10-2016

Implementation of Course Scoring System Based on Spring MVC and Hibernate

Mustafa Ahmed
ahmu1401@stcloudstate.edu

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

Part of the Computer Sciences Commons

Recommended Citation
Ahmed, Mustafa, "Implementation of Course Scoring System Based on Spring MVC and Hibernate" (2016). Culminating Projects in Computer Science and Information Technology. 13.
https://repository.stcloudstate.edu/csit_etds/13
Implementation of Course Scoring System Based on Spring MVC and Hibernate

by

Mustafa Ahmed

A Starred Paper

Submitted to the Graduate Faculty of

St. Cloud State University

in Partial Fulfillment of the Requirements

for the Degree of

Master of Science

in Computer Science

October, 2016

Starred Paper Committee:
Dr. Jie Hu Meichsner, Chairperson
Dr. Omar Al-Azzam
Dr. David Robinson
Abstract

With the growing complexity of enterprise applications, maintainability and extensibility became an expensive task. Model-View-Controller (MVC) architecture became a solution to these problems as it isolates the application logic from the presentation layer. Spring MVC became popular as it realized the MVC architecture for developing enterprise level applications. Hibernate is a powerful, high performance Object-Relational Persistence and Query service for Java applications. Applications developed using Spring MVC and Hibernate are complete, modular and easy to use. In this project, a Course Scoring System is implemented based on both Spring MVC and Hibernate architectures. JSP, HTML and CSS are used in the presentation layer, Spring MVC handles the application logic and Hibernate manages the data persistence. The system can be used to organize a teacher selection competition at the University level. Assigning of judges/ voter, registration of teachers to a competition, assigning scores to the teachers and winner selection are the different features of the application. This project would be a good reference for anyone trying to gain practical knowledge about Spring MVC and Hibernate architectures and build a real time web application using them.
Table of Contents

Chapter 1 INTRODUCTION ............................................................................................................. 11

Chapter 2 TECHNOLOGY OVERVIEW ............................................................................................. 14

2.1 Spring Framework ..................................................................................................................... 14

2.2 Spring MVC .................................................................................................................................. 16

2.2.1 Spring MVC Architecture ..................................................................................................... 17

2.2.2 Features of Spring MVC framework ..................................................................................... 18

2.2.3 Advantages of Spring MVC framework ............................................................................... 19

2.3 Hibernate ....................................................................................................................................... 20

2.3.1 Hibernate Architecture .......................................................................................................... 21

2.3.2 Features of Hibernate ............................................................................................................ 23

2.3.3 Advantages of Hibernate ...................................................................................................... 23

Chapter 3 COURSE SCORING SYSTEM ......................................................................................... 24

3.1 System Functionality .................................................................................................................. 24

3.2 System Design ............................................................................................................................ 26

3.2.1 Use Case Diagrams ............................................................................................................... 26

3.2.2 Statecharts ............................................................................................................................ 35

3.2.3 Use Case Realization (Scenarios and Sequence Diagrams) .................................................. 39

3.2.4 Class Diagrams .................................................................................................................... 67
3.2.5 Application Architecture ................................................................. 76

3.3 Application Screenshots........................................................................... 78

Chapter 4: CONCLUSION ............................................................................. 88

References.................................................................................................... 89

Bibliography ............................................................................................... 90
List of Tables

Table 1: Login use case description for the Course Scoring System .......................................................... 27
Table 2: Manage Competition Details use case description for the Course Scoring System ...... 28
Table 3: Manage Users use case description for the Course Scoring System ............................ 29
Table 4: Register Teacher use case description for the Course Scoring System ........................... 30
Table 5: Query Scores use case description for the Course Scoring System .................................. 32
Table 6: Assign Score use case description for the Course Scoring System ............................... 33
Table 7: Vote for better Teacher use case description for the Course Scoring System .............. 34
Table 8: ‘Admin login - Success’ scenario of Login use case ............................................................... 41
Table 9: ‘Admin login – Failure’ scenario of Login use case ............................................................... 42
Table 10: ‘Judge login – Success’ scenario of Login use case ............................................................... 44
Table 11: ‘Judge login – Failure’ scenario of Login use case ............................................................... 45
Table 12: ‘Voter login – Success’ scenario of Login use case ............................................................... 46
Table 13: ‘Voter login – Failure’ scenario of Login use case ............................................................... 47
Table 14: ‘Create New Competition’ scenario of Manage Competition Details use case ............. 48
Table 15: ‘Update Competition’ scenario of Manage Competition Details use case .................. 50
Table 16: ‘Delete Competition’ scenario of Manage Competition Details use case ................. 51
Table 17: ‘Update Teacher Order’ scenario of Manage Competition Details use case ............ 53
Table 18: ‘Create New User’ scenario of Manage Users use case ..................................................... 54
Table 19: ‘Update User Info’ scenario of Manage Users use case ..................................................... 56
Table 20: ‘Search Users’ scenario of Manage Users use case ............................................................. 57
Table 21: ‘Register New Teacher’ scenario of Register Teacher use case ..................................... 58
Table 22: ‘Edit Teacher Info’ scenario of Register Teacher use case ........................................ 60
Table 23: ‘Query Competition Summary’ scenario of Query Scores use case ............................... 61
Table 24: ‘Assign Scores to Teachers’ scenario of Assign Score use case ........................................ 63
Table 25: ‘Select Winner’ scenario of Vote for better Teacher use case ........................................... 65
List of Figures

Figure 1: Modules of Spring Framework ................................................................. 14

Figure 2: High level view of request processing workflow in Spring MVC ............... 16

Figure 3: Spring MVC Architecture ......................................................................... 17

Figure 4: ORM Mapping Concept ........................................................................... 20

Figure 5: Role of Hibernate in Java application ...................................................... 21

Figure 6: Hibernate Architecture ........................................................................... 22

Figure 7: Use Case diagram for the Course Scoring System .................................... 26

Figure 8: Login use case for the Course Scoring System ......................................... 27

Figure 9: Manage Competition Details use case for the Course Scoring System ....... 28

Figure 10: Manage Users use case for the Course Scoring System ......................... 29

Figure 11: Register Teacher use case for the Course Scoring System ..................... 30

Figure 12: Query Scores use case for the Course Scoring System .......................... 31

Figure 13: Assign Score use case for the Course Scoring System ......................... 33

Figure 14: Vote for better Teacher use case for the Course Scoring System .......... 34

Figure 15: Complete Statechart of the Course Scoring System .............................. 36

Figure 16: Statechart of the Admin Subsystem ...................................................... 37

Figure 17: Statechart of the Judge Subsystem ....................................................... 38

Figure 18: Statechart of the Voter Subsystem ....................................................... 38

Figure 19: Different Classes for the Course Scoring System .................................... 40

Figure 20: Interaction of Classes to realize the Login use case ............................... 41

Figure 21: Sequence diagram for ‘Admin login – Success’ scenario ....................... 42
Figure 22: Sequence diagram for ‘Admin login – Failure’ scenario ................................................. 43
Figure 23: Sequence diagram for ‘Judge login – Success’ scenario .................................................. 44
Figure 24: Sequence diagram for ‘Judge login – Failure’ scenario .................................................. 45
Figure 25: Sequence diagram for ‘Voter login – Success’ scenario .................................................. 46
Figure 26: Sequence diagram for ‘Voter login – Failure’ scenario .................................................. 47
Figure 27: Interaction of Classes to realize the Manage Competition Details use case ................ 48
Figure 28: Sequence diagram for ‘Create New Competition’ scenario ............................................. 49
Figure 29: Sequence diagram for ‘Update Competition’ scenario .................................................... 51
Figure 30: Sequence diagram for ‘Delete Competition’ scenario ..................................................... 52
Figure 31: Sequence diagram for ‘Update Teacher Order’ scenario ............................................... 53
Figure 32: Interaction of Classes to realize the Manage Users use case .......................................... 54
Figure 33: Sequence diagram for ‘Create New User’ scenario ....................................................... 55
Figure 34: Sequence diagram for ‘Update User Info’ scenario ....................................................... 56
Figure 35: Sequence diagram for ‘Search Users’ scenario .............................................................. 57
Figure 36: Interaction of Classes to realize the Register Teacher use case ...................................... 58
Figure 37: Sequence diagram for ‘Register New Teacher’ scenario ................................................ 59
Figure 38: Sequence diagram for ‘Edit Teacher Info’ scenario ....................................................... 60
Figure 39: Interaction of Classes to realize the Query Scores use case ............................................ 61
Figure 40: Sequence diagram for ‘Query Competition Summary’ scenario ..................................... 62
Figure 41: Interaction of Classes to realize the Assign Score use case ............................................ 63
Figure 42: Sequence diagram for ‘Assign Scores to Teachers’ scenario ......................................... 64
Figure 43: Interaction of Classes to realize the Vote for better Teacher use case ......................... 65
Figure 44: Sequence diagram for ‘Select Winner’ scenario ........................................ 66
Figure 45: Complete Class diagram for Course Scoring System .................................. 67
Figure 46: Part of the overall class diagram showing Login Controller .......................... 68
Figure 47: Part of the overall class diagram showing Admin Controller ......................... 69
Figure 48: Part of the overall class diagram showing Scoring Controller ......................... 70
Figure 49: Part of the class diagram showing Login Controller with attributes and methods .... 71
Figure 50: Part of the class diagram showing Admin Controller with attributes and methods ... 72
Figure 51: Part of the class diagram showing Scoring Controller with attributes and methods... 73
Figure 52: Part of the class diagram showing Model Classes with attributes and methods ....... 75
Figure 53: Course Scoring System Architecture .......................................................... 76
Figure 54: Login Screen of the Course Scoring System ............................................... 79
Figure 55: Admin Screen of the Course Scoring System ............................................... 79
Figure 56: Manage Users – Create new user ................................................................. 80
Figure 57: Manage Users – Success message after creating a new user ............................ 80
Figure 58: Password Reset option for new user ............................................................ 81
Figure 59: Manage Users – Search users ....................................................................... 81
Figure 60: Manage Competition Details – Create new Competition ............................... 82
Figure 61: Manage Competition Details – New competition created successfully ............... 82
Figure 62: Manage Competition Details – Update teacher order ..................................... 83
Figure 63: Register new teacher .................................................................................... 83
Figure 64: Success message after registering a teacher ..................................................... 84
Figure 65: Judge Screen ............................................................................................... 84
Figure 66: Assign Scores to participants .............................................................. 85
Figure 67: Assign Scores to participants– alert message........................................ 85
Figure 68: Assign Scores to participants– scores saved successfully............................. 86
Figure 69: Voter Screen – voting needed .................................................................. 86
Figure 70: Voter Screen – winner selected by voting..................................................... 87
Chapter 1 INTRODUCTION

In a traditional Java based web application, Java Server Pages (JSP) are responsible for all the activities like receiving request, processing the business logic, choosing the next page and sending the response back to the user. So a small change in the application feature would need the changes in existing JSP code, which may more likely cause regression faults (change in one part of the code inducing errors in some other part of the code). With the increasing complexity of the enterprise applications, this approach became expensive for the maintenance and extension of the applications. As a solution to this problem, the Model- View-Controller (MVC) architecture gained popularity.

In the MVC architecture, different aspects of an application are as below:

- Model part consists of the application data
- View part is responsible for rendering the model data and generating HTML output that browsers can display
- Controller is responsible for processing user requests, building appropriate models and passing them to the view for rendering

In an MVC architecture, no change would be needed in the view part if the business logic needs to be changed or vice versa, thereby enhancing the maintainability and extensibility of the applications. Two frameworks – Struts2 and Spring MVC became popular as they realized this MVC architecture for building the enterprise level applications.

In this project, Spring MVC is preferred over the Struts2 framework as it has some general advantages compared to the Struts2 framework:
- Spring MVC provides a clear separation of roles between controllers, JavaBean models and views.
- Spring MVC offers better integration with other view technologies, other than JSP, like Velocity, FreeMarker, etc.
- The number of configurations needed in Spring MVC is lesser than that in the Struts2, increasing the development efficiency.
- By design, Struts2 makes class-level interceptors and Spring MVC makes method-level interceptors. Struts2 injects all the attributes of the class which the Action is in but Spring MVC only injects the request data of current Action. So the operating efficiency of Spring MVC is higher than that of Struts2.
- Struts2 imposes dependencies on the controllers (that it must extend a Struts class) but Spring MVC does not do this, instead convenience controllers are available to extend if needed.
- Struts2 forces the Action and Form objects into concrete inheritance (taking away the chance of inheriting further desired classes, as only one class can be explicitly inherited in Java), but Spring MVC is based on interfaces making it more flexible to inherit the desired class.

Although MVC architecture frameworks solved many development issues, the interaction of the application directly with the database was still a bottleneck for the application’s performance. The reason being that Object-oriented languages like Java and C# represent data in the form of objects, whereas traditional database systems store data in the form of tables. So data storage and retrieval from an object oriented language to a traditional database system was complex and an inefficient
As an efficient solution, Object-Relational Mapping (ORM) frameworks gained importance. ORM is a programming technique that maps the application objects to the data stored in the database. In other words, they introduce a direct mapping mechanism between the object and the database. Hibernate is one such ORM solution, which is used in this project. As Hibernate has in-built support for wide variety of relational databases, the application can be easily configured to use different relational database in the future.

The application ‘Course Scoring System’ can be used at a university level to organize a teacher selection competition for a given course. This system has the advantages offered by both Spring MVC and Hibernate architectures as its design is based on the integration of these two frameworks. The application features are based on the ideas proposed in the paper ‘Design and Implementation of High-quality Course Scoring System Based on Struts and Spring and Hibernate Architecture’ [1]. The features of the application include management of users (judges/ voter/ admin) and competitions, registration of the teachers in a competition, assigning scores to the teachers and finally voting for the better teacher if there is a tie for the winning position at the end of the competition.

The rest of this document is organized as follows: Chapter 2 gives an overview of the Spring and Hibernate frameworks. Chapter 3 presents the system design and functionalities of the Course Scoring System along with the application screenshots. Chapter 4 concludes the project and proposes possible future enhancements. Finally, the references for this project are listed.
Chapter 2 TECHNOLOGY OVERVIEW

This chapter presents detailed information about the Spring and Hibernate frameworks – their overview, features and advantages. The first section of this chapter explains the Spring framework, the second section explains the Spring MVC framework and the third section explains the Hibernate framework.

2.1 Spring Framework

Spring is an open source framework created to address the complexity of enterprise application development. It is lightweight, has negligible processing overhead and uses Java beans. It is modular and complete due to its layered architecture. Spring based applications are loosely coupled wherein their dependencies are listed in configuration files and injected by the framework. Spring framework is a collection of modules, allowing the programmers to choose the modules which are needed for their applications.

The different modules provided by the Spring framework are shown in Figure 1. This figure is adapted from ‘Spring Framework Reference Documentation’ [2].

Figure 1: Modules of Spring Framework
A brief description of the most commonly used Spring modules [2] is provided below:

- **Core** module provides the Inversion of Control, Dependency Injection and other fundamental features.
- **Bean** module is the implementation of the factory pattern.
- **Context** module is a medium to access any objects defined and configured.
- **SpEL** module is an extension of unified expression language that is used to manipulate the object graphs at runtime.
- **JDBC** module provides a JDBC-abstraction layer that eliminates the need for the programmer to write complex code to handle database specific error codes.
- **ORM** module provides layers for the integration of ORM tools like JPA, Hibernate and iBatis.
- **OXM** module provides an abstraction layer that supports Object/XML mapping implementations for JAXB, JiBX and XStream.
- **JMS** (Java Messaging Service) module provides the foundation for messaging-based applications.
- **Transaction** module provides declarative support for transaction management.
- **Web** module provides basic web-oriented integration features such as the Inversion of Control container initialization using servlet listeners.
- **Web-MVC** module provides Spring's model-view-controller (MVC) implementation for developing web applications.
- **Web-Socket** module provides support for web socket-based communication between the client and the server in web applications.
- **Web-Portlet** module depicts the functionality of Web-Servlet module and has the MVC implementation to be used in a portlet environment.

- **AOP** module provides aspect-oriented programming to cleanly decouple the code that implements different functionalities.

- **Test** module provides the support for testing the Spring components with JUnit framework.

### 2.2 Spring MVC

Spring MVC is a web module of the Spring framework. It is an open source framework based on the Model-View-Controller (MVC) architecture, used to build robust and loosely coupled applications.

On a high level, the request processing workflow in Spring MVC is shown in Figure 2. This figure is adapted from ‘Spring Framework Reference Documentation’ [2].

![Diagram of Spring MVC workflow](image)

**Figure 2:** High level view of request processing workflow in Spring MVC
Spring MVC provides a clean separation between domain model code, web forms and allows use of all the other features of the Spring Framework. The request processing workflow in Spring MVC is as follows:

- The Front Controller (or DispatcherServlet) receives the user request.
- The Front Controller hands over the user request to the appropriate controller, which processes the request and hands back the model (application data) to the Front Controller.
- The Front controller now passes the model to the View Template and receives the rendered data (to generate the output response) from the View Template.
- The output response is passed to the user’s browser by the Front Controller.

### 2.2.1 Spring MVC Architecture

Spring MVC is built on the notion of a Central servlet (Dispatcher Servlet) that handles all the HTTP requests and responses.

![Spring MVC Architecture](image)

Figure 3: Spring MVC Architecture
Spring MVC architecture is shown in Figure 3. This figure is adapted from ‘Spring MVC architecture’ [3]. MVC is implemented in Spring using these five components [3]:

- **Spring’s Dispatcher Servlet**: This serves as the front controller in a Spring MVC based application. It acts as an intermediate between the application and its clients. All the requests to the application are handled by the Dispatcher servlet, which consults the Handler Mapping to determine which controller should be invoked for a specific request.

- **Handler Mapping**: This finds the appropriate controllers for a given request. The URL’s can be mapped to the controller classes using XML configuration or annotations.

- **Controller**: This processes a given request using other business classes if needed. It returns the model object and the logical view name which can render the information present in the model object.

- **View Resolver**: It returns the physical name (fully qualified name and location) of the view from the logical view name.

- **View**: These can be html or JSP pages. The information from the model object is rendered and the response is sent back to the Dispatcher servlet, which serves the client browser.

### 2.2.2 Features of Spring MVC framework

Spring MVC provides a variety of web support features [2] as mentioned below:

- **Clear separation of roles**: Each role – controller, command object, form object or DispatcherServlet can be served by a specialized object.
- **Adaptability and Flexibility**: Any controller method signature can be defined by using one of the parameter annotations (such as `@RequestParam`, `@RequestHeader`, `@PathVariable`, and more).

- **Flexible model transfer**: Model transfer with a key/value Map helps easy integration with any view technology.

- **Code reusability**: Existing business objects can be reused as command or form objects instead of recreating the objects for a particular class.

- **Integrated MVC pattern**: Special annotations for MVC allow the separation of code in groups like services or repositories, and making it easier to load and test them using dependency injection.

- Powerful configuration of both framework and application classes helps easy referencing of objects across contexts, like referencing the web controllers to business objects and validators.

- Interceptors as well as controllers are provided, making it easy to configure the behavior common to the handling of many requests.

### 2.2.3 Advantages of Spring MVC framework

The main advantages of Spring MVC [2] are listed below:

- **Predefined templates**: It provides templates for Hibernate, JDBC, etc. which can be used to easily integrate these technologies into the application.

- **Lightweight**: It is lightweight because of its POJO (Plain Old Java Object or normal java class) implementation. It does not force the programmer to inherit any special classes or interfaces.
- **Fast Development**: Its support to various frameworks makes the development of JavaEE (Java Platform, Enterprise Edition) applications easy.

- **Declarative support**: It provides declarative support for different features like caching, transactions and validation.

- No need for the creation of Singleton and factory classes.

- It provides a flexible transaction management interface that scale down to a local transaction and scale up to global transactions (using JTA).

### 2.3 Hibernate

Object-Relational Mapping (ORM) framework is used to map objects from the object-oriented language into the relational model structure based on SQL. ORM frameworks work by transforming one data representation to another as shown in Figure 4. This figure is adapted from ‘Research on data persistence layer based on hibernate framework’ [8]. The ‘*.hbm.xml’ file defines the association between the Java class and the database table, Java class attributes and the table fields.

![Figure 4: ORM Mapping Concept](image)

Hibernate is an open source, lightweight, ORM (Object Relational Mapping) solution for Java applications. It is a powerful, high performance Object-Relational Persistence and Query service for any Java application [9]. It solves object-relational impedance mismatch problems by
replacing direct persistence-related database accesses with high-level object handling functions. It generates the SQL calls depending on the database configured, keeping the application portable to all supported SQL databases with very little performance overhead. So, the Java programmers can control database in object-oriented method easily. Programmers can dedicate all the efforts in the development of business logic as Hibernate relieves them from the burden of writing complex JDBC code to persist data.

Hibernate resides between the Java objects and database system to perform all the work needed to persist (or save) the objects in the database as shown in Figure 5. This figure is adapted from ‘Hibernate Overview- Tutorialspoint’ [9].

Figure 5: Role of Hibernate in Java application

2.3.1 Hibernate Architecture

In a typical Java based application, the Hibernate would be present between the Java application and the backend database as shown in Figure 6. This figure is adapted from ‘Hibernate Architecture Tutorial - Javatpoint’ [5].

Hibernate Framework has the below core objects [5]:

- **SessionFactory**: It plays a role of buffer zone to buffer Hibernate generated SQL statements and other mapping data. When applications need to use multiple databases, a ‘SessionFactory’ must be specified for each database. It holds second level cache of data. It is a heavyweight object, so it is initially created during application startup.
- **Session**: This object holds the first level cache of data. It provides an interface between the application code and data stored in the database tables. The `org.hibernate.Session` interface provides methods to insert, update and delete the object. It is sometimes referred to as the persistence layer manager.

- **Transaction Factory**: It is a factory of transactions.

- **Transaction**: This object represents a unit of work with database and almost all the relational databases support transaction functionality. This is an optional object. Developers achieve reunification through Transaction Management to ensure that their systems can easily move value between different environments and the container.

- **Connection Provider**: It is a factory of JDBC connections. It abstracts the application from data source.

Figure 6: Hibernate Architecture
2.3.2 Features of Hibernate

The following are a few features [6] of Hibernate framework:

- **Transparent persistence**: No build-time source or byte code generation during data persistence.

- **Object oriented query language**: It supports both native SQL queries and complex criteria queries.

- **High performance**: This is due to multiple features like Lazy initialization, Outer join fetching and support for optimistic locking with versioning/timestamping.

- **Highly scalable**: This is due to its dual layer cache structure, making it usable in a cluster.

- Supports inheritance, polymorphism, composition, and the Java Collections Framework.

- Offers automatic primary key generation and support for composite keys.

- Configurable condition to generate SQL statements for easy debugging.

2.3.3 Advantages of Hibernate

The key advantages of Hibernate [7] are listed below:

- Cache is internally used in the Hibernate framework and the first level cache is enabled by default, so it has high performance.

- It takes care of mapping Java classes to database tables using XML files, without the need to write complex code.

- If anything is changed in the database, then only changing the XML file properties is sufficient.
Chapter 3 COURSE SCORING SYSTEM

This chapter describes the Course Scoring System in detail. The first section of this chapter explains the different functionalities of the system. The second section presents the system design – Use Case diagrams, Statecharts, Use Case realization, Sequence diagrams, Class diagrams and Application architecture. The third section presents the application screenshots.

3.1 System Functionality

The Course Scoring System can be used at the University level to organize a teacher selection competition. The system has three kinds of roles- Judge role, Admin role and Voter role. In a competition there can be any number of participating teachers and judges but there will be only one voter, who would make the decision in case of a tie at the end of the competition.

The Admin initially sets up the competition and assigns judges and a voter to that competition. Then the Admin registers teachers for the competition. The Judges can login into their accounts and assign scores for the participating teachers. The final score for a teacher is the average of the scores from all the judges. The final score for a teacher is calculated by the system once all the judges have finished assigning the scores for that particular teacher. After the final scores for all the teachers are calculated, if only one teacher has the highest final score, then that teacher would be declared as the Winner of the competition by the system. In case of more than one teacher having the same highest score, the Voter has to login into his account and select one teacher to be the Winner of the competition. After a competition is completed, the Admin can query the scores of the teachers in the competition and sort them by name, or title or affiliation.

The Course Scoring System has the following features:
- **Users Management**: This operation is performed by the Admin role. The Admin can search for the existing list of users using email id or first name or last name or user role. Also, the Admin can create new users and update the user information.

- **Competition Management**: This operation is performed by the Admin role. The Admin can create/edit competition details, including the judges and voter for that competition. A competition can be deleted only if no teachers have been assigned to it. The order of the teachers in the competition can also be edited.

- **Teacher Registration**: This operation is performed by the Admin role. The Admin can register the teachers to a competition. Also, the Admin can modify the affiliation, presentation title and presentation description for the registered teachers.

- **Assigning Scores**: This operation is performed by the Judge role. The Judge can give a score to each teacher in the competition, for different evaluation indexes. The scores, once submitted, cannot be modified. The judge can view the final score for a teacher once all the judges have submitted their scores for that teacher.

- **Special Voting**: This operation can be performed by the Voter role. In case of a tie for the first place at the end of the competition, the Voter can select a better teacher (by his random choice) to be the Winner of the competition.

- **Result Querying**: This operation is performed by the Admin role. The Admin can query the scores of the teachers in a competition. Also, the Admin can sort the teachers list in the competition by scores or teachers' titles or affiliations.


3.2 System Design

3.2.1 Use Case Diagrams

A use case models an interaction between the software product itself and the users of that software product (actors). The system has three Actors – Admin, Judge and Voter.

- Admin user is responsible for the following Use Cases: Login, Manage Competition Details, Manage Users, Register Teacher and Query Scores.
- Judge user is responsible for the following Use Cases: Login and Assign Score.
- Voter user is responsible for the following Use Cases: Login and Vote for better Teacher.

The Use Case diagram for the Course Scoring System is shown in Figure 7.

![Use Case Diagram](image)

Figure 7: Use Case diagram for the Course Scoring System
The description for the various Use Cases is presented below:

**Login Use Case:**

Use case *Login* is shown in Figure 8.

![Diagram](image)

Figure 8: *Login* use case for the Course Scoring System

Table 1: *Login* use case description for the Course Scoring System

<table>
<thead>
<tr>
<th>Brief Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>The <em>Login</em> use case enables the Course Scoring System Admin/ Judge/ Voter to login into the Course Scoring System.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Step-by-Step Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Validate the UserId and Password entered by the user to display Admin/ Judge/ Voter screen based on the user role.</td>
</tr>
</tbody>
</table>

**Manage Competition Details Use Case:**

Use case *Manage Competition Details* is shown in Figure 9.
Figure 9: Manage Competition Details use case for the Course Scoring System

Table 2: Manage Competition Details use case description for the Course Scoring System

<table>
<thead>
<tr>
<th>Brief Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Manage Competition Details use case enables the Course Scoring System Admin to create a new competition and manage the existing competitions.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Step-by-Step Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Create a new competition.</td>
</tr>
<tr>
<td>2. Edit the new competition to allow updates to the following attributes:</td>
</tr>
<tr>
<td>• Competition Description</td>
</tr>
<tr>
<td>• Competition Date</td>
</tr>
<tr>
<td>• Voter</td>
</tr>
<tr>
<td>• Judge(s)</td>
</tr>
<tr>
<td>3. Delete the competition if no teachers are assigned to it.</td>
</tr>
<tr>
<td>4. Update the order of the teachers in the competition.</td>
</tr>
</tbody>
</table>

Manage Users Use Case:

Use case Manage Users is shown in Figure 10.
Table 3: **Manage Users** use case description for the Course Scoring System

<table>
<thead>
<tr>
<th>Brief Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>The <strong>Manage Users</strong> use case enables the Course Scoring System Admin to create a new user and manage the existing users.</td>
</tr>
</tbody>
</table>

**Step-by-Step Description**

1. Create a new Judge/ Voter/ Admin user.

2. Search for the existing users using email id or first name or last name or user role. For each user, the following attributes are printed:
   - Email/User id
   - First Name
   - Last Name
   - Phone Number
   - Address
   - User Role
   - Action (Update)

3. For the existing users, allow updates to the following attributes:
   - Email/User id
Register Teacher Use Case:

Use case Register Teacher is shown in Figure 11.

Figure 11: Register Teacher use case for the Course Scoring System

Table 4: Register Teacher use case description for the Course Scoring System

<table>
<thead>
<tr>
<th>Brief Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Register Teacher use case enables the Course Scoring System Admin to register the teachers and assign them to a competition. Also, edit the registered teacher information.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Step-by-Step Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Enter the below details to register a teacher to the competition:</td>
</tr>
<tr>
<td>• First Name</td>
</tr>
<tr>
<td>• Last Name</td>
</tr>
<tr>
<td>• Email/User id</td>
</tr>
<tr>
<td>• Address</td>
</tr>
</tbody>
</table>
2. For the registered teachers, allow updates to the following attributes:

- First Name
- Last Name
- Email/User id
- Address
- Phone Number
- Affiliation
- Presentation Title
- Presentation Description
- Years of Experience

**Query Scores Use Case:**

Use case **Query Scores** is shown in Figure 12.

![Query Scores Use Case](image-url)
Table 5: *Query Scores* use case description for the Course Scoring System

**Brief Description**

The *Query Scores* use case enables the Course Scoring System Admin to view the summary of the competitions and query the scores of the teachers within a competition.

**Step-by-Step Description**

1. List of competitions held must be displayed with the following attributes:
   - Competition Description
   - Competition Date
   - Winner
   - Winner Score
   - Status

2. For the selected competition, summary should be printed with the following attributes:
   - First Name
   - Last Name
   - Affiliation
   - Presentation Title
   - Years of experience
   - Content clarity score
   - Content organization score
   - Communication skills score
   - Presentation score
- Expertise score
- Final Score
- Winner (yes/ no)

a. All the column headers in the competition summary should be sortable to help sort the result of competition ordered by the scores or grouped by teachers’ titles/ affiliations.

**Assign Score Use Case:**

Use case *Assign Score* is shown in Figure 13.

![Assign Score use case for the Course Scoring System](image)

Figure 13: *Assign Score* use case for the Course Scoring System

**Table 6: Assign Score use case description for the Course Scoring System**

<table>
<thead>
<tr>
<th>Brief Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>The <em>Assign Score</em> use case enables the Course Scoring System Judge to assign the scores to all the teachers in a competition.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Step-by-Step Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Assign the scores to the teachers in a competition for the following evaluation indexes:</td>
</tr>
<tr>
<td>- Content clarity score (Max 10)</td>
</tr>
</tbody>
</table>
- Content organization score (Max 10)
- Communication skills score (Max 10)
- Presentation score (Max 10)
- Expertise score (Max 10)

2. Scores once submitted should not be editable by the user.

3. Final score for the teacher should be the average of all the scores from all the Judges in a competition. Final score for the teacher should be displayed after all the judges submit the score for that particular teacher.

Vote for better Teacher Use Case:

Use case Vote for better Teacher is shown in Figure 14.

Figure 14: Vote for better Teacher use case for the Course Scoring System

Table 7: Vote for better Teacher use case description for the Course Scoring System

<table>
<thead>
<tr>
<th>Brief Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Vote for better Teacher use case enables the Course Scoring System Voter to select a winner in case of a tie between the teacher’s final scores.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Step-by-Step Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>
1. Display the teachers of the competition with the same highest scores, printing the following attributes:
   - First Name
   - Last Name
   - Affiliation
   - Presentation Title
   - Years of Experience
   - Final Score

2. Select one teacher from the teachers displayed in step 1, to be the Winner of the competition.

### 3.2.2 Statecharts

A Statechart diagram reflects all the operations performed by or to that system, indicating the events that cause the transition from state to state. The major source of information regarding the relevant operations is the scenarios.

The operations of the complete Course Scoring System are represented in the Statechart diagram shown in Figure 15. The system is represented as a combination of three subsystems—Admin Subsystem, Judge Subsystem and Voter Subsystem.

The arrow from the initial state leads to the state labeled Course Scoring System Event Loop. From this state, the user can login into the application or quit the system. The trigger ‘application started’ takes the Course Scoring System Event Loop to the state ‘Providing Login Details’. In this state, the user can login into the system using his user id and password. The trigger ‘admin credentials
entered’ takes the system to the state ‘Performing Admin Events’. In this state, user can perform events like Manage Users, Manage Competition Details, Register Teacher and Query Scores. The trigger ‘judge credentials entered’ takes the system to the state ‘Performing Judge Events’, in which the event ‘Assign Score’ can be performed. The trigger ‘voter credentials entered’ takes the system to the state ‘Performing Voter Events’, in which the event ‘Vote for better Teacher’ can be performed.

![Statechart Diagram](image)

**Figure 15: Complete Statechart of the Course Scoring System**

The Statechart diagram for the Admin subsystem is shown in the Figure 16. The trigger ‘manage users selected’ transforms the system from the Course Scoring System Admin Event Loop to the
state ‘Managing Users’. In this state, ‘list existing users’ and ‘register new user’ events can be performed.

![Statechart of the Admin Subsystem](image)

**Figure 16: Statechart of the Admin Subsystem**

The trigger ‘manage competition details selected’ transforms the system from the Course Scoring System Admin Event Loop to the state ‘Managing Competition Details’, in which ‘add/ edit/ delete competition’ and ‘edit lecture order’ events can be performed. The trigger ‘register teacher selected’ transforms the system from the Course Scoring System Admin Event Loop to the state ‘Registering Teacher’, in which the event ‘register teacher and assign them to competition’ can be performed. Finally, the trigger ‘query scores selected’ leads the system to the state ‘Querying Scores’, in which the ‘view competition summary’ event can be performed.
The Statechart diagram for the Judge subsystem is shown in the Figure 17. The trigger ‘competition selected’ transforms the system from the Course Scoring System Judge Event Loop to the state ‘Assigning Scores’. In this state, ‘assign scores to all the participants in the competition’ event can be performed.

Figure 17: Statechart of the Judge Subsystem

Figure 18: Statechart of the Voter Subsystem
The Statechart diagram for the Voter subsystem is shown in the Figure 18. The trigger ‘competition selected’ transforms the system from the Course Scoring System Voter Event Loop to the state ‘Voting for better Teacher’. In this state, the event ‘select one winner in case of a tie’ can be performed.

3.2.3 Use Case Realization (Scenarios and Sequence Diagrams)

The process of extending and refining use cases is called Use Case Realization. A Sequence diagram depicts the realization of a specific scenario of the use case. To realize the Use Cases mentioned in the Section 3.1.2, interaction between the classes is necessary. The classes in the Course Scoring System have been classified into three categories: Entity Classes, Boundary Classes and Controller Classes.

- An Entity class models the information that is long lived.
- A Boundary class models the interaction between the software product and its actors.
- A Control class models complex computations and algorithms.

The different classes for the Course Scoring System is shown in the Figure 19.
Figure 19: Different Classes for the Course Scoring System

**Use Case Realization for the Login Use Case:**

*Login* use case is realized by the interaction between the different Entity, Boundary and Control classes as shown in Figure 20.
Scenarios and Sequence diagrams for the Login Use Case:

Scenario 1 (Admin login - Success)

‘Admin login - Success’ scenario of Login use case is described in Table 8.

Table 8: ‘Admin login - Success’ scenario of Login use case

The Course Scoring System Admin wants to login into the system.

1. The admin enters the valid user id and password at the login screen.
2. The system validates the entered credentials.
3. After successful authentication, the admin screen options are displayed on the admin screen.

The Sequence diagram for the realization of the ‘Admin login – Success’ scenario of the Login use case is shown in Figure 21.
Scenario 2 (Admin login - Failure)

‘Admin login – Failure’ scenario of Login use case is described in Table 9.

Table 9: ‘Admin login – Failure’ scenario of Login use case

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>The admin enters the invalid user id and password at the login screen.</td>
</tr>
<tr>
<td>2.</td>
<td>The system validates the entered credentials.</td>
</tr>
</tbody>
</table>
3. After the authentication failure, error message is displayed on the login screen informing ‘Invalid user name/password’.

The Sequence diagram for the realization of the ‘Admin login – Failure’ scenario of the Login use case is shown in Figure 22.

Figure 22: Sequence diagram for ‘Admin login – Failure’ scenario

Scenario 3 (Judge login – Success)

‘Judge login – Success’ scenario of Login use case is described in Table 10.
Table 10: ‘Judge login – Success’ scenario of Login use case

The Course Scoring System Judge wants to login into the system.

1. The judge enters the valid user id and password at the login screen.

2. The system validates the entered credentials.

3. After successful authentication, the judge screen options are displayed on the judge screen.

The Sequence diagram for the realization of the ‘Judge login – Success’ scenario of the Login use case is shown in Figure 23.

Figure 23: Sequence diagram for ‘Judge login – Success’ scenario
Scenario 4 (Judge login – Failure)

‘Judge login – Failure’ scenario of Login use case is described in Table 11.

Table 11: ‘Judge login – Failure’ scenario of Login use case

The Course Scoring System Judge wants to login into the system.

1. The judge enters the invalid user id and password at the login screen.
2. The system validates the entered credentials.
3. After the authentication failure, error message is displayed on the login screen informing ‘Invalid user name/password’.

The Sequence diagram for the realization of the ‘Judge login – Failure’ scenario of the Login use case is shown in Figure 24.

Figure 24: Sequence diagram for ‘Judge login – Failure’ scenario
Scenario 5 (Voter login - Success)

‘Voter login – Success’ scenario of Login use case is described in Table 12.

Table 12: ‘Voter login – Success’ scenario of Login use case

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>The voter enters the valid user id and password at the login screen.</td>
</tr>
<tr>
<td>2.</td>
<td>The system validates the entered credentials.</td>
</tr>
<tr>
<td>3.</td>
<td>After successful authentication, the voter screen options are displayed on the voter screen.</td>
</tr>
</tbody>
</table>

The Sequence diagram for the realization of the ‘Voter login – Success’ scenario of the Login use case is shown in Figure 25.

Figure 25: Sequence diagram for ‘Voter login – Success’ scenario
Scenario 6 (Voter login – Failure)

‘Voter login – Failure’ scenario of Login use case is described in Table 13.

Table 13: ‘Voter login – Failure’ scenario of Login use case

<table>
<thead>
<tr>
<th>The Course Scoring System Voter wants to login into the system.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. The voter enters the invalid user id and password at the login screen.</td>
</tr>
<tr>
<td>2. The system validates the entered credentials.</td>
</tr>
<tr>
<td>3. After the authentication failure, error message is displayed on the login screen informing ‘Invalid user name/password’.</td>
</tr>
</tbody>
</table>

The Sequence diagram for the realization of the ‘Voter login – Failure’ scenario of the Login use case is shown in Figure 26.
**Use Case Realization for the Manage Competition Details Use Case:**

*Manage Competition Details* use case is realized by the interaction between the different Entity, Boundary and Control classes as shown in Figure 27.

![Interaction of Classes to realize the Manage Competition Details use case](image)

**Figure 27:** Interaction of Classes to realize the Manage Competition Details use case

**Scenarios and Sequence diagrams for the Manage Competition Details Use Case:**

Scenario 1 (Create New Competition)

‘Create New Competition’ scenario of *Manage Competition Details* use case is described in Table 14.

<table>
<thead>
<tr>
<th>The Course Scoring System Admin wants to create a new competition.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. The admin selects ‘Create New Competition’ option from the admin screen.</td>
</tr>
</tbody>
</table>
2. The admin enters competition description and selects the competition date, voter and judges for the competition and finally submits the data.

3. The system creates the new competition and displays the success message on the admin screen.

The Sequence diagram for the realization of the ‘Create New Competition’ scenario of the Manage Competition Details use case is shown in Figure 28.
Scenario 2 (Update Competition)

‘Update Competition’ scenario of Manage Competition Details use case is described in Table 15.

Table 15: ‘Update Competition’ scenario of Manage Competition Details use case

<table>
<thead>
<tr>
<th>The Course Scoring System Admin wants to edit an existing competition.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. The admin selects ‘Update’ option from the admin screen, for the competition to be edited.</td>
</tr>
<tr>
<td>2. The system enables the competition description, competition date, voter and judges for the competition, which can be edited by the admin.</td>
</tr>
<tr>
<td>3. The admin edits and submit the data.</td>
</tr>
<tr>
<td>4. The system edits the competition and displays the success message on the admin screen.</td>
</tr>
</tbody>
</table>

The Sequence diagram for the realization of the ‘Update Competition’ scenario of the Manage Competition Details use case is shown in Figure 29.
Figure 29: Sequence diagram for ‘Update Competition’ scenario

Scenario 3 (Delete Competition)

‘Delete Competition’ scenario of Manage Competition Details use case is described in Table 16.

Table 16: ‘Delete Competition’ scenario of Manage Competition Details use case

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>The admin selects ‘Delete’ option from the admin screen, for the competition with no teachers assigned to it.</td>
</tr>
<tr>
<td>2.</td>
<td>The system deletes this competition.</td>
</tr>
</tbody>
</table>
3. The system displays the success message on the admin screen.

The Sequence diagram for the realization of the ‘Delete Competition’ scenario of the *Manage Competition Details* use case is shown in Figure 30.

![Sequence diagram for ‘Delete Competition’ scenario](image)

**Figure 30**: Sequence diagram for ‘Delete Competition’ scenario

Scenario 4 (Update Teacher Order)

‘Update Teacher Order’ scenario of *Manage Competition Details* use case is described in Table 17.
Table 17: ‘Update Teacher Order’ scenario of Manage Competition Details use case

The Course Scoring System Admin wants to delete a competition.

1. The admin selects ‘Update Teacher Order’ option from the admin screen, for the competition with teachers assigned to it.
2. The admin chooses the order of the teachers and submits the data.
3. The system updates the teacher order and displays the success message on the admin screen.

The Sequence diagram for the realization of the ‘Update Teacher Order’ scenario of the Manage Competition Details use case is shown in Figure 31.
**Use Case Realization for the Manage Users Use Case:**

*Manage Users* use case is realized by the interaction between the different Entity, Boundary and Control classes as shown in Figure 32.

![Figure 32: Interaction of Classes to realize the Manage Users use case](image)

**Scenarios and Sequence diagrams for the Manage Users Use Case:**

Scenario 1 (Create New User)

‘Create New User’ scenario of *Manage Users* use case is described in Table 18.

<table>
<thead>
<tr>
<th>Table 18: ‘Create New User’ scenario of <em>Manage Users</em> use case</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Course Scoring System Admin wants to create a new user.</td>
</tr>
<tr>
<td>1. The admin selects ‘Create New User’ option from the admin screen</td>
</tr>
<tr>
<td>2. The admin enters the details: Email, First Name, Last Name, Phone Number, Address, User Role and submits the data.</td>
</tr>
</tbody>
</table>
3. The system creates the new user and displays the password reset link on the admin screen.

The Sequence diagram for the realization of the ‘Create New User’ scenario of the Manage Users use case is shown in Figure 33.
Scenario 2 (Update User Info)

‘Update User Info’ scenario of Manage Users use case is described in Table 19.

Table 19: ‘Update User Info’ scenario of Manage Users use case

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>The admin selects ‘Update’ option from the admin screen, for the existing user.</td>
</tr>
<tr>
<td>2.</td>
<td>The admin edits the details: Email, First Name, Last Name, Phone Number, Address and submits the data.</td>
</tr>
<tr>
<td>3.</td>
<td>The system edits the user information and displays the success message on the admin screen.</td>
</tr>
</tbody>
</table>

The Course Scoring System Admin wants to update the existing user information.

The Sequence diagram for the realization of the ‘Update User Info’ scenario of the Manage Users use case is shown in Figure 34.
Scenario 3 (Search Users)

‘Search Users’ scenario of Manage Users use case is described in Table 20.

Table 20: ‘Search Users’ scenario of Manage Users use case

<table>
<thead>
<tr>
<th>The Course Scoring System Admin wants to list the existing users.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. The admin specifies the search criteria: Email id or First Name or Last Name or User Role and selects ‘Search’ option from the admin screen.</td>
</tr>
<tr>
<td>2. The system displays the list of users with the following details: Email id, First Name, Last Name, Phone Number, Address, User Role, Action (Update).</td>
</tr>
</tbody>
</table>

The Sequence diagram for the realization of the ‘Search Users’ scenario of the Manage Users use case is shown in Figure 35.

Figure 35: Sequence diagram for ‘Search Users’ scenario
Use Case Realization for the Register Teacher Use Case:

Register Teacher use case is realized by the interaction between the different Entity, Boundary and Control classes as shown in Figure 36.

Figure 36: Interaction of Classes to realize the Register Teacher use case

Scenarios and Sequence diagrams for the Register Teacher Use Case:

Scenario 1 (Register New Teacher)

‘Register New Teacher’ scenario of Register Teacher use case is described in Table 21.

Table 21: ‘Register New Teacher’ scenario of Register Teacher use case

<table>
<thead>
<tr>
<th>The Course Scoring System Admin wants to register a teacher to a competition.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. The admin specifies ‘Teacher Registration’ option from the admin screen.</td>
</tr>
<tr>
<td>2. The admin enters the following details: First Name, Last Name, Email id, Address, Phone Number, Affiliation, Presentation Title, Presentation Description, Years of Experience and Competition name and submits data.</td>
</tr>
</tbody>
</table>
3. The system registers the teacher to the competition and displays the success message on the admin screen.

The Sequence diagram for the realization of the ‘Register New Teacher’ scenario of the *Register Teacher* use case is shown in Figure 37.

![Sequence diagram for Register New Teacher scenario](image-url)

Figure 37: Sequence diagram for ‘Register New Teacher’ scenario

Scenario 2 (Edit Teacher Info)

‘Edit Teacher Info’ scenario of *Register Teacher* use case is described in Table 22.
**Table 22: ‘Edit Teacher Info’ scenario of Register Teacher use case**

The Course Scoring System Admin wants to edit the information of a teacher.

1. The admin specifies ‘Edit Teacher Info’ option from the admin screen and selects the competition name to edit the teacher details in that competition.
2. The admin edits the details: First Name, Last Name, Email id, Address, Phone Number, Affiliation, Presentation Title, Presentation Description, Years of Experience and submits data.
3. The system saves the updated information for the teacher and displays the success message on the admin screen.

The Sequence diagram for the realization of the ‘Edit Teacher Info’ scenario of the Register Teacher use case is shown in Figure 38.

![Sequence diagram for ‘Edit Teacher Info’ scenario](image)

Figure 38: Sequence diagram for ‘Edit Teacher Info’ scenario
**Use Case Realization for the Query Scores Use Case:**

*Query Scores* use case is realized by the interaction between the different Entity, Boundary and Control classes as shown in Figure 39.

![Interaction of Classes to realize the Query Scores use case](image)

**Figure 39: Interaction of Classes to realize the Query Scores use case**

**Scenarios and Sequence diagrams for the Query Scores Use Case:**

Scenario 1 (Query Competition Summary)

‘Query Competition Summary’ scenario of *Query Scores* use case is described in Table 23.

<table>
<thead>
<tr>
<th>The Course Scoring System Admin wants to query the competition summary.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. The admin selects ‘Query Scores’ option from the admin screen and selects a competition to view its summary.</td>
</tr>
<tr>
<td>2. The system displays the summary of the competition with the following details: First Name, Last Name, Affiliation, Presentation Title, Years of Experience, Content</td>
</tr>
</tbody>
</table>
Clarity score, Content Organization score, Communication Skills score, Presentation Score, Expertise Score, Final Score and Winner (yes/no).

The Sequence diagram for the realization of ‘Query Competition Summary’ scenario of Query Scores use case is shown in Figure 40.

**Use Case Realization for the Assign Score Use Case:**

*Assign Score* use case is realized by the interaction between the different Entity, Boundary and Control classes as shown in Figure 41.
Figure 41: Interaction of Classes to realize the Assign Score use case

**Scenarios and Sequence diagrams for the Assign Score Use Case:**

Scenario 1 (Assign Scores to Teachers)

‘Assign Scores to Teachers’ scenario of Assign Score use case is described in Table 24.

Table 24: ‘Assign Scores to Teachers’ scenario of Assign Score use case

The Course Scoring System Judge wants to assign scores to the teacher participating in the competition.

1. The judge selects ‘Assign Score’ option from the judge screen.
2. The system displays the list of competitions assigned to this judge.
3. The judge selects a competition and assigns scores to the teachers in the competition.
4. The system calculates the final score for the teacher by averaging the scores from all the judges for that teacher and displays this final score to the judge on the judge screen.

The Sequence diagram for the realization of ‘Assign Scores to Teachers’ scenario of Assign Score use case is shown in Figure 42.

Figure 42: Sequence diagram for ‘Assign Scores to Teachers’ scenario
**Use Case Realization for the Vote for better Teacher Use Case:**

*Vote for better Teacher* use case is realized by the interaction between the different Entity, Boundary and Control classes as shown in Figure 43.

![Diagram](image)

**Figure 43: Interaction of Classes to realize the Vote for better Teacher use case**

**Scenarios and Sequence diagrams for the Vote for better Teacher Use Case:**

Scenario 1 (Select Winner)

‘Select Winner’ scenario of *Vote for better Teacher* use case is described in Table 25.

**Table 25: ‘Select Winner’ scenario of Vote for better Teacher use case**

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>The voter selects ‘Competition List’ option from the voter screen.</td>
</tr>
<tr>
<td>2.</td>
<td>The system displays the list of tied competitions assigned to this voter.</td>
</tr>
<tr>
<td>3.</td>
<td>The voter selects a competition and selects a better teacher to be a winner of the competition.</td>
</tr>
</tbody>
</table>
4. The system records the winner and displays a success message to the voter on the voter screen.

The Sequence diagram for the realization of ‘Select Winner’ scenario of *Vote for better Teacher* use case is shown in Figure 44.
3.2.4 Class Diagrams

Complete Class Diagram:

Complete (overall) class diagram of the Course Scoring System is shown in Figure 45.
Part of the Class Diagram - Login Controller:

Part of the overall class diagram showing Login Controller is shown in Figure 46.

Figure 46: Part of the overall class diagram showing Login Controller
Part of the Class Diagram - Admin Controller:

Part of the overall all class diagram showing Admin Controller is shown in Figure 47.

Figure 47: Part of the overall class diagram showing Admin Controller
Part of the Class Diagram - Scoring Controller:

Part of the overall class diagram showing Scoring Controller is shown in Figure 48.

Figure 48: Part of the overall class diagram showing Scoring Controller
Part of the Class Diagram - Login Controller with attributes and methods:

Part of the overall all class diagram showing Login Controller with attributes and methods is shown in Figure 49. In the figure, symbol C represents Java Class and I represents Java Interface.

Figure 49: Part of the class diagram showing Login Controller with attributes and methods
Part of the Class Diagram - Admin Controller with attributes and methods:

Part of the overall class diagram showing Admin Controller with attributes and methods is shown in Figure 50. In the figure, symbol C represents Java Class and I represents Java Interface.
Part of the Class Diagram - Scoring Controller with attributes and methods:

Part of the overall all class diagram showing Scoring Controller with attributes and methods is shown in Figure 51. In the figure, symbol C represents Java Class and I represents Java Interface.

Figure 51: Part of the class diagram showing Scoring Controller with attributes and methods
Part of the Class Diagram – Model Classes with attributes and methods:

Part of the overall all class diagram showing Model Classes with attributes and methods is shown in Figure 52. In the figure, symbol C represents Java Class and I represents Java Interface.
Figure 52: Part of the class diagram showing Model Classes with attributes and methods
3.2.5 Application Architecture

Course Scoring System is complete and modular due to its layered architecture. The different layers in this application are shown in the Figure 53.

![Course Scoring System Architecture Diagram]

Figure 53: Course Scoring System Architecture

The following are the layers involved in the Course Scoring System development:

- **Presentation Layer**: This layer corresponds to the View part in the MVC pattern. It involves the user interaction. It is implemented using three technologies – JSP, HTML and CSS. Java Server Pages (JSP) is a technology used for creating platform independent and dynamic web pages. HTML is a markup language for content presentation in web pages. Cascading Style Sheets (CSS) is used to control the layout of multiple web pages.

- **Controller Layer**: This layer corresponds to the Controller in the MVC pattern. The
user requests are handled in this layer. This layer is handled by the Spring Controller. It translates the incoming requests, takes the request data and passes it to the appropriate service in the Service layer. Controllers provide access to the application behavior without knowing the exact implementation of the service logic. Upon receiving the response from the Service layer, the model data is rendered to the appropriate view for displaying it to the user. In the current application, three Controllers are defined – LoginController, AdminController and ScoringController.

- **Service Layer:** This layer is used by the Controller layer to execute the application functionality. It has all the business logic of the application. Each service should encapsulate a logical collection of transactions. Each service method can be defined as the smallest atomic operation the application does to the database – so a service method either completes the entire transaction and commits or rolls back the entire transaction to maintain the database in a consistent state at any given time. This layer must be separate from the Presentation or the Controller layers to enable one service to be invoked by multiple sources. Inside the Service layer, one service may invoke another service in order to execute the business functionality. The service layer in this application utilizes the concept of interfaces and their implementation classes. Interfaces in Java are just like classes, except that an interface only contains method signatures and fields. The implementation of these methods are defined in the Java class that implements this interface. Implementation using Java interfaces and their implementation classes results in loose coupling as only a change in the implementation class would be needed to change the system functionality and no change to the service
interface definition is needed. In other words, if a better implementation class is written in future, the new class should implement the interface and provide definitions for the methods in the interface. This new implementation class can then replace the earlier implementation class. In the current application, three Service interfaces – LoginService, CSSAdminService and ScoringService and their implementation classes – LoginServiceImpl, CSSAdminServiceImpl and ScoringServiceImpl are defined.

- **Data Access Layer**: This layer corresponds to the Model in the MVC pattern. It handles the database operations. It is invoked by the Service layer. This layer uses Hibernate API’s (or methods) to interact with MySQL database. Dao interface specifies what operations are performed by the application to the database and their implementation class have the necessary code for data persistence logic. In the future, if a change in code persistence logic is needed, only the implementation classes need to be modified and no change would be needed in the Dao interface. In this project, five interfaces – LoginDao, AdminDao, CompetitionDetailsDao, ScoringDao, TeacherRegistrationDao and their implementation classes- LoginDaoImpl, AdminDaoImpl, CompetitionDetailsDaoImpl, ScoringDaoImpl, TeacherRegistrationDaoImpl are defined in the DAO layer.

### 3.3 Application Screenshots

Following are the screenshots from the Course Scoring System showing the different System functionalities.
Login Screen:

![Login Screen of the Course Scoring System](image)

Figure 54: Login Screen of the Course Scoring System

Admin Screen:

![Admin Screen of the Course Scoring System](image)

Figure 55: Admin Screen of the Course Scoring System
Manage Users – Create new user:

![Image of user registration form](https://example.com/image.png)

Figure 56: Manage Users – Create new user

Manage Users – Success message after creating a new user:

When a new user is created, a link (as shown in Figure 57) is generated that can be used by the new user to reset the password (as shown in Figure 58). For security reasons, this link will only be active for 24 hours after it is generated.

![Image of success message](https://example.com/image.png)

Figure 57: Manage Users – Success message after creating a new user
Figure 58: Password Reset option for new user

Manage Users – Search users:

Figure 59: Manage Users – Search users
Manage Competition Details – Create new Competition:

![Figure 60: Manage Competition Details – Create new Competition](image)

Manage Competition Details – New competition created successfully:

![Figure 61: Manage Competition Details – New competition created successfully](image)
Manage Competition Details – Update teacher order:

Figure 62: Manage Competition Details – Update teacher order

Register new teacher:

Figure 63: Register new teacher
Success message for Register new teacher:

Figure 64: Success message after registering a teacher

Judge Screen:

Figure 65: Judge Screen
Assign Scores to participants:

Figure 66: Assign Scores to participants

Assign Scores to participants – alert message:

Figure 67: Assign Scores to participants – alert message
Assign Scores to participants – scores saved successfully:

![Assign Scores to participants](image1)

Figure 68: Assign Scores to participants – scores saved successfully

Voter Screen – voting needed:

![Voter Screen](image2)

Figure 69: Voter Screen – voting needed
Voter Screen – winner selected by voting:

Figure 70: Voter Screen – winner selected by voting
Chapter 4: CONCLUSION

Spring MVC is a powerful framework for developing enterprise level Java applications. It can easily be integrated with Object Relational Mapping (ORM) tools such as Hibernate to make the application development faster by eliminating the need for the programmers to write complex SQL code for data persistence. Applications developed with Spring MVC and Hibernate are robust, loosely coupled and easy to use. As Hibernate has in-built support for wide variety of relational databases, the application can be configured to use a different relational database with minimal changes to the application code. Hibernate provides caching feature for fast retrieval of data from the database, thus improving the performance of the system.

The objective of this project is to understand the Spring MVC and Hibernate frameworks (both of these frameworks have become popular for enterprise level applications in the recent times) and implement a ‘Course Scoring System’ integrating these two frameworks. The application can be used to organize a teacher selection competition at the University level. The various features of the application include teacher registration, assigning scores to the teachers and winner selection. This project would be a good reference for anyone trying to gain the practical knowledge of Spring MVC and Hibernate frameworks and develop an application integrating them.

The Course Scoring System can be further enhanced by implementing some additional features such as having multiple voters to vote in case of a tie at the end of the competition to select the competition winner, email service to send the notifications to the participants reminding about the competition time, email service to send the password reset link automatically to the Judges and Voters when they are assigned roles and ability to assign the teacher to more than one competition during the registration process.
References


http://docs.spring.io/spring/docs/current/spring-framework-reference/html/


http://www.java-overview.thiyagaraaj.com/features-of-hibernate


[9] "Hibernate Overview- Tutorialspoint",
https://www.tutorialspoint.com/hibernate/hibernate_overview.htm
Bibliography

