    {
        var check = (string)Session["PasswordCheck"];
        if (check == "yes")
        {
            Session.Remove("PasswordCheck");
            return View();
        }
        return RedirectToAction("ChangePassword", "Instructor");
    }
    else
    {
        return RedirectToAction("Login", "Instructor");
    }
}
```



```
    }  
  
    [HttpPost]  
    public ActionResult change>PasswordViewModel ins)  
    {  
        int instructorID = (int)Session["InstructorID"];  
        if (Session["InstructorID"] != null)  
        {  
            var s = db.Instructor.First(x => x.ID == instructorID);  
  
            s.password = ins.password;  
            s.Repassword = ins.RePassword;  
  
            db.SaveChanges();  
            TempData["Password"] = "Password changed successfully !";  
            return RedirectToAction("Profile", "Instructor");  
        }  
        return View();  
    }  
  
    }  
}
```

Studentcontroller.cs

```
    /// home page of student module for LME4U

using E_Learning_Managment_System.Models;

using E_Learning_Managment_System.Models.ViewModel;

using E_Learning_Managment_System.ViewModel;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

namespace E_Learning_Managment_System.Controllers
{
    public class StudentController : Controller
    {
        DatabaseOperations obj = new DatabaseOperations();

        DbConnect db = new DbConnect();

        //variable to store the ids of questions during quiz

        public static List<int> rand_question = new List<int>();

        public static int marks;

        Random random = new Random();
```

```
public ActionResult Index()
{
    if (Session["StudentID"] != null)
    {
        List<Course> courseList = new List<Course>();
        List<Result> resultList = new List<Result>();

        var studentId = (int)Session["StudentID"];

        try
        {
            var courses = db.StudentCourseAssignment.Where(x => x.StudentID ==
studentId).ToList();

            foreach (var c in courses)
            {
                var course = db.Course.First(x => x.CourseID == c.CourseID);
                courseList.Add(course);

                try
                {
                    var quizzes = db.Quiz.Where(x => x.CourseID ==
course.CourseID).ToList();

                    foreach (var q in quizzes)
                    {
                        try
```

```
    {  
        var results = db.Result.Where(x => x.QuizID == q.ID &&  
x.StudentID == studentId).ToList();  
        foreach (var r in results)  
        {  
            resultList.Add(r);  
        }  
    }  
    catch { }  
}  
}  
catch { }  
}  
}  
catch { }  
  
ViewBag.mycourses = courseList;  
ViewBag.results = resultList;  
  
return View();  
}  
else  
{
```

```
        return RedirectToAction("Login", "Student");
    }
}

/// Function for downloading files
public ActionResult DownloadsFile()
{
    if (Session["StudentID"] != null)
    {
        var dir = new System.IO.DirectoryInfo(Server.MapPath("~/App_Data/Files"));
        System.IO.FileInfo[] fileNames = dir.GetFiles("*.*.");
        List<string> items = new List<string>();

        foreach (var file in fileNames)
        {
            items.Add(file.Name);
        }

        return View(items);
    }
    else
    {
        return RedirectToAction("Login", "Student");
    }
}
```

```
    }  
}  
  
/// function for downloading the files  
public ActionResult Download(string FileName)  
{  
    return new FilePathResult("~/App_Data/Files/" + FileName,  
System.Net.Mime.MediaTypeNames.Application.Octet)  
    {  
        FileNameDownload = FileName  
    };  
}  
  
/// login for student module  
public ActionResult Login()  
{  
    if (Session["StudentID"] != null)  
    {  
        return RedirectToAction("Index", "Student");  
    }  
    else  
    {  
        return View();  
    }  
}
```

```
    }  
}  
  
[HttpPost]  
public ActionResult Login(LoginViewModel std)  
{  
    Session.Remove("InstructorID");  
    if (ModelState.IsValid)  
    {  
        try  
        {  
            DbConnect dbc = new DbConnect();  
            var s = dbc.Student.First(m => m.Email == std.Email && m.password ==  
std.password);  
            Session["StudentID"] = s.ID;  
            Session["Name"] = s.FirstName + " " + s.LastName;  
            return RedirectToAction("Index", "Student");  
        }  
        catch  
        {  
            ViewBag.Message = "Invalid Email or Password !";  
            return View();  
        }  
    }  
}
```

```
    }  
    return View();  
}  
  
/// signup page for student module  
public ActionResult SignUp()  
{  
    if (Session["StudentID"] != null)  
    {  
        return RedirectToAction("Login", "Student");  
    }  
    else  
    {  
        return View();  
    }  
}  
  
[HttpPost]  
public ActionResult SignUp(StudentSignUpViewModel std)  
{  
    if (ModelState.IsValid)  
    {  
        Student s = new Student();
```



```
s.FirstName = std.FirstName;

s.LastName = std.LastName;

s.Email = std.Email;

s.password = std.password;

s.RePassword = std.RePassword;

obj.StudentSignUp(s);

return RedirectToAction("Login", "Student");

}

//ViewBag.Message = "Something went wrong either you are already registered,
or you have entered wrong values";

return View();

}

/// this function checks that if email already exists or not during signup

[HttpPost]

public JsonResult CheckEmail(string Email)

{

    return Json(!db.Student.Any(a => a.Email.Equals(Email)));

}

/// funtion for logout

public ActionResult Logout()

{
```

```
        Session.Abandon();

        return RedirectToAction("Login","Student");
    }

    /// Function to view the student profile page
    public ActionResult Profile()
    {
        if (Session["StudentID"] != null)
        {
            Student std =
obj.SearchAndReturnStudent(Convert.ToInt32(Session["StudentID"]));

            ViewBag.password = TempData["Password"];

            return View(std);
        }
        else
        {
            return RedirectToAction("Login", "Student");
        }
    }

    /// page for chatting in student module
    public ActionResult Messages()
    {
```

```
if (Session["StudentID"] != null)
{
    return View();
}
else
{
    return RedirectToAction("Login", "Student");
}
}

/// this function is used to show all courses in student module
/// <returns>list of courses</returns>
public ActionResult ShowAllCourses()
{
    if (Session["StudentID"] != null)
    {
        //List<Course> semesterCourseList = new List<Course>();
        List<Course> courseList = new List<Course>();

        var allCourses = db.Course.ToList();
        foreach (var course in allCourses)
        {
```

```
        courseList.Add(course);
    }

    ViewBag.Courses = courseList;
    return View();
}
else
{
    return RedirectToAction("Login", "Student");
}
}
```

/// this function is used to show those courses to students in which they are registered

```
/// <returns>list of courses</returns>
public ActionResult ShowMyCourses()
{
    if (Session["StudentID"] != null)
    {
        //List<Course> semesterCourseList = new List<Course>();
        List<Course> courseList = new List<Course>();
        var studentId = (int)Session["StudentID"];
        if (Session["StudentID"] != null)
```

```
{  
    var myCourses = db.StudentCourseAssignment.Where(x => x.StudentID ==  
studentId).ToList();  
  
    foreach (var course in myCourses)  
    {  
        var searchedCourse = db.Course.First(x => x.CourseID ==  
course.CourseID);  
    }  
}  
  
ViewBag.Courses = courseList;  
  
return View();  
  
}  
else  
{  
    return RedirectToAction("Login", "Student");  
}  
}
```

```
/// this function is used to register a student in course

public ActionResult TakeCourse()
{
    if (Session["StudentID"] != null)
    {
        List<Course> allCourses = new List<Course>();

        List<Course> courses = new List<Course>();

        int studentID = (int)Session["StudentID"];

        allCourses = db.Course.ToList();

        courses = db.Course.ToList();

        var myCourses = db.StudentCourseAssignment.Where(x => x.StudentID ==
studentID);

        if (myCourses != null)
        {
            foreach (var i in allCourses)
            {
                foreach (var j in myCourses)
                {
                    if (i.CourseID == j.CourseID)
                    {
                        try
```

```
        {  
            var c = courses.First(x => x.CourseID == i.CourseID);  
            courses.Remove(c);  
        }  
        catch { }  
    }  
}  
}  
}  
}  
  
    return View(courses);  
}  
else  
{  
    return RedirectToAction("Login", "Student");  
}  
}  
  
[HttpPost]  
public ActionResult TakeCourse(string CourseID)  
{  
    StudentCourseAssignment studentCourseAssignment = new  
StudentCourseAssignment();
```

```

studentCourseAssignment.CourseID = CourseID;

studentCourseAssignment.StudentID = (int)Session["StudentID"];

db.StudentCourseAssignment.Add(studentCourseAssignment);

db.SaveChanges();

return RedirectToAction("ShowMyCourses","Student");
}

/// this shows all the quizzes of those subjects in which the student is enrolled
/// <returns>quizes list</returns>
public ActionResult StartQuiz()
{
    if (Session["StudentID"] != null)
    {

        List<Quiz> quizList = new List<Quiz>();

        var studentId = (int)Session["StudentID"];

        var myCourses = db.StudentCourseAssignment.Where(x => x.StudentID ==
studentId).ToList();

        foreach (var course in myCourses)
        {

            var searchedCourse = db.Course.First(x => x.CourseID ==
course.CourseID);

```



```
var quizzes = db.Quiz.Where(x => x.CourseID ==  
searchedCourse.CourseID);
```

```
    foreach (var q in quizzes)  
    {  
        quizList.Add(q);  
    }  
}  
ViewBag.Quizes = quizList;  
  
return View();  
}  
else  
{  
    return RedirectToAction("Login", "Student");  
}  
}
```

```
/// this function is used to conduct the quiz
```

```
/// <param name="QuizID"></param>
```

```
/// <returns>question</returns>
```

```
public ActionResult Quiz(int QuizID)
```

```
{
```

```
if (Session["StudentID"] != null)
{
    try
    {
        var quiz = db.Quiz.First(x => x.ID == QuizID);

        int quizID = quiz.ID;

        marks = 0;

        rand_question.Clear();

        var total_questions = db.QuizQuestions.Where(x => x.QuizID == quiz.ID);
        var count_total_questions = total_questions.Count();
        int count_questions = rand_question.Count();

        int number;

        do
        {
            number = random.Next(1, (count_total_questions + 1));
        } while (rand_question.Contains(number));

        rand_question.Add(number);

        var question = total_questions.OrderBy(a => a.QuestionID).Skip(number -
1).Take(1).ToList();
```

```
foreach (var s in question)
{
    ViewBag.Question = db.Questions.First(x => x.ID == s.QuestionID);
    var options = db.Options.Where(x => x.QuestionID == s.QuestionID);
    List<Options> optionList = new List<Options>();
    foreach (var o in options)
    {
        optionList.Add(o);
    }
    ViewBag.Options = optionList;
}
ViewBag.QuizID = quizID;
}
catch
{
    ViewBag.Question = null;
}
return View();
}
else
{
    return RedirectToAction("Login", "Student");
}
```

```

    }
}

[HttpPost]
public ActionResult Quiz(int QuizID, int QuestionID, int? SelectedOption)
{
    var correctOption = db.Options.First(x => x.QuestionID == QuestionID &&
x.Status == "correct");
    if(SelectedOption.HasValue)
    {
        if (correctOption.ID == SelectedOption)
        {
            marks = marks + 1;
        }
    }

    var total_questions = db.QuizQuestions.Where(x => x.QuizID == QuizID);
    var count_total_questions = total_questions.Count();
    int count_questions = rand_question.Count();

    if (count_questions < count_total_questions)
    {
        int number;

```

```
do
{
    number = random.Next(1, (count_total_questions + 1));
} while (rand_question.Contains(number));

rand_question.Add(number);

int pick = number - 1;

var question = total_questions.OrderBy(a =>
a.QuestionID).Skip(pick).Take(1).ToList();

foreach (var s in question)
{
    ViewBag.Question = db.Questions.First(x => x.ID == s.QuestionID);
    var options = db.Options.Where(x => x.QuestionID == s.QuestionID);
    List<Options> optionList = new List<Options>();
    foreach (var o in options)
    {
        optionList.Add(o);
    }
    ViewBag.Options = optionList;
}
```

```
ViewBag.QuizID = QuizID;

ViewBag.finish = "no";

}

else

{

    ViewBag.finish = "yes";

    ViewBag.TotalMarks = count_total_questions;

    ViewBag.ObtainedMarks = marks;

    var studentId = (int)Session["StudentID"];

    var quiz = db.Quiz.First(x => x.ID == QuizID);

    var course = db.Course.First(x => x.CourseID == quiz.CourseID);

    var instructor = db.InstructorCourseAssignment.First(x => x.CourseID ==
course.CourseID);

    Result result = new Result();

    result.QuizID = QuizID;

    result.StudentID = studentId;

    result.InstructorID = instructor.InstructorID;

    result.CourseID = course.CourseID;

    result.TotalMarks = ViewBag.TotalMarks;

    result.ObtainedMarks = ViewBag.ObtainedMarks;
```

```
        db.Result.Add(result);

        db.SaveChanges();
    }

    return View();
}

/// this function is used to show quiz result after the quiz is finished
public ActionResult QuizResult()
{

    if (Session["StudentID"] != null)
    {

        var studentId = (int)Session["StudentID"];

        var result = db.Result.Where(x => x.StudentID == studentId);

        return View(result);
    }

    else
    {

        return RedirectToAction("Login", "Student");
    }

}

/// page to confirm old password
```

```
public ActionResult ChangePassword()
{
    if (Session["StudentID"] != null)
    {
        return View();
    }
    else
    {
        return RedirectToAction("Login", "Student");
    }
}

[HttpPost]
public ActionResult ChangePassword(string password)
{
    int studentID = (int)Session["StudentID"];
    if (Session["StudentID"] != null)
    {
        var std = db.Student.First(x => x.ID == studentID);
        if (std.password == password)
        {
            Session["PasswordCheck"] = "yes";
            return RedirectToAction("Change", "Student");
        }
    }
}
```



```
    }  
    else  
    {  
        Session["PasswordCheck"] = "no";  
        ViewBag.wrongpassword = "Wrong password !";  
        return View();  
    }  
}  
return View();  
}
```

```
/// <summary>
```

```
/// function to change the student's account password
```

```
/// </summary>
```

```
/// <returns></returns>
```

```
public ActionResult Change()
```

```
{  
    if (Session["StudentID"] != null)  
    {  
        var check = (string)Session["PasswordCheck"];  
        if (check == "yes")  
        {  
            Session.Remove("PasswordCheck");  
        }  
    }  
}
```

```
        return View();
    }

    return RedirectToAction("ChangePassword", "Student");
}

else
{
    return RedirectToAction("Login", "Student");
}
}
```

[HttpPost]

```
public ActionResult change(PasswordViewModel std)
{
    int studentID = (int)Session["StudentID"];
    if (Session["StudentID"] != null)
    {
        var s = db.Student.First(x => x.ID == studentID);

        s.password = std.password;
        s.RePassword = std.RePassword;

        db.SaveChanges();

        TempData["Password"] = "Password changed successfully !";
    }
}
```

```

        return RedirectToAction("Profile","Student");
    }
    return View();
}
}
}

```

Description of Code

Database Updates:

- Tables in DB(Course, InstructorCourseAssignment, StudentCourseAssignment, Question, Option, Quiz, Result)

Course Module:

- In Instructor module, the instructor can view all of the courses from DB by clicking on the “Courses List” in navigation bar.(Action method in InstructorController = ShowAllCourses()).
- In Instructor module, the instructor can view the courses that are assigned to him by clicking on the “My Courses” in left side bar. (Action method in InstructorController = ShowMyCourses())
- In “My Courses” the instructor can add new questions with options inside the course and can show the questions of a course. (Action method in InstructorController = AddQuestions())

- In Student module, the student can also view all of the courses from DB by clicking on the “Courses List” in navigation bar. (Action method in StudentController = ShowAllCourses())
- In Student module, the student can also view all of the courses he/she has taken by clicking on the “My Courses” in left side bar. (Action method in StudentController = ShowMyCourses())
- The Student can also take courses the he hasn’t taken already, as a drop-down list is provided on the page to select a course. (Action method in StudentController = TakeCourse())

Quiz Module:

- In Instructor module, the instructor can generate the quiz against a course using “Quiz Generator” page and there can only be one quiz against one course. In this, the instructor has to select the questions from the question bank of that course. (Action method in InstructorController = QuizGenerator(), SelectQuestions())
- In student module, the student can give quiz of every course he/she has taken. The result will be shown to the student at the end of quiz. (Action method in StudentController = Quiz(), StartQuiz())

Messaging Module:

- “DatabaseOperations” viewmodel and HomeController’s action methods
- Student can send messages to instructors and vice versa. (Action method in StudentController and InstructorController = Messages())

File Uploading/Downloading Module:

- In Instructor module, the instructor can upload file using UploadFile() action method in InstructorController.
- In student module, the student can download file using DownloadsFile() and Download() action method in StudentController.