**A WEB APPLICATION BASED ON THE MVC ARCHITECTURE USING THE**

**SPRING FRAMEWORK**

A PROJECT REPORT

Presented to the Department of Computer Engineering and Computer Science

California State University, Long Beach

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Computer Science

Committee Members:

Frank Murgolo, Ph.D. (Chair)

Birgit Penzenstadler, Ph.D.

Phuong Nguyen, M.S.

College Designee:

Antonella Sciortino, Ph.D.

By Hardikkumar B. Panchal

B.E., 2014, Gujarat Technology University, India

December 2016

ABSTRACT

**A WEB APPLICATION BASED ON THE MVC ARCHITECTURE USING THE**

**SPRING FRAMEWORK**

By

Hardikkumar B. Panchal

December 2016

In the field of software engineering, the Model-View-Controller (MVC) architecture was a great breakthrough that was introduced in the 1970s. It is an architectural pattern that is useful for structuring software applications. MVC divides an application into three components: 1) Model, 2) View, and 3) Controller. This separation provides great modularity, which results in ease of maintenance for systems employing this architecture. A software framework is a collection of modules and connectors that embody a given software architecture. In the spirit of software reuse, the Spring Framework has been integrated for implementing the MVC architecture.

In this document, a Web-Application is presented that demonstrates the features and characteristics of the MVC architecture and Spring Framework. All the development technologies that are used in the Web-Application will be discussed along with the Object-Oriented-Analysis and Design methodology.

**TABLE OF CONTENTS**

ABSTRACT ii

LIST OF TABLES iv

LIST OF FIGURES v

LIST OF ABBREVIATIONS vii

1. INTRODUCTION 1
2. SYSTEM ANALYSIS 7
3. CONCEPT BREAKDOWN 10
4. METHODOLOGY 15
5. RESULTANT SYSTEM 31
6. CONCLUSION 43

APPENDIX: USE CASES 45

REFERENCES 66

**LIST OF TABLES**

1. Installing and Configuring JDK 46
2. Installing and Configuring Apache-Tomcat 47
3. Installing Spring Framework 48
4. Student Registration 49
5. Student Login 50
6. Student Enrollment 51
7. Student Drop the Course 52
8. View Grade 53
9. View Schedule 54
10. Change Personal Information 55
11. Teacher Registration 56
12. Teacher Login 57
13. Drop Student 58
14. Add Grade 59
15. Add Term 60
16. Add Department 61
17. Add Course 62
18. Manage Class Location 63
19. Add Time-Slot 64
20. Manage Course Allocation 65

**LIST OF FIGURES**

1. MVC architecture 12
2. Class diagram 19
3. Sequence diagram: Add term 20
4. Sequence diagram: Course allocation 21
5. Sequence diagram: Registration 22
6. Sequence diagram: Course enrollment 23
7. Database schema 24
8. ORM 26
9. Spring Framework architecture 26
10. Working of Spring Web MVC Framework 29
11. Processing of a JSP page 30
12. Role page 31
13. Login page 32
14. Login error 33
15. Profile page 33
16. Add class 34
17. Enrollment page 34
18. Enrollment error 35
19. Grade page 35
20. Schedule page 36
21. Teacher Grade page 36
22. Grade assignment 1 37
23. Grade assignment 2 37
24. Grade change 38
25. Student list 38
26. Home page 39
27. Add course 39
28. Course allocation 40
29. Change course allocation 41
30. Implemented Spring MVC 41

**LIST OF ABBREVIATIONS**

DAO Data Access Object

GUI Graphical User Interface

HTTP Hyper Text Transfer Protocol

HQL Hibernate Query Language

JDBC Java Database Connection

JDK Java Development Kit

JSP Java Server Page

MVC Model-View-Controller

ORM Object Relation Mapping

RDBM S Relational Database Management System

SQL Structured Query Language

XML Extensible Markup Language

UI User Interface

**CHAPTER 1**

**INTRODUCTION**

In today’s world, there is a drastic change in each and every field due to rapid advancement in technology. The computer was the most revolutionary discovery until the end of 20th century, and after the advent of the internet, web applications became an inherent part of most of the software systems. The internet is growing at such a rapid pace that it has become an essential part of our day-to-day lives. It has been adopted by not only the software industry but also by nearly every other application area that desires a flexible and global nature. The internet has given rise to the development of a Web-Application in which the client–server communication model is used where the user interface (Client) runs on the web-browser.

The Web-Applications are dominating desktop based applications. Desktop applications have their own advantages, but Web-Applications are more in demand than desktop applications because users want an up-to-date application that satisfies their requirements. Web-Applications are becoming so popular that many organizations are growing their businesses by deploying their resources and services on the web. Websites have become an absolute requirement for driving an organization with a prolific profit, besides this, it becomes really easy to span a wide range of customers from all over the world by going on the web. Also, in areas such as customer relations, requirements and interactions are efficiently maintained with an adequate expertise in a timely manner. Up to now, the impact of the internet and a Web-Application on a computer system have been discussed. There are various new concepts that work together and create a Web-Application that runs on the web. Mainly there are three components for a Web-Application. The three components are discussed in the following list:

1. User Interface: It is the view of the Web-Application created by using technologies such as HTML [1], CSS [2], JavaScript [3], JQuery [4], and Bootstrap [5]. Many user–interfaces are developed using these designing technologies.
2. Server Side Code: It is the core part of the Web-Application where whole business logic resides. It is also responsible for handling requests–response from the client. Various programming languages, architectures, and frameworks fall into this category, which will be discussed later in the report.
3. Database: It is responsible for storing and retrieving the data. The database is a separate entity that is stored externally outside the programming workspace and it is usually managed by 3rd party software such as MySQL or Oracle, the two most prominent database tools that are being used in software development.

These three aspects have become the defacto standard for designing the architecture of the Web-Application.

This project focuses on a Web-Application, in which its specific architecture and framework are used for coding. Thinking from the general perspective, in order to run a nation in an orderly manner, a government has defined certain rules and regulations, which every citizen has to follow, to create a law abiding society. Similarly, software engineers have to architect and develop a software system to create an efficient and maintainable system, according to Design Patterns [6]. Design patterns are general repeatable solutions to a commonly occurring problems within a given context in software design. For example, The Model-View-Controller (MVC) architectural pattern is one of the most useful design patterns for developing applications. Primarily, the MVC architecture aims to separate business logic, data model, and user interface. In the presented Web-Application, the Java Spring Framework is used. The Spring Framework is an open source Java platform for MVC that was initially written by Rod Johnson and was first released under the Apache 2.0 license in June 2003. It has gained a significant amount of attention from the Java application industry because its stable release is packed with interactive features such as Inversion of Control, Dependency Injection, and Auto-Wiring, which will be discussed later in the report. The presented Web-Application, Student Registration System, will serve as an example to demonstrate the implementation and integration of MVC architecture and Java Spring Framework on a web platform.

**1.1 Goal**

This report aims to provide a perspicacious understanding of the MVC architecture and its features by demonstrating a Web-Application in which Spring Framework is integrated. This system contains many of the necessary functionalities that can be usually found in an university enrollment system. This Web-Application has sufficient complexity to illustrate all aspects of the MVC architecture. Moreover, the Spring Framework is also keenly depicted by the Web-Application thorough which all the crucial characteristics of Spring Framework will be described.

Moreover, the ultimate goal of this project is not to improve upon a current enrollment system but to demonstrate the working MVC and Spring Framework.

**1.2 Scope of the System**

Here, the implicit purpose of developing the Student Registration System is to explain the MVC architecture and functioning while integrated with the Spring Framework, and this goal is efficiently accomplished. Although, there are few limitations of the purposed Student Registration System:

* The presented Student Registration System does not have any payment module to pay the fees for enrolled courses.
* The credit management system for courses is not yet developed.
* This system works only on web-browser, so it requires reliable internet connection at the user end.

**1.3 Functional Objectives**

The success of a project depends on the fulfillment of functional requirements and non-functional requirements accompanied by objectives of the project that also should be satisfied without any error. The functional objectives are listed below:

* Student Registration
* Teacher Registration
* Student and Teacher Profile Management
* Spring Security for Student and Teacher Login Module
* Add Course
* Drop Course
* Schedule Updating for both Student and Teacher
* Assignment of Grades
* Change of Grades
* Drop Student
* Admin Assigning Courses to Teacher
* Admin Manage Class Location
* Admin Manage Time-Slot
* Admin Manage Term
* Admin Manage Department
* Admin Manage Course
* Admin Manage Students and Teachers

**1.4 Non-Functional Objectives (Acceptance Criteria)**

The pivotal acceptance condition is to effectively demonstrate all the characteristics of the MVC architecture and its involvement with the Spring Framework. Besides this, following are the remainders of the non-functional requirements for the proposed Student Registration System:

* Probability that website becomes unreachable or unresponsive is less than 1%.
* The user response time is less than 4 seconds.
* The Web-Application is compatible with all browsers.
* The system does not affect the other stored or running programs in malicious ways.
* The website is adaptable to maintenance or updating at later stages.

**1.6 Synopsis**

Summaries and detailed descriptions of the proposed application then the Functional and Non-Functional requirements will be addressed in this documentation. Possible scenarios, use cases, database-schema, class-diagram, and database-dictionary will also be included.

This project report outlines all of the crucial steps taken to develop the Student Registration System. Chapter 2 consists of a detailed requirement analysis that reviews the functional and non-functional requirements. These requirements represent features of the Web-Application. Then in the subsequent chapter, to construct formation of understanding, two pivotal concepts, Model-View-Controller architecture and Spring Framework, will be discussed, which are the intrinsic parts of the presented Web-Application. The next chapter is the heart of the report in which methodology used in developing the Web-Application will be discussed. In this chapter various kinds of developing technologies will be briefly covered. Also, Object Oriented Analysis and Design will be presented, which includes use-cases, class diagram, sequence diagrams, and database schema. Then finally, resultant system will be shown via mock-ups to perceive the look and feel of the Web-Application.

**CHAPTER 2**

**SYSTEM ANALYSIS**

The proposed Web-Application (Student Registration System) shows the working of the MVC architecture and its functioning with respect to the Java Spring Framework. In this system there are three types of users: 1. Admin, 2. Student, and 3. Teacher.The admin is the essential key to all functions, who manages all the modules and is also responsible for assigning courses to teachers. The Web-Application is focused on students and teachers, so that they can use this application for their registration, profile management, student management, schedule management, and course management with great finesse. There are two types of requirements in the Web-Application: 1. functional requirement and 2. non-functional requirement.

**2.1. Functional Requirements**

Functional requirements are the features of a system, which specify its capabilities. They are the user stories that describe how users will use the system. The functional requirements of the proposed system are the following:

* Registration: Student and Teacher will both register in the system to access their account and features.
* Login: Login module provides access only to authorized users who complete the

registration process and set their username and password.

* Course Enrollment: Student should be able to add a course and, if the class size is less than or equal to 20, enroll in that course.
* Grade Assignment: Teachers should be able to assign grades a range from A to F to their students. Occasionally, they can also change the assigned grade.
* Manage Student List: Teachers can see the information of every student who enrolled into the courses but only in the courses to which that teacher is assigned. Teachers can also drop students from courses if necessary. The rules for dropping students differ among different university policies.
* Profile Management: Student and Teacher both can edit their profile whenever they want.
* Manage Term: Admin can add, update, and remove a term.
* Manage Department: Admin can add, update, and remove a department.
* Manage Course: After selecting term and department, the admin can add, update, and remove a course.
* Manage Class Location: Admin can add, update, and remove a lecture hall.
* Manage Time-Slot: Admin can add, update, and remove a Time-Slot.
* Manage Class Allocation: Admin can assign one or more courses to teacher and can also remove and change course assignments.

**2.2 Non** - **Functional Requirements**

Non-functional requirements define the behavior of the system; they also describe how the system is working. The non-functional requirements of the proposed system are following:

* All inserted data should be properly stored in the database.
* The database should reflect the changes.
* Class-list should be updated.
* Teachers’ profiles should be updated after course allocated to them.
* Students’ profile should be updated when teacher assigns grades to them.
* All form-validations should run correctly.
* Response time of the Schedule change should be 5ms.
* Grade should be reflected as soon as teacher adds it.
* Website should be secured with the SSL certificate.
* System does not allow unauthorized user to log in.
* Server must operate all the time.
* Website should be compatible with every web-browser.

**CHAPTER 3**

**CONCEPT BREAKDOWN**

In this chapter, Web-Application, Model-View-Controller architecture, and Software Framework will be discussed in order to assimilate the reason behind selecting these three concepts for developing the project and considering them as intrinsic parts of this developed system.

**3.1 Web-Application**

As discussed in the introduction, Web-Applications have had a significant impact on information processing. Due to this revolutionary breakthrough, Web-Applications have become the most favorable tool for widening the scopes of businesses for each global organization. Web-Applications require web-browsers with which users interact. Therefore, to serve the majority of users, a Web-Application should be compatible with all web-browsers such as Google Chrome, Mozilla Firefox, and Internet Explorer.

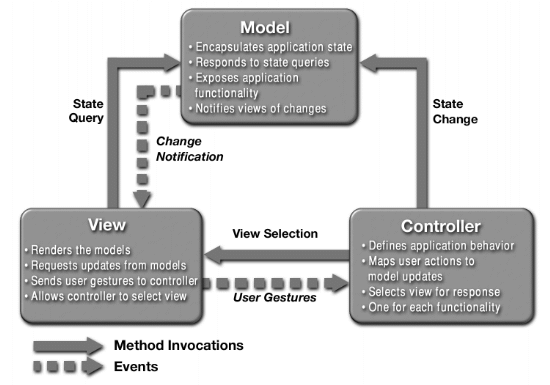
From user-interface perspective, Web-Applications currently appear more stunning and attractive compared to desktop applications because there are many efficient designing technologies that are being used for designing Web-Applications, such as HTML, CSS, JavaScript, JQuery, Bootstrap, Angular JS, and Wordpress. Each technology serves a different purpose, yet they are compatible and dependent on each other for a great outcome. Web-Applications are gaining dominance over desktop applications because they have lots of advantages over desktop applications. These advantages are represented in the following list:

* Internet use has permeated the globe, which has resulted in enormous increase in the use of Web-Applications because the internet is the only crucial requirement for the execution of a Web-Application.
* A Web-Application is platform independent, so it does not rely on an operating system such as a desktop application [7].
* A Web-Application does not have exacting requirement specifications for hardware because it can utilize its required resources through the internet, including storage needs and CPU processing power [7].
* A Web-Application is easy to maintain and update on the web after its release.
* In a Web-Application, the user interface is integrated with numerous visual effects and rich media content [7].

**3.2 Model-View-Controller Architecture**

MVC stands for Model-View-Controller, which is an architecture for developing software systems. This architecture was introduced in Smalltalk-76 in the 1970s, but it was implemented in Smalltalk-80 in 1980 [8][9]. MVC divides an application into three different parts called Model, View, and Controller. These modules partition the system into: 1. User Interface, 2. Application Logic [10], and 3. Application Data; therefore, these three parts are independent and can be treated as individual modules, yet they work together. MVC supports separation of concerns, which is the way data is processed and represented on the user interface, both have discrete implementations [10], meaning that MVC separated them.

The architecture of MVC is shown in Figure 1. The three aforementioned individual components have been depicted in the figure. These individual components interact with each other and process all requests from user through each module. It is important to notice that MVC is not similar to the framework, which typically provides partial code or some basic functionalities. It is just an architectural concept through which an application code can be organized and executed in an efficient manner. Three parts of MVC are explained in the following section:



**FIGURE 1. MVC architecture [11].**

**3.2.1 Model**

Model is the most basic level of the architecture that is responsible for maintaining the application’s data [10]. As shown in Figure 1, controller requests for specific data on behalf of the user, the model then processes that input according to business logic implemented in it and returns the resultant (processed data) data back to the controller. MVC decouples model and view by employing a specific protocol called subscribe/notify between them [12]. A view updates its representation according to the model data. Whenever the model modifies the data, it notifies the related view regarding the change and the view reflects those changes [12].

Separation of concerns is the salient feature of MVC and idea behind this is to make an evident isolation between domain objects and presentation objects [13]. Domain objects are the actual representation of the data that is stored in the database in the form of entity where presentation objects are the Graphical User Interface (GUI) components that are displayed on the screen [13]. These domain objects are known as models of the MVC; these objects are implemented independently from the other two parts so they are not aware of any implementation subject to UI components [13].

**3.2.2 View**

View is the only module that directly interacts with the end user and receives requests and input from the user. It also deals with the representation of the processed output to the user, using technologies such as HTML and JSP. The presentation objects or GUI elements that have just been discussed are part of the view module [13]. As shown in Figure 1, view sends user request to the controller and receives the result back for display. Moreover, it is possible that a model has multiple views, therefore after processing business logic model will decide which view to invoke through controller. According to subscribe/notify protocol view has to ensure that its representation reflects the current state of the model.

**3.2.3 Controller**

TheController fills the gap between the model and the view because the view cannot directly interact with the model. The controller maps the user request with appropriate model and forwards the inputs to the model. After receiving response from the model, it invokes the related view. Therefore, the controller interprets the user action to the model, then the request is either forwarded to business logic layer or data access layer through the model. So, in a way, the controller is the request handler for the MVC. Moreover, it is important to notice that between the model and the controller, the model cannot interact with the controller directly.

**3.3 Software Framework**

In general, framework means an organized structure that can be real or conceptual intended to support building a target system. Similarly, software framework provides a platform integrated with reusable code and generic functions that can be selectively customized by developers. It is a set of predefined components that are interconnected with each other. So by integrating the framework in the application, developers’ productivity will be augmented because now they can concentrate more on the efficiency and implementation of the application logic rather than utilizing their time in designing the application infrastructure.

In the presented Web-Application the Spring Framework has been used, which will be discussed in the following chapter.

**CHAPTER 4**

**METHODOLOGY**

In previous chapter the core concepts used in this project have been discussed. In this chapter, Use Cases and Sequence Diagrams have been depicted to demonstrate the flow of execution for different functionalities. Moreover, various kinds of technologies such as Hibernate, JSP, and Java Spring Framework will be briefly discussed which have been used in the development. Apart from this, database structure will be shown through class diagram and database schema.

**4.1 Use-Case**

Use-case means a written description of how to perform tasks on the Web-Application. The description will be in a language understandable by the end user. A use-case demonstrates the execution flow for a specific task that a user follows, along with desired output and some more properties which are listed below:

* Actor: Users which performs the task [14].
* Pre-Condition: Prerequisites which should be met prior to access a particular use-case [14].
* Flow of Control: Sequence of steps to access a specific functionality of the system [14].
* Post-Condition: Desired outcome after the successful execution of a use case [14].
* Error-Condition: List of errors that might occur during the execution [14].
* Non-Functional Requirement: A particular requirement associated with a use case that defines behavior of the system during the execution [14].

In the following list, the use-cases that are designed to illustrate various functionalities of the Student Registration System have been briefly explained. All these use-cases have been included in the Appendix A.

1. Installing and Configuring JDK: This use-case will be executed when the developer installs and configures Java Development Kit (JDK) in the system. JDK support Java development and compiles Java code. The related use-case in presented in Table 1.
2. Installing and Configuring Apache-Tomcat: The developer installs the Apache-Tomcat server that handles the HTTP requests and provides support to the Java applications. The related use-case is presented in Table 2.
3. Installing Spring Framework: The use-case for installing Spring framework is presented in Table 3. This framework is one of the most crucial technologies that is used in the development of the Web-Application.
4. Student Registration: This use-case reflects the registration process that a student follows in order to create a profile. The use-case is presented in Table 4.
5. Student Login: In order to login into the system, the students follow the use-case presented in Table 5.
6. Course Enrollment: The students enroll into various courses by following the procedure illustrated in the use-case shown in Table 6.
7. Student Drop the Course: The students drop out from the enrolled courses using the use-case that is shown in Table 7.
8. View Grade: The students can see their grades by following steps listed in the use-case shown in Table 8.
9. View Schedule: The Schedule will be generated, once the students enroll into courses. The students can access their schedule as shown in the use-case presented in Table 9.
10. Change Personal Information: The student can edit their profile by following the use-case presented in Table 10.
11. Teacher Registration: This use-case reflects the registration process that a teacher follows in order to create a profile. The use-case is presented in Table 11.
12. Teacher Login: In order to login into the system, the teachers follow the use-case presented in Table 12.
13. Drop Student: The teachers follow the use-case presented in Table 13, for dropping the students.
14. Add Grade: The teachers can add grades using the use-case shown in Table 14.
15. Add Term: The admin adds terms. And for this functionality, the use-case shown in Table 15 will be referred.
16. Add Department: The admin adds department. The use-case shown in Table 16 reflects how this function will be handle.
17. Add Course: The use-case shown in Table 17, will presents the add course functionality, which is done by the admin.
18. Manage Class Location: The admin manages class location. This functionality is explained in the use-case shown in Table 18.
19. Add Time-Slot: The admin adds Time-Slots. The use-case shown in Table 19 reflects how this function will be performed.
20. Manage Course Allocation: The admin assigns course to the teachers by following the use-case shown in Table 20.

**4.2 Class Diagram**

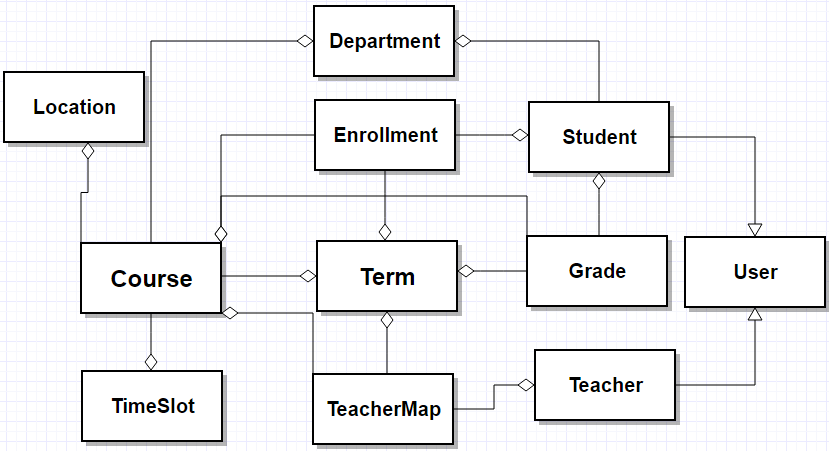
Objects in software system have similar characteristics like real-world objects, they also have their own state and behavior. In object-oriented programming [15] state refers to variable and behavior refers to method. Class is a blueprint from where object is created.

Class diagram shows relationships between classes, it also represents the static view of the Web-Application that is fruitful to develop and structure the code [16]. A class diagram can consist of different kind of relationships which are explained below:

* Association: It is the most general form of a relation that specifies any kind of logical connection between classes [16].
* Aggregation: This kind of relationship is established when a class becomes a result of the collection of another class. These classes are loosely coupled, therefore if one class dissolved, other class will not get affected [16].
* Composition: It is same as aggregation but here classes are tightly coupled so existence is dependent on both classes [16].
* Inheritance: If there is a parent-child (Super-Sub Class) relationship between classes then it is known as inheritance [16].

In Figure 2, a class diagram is shown for the proposed Student Registration System.

All the classes that have been created for the Web-Application are shown in the figure. These classes are related with each other through a relationship of aggregation and inheritance. The aggregation is denoted by a line with a hollow diamond at the end of parent class. The inheritance in denoted by a line with hollow triangle towards the parent class.

­

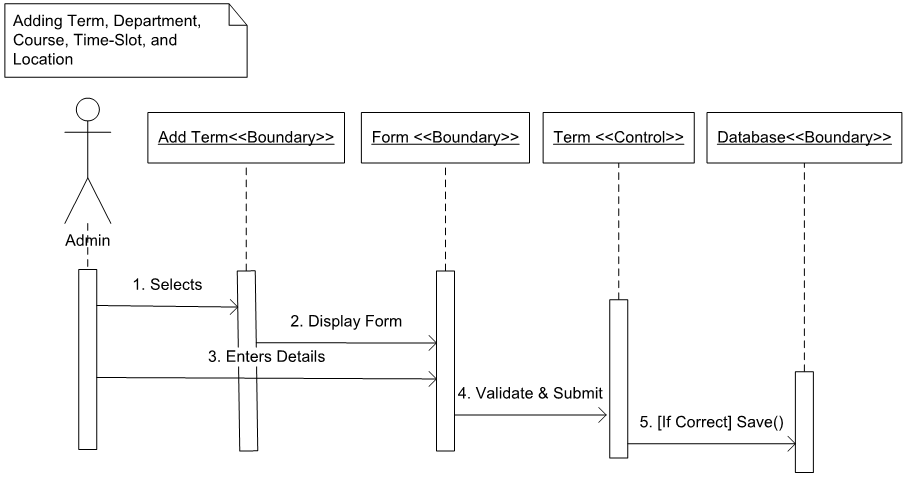
**FIGURE 2. Class diagram.**

**4.3 Sequence Diagram**

Sequence diagram shows the interaction between objects [17] and in what order they communicate with each other. These objects are of three types:

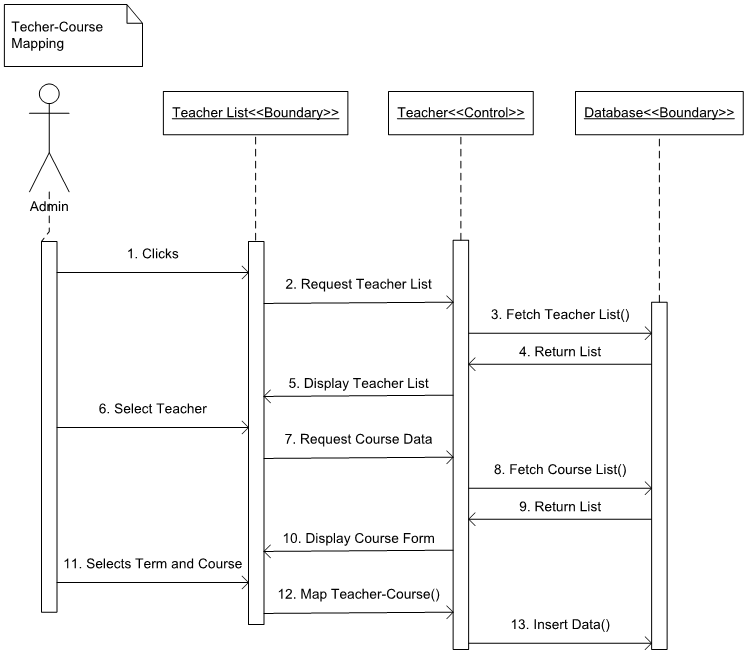
* Entity Object: These objects represent real world entity such as a thing or person [18].
* Control Object: These kind of objects manage the flow of interaction; they process the inputs using functions that are implemented inside their scope and forward the output to the appropriate object [18].
* Boundary Object: These objects lie on the boundary of the system; these are the front-end elements such as buttons, forms, and external link [18].

Moreover, sequence diagram depicts lifetime of the object by a vertical bar connected to a rectangular box on top of it, which represents the object itself. The horizontal arrows show the messages exchanged between the objects. This diagram consists of an actor that invokes the function and triggers the execution of sequence diagram. In the following figures, few sequence diagrams for the proposed system have been demonstrated, which provides insight to the orderly execution of various functions [18].



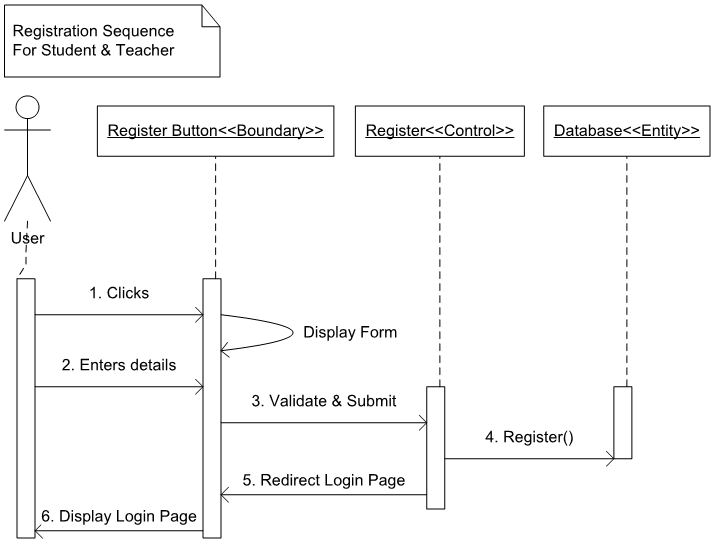
**FIGURE 3. Sequence diagram: Add term.**

The sequence diagram shown in Figure 3 is demonstrating the sequence of execution whenever admin wants to add Term, Department, Course, Time-Slot and Location of the course.



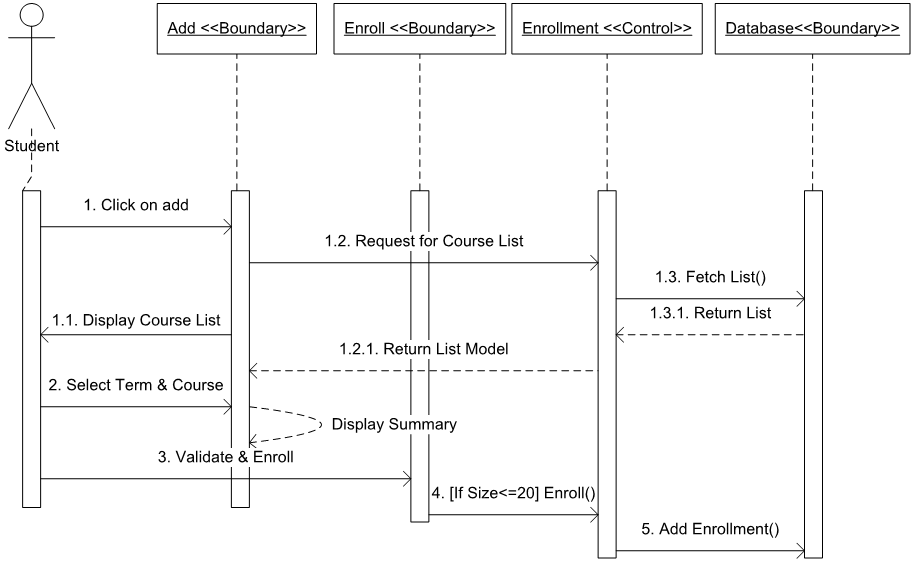
**FIGURE 4. Sequence diagram: Course allocation.**

The sequence diagram shown in Figure 4 is demonstrating the sequence of execution when admin assigns one or more courses to teacher.



**FIGURE 5. Sequence diagram: Registration.**

The sequence diagram shown in Figure 5 is demonstrating the execution sequence of student or teacher registering to the system.



**FIGURE 6. Sequence diagram: Course enrollment.**

The sequence diagram shown in Figure 6 will be executed when students want to enroll in the course.

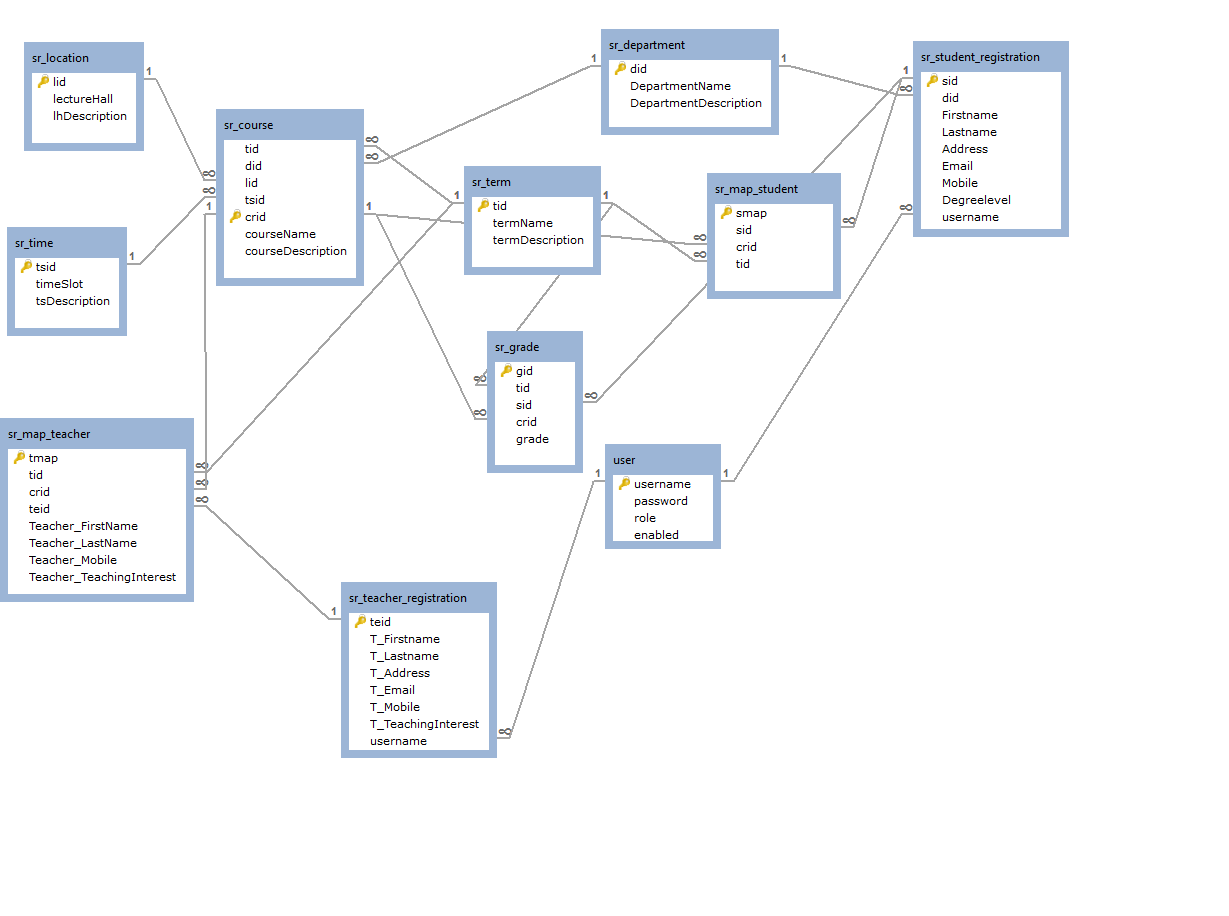
**4.4 Database**

Adatabase is an organized collection of data, it consists of tables and schemas. In RDBMS (Relational Database Management System) [19] data will be stored in the form of tuples and these tuples will be stored in various tables. The Table sometimes refer as a relation which contains unique rows of data. RDBMS is a structure approach to manage the database in which SQL (Structured Query Language) is often used to create and update database [19].

**4.4.1 Database Schema**

The database schema provides an overview of the database. It offers a detailed analysis on how the tables are associated with each other and how the data is structured. Database schema presents a logical and structural view of the database including relationship between entities (table) [20].

In figure 7, the database schema for the student registration system is depicted.



**FIGURE 7. Database schema.**

**4.4.2 Database Dictionary**

Previously, the database schema that has been discussed gives a logical view of the database. But to get more details about the characteristics of the data attributes, the database dictionary has to be created. A database dictionary is a repository. It stores descriptions of all the data objects, which are used in data models.

**4.5 Hibernate**

The management of persistent data is a crucial requirement that has to be fulfilled to develop most of the software systems. In recent years, Object Relation Mapping (ORM) is a popular solution for maintaining persistent data. ORM represents and converts the data from the database to an object type so that any object-oriented language can interpret and manipulate it with ease. These objects will be converted back to their original form while saving them to the database. Hibernate is the most prominent ORM because it is compatible with many programming languages [21].

Moreover, hibernate provides various features when integrated with an object-oriented language. It has its own query language, Hibernate Query Language (HQL), to interact with the relational database that is similar to SQL (Structured Query Language). HQL is written on top of SQL, so Hibernate supports both of these query languages. Apart from this, the Hibernate ORM has been developed above the JDBC [22] layer, so in the core of Hibernate it uses JDBC call to establish a connection with a database. However, it is more efficient and less cumbersome than the JDBC because it is database independent while JDBC is database specific.

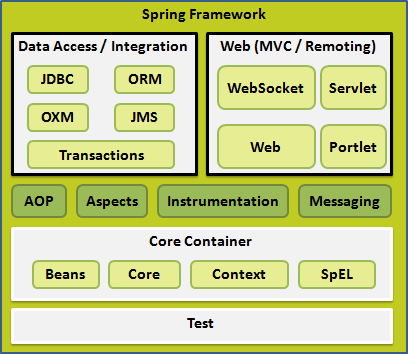
HQL is compatible with both Oracle [23] and MySQL [24] databases, so underlying code is same for both databases, while in JDBC majority of code is different. As shown in Figure 8, Hibernate ORM acts as a bridge between application and relational database. All the configurations and mapping will take place in XML (Extensible Markup Language) [25] files. A Java class will be mapped to a specific table in the database and all the query operation will be performed on it. These types of classes will have getter and setter methods for each variable, essentially these variables will be mapped to the columns of the table.



**FIGURE 8. ORM [26].**

**4.6 Spring Framework**

As discussed in Chapter 1 and Section 3.3, Spring Framework is an application framework that supports the robust development of Java applications by providing a comprehensive infrastructure. The basic version of Spring Framework occupies only 2 Mega Bytes (MB) in the workspace, so it is a light weight framework [27].



**FIGURE 9. Spring Framework architecture [28].**

The architecture of Spring Framework is shown in Figure 9. It is evident from the figure that Spring Framework has modular architecture that gives freedom of implementing only those modules that a developer has selected for the application. This feature of Spring Framework is known as “pick and choose.” In addition, it is not required to handle or integrate the unused modules, so it is completely flexible framework that is adaptable according to developer’s requirement. The Spring Framework consists of around 20 modules [27], which are grouped into various categories. These categories are briefly explained in the following list:

* *Core Container*: The modules of this section are responsible for providing configurational support. These modules are the intrinsic part of the framework on which the key concepts such as Inversion of Control and Dependency Injection are dependent [27].
* *Data Access/Integration*: It is the data access layer of the framework where Spring offers support for diverse database access technologies. As shown in Figure 9, each module represents a different data access medium. These modules have been implemented by Spring so that developers can directly integrate and use them [27].
* *Web*: This layer provides support for web-based applications. All the configurations and mappings for web-development will take place in different modules. Developers choose modules from this layer according to their Web-Application’s requirement [27].

Apart from the modules of these layers, there are some other modules which satisfy miscellaneous purposes such as providing support for testing components. The Spring Framework contains several useful features; among them Dependency Injections is the most prominent characteristic of the framework. In the following section, Dependency Injections (DI) will be explained.

**4.6.1 Dependency Injection**

The Dependency Injection (DI) is a demonstration of the inversion of control mechanism; this concept can be exemplified through various methods but Dependency Injection is the most concrete and relevant method to express the inversion of control.

In the Spring Framework, for a complex Java application, to increase maintainability and testability of Java classes, it is very important to decrease coupling (dependency) between classes. Although, a developer should minimize the dependency between Java classes, but it is usually not possible to achieve completely independent application classes. Dependency Injection plays an important role by connecting Java classes together and at the same time maintain every classes as a discrete unit. To implement this kind of behavior, Spring maintains an XML [25] file that is also known as application-context file, where it maps Java classes. Therefore, if class A wants to create an instance [29] of class B, then class A will invoke the XML file and request for an instance of class B [27].

**4.6.2 Spring Web MVC** **Framework**

In chapter 3, the Model-View-Controllers architecture is explained. MVC is a core concept that has been implemented using different approaches on various platforms such as .NET and Spring. Both of these frameworks have implemented MVC, but its functioning is not similar in both the frameworks. However, both frameworks achieved the same goal, but using different methodology.

As shown in Figure 10, the Spring Framework is implemented using the MVC architecture; this arrangement helps to create loosely coupled applications. It is evident from the figure that every request is intercepted and routed through DispatcherServlet. After receiving incoming HTTP request DispatcherServlet invokes a specific controller suggested by HandlerMapping. Then the controller processes the request and calls the service method associated with the request mapping. The service method executes the business logic and returns the resultant model data to DispatcherServlet. Subsequently, with the help of View Resolver the view file will be invoked and model data will be sent to the view. Finally, browser will display the view including processed model data [30].



**FIGURE 10. Working of Spring MVC Framework [30].**

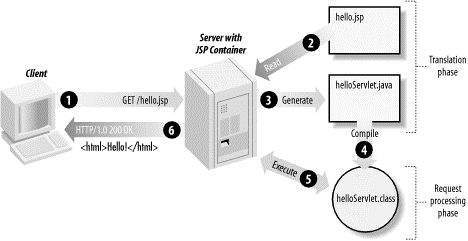
**4.7 Java Server Pages**

In this section, JavaServer Pages (JSP) technology will be introduced and its working will be explained. It is a server side programming technology that helps create dynamic web-pages [31]. For developing Web-Applications in Java, JSP is the crucial technology that is integrated with HTML for supporting Java code and dynamic content of the application. HTML does not support dynamic content, but by merging HTML and JSP the web-page will have both an interactive UI and the dynamic data support.

JSP is a type of Java Servlet [32] because the ultimate purpose behind developing JSP was to satisfy the requirement of a UI (User Interface), which supports the dynamic content of the Web-Application. Therefore, JSP can be treated as an abstraction of Java Servlets because during the execution time JSP files are converted to Java Servlets. JSP supports all potential Java APIs such as JDBC, JNDI [33], EJB [34], and JAXP [35]. Among these APIs, in the Student Registration System, JDBC has been used extensively for the representation of retrieved data from the database [31].

**4.7.1 Working**

The step-by-step processing of JSP page is shown in Figure 11. All steps are described in following list:



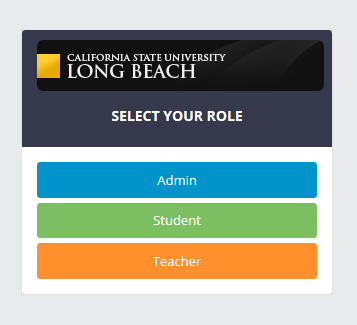
**FIGURE 11. Processing of a JSP page [31].**

* The web-browser sends HTTP request to the web-server [31].
* The web-server with a JSP container interprets the page and will eventually recognize that the web-page is JSP page by reading the “. jsp” extension of the page. The web- browser then forwards the JSP page to a JSP engine (container) [31].
* The JSP engine converts the JSP page into an appropriate Java Servlet [31].
* The JSP engine creates an executable class file after the compilation of the JSP page and forwards the request to the servlet engine (a part of web-server) [31].
* Servlet engine executes the class file and generates a HTML output, which is forwarded to the web-browser along with the HTTP response [31].

**CHAPTER 5**

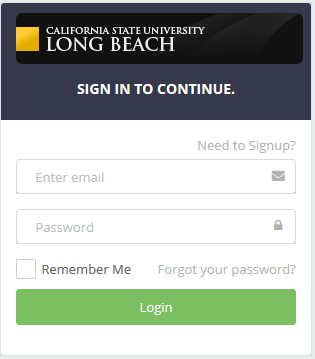
**RESULTANT SYSTEM**

In the previous chapters, the technologies that were used in developing the Student Registration System have been discussed. The MVC architecture and Java Spring Framework are the core technology on which the whole system relies. Demonstration of these technologies is the main goal behind the development of the Student Registration System. All the discussed technologies are successfully integrated and implemented with each other. As a result, a robust and efficient Web-Application has been developed in which all aspects of Model-View-Controller architecture and Spring Framework have been distinctly exemplified.

In this chapter, the Student Registration System will be presented using mock-ups. All the crucial functionalities of the system will be exemplified in these mock-ups. In the following figures these mock-ups have been presented with their descriptions:

**FIGURE 12. Role page.**

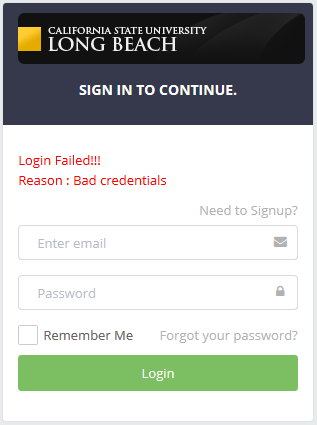
In Figure 12, the role page has been displayed. This page is the welcome page of the Web-Application, which will open first whenever the Web-Application is executed. As it can be seen from the figure that there are three types of users: Admin, Student, and Teacher. Users will click a button and select their appropriate role.



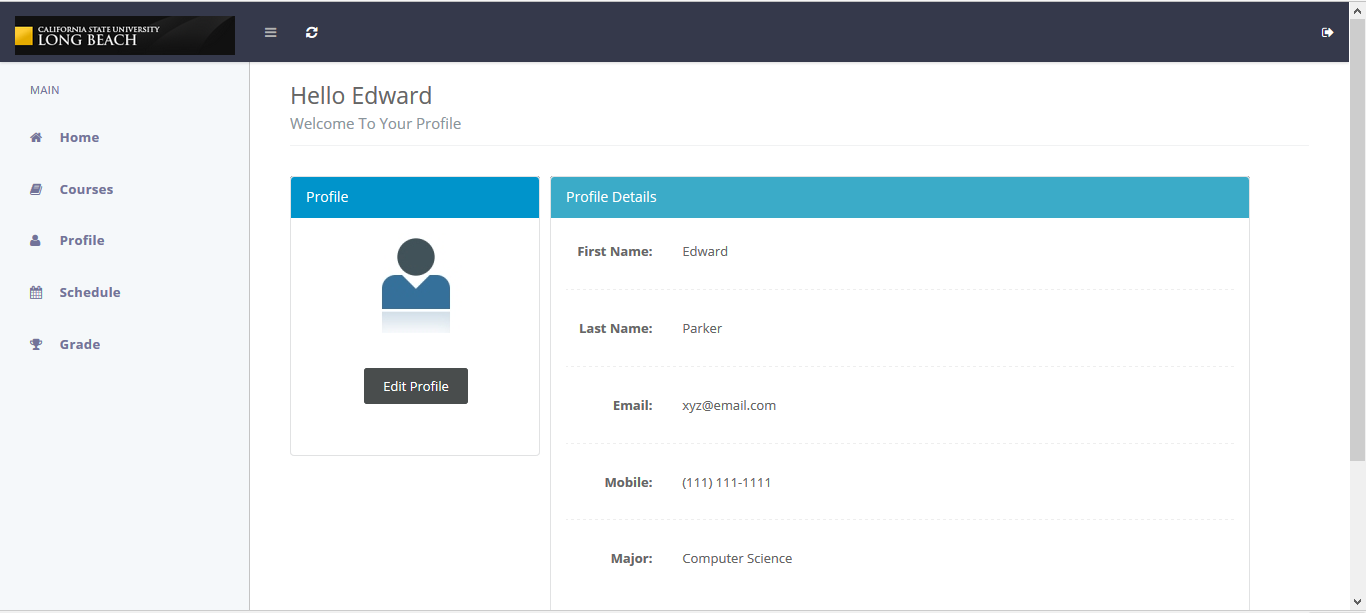
**FIGURE 13. Login page.**

After selecting the role, users will be redirected to the login page that is shown in Figure 13. Then they will enter their credentials and if entered username and password are correct then users can access their profile. However, if the entered credentials were incorrect, then the system will deny the access and shows an error message as shown in Figure 14. To implement this login module an efficient technology known as Spring Security [36] has been used.

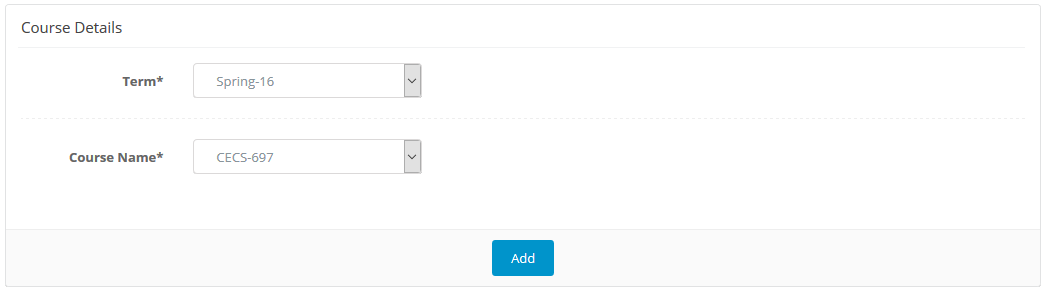
Once the users (teacher or student) are successfully logged into the system, their profile page will be opened as shown in Figure 15. The menu is also displayed in the figure from which user can select any option to perform their tasks.



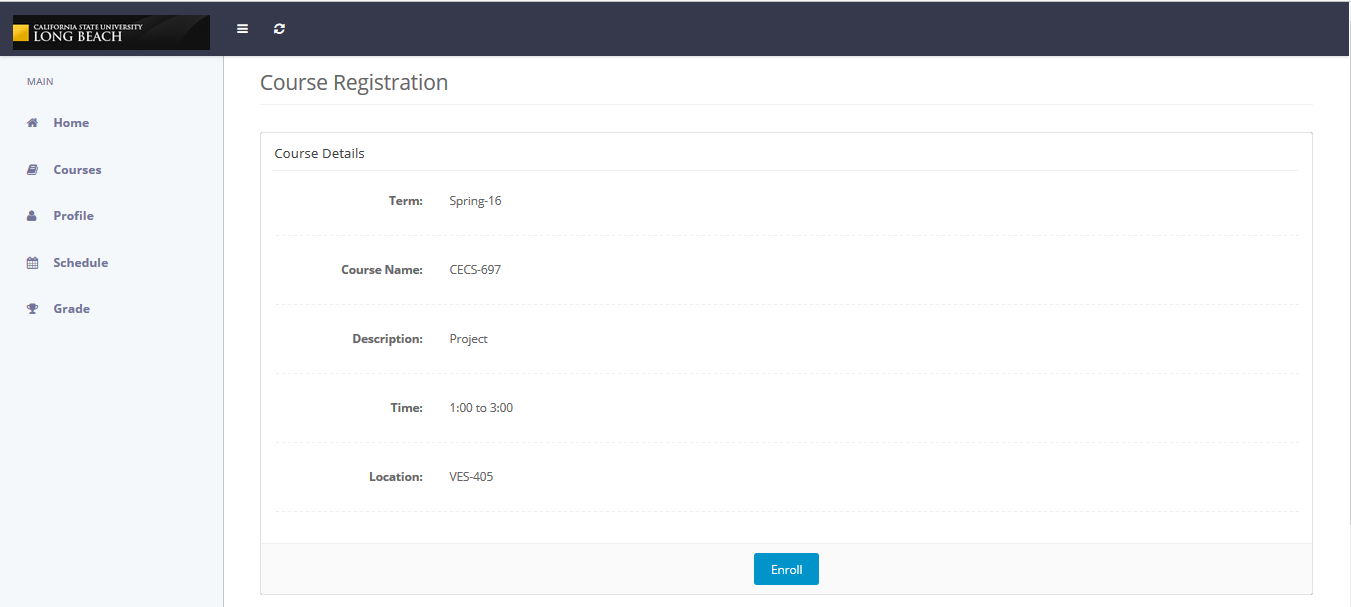
**FIGURE 14. Login error**

****

**FIGURE 15. Profile page**

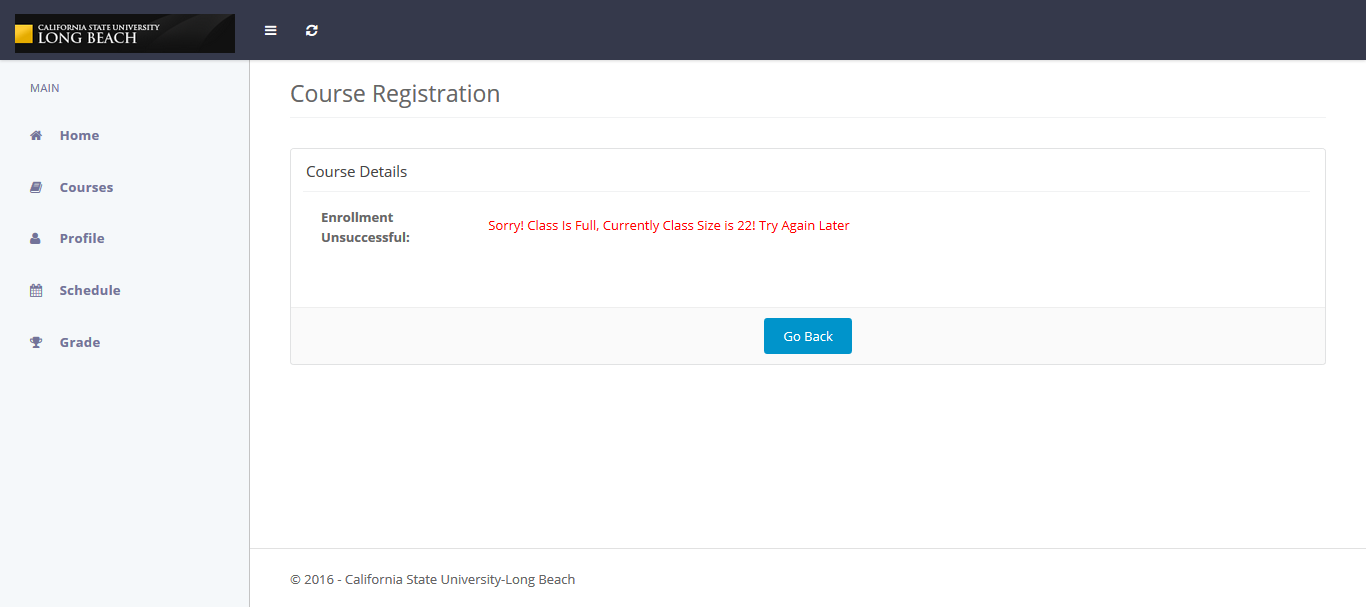
****

**FIGURE 16. Add class**

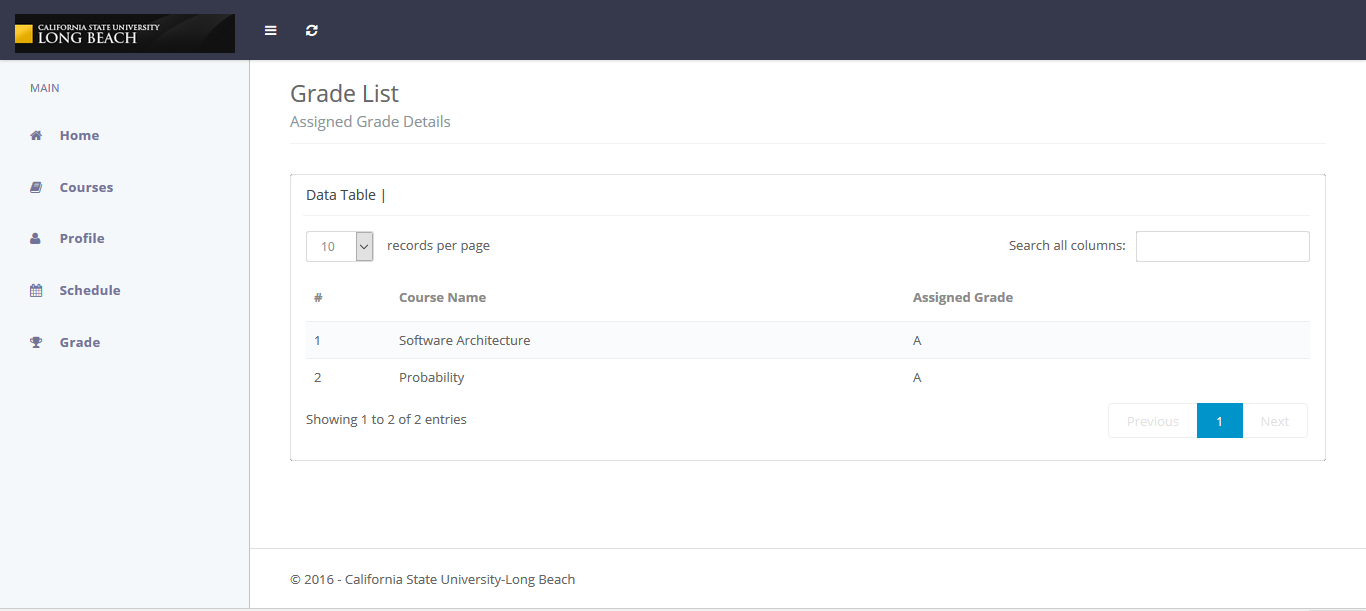
Figure 16 shows the page when students select the add class option from the menu. From this page, they will select the Term Name and Course Name to enroll in a particular course. After clicking on the Add button, the system will validate that the entered course has available seats because in a particular course no more than 20 students can enroll for one term. 

**FIGURE 17. Enrollment page**

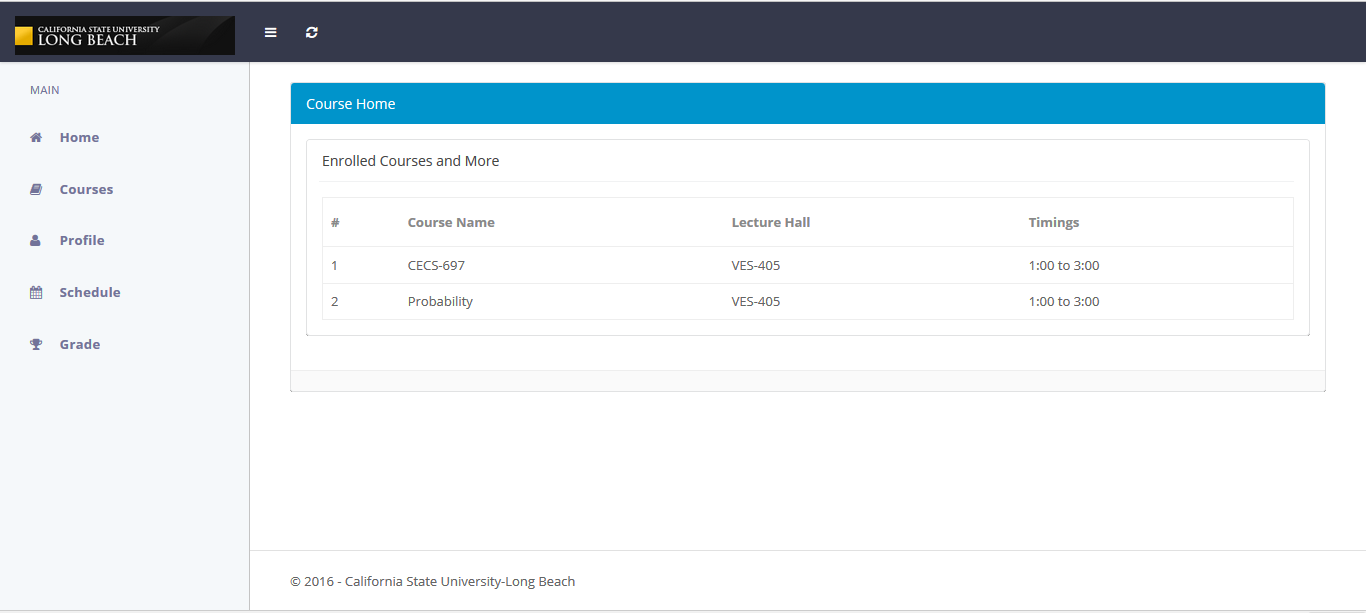
If the validation returns with any available seats, then the summary page shown in Figure 17 will be displayed, and after clicking on the enroll button, the course will be successfully added in the student’s schedule. In the opposite condition, the validation will return with the error message shown in Figure 18.



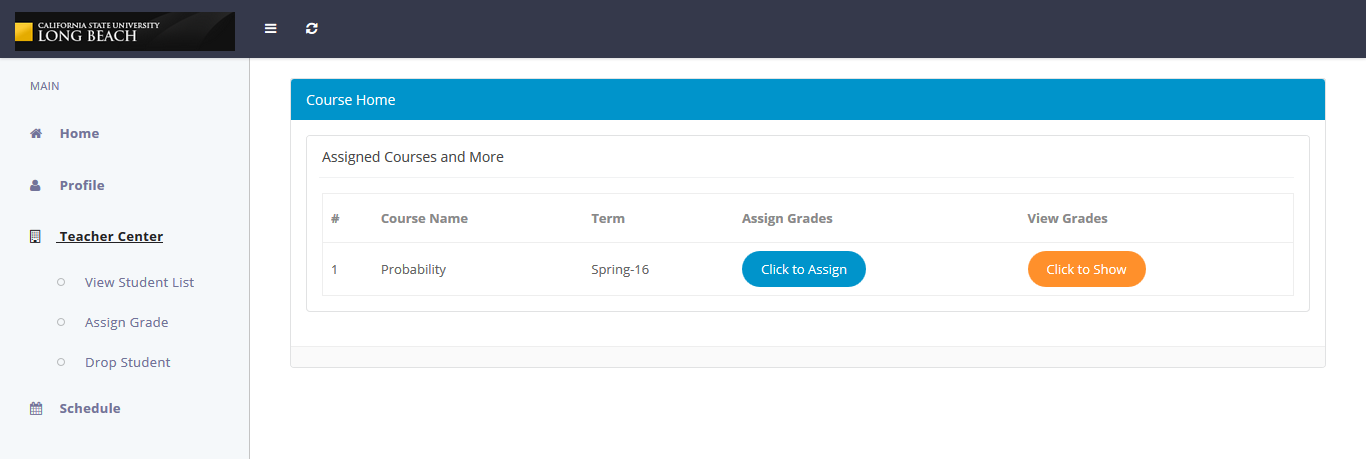
**FIGURE 18. Enrollment error.**

****

**FIGURE 19. Grade page.**

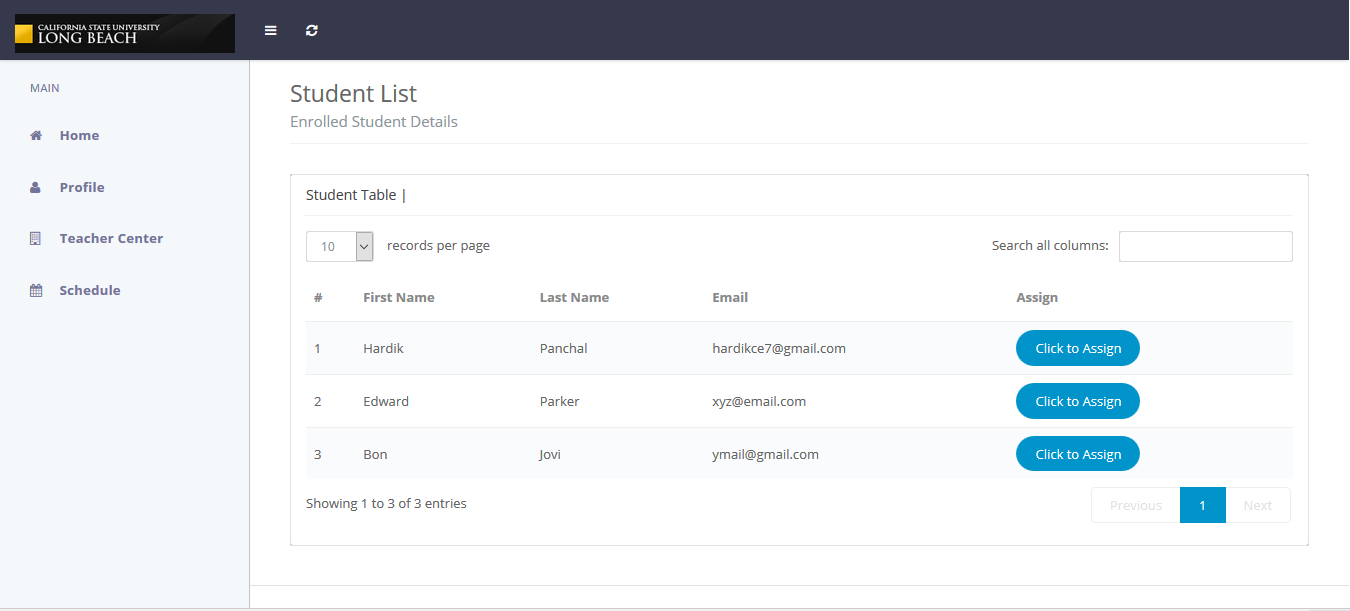
**FIGURE 20. Schedule page.**

Students can see their grades by selecting the Grade option from the menu as shown in Figure 19. The grade page shows the assigned grades once the allocated teacher assigns them. In Figure 20 the schedule page is shown, which has a common structure for both students and teachers. This page shows locations and timings for the assigned or enrolled courses.

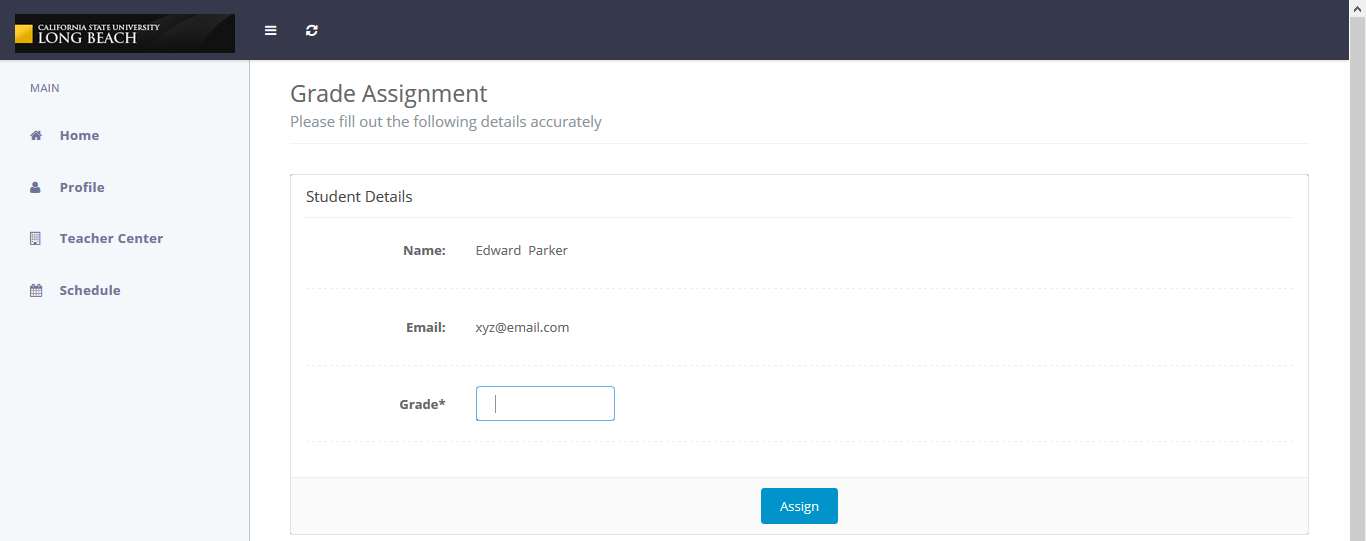


**FIGURE 21. Teacher Grade page.**

According to Figure 21, teachers can perform different tasks by selecting appropriate option from the menu list. The Teacher Center option is filled with all the featured functions that teachers can perform. When a teacher selects the Assign Grade option from the menu, the page shown in figure will be displayed. If a teacher clicks on the assign button, the page shown in Figure 22 will be displayed.

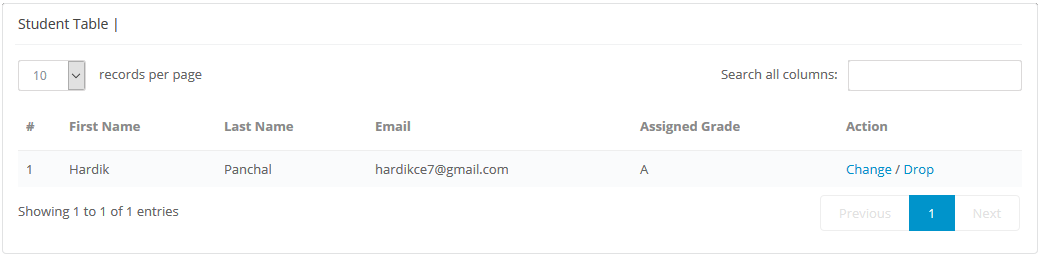


**FIGURE 22. Grade assignment 1.**



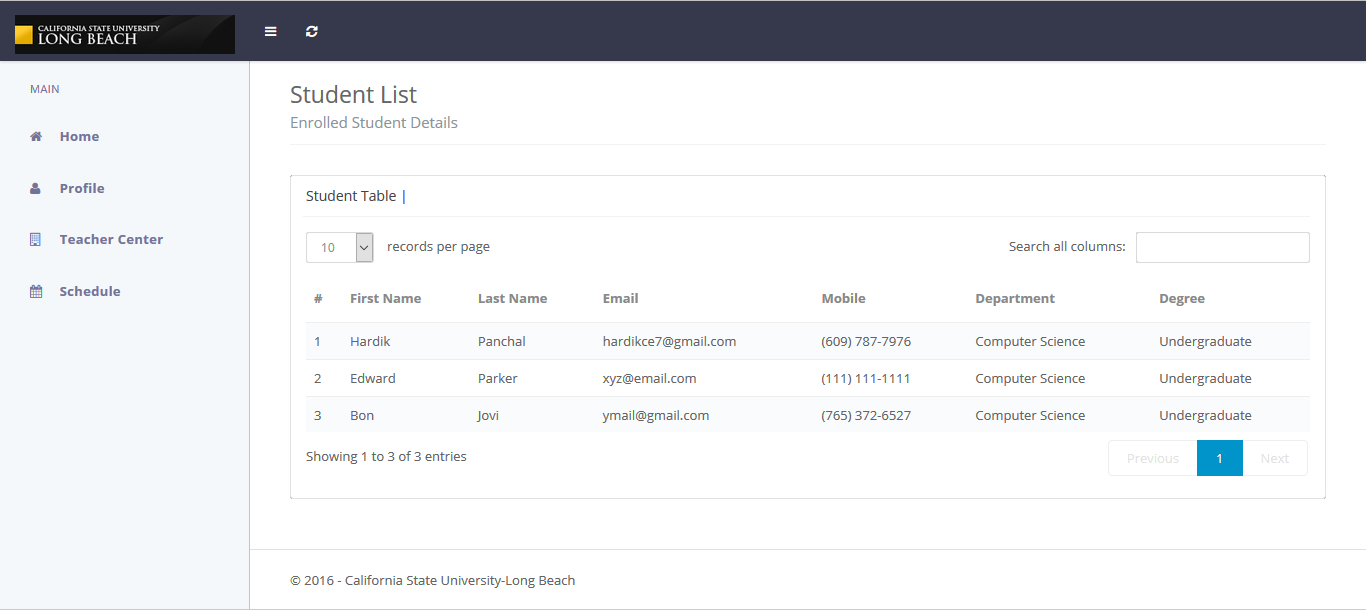
**FIGURE 23. Grade assignment 2.**

A teacher will select the student and enters an appropriate grade in the Grade field as shown in Figure 23. Afterwards, if a teacher wants to change the assigned grade, then by selecting the View Grade option this modification can be done as shown in Figure 24.

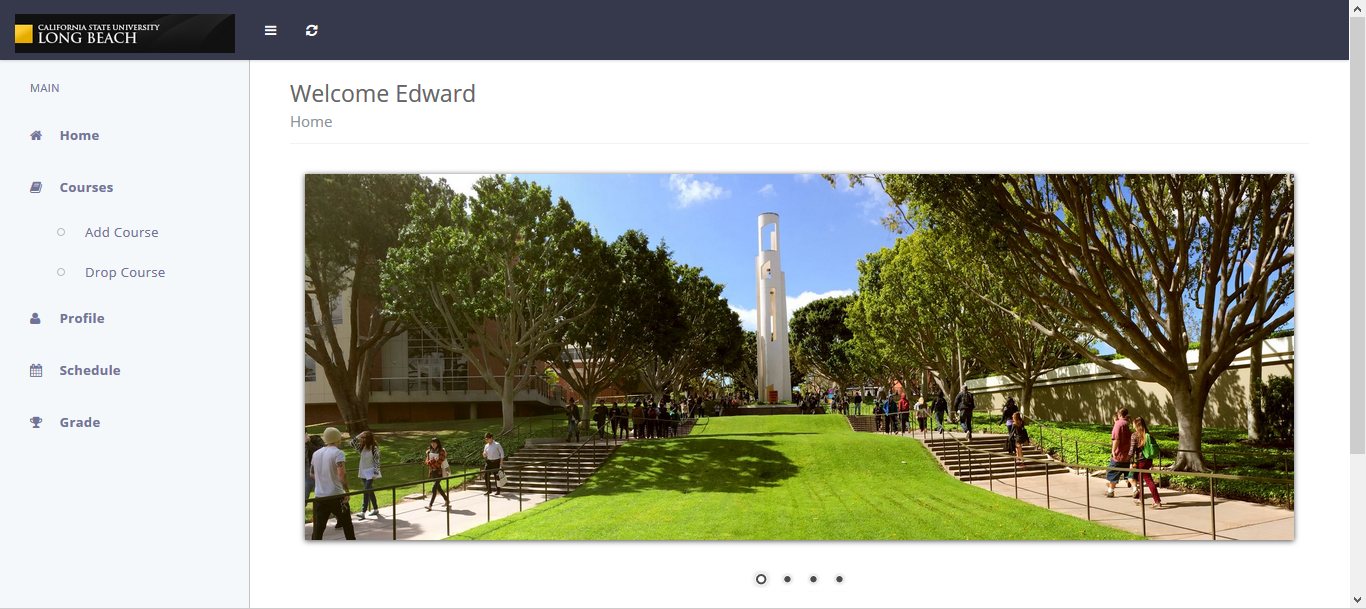


**FIGURE 24. Grade change.**

To acquire the information about all the enrolled students, teachers have to select the View Student List option from the teacher Center option shown in Figure 21, and the student list shown in Figure 25 will be displayed.

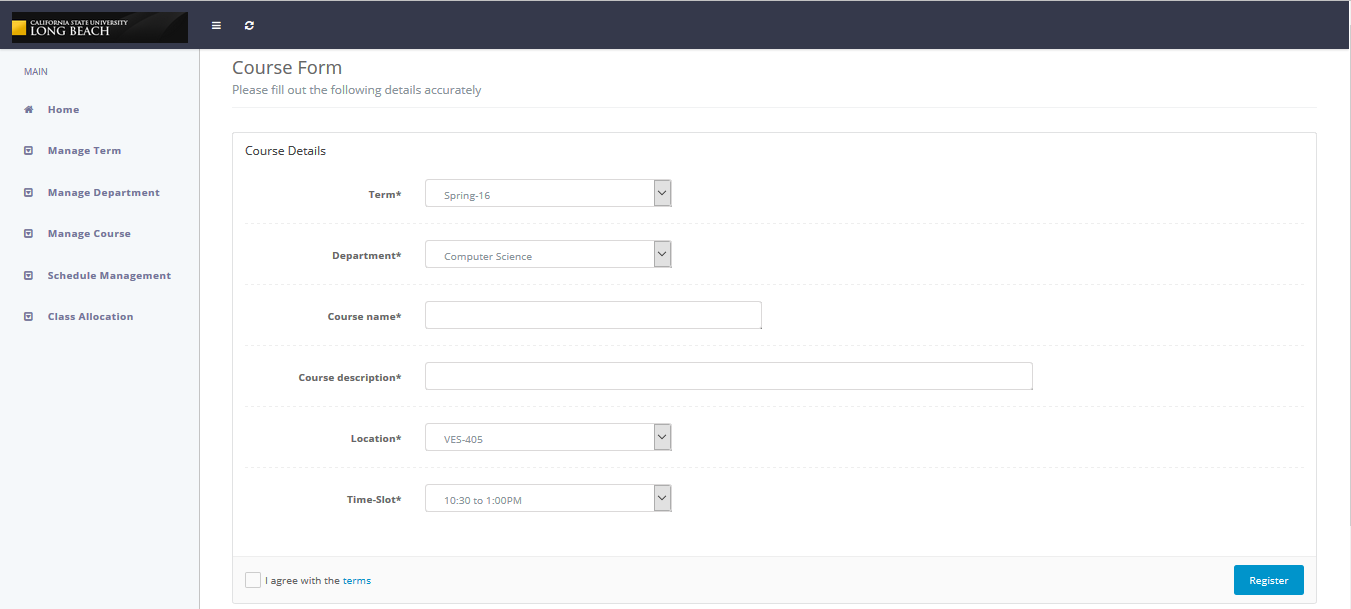


**FIGURE 25. Student list.**

****

**FIGURE 26. Home page.**

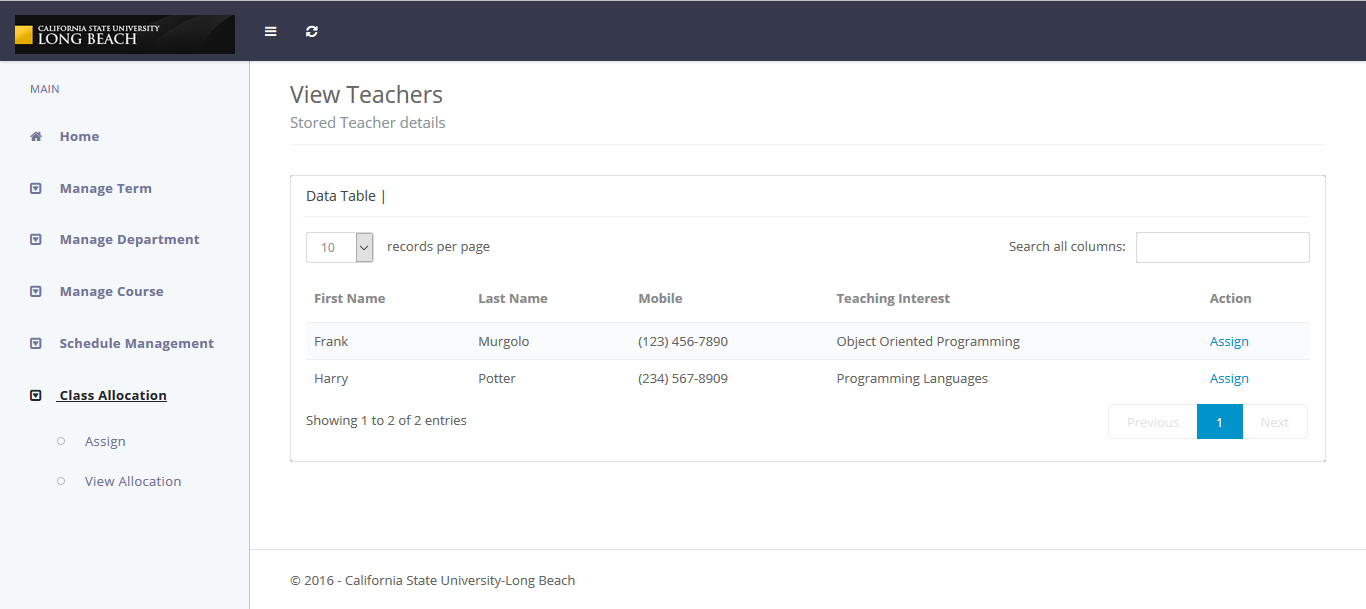
Thehome option is given to all three types of users: admin, student, and teacher. The home page is shown in Figure 26, which is similar for all types of users. Admin is the one who manages all features provided to students and teachers.



**FIGURE 27. Add course.**

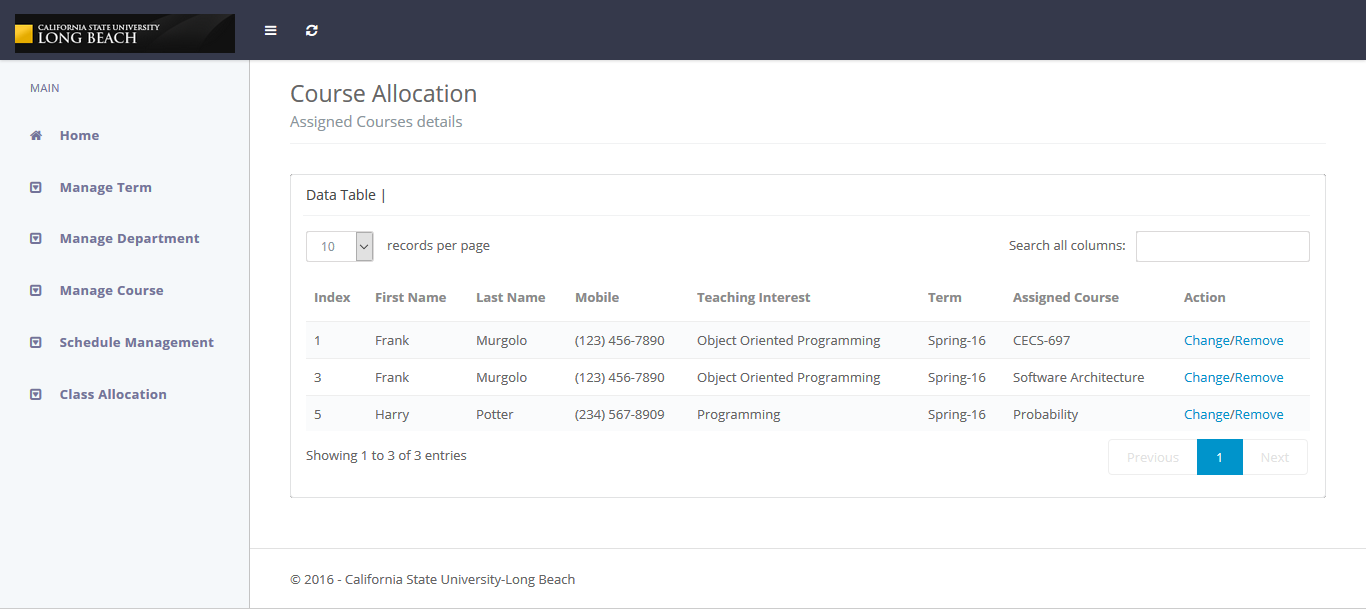
The admin is responsible for the management of various modules such as term, department, course, location, time, and course allocation. To manage these modules, the admin has separate options as shown in Figure 27. The admin adds a course by selecting the Manage Course option, and then the page shown in the figure will be displayed. After entering all the required information in the form, clicking on the Register button will add the course in the database.

In the Student Registration System, the admin assigns the added courses to teachers, and once the assignment is done, teachers cannot change the assigned course on their own. Only the admin can make modifications in the course allocations. Selecting the Assign option from the Course Allocation option will open the page shown in Figure 28. In this figure, the teacher list is displayed. From this list, the admin selects a teacher and clicks on the assign button.

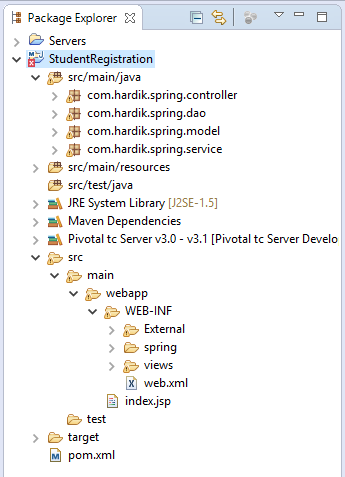


**FIGURE 28. Course allocation.**

Moreover, if the admin wants to change the course allocation, then by selecting the View Allocation option, a page shown in Figure 29, will be displayed. As shown in the figure, system provides two options: Change and Remove to manage the course allocation.



**FIGURE 29. Change course allocation.**

****

**FIGURE 30. Implemented Spring MVC.**

The implemented Spring MVC architecture for the Student Registration System is shown in Figure 30. The concept behind the working of this architecture is exactly same as discussed in Chapter 4. The directory structure shown in the figure is organized according to MVC architecture. It contains the three components of MVC. Under src/main/java folder there are four folders, and among them controller and model have same significance as discussed in the previous chapters.

Data access objects (DAO) are shown the figure, which are responsible for interacting with the database. The service folder is an added layer above the data access layer so that the controllers cannot directly access the data access layer. The service folder implements an important feature of the Object Oriented Programming [15] known as Encapsulation [37].

**CHAPTER 6**

**CONCLUSION**

The Student Registration System is a tool that is built on the MVC architecture and the Spring Framework. The ultimate goal is the demonstration of the MVC architecture with integration of the Spring Framework. This goal is successfully achieved by developing a Web-Application: The Student Registration System, which has implemented all aspects and characteristics of the MVC architecture and the Spring Framework.

The Student Registration System is presented through mock-ups. These mock-ups have been designed using the latest designing technologies to provide an interactive UI (User Interface) for students, teachers, and admins. Students and teachers can register into the system to manage their profiles. Students can enroll into their desired courses during the beginning of the new term and can perform various functions such as manage courses, schedules, grades, and personal information. Similarly, teachers can also manage their courses by adjusting their student lists, schedules, grade assignments, and saved information. In the Web-Application, the admin manages the functions that are provided to teachers and students. The Student Registration System mainly focuses on the enrollment system because it is the core functionality of the system. The system has been successfully developed to be user-friendly so that students, teachers, and admins can perform various functions with ease.

The MVC architecture provides a better management of coding by isolating the model, view, and controller of the software application. Applications that are structured using the MVC architecture support effortless modifications. Also, the Spring Framework supports the robust development and dependency injection; therefore, various modules can easily interact with each other without being involved in complex coding. As a result, the implementation of the MVC architecture and Spring Framework will immensely help developers to create an efficient application with a rich infrastructure.

**APPENDIX**

**USE CASES**

|  |  |
| --- | --- |
| **USE CASE NAME** | 1. **Installing and Configuring JDK** |
| **Actors** | Developer |
| **Pre-Conditions** | 1. Developer should have reliable internet connection. |
| **Flow of Control** | 1. Navigate to http://www.oracle.com/ in browser. 2. Download the latest version. 3. Save the downloaded folder in the C drive. 4. Open “System Property” 5. Select “Advanced system setting” 6. Select “Environment variable” 7. Add new variable “JAVA\_HOME” under User variable and set path of the folder. 8. Set “path” variable under System variable. |
| **Post Conditions** | 1. JDK installed and configured with the system. 2. Java Programs runs successfully. |
| **Error-Conditions** | 1. Internet connectivity lost during downloading apache-tomcat. 2. Program is not running because of incorrect path. |

**TABLE 1. Installing and Configuring JDK**

**TABLE 2. Installing and Configuring Apache-Tomcat**

|  |  |
| --- | --- |
| **USE CASE NAME** | 1. **Installing and Configuring Apache-Tomcat** |
| **Actors** | Developer |
| **Pre-Conditions** | 1. Developer should have reliable internet connection. 2. JDK is already configured with the system |
| **Flow of Control** | 1. Navigate to http://tomcat.apache.org/ in browser. 2. Download the latest version. 3. Save the downloaded folder in the C drive. 4. Open “System Property” 5. Select “Advanced system setting” 6. Select “Environment variable” 7. Add new variable “CATALINA\_HOME” under User variable and set path of the folder. |
| **Post Conditions** | 1. Apache tomcat installed and configured with the system. 2. Web-application runs successfully on tomcat server. |
| **Error-Conditions** | 1. Internet connectivity lost during downloading apache-tomcat. 2. Program is not running because of “no server found” error. |

**TABLE 3. Installing Spring Framework**

|  |  |
| --- | --- |
| **USE CASE NAME** | 1. **Installing Spring Framework** |
| **Actors** | Developer |
| **Pre-Conditions** | 1. Developer should have reliable internet connection. 2. JDK is already configured with the system. 3. Apache-tomcat server is configured with the system. |
| **Flow of Control** | 1. Navigate to http://spring.io/ in browser. 2. Download the latest version. 3. Save the downloaded folder in the appropriate drive. 4. Set libraries files in the lib folder |
| **Post Conditions** | 1. Spring libraries are being used in the code without raising any issue. |
| **Error-Conditions** | 1. Internet connectivity lost during downloading Spring framework. 2. Error in code because library file missing. |

**TABLE 4. Student Registration**

|  |  |
| --- | --- |
| **USE CASE NAME** | 1. **Student Registration** |
| **Actors** | Student |
| **Pre-Conditions** | 1. Student should have reliable internet connection. 2. The database connection is required. |
| **Flow of Control** | 1. Student clicks on “Registration” button. 2. Student will enter his all information like Firstname, Lastname, Address, Mobile, Birth-date, Course, Degree Level, and Email. 3. Student will select a unique username and password. 4. Student will click on “Submit” button |
| **Post Conditions** | 1. Student will successfully register in the system. 2. Student will now able to access his account his personal account. 3. Student can enroll and drop courses. |
| **Error-Conditions** | 1. Internet connectivity lost during registration. 2. Entered information is not in proper format- validation error. 3. System gets disconnected from the database server. 4. Password chosen does not have the proper specification. (Password requirements not met) 5. Characters entered in “Password” and “Confirm Password” does not match. 6. E-mail entered is invalid. 7. Contact number has less than ten digits. 8. Data entered is not stored in the database- error: server down. |
| **Non-Functional Requirement** | 1. **Reliability:** All inserted data should be properly stored in the database. 2. **Performance:** All validations should run correctly. |

**TABLE 5. Student Login**

|  |  |
| --- | --- |
| **USE CASE NAME** | 1. **Student Login** |
| **Actors** | Student |
| **Pre-Conditions** | 1. Student should have reliable internet connection. 2. The database connection is required. 3. Student is a registered user. |
| **Flow of Control** | 1. Student clicks on “Login” button. 2. Student will enter his username and password. 3. Student will click on “Submit” button |
| **Post Conditions** | 1. Student will successfully log in to the system. 2. Student will now able to access his account his personal account. 3. Student can enroll and drop courses. |
| **Error-Conditions** | 1. Internet connectivity lost during registration. 2. System gets disconnected from the database server. 3. Entered username and password is incorrect. |
| **Non-Functional Requirement** | **Security:** 1. Website is secured with the SSL certificate. 2. System does not allow unauthorized user to log in.  **Availability:** Server must operate all the time.  **Interoperability:** 1. Website is compatible with every web-browser. |

**TABLE 6. Student Enrollment**

|  |  |
| --- | --- |
| **USE CASE NAME** | 1. **Student Enrollment** |
| **Actors** | Student |
| **Pre-Conditions** | 1. Student should have reliable internet connection. 2. The database connection is required. 3. Students should have already register themselves with system. 4. Student is logged in during enrollment. |
| **Flow of Control** | 1. Student clicks on “Enroll” button. 2. Student selects the term for which he wants to enroll. 3. Student selects the course in which he wants to enroll. 4. Student clicks on “Enroll” button |
| **Post Conditions** | 1. Student will successfully enroll into that particular class. 2. Schedule will be updated |
| **Error-Conditions** | 1. Internet connectivity lost during registration. 2. Entered information is not in proper format- validation error. 3. Violating Course requirement constraint- crossed max limit 4. Class limit exceeded. |
| **Non-Functional Requirement** | 1. **Performance:** Response time of the Schedule change should be 5ms. |

**TABLE 7. Student Drop the Course**

|  |  |
| --- | --- |
| **USE CASE NAME** | 1. **Student Drop the course** |
| **Actors** | Student |
| **Pre-Conditions** | 1. Student should have reliable internet connection. 2. The database connection is required. 3. Students should have already register themselves with system. 4. Student is logged in during enrollment. |
| **Flow of Control** | 1. Student selects the “Drop Course” option. 2. Student selects the course which he wants to drop. 3. Student clicks on “finish dropping” button |
| **Post Conditions** | 1. Student will successfully drop particular subject. 2. Schedule will be updated. |
| **Error-Conditions** | 1. Internet connectivity lost during registration. 2. Entered information is not in proper format- validation error. 3. Deadline of dropping class has been gone. |
| **Non-Functional Requirement** | 1. **Performance:** Response time of the Schedule change should be 5ms. |

**TABLE 8. View Grade**

|  |  |
| --- | --- |
| **USE CASE NAME** | 1. **View grade** |
| **Actors** | Student |
| **Pre-Conditions** | 1. Student should have reliable internet connection. 2. The database connection is required. 3. Students should have already register themselves with system. 4. Student is already logged in into the system. |
| **Flow of Control** | 1. Student selects the “Grade” option. |
| **Post Conditions** | 1. Student will able to see the final grades of his courses if teachers have uploaded it. |
| **Error-Conditions** | 1. Internet connectivity lost during registration. 2. System gets disconnected from the database server. |
| **Non-Functional Requirement** | 1. **Performance:** Grade should be reflected as soon as teacher adds it. |

**TABLE 9. View Schedule**

|  |  |
| --- | --- |
| **USE CASE NAME** | 1. **View Schedule** |
| **Actors** | Student |
| **Pre-Conditions** | 1. Student should have reliable internet connection. 2. The database connection is required. 3. Students should have already register themselves with system. 4. Student is already logged in into the system. |
| **Flow of Control** | 1. Student clicks on “View Schedule” button. 2. Select the term. |
| **Post Conditions** | 1. Student will able to see their class location and timings. |
| **Error-Conditions** | 1. Internet connectivity lost during registration. 2. Entered a wrong term. |
| **Non-Functional Requirement** | 1. **Performance:** Response time of the Schedule change should be 5ms. |

**TABLE 10. Change Personal Information**

|  |  |
| --- | --- |
| **USE CASE NAME** | 1. **Change personal information** |
| **Actors** | Student |
| **Pre-Conditions** | 1. Student should have reliable internet connection. 2. The database connection is required. 3. Students should have already register themselves with system. 4. Student is already logged in into the system. |
| **Flow of Control** | 1. Student clicks on “Edit Profile” button. 2. Edit the information which they want to change. |
| **Post Conditions** | 1. Student’s profile will reflect changes. |
| **Error-Conditions** | 1. Internet connectivity lost during registration. 2. Entered information is not in proper format- validation error. 3. System gets disconnected from the database server. |
| **Non-Functional Requirement** | 1. **Reliability:** The database should reflect the changes. |

|  |  |
| --- | --- |
| **USE CASE NAME** | 1. **Teacher Registration** |
| **Actors** | Teacher |
| **Pre-Conditions** | 1. Teacher should have reliable internet connection. 2. The database connection is required. |
| **Flow of Control** | 1. Teacher clicks on “Registration” button. 2. Teacher will enter his all information like Firstname, Lastname, Address, Mobile, Birth-date, Teaching-Interest, Email. 3. Teacher will select a unique username and password. 4. Teacher will click on “Submit” button |
| **Post Conditions** | 1. Teacher successfully registered in the system. 2. Teacher will now able to access his account his personal account. 3. Teacher can drop a student. |
| **Error-Conditions** | 1. Internet connectivity lost during registration. 2. Entered information is not in proper format- validation error. 3. System gets disconnected from the database server. 4. Password chosen does not have the proper specification. (Password requirements not met) 5. Characters entered in “Password” and “Confirm Password” does not match. 6. E-mail entered is invalid. 7. Contact number has less than ten digits. 8. Data entered is not stored in the database- error: server down. |
| **Non-Functional Requirement** | 1. **Reliability:** All data should be properly stored in the database. 2. **Performance:** All validations should run correctly. |

**TABLE 11. Teacher Registration**

**TABLE 12. Teacher Login**

|  |  |
| --- | --- |
| **USE CASE NAME** | 1. **Teacher Login** |
| **Actors** | Teacher |
| **Pre-Conditions** | 1. Teacher should have reliable internet connection. 2. The database connection is required. 3. Teacher is a registered user. |
| **Flow of Control** | 1. Teacher clicks on “Login” button. 2. Teacher will enter his username and password. 3. Teacher will click on “Submit” button |
| **Post Conditions** | 1. Teacher will successfully log in to the system. 2. Teacher will now able to access his account his personal account. 3. Teacher can drop students as well as manage his account. |
| **Error-Conditions** | 1. Internet connectivity lost during registration. 2. System gets disconnected from the database server. 3. Entered username and password is incorrect. |
| **Non-Functional Requirement** | **Security:** 1. Website is secured with the SSL certificate. 2. System does not allow unauthorized user to log in.  **Availability:** Server must operate all the time.  **Interoperability:** 1. Website is compatible with every web-browser. |

**TABLE 13. Drop Student**

|  |  |
| --- | --- |
| **USE CASE NAME** | 1. **Drop Student** |
| **Actors** | Teacher |
| **Pre-Conditions** | 1. Teacher should have reliable internet connection. 2. The database connection is required. 3. Teacher is a registered user. 4. Teacher is already logged in into the system. |
| **Flow of Control** | 1. Teacher clicks on “Class-List” button. 2. Teacher will select the student which he wants to drop. 3. Teacher will click on “Drop” button |
| **Post Conditions** | 1. Teacher successfully drop the student. 2. Class-List got updated. 3. Student will be notified and course will be removed from his schedule. |
| **Error-Conditions** | 1. Internet connectivity lost during registration. 2. System gets disconnected from the database server. |
| **Non-Functional Requirement** | 1. **Reliability:** Class-list should be updated. |

**TABLE 14. Add Grade**

|  |  |
| --- | --- |
| **USE CASE NAME** | 1. **Add Grade** |
| **Actors** | Teacher |
| **Pre-Conditions** | 1. Teacher should have reliable internet connection. 2. The database connection is required. 3. Teacher is a registered user. 4. Teacher is already logged in into the system. |
| **Flow of Control** | 1. Teacher clicks on “Class-List” button. 2. Teacher will select the student for which he wants to add grade. 3. Teacher will click on “Submit” button |
| **Post Conditions** | 1. Teacher successfully Add the grade. 2. Grade in Class-List got updated. 3. Student will be notified and course history will be updated. |
| **Error-Conditions** | 1. Internet connectivity lost during registration. 2. System gets disconnected from the database server. |
| **Non-Functional Requirement** | 1. **Reliability:** Student’s profile should be updated with grade. |

**TABLE 15. Add Term**

|  |  |
| --- | --- |
| **USE CASE NAME** | 1. **Add Term** |
| **Actors** | Admin |
| **Pre-Conditions** | 1. Admin should have reliable internet connection. 2. The database connection is required. 3. Admin is already logged in into the system. |
| **Flow of Control** | 1. Admin selects the “Add Term” option from the menu. 2. Admin enters the name of the term in the name field. 3. Admin enters the description of the term in the description field 4. Admin will click on “Submit” button |
| **Post Conditions** | 1. Admin successfully added the term. 2. Every student will now able to enroll for that particular term. |
| **Error-Conditions** | 1. Internet connectivity lost during registration. 2. System gets disconnected from the database server. |
| **Non-Functional Requirement** | 1. **Reliability:** All data should be properly store in the database so that it can be used via system. |

**TABLE 16. Add Department**

|  |  |
| --- | --- |
| **USE CASE NAME** | 1. **Add Department** |
| **Actors** | Admin |
| **Pre-Conditions** | 1. Admin should have reliable internet connection. 2. The database connection is required. 3. Admin is already logged in into the system. |
| **Flow of Control** | 1. Admin selects the “Add Department” option from the menu. 2. Admin enters the name of the Department in the name field. 3. Admin enters the description of the Department in the description field 4. Admin will click on “Submit” button |
| **Post Conditions** | 1. Admin successfully added the Department. 2. Every student will now able to select their Department. |
| **Error-Conditions** | 1. Internet connectivity lost during registration. 2. System gets disconnected from the database server. |
| **Non-Functional Requirement** | 1. **Reliability:** All data should be properly store in the database so that it can be used via system. |

|  |  |
| --- | --- |
| **USE CASpE NAME** | 1. **Add Course** |
| **Actors** | Admin |
| **Pre-Conditions** | 1. Admin should have reliable internet connection. 2. The database connection is required. 3. Admin is already logged in into the system. |
| **Flow of Control** | 1. Admin selects the “Add Course” option from the menu. 2. Admin select the term from the drop down. 3. Admin select the department from the drop down. 4. Admin enters the name of the Course in the name field. 5. Admin enters the description of the Course in the description field 6. Admin selects the location from the drop down. 7. Admin select the time slot from the drop down. 8. Admin will click on “Submit” button |
| **Post Conditions** | 1. Admin successfully added the course. 2. Every student will now able to select the course. |
| **Error-Conditions** | 1. Internet connectivity lost during registration. 2. System gets disconnected from the database server. |
| **Non-Functional Requirement** | 1. **Reliability:** All data should be properly store in the database so that it can be used via system. |

**TABLE 17. Add Course**

**TABLE 18. Manage Class Location**

|  |  |
| --- | --- |
| **USE CASE NAME** | 1. **Manage class Location** |
| **Actors** | Admin |
| **Pre-Conditions** | 1. Admin should have reliable internet connection. 2. The database connection is required. 3. Admin is already logged in into the system. |
| **Flow of Control** | 1. Admin selects the “Manage Location” option from the menu. 2. Admin enters the name of the Lecture Hall 3. Admin enters the description of the Lecture Hall 4. Admin clicks “Submit” button |
| **Post Conditions** | 1. Admin successfully added the Lecture Hall to Location. 2. Class allocation table will reflect this change. 3. Student schedule will be updated. 4. Teacher profile will reflect this location. 5. Course will be added using this location. |
| **Error-Conditions** | 1. Internet connectivity lost during registration. 2. System gets disconnected from the database server. |
| **Non-Functional Requirement** | 1. **Performance:** Teachers and students schedule should reflect changes. |

**TABLE 19. Add Time-Slot**

|  |  |
| --- | --- |
| **USE CASE NAME** | 1. **Add Time-Slot** |
| **Actors** | Admin |
| **Pre-Conditions** | 1. Admin should have reliable internet connection. 2. The database connection is required. 3. Admin is already logged in into the system. |
| **Flow of Control** | 1. Admin selects the “Add Time-Slot” option from the menu. 2. Admin enters the time-range 3. Admin enters the description of the time-range 4. Admin clicks “Submit” button |
| **Post Conditions** | 1. Admin successfully added the Time-Slot. 2. Class allocation table will reflect this change. 3. Student schedule will be updated. 4. Teacher Schedule will be updated. 5. Course will be added using this Time-Slot. |
| **Error-Conditions** | 1. Internet connectivity lost during registration. 2. System gets disconnected from the database server. |
| **Non-Functional Requirement** | 1. **Performance:** Teachers and students schedule should reflect changes. |

**TABLE 20. Course Allocation**

|  |  |
| --- | --- |
| **USE CASE NAME** | 1. **Course allocation** |
| **Actors** | Admin |
| **Pre-Conditions** | 1. Admin should have reliable internet connection. 2. The database connection is required. 3. Admin is already logged in into the system. |
| **Flow of Control** | 1. Admin selects the “Manage Course” option from the menu. 2. Admin selects particular teacher from the list. 3. Admin assigns courses to that teacher. 4. Admin clicks “Submit” button |
| **Post Conditions** | 1. Admin successfully added courses to that teacher. 2. Teacher gets notify and profile will be updated. |
| **Error-Conditions** | 1. Internet connectivity lost during registration. 2. System gets disconnected from the database server. |
| **Non-Functional Requirement** | 1. **Reliability:** Teachers’ profiles should be updated after course allocated to them. |

**REFRENCES**

**REFERENCES**

1. W3Schools, “HTML5 Tutorial,” 2016; http://www.w3schools.com/html/.

2. W3schools, “CSS Tutorial,” 2016; http://www.w3schools.com/css/default.asp.

3. W3schools, “JavaScript Tutorial,” 2016; http://www.w3schools.com/js/default.asp.

4. W3schools, “JQuery Tutorial,” 2016; http://www.w3schools.com/jquery/default.asp.

5. W3schools, “Bootstrap Tutorial,” 2016; http://www.w3schools.com/bootstrap/default.asp.

6. Source Making, “Design Patterns,” 2016; https://sourcemaking.com/design\_patterns.

7. Segue Technologies, “Desktop vs. Web Applications: A Deeper Look and Comparison,” 2016; http://www.seguetech.com/desktop-vs-web-applications-a-deeper-look-and -comparison/.

8. Cunningham and Cunningham, Inc., “Smalltalk Language,” n.d.; http://c2.com/cgi/wiki ?SmalltalkLanguage.

9. Trygve/MVC, “MVCXEROX PARC 1978-79,” n.d.; http://heim.ifi.uio.no/~trygver/themes /mvc/mvc-index.html.

10. Tutorials Point, “Basic MVC Architecture,” 2016; http://www.tutorialspoint.com/struts\_2 /basic\_mvc\_architecture.htm.

11. L. Bass, P. Clements, and R. Kazman, “Architectural Tactics and Patterns,” *Software Architecture in Practice,* Boston: Addison-Wesley, 2003, pp. 214.

12. I. Davis, “What Are the Benefits of MVC?” Dec. 2008; http://blog.iandavis.com/2008/12 /what-are-the-benefits-of-mvc/.

13. M. Fowler, “GUI Architectures,” July 2006; http://martinfowler.com/eaaDev/uiArchs.html.

14. Usability, “Use Cases,” Oct. 2016; https://www.usability.gov/how-to-and-tools/methods /use-cases.html.

15. Oracle, “Lesson: Object-Oriented Programming Concepts,” 2015; https://docs.oracle.com /javase/tutorial/java/concepts/.

16. Creately, “Class Diagram Relationships in UML with Examples,” 2016; http://creately.com /blog/diagrams/class-diagram-relationships/#Association.

17. Oracle, “What is an Object?” 2015; https://docs.oracle.com/javase/tutorial/java/concepts /object.html

18. Eclipse, “Conceito: Entity-Control-Boundary Pattern,” n.d.; http://epf.eclipse.org/wikis /openuppt/openup\_basic/guidances/concepts/entity\_control\_boundary\_pattern,\_uF -QYEAhEdq\_UJTvM1DM2Q.html.

19. TechTarget, “relational database,” April 2006; http://searchsqlserver.techtarget.com /definition/relational-database.

20. Tutorials Point, “DBMS - Data Schemas,” 2016; http://www.tutorialspoint.com/dbms /dbms\_data\_schemas.htm.

21. Hibernate, “Hibernate ORM documentation (5.2),” n.d.; http://hibernate.org/orm /documentation/5.2/.

22. Oracle, “Lesson: JDBC Basics,” 2015; https://docs.oracle.com/javase/tutorial/jdbc/basics/.

23. Oracle, “Introduction to the Oracle Database,” 2016; https://docs.oracle.com/cd/B19306\_01 /server.102/b14220/intro.htm.

24. MySQL, “1.3.1 What is MySQL?” 2016; https://dev.mysql.com/doc/refman/5.7/en/what-is-mysql.html.

25. W3C Information and Knowledge Domain, “Extensible Markup Language (XML),” May 2015; https://www.w3.org/XML/.

26. JavaTpoint, “Hibernate Framework,” 2014; http://www.javatpoint.com/hibernate-tutorial.

27. Tutorials Point, “Spring Framework – Overview,” 2016; https://www.tutorialspoint.com /spring/spring\_overview.htm.

28. Tutorials Point, “Spring Framework – Architecture,” 2016; https://www.tutorialspoint.com /spring/spring\_architecture.htm.

29. Oracle, “Understanding Class Members,” 2015; https://docs.oracle.com/javase/tutorial /java/javaOO/classvars.html.

30. Tutorials Point, “Spring - MVC Framework Tutorial,” 2016; https://www.tutorialspoint.com /spring/spring\_web\_mvc\_framework.htm.

31. Tutorials Point, “JSP – Architecture,” 2016; https://www.tutorialspoint.com/jsp /jsp\_architecture.htm.

32. Oracle, “Java Servlet Technology,” 2015; http://docs.oracle.com/javaee/6/tutorial/doc /bnafd.html.

33. Oracle, “Java Naming and Directory Interface (JNDI),” 2015; http://www.oracle.com /technetwork/java/jndi/index.html.

34. Oracle, “Enterprise JavaBeans Technology,” 2015; http://www.oracle.com/technetwork/java /javaee/ejb/index.html.

35. Oracle, “Lesson: Introduction to JAXB,” 2015; https://docs.oracle.com/javase/tutorial/jaxb /intro/.

36. Spring, “Spring Security,” 2016; http://projects.spring.io/spring-security/.

37. Tutorials Point, “Java – Encapsulation,” 2016; https://www.tutorialspoint.com/java

/java\_encapsulation.htm.