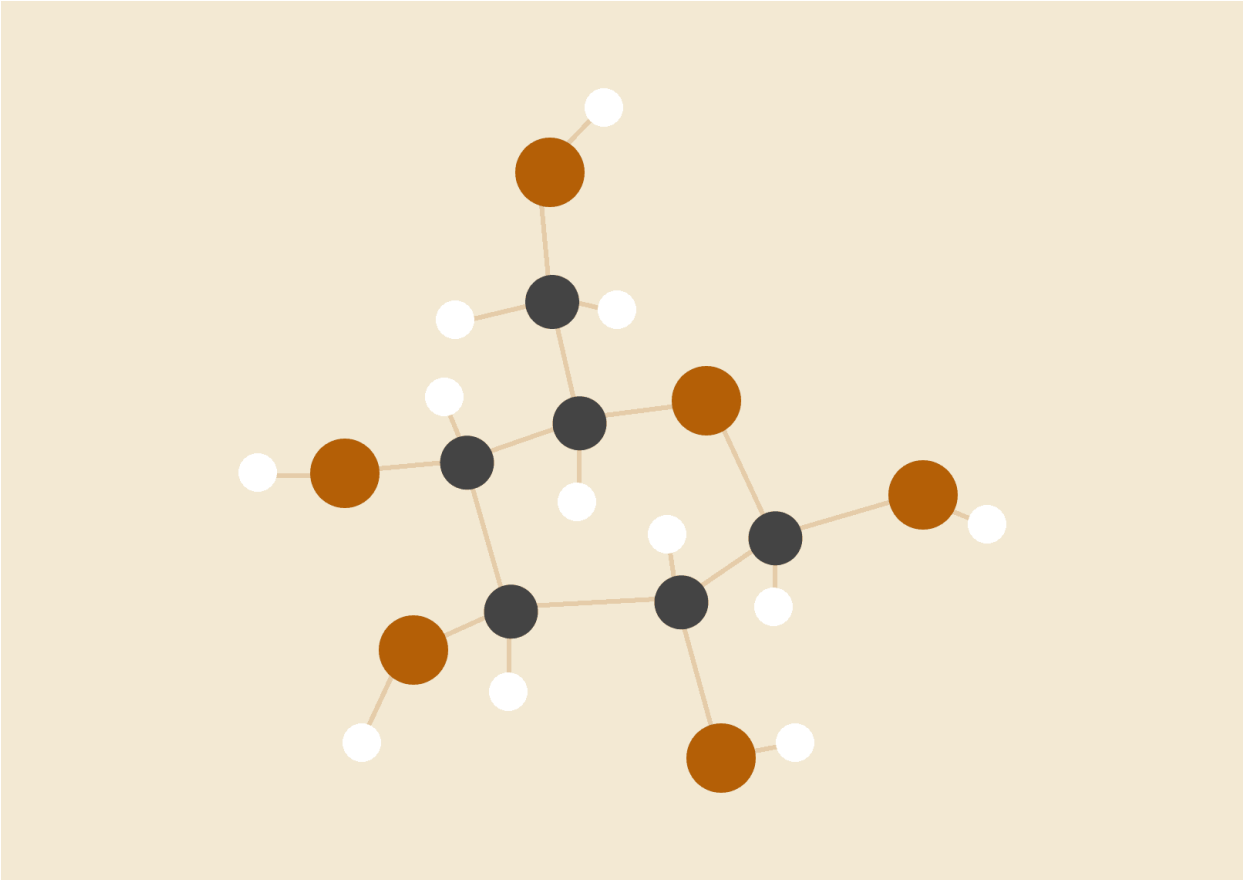
Student Grading System



Software is a great combination between artistry and engineering

Ahmad Omar Mohammad

Table of Contets

OVERVIEW

In this project I built a simple student grading system using various java technologies and frameworks. The development of this system took variant forms, built firstly using Java socket Programing methods. Then upgraded to a webapplication it using MVC Servlets and Jsp. Lastly it was built using the most modern framework called spring boot.

SOCKET PROGRAMMING

Java socket programming allows you to create networked applications that can communicate with each other over a network.

Application Specification

This application consists of multiple components in order to operate.

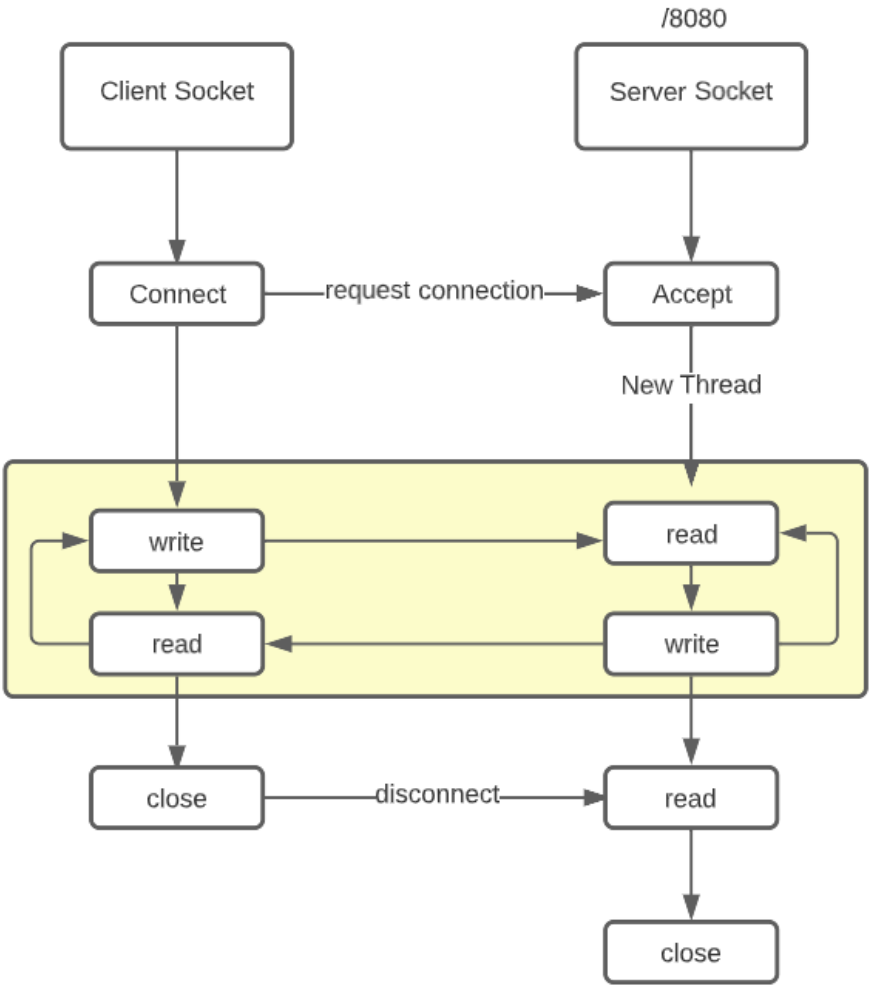
- **Client Socket** this is responsible of connecting the client to the server socket. This socket sends the username and password entered by the client to the server, if the login succeeds, this socket will receive all the student information from the server.
- **Server Socket** this is responsible of receiving a connection request, this request is consisting of an account for authentication, if the account is authenticated, the server will fetch all the student information from the database, after that the server will store the data in a object then sends it back to the client socket.
- **JavaFX GUI** this interface is created to facilitates the interaction of the user with the system.
- **MySql Database** for storing all student related data. **Data Access Object** is implemented to make the programmer life easier when interacting with the database.

This application is organized into three packages:

- **Data**: contains all DAO classes that are responsible of accessing the database.
- **Models:** contains all the classes that are using as beans to interact with data.
- **Network:** contains server and client sockets and the GUI.

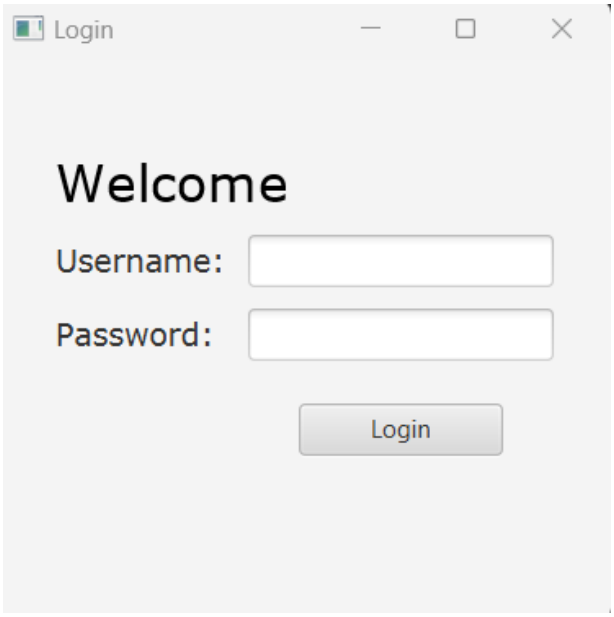
Application Life Cycle

This figure below demonstrates the interaction between the client socket and the server socket, when a request is made to the server, the server creates a new thread so serve this client, then an exchange of data read and write is made between the sockets. Once client finishes reading data from the server, the socket will close.

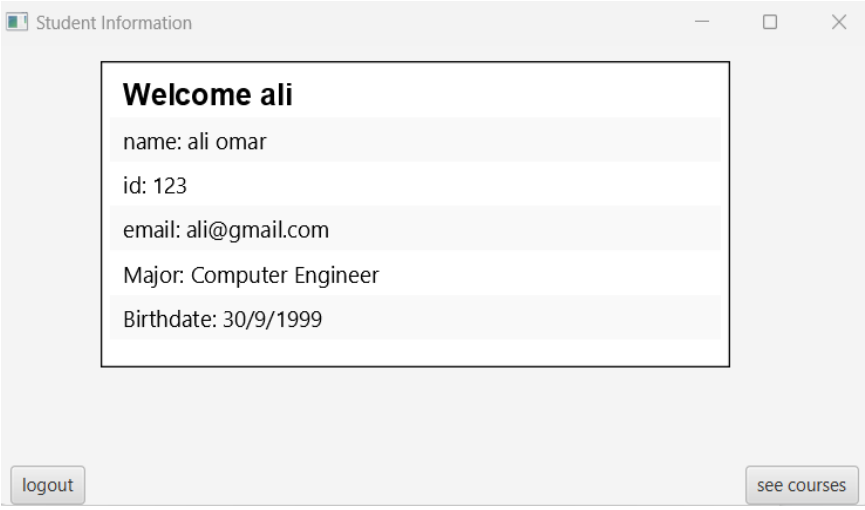


Client Side

First of all a login screen is displayed to the user to enter his credentials. If the credentials are correct, the user will be directed their account.

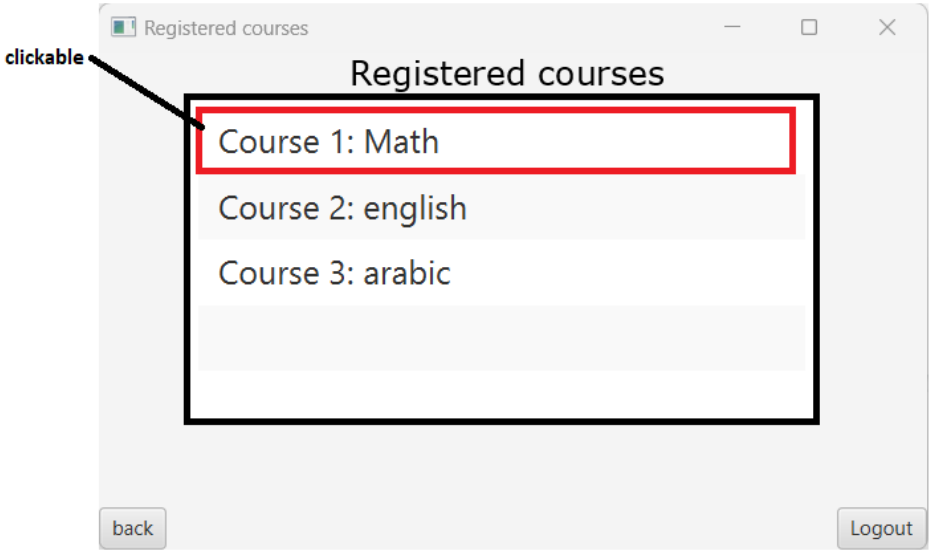


After the user successfully logs in, the server will check the student id and fetch all related info about him/her from the database.

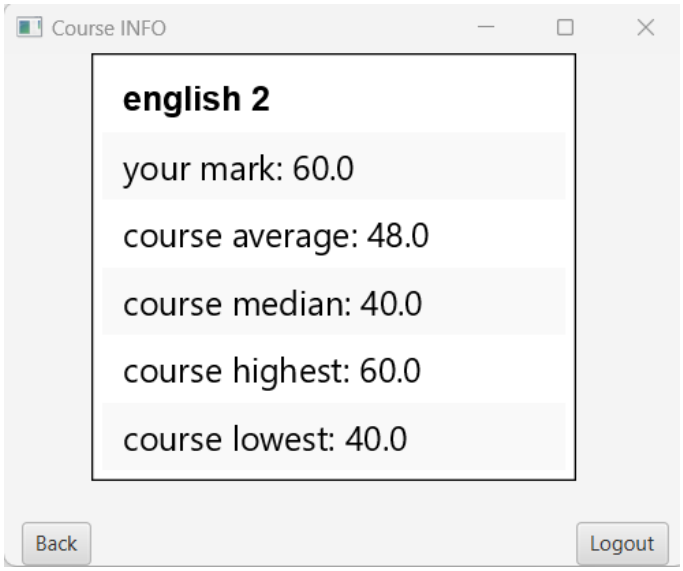


If the user clicks on **see courses** button, he will be directed to next page.

Now a clickable list of courses will be viewed to the student that he's registered to.



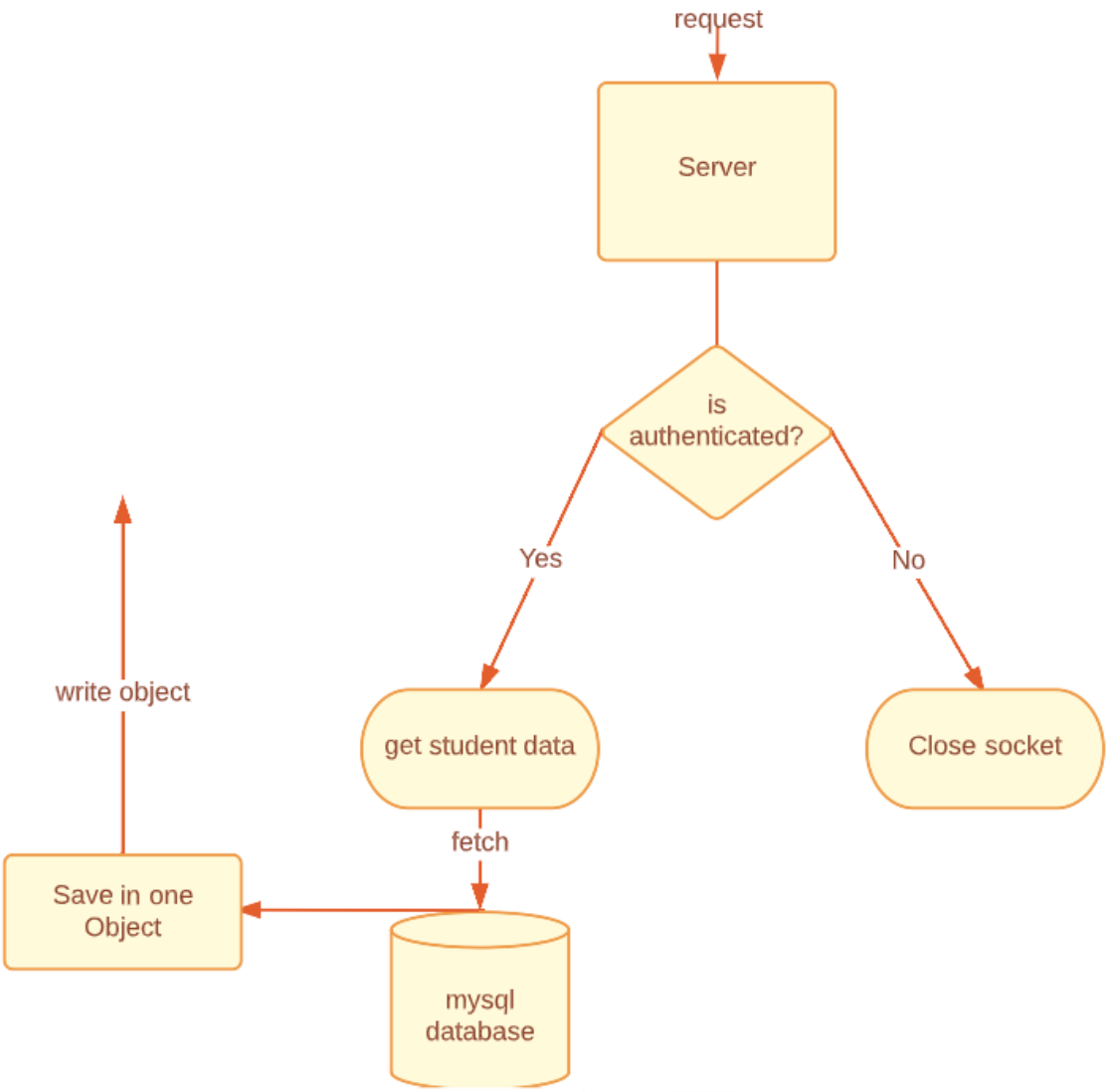
Now when you click for example on english course, this list will be viewed, including the student mark. and a statistics about the course marks. If the student mark isn't listed yet by the instructor, it will be displayed as NaN.



The student can use the back button to traverse between the pages.

Server side

The figure below illustrates what does the server do when he receives a request. Firstly he authenticates the request, if the request is authenticated he sends “SUCCESS” to the client socket so it knows that it should prepare for data to receive. Then the server fetches all the student data from the database and saves all of it in one serializable object. This object then will be sent to the client socket using write object.

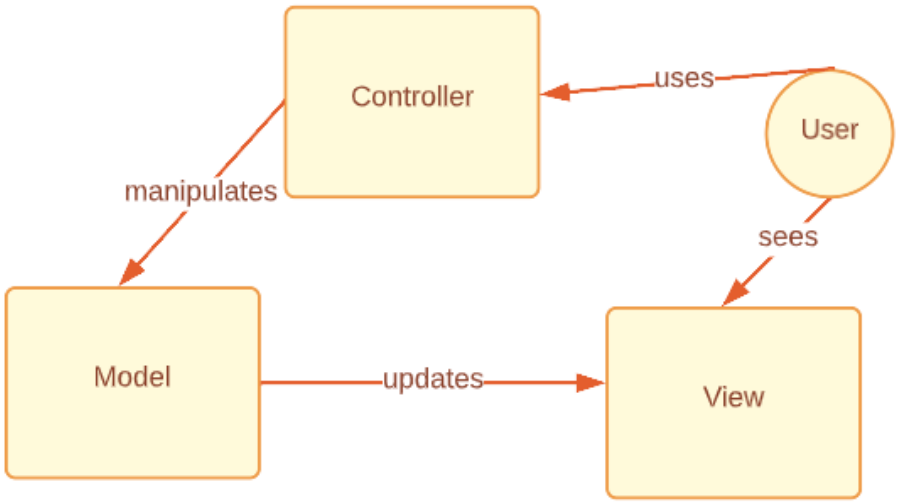


SERVLETS AND JSP

Servlets and JSPs are two technologies used in Java web development to create dynamic web pages. Servlets are Java classes that are used to handle HTTP requests and generate responses. JSPs, on the other hand, allow you to embed Java code directly into HTML pages, making it easier to generate dynamic content.

MVC Architecture is used to implement the webapplication in this stage. The MVC follows the separation of concerns principle. This architecture is divided into 3 main categories.

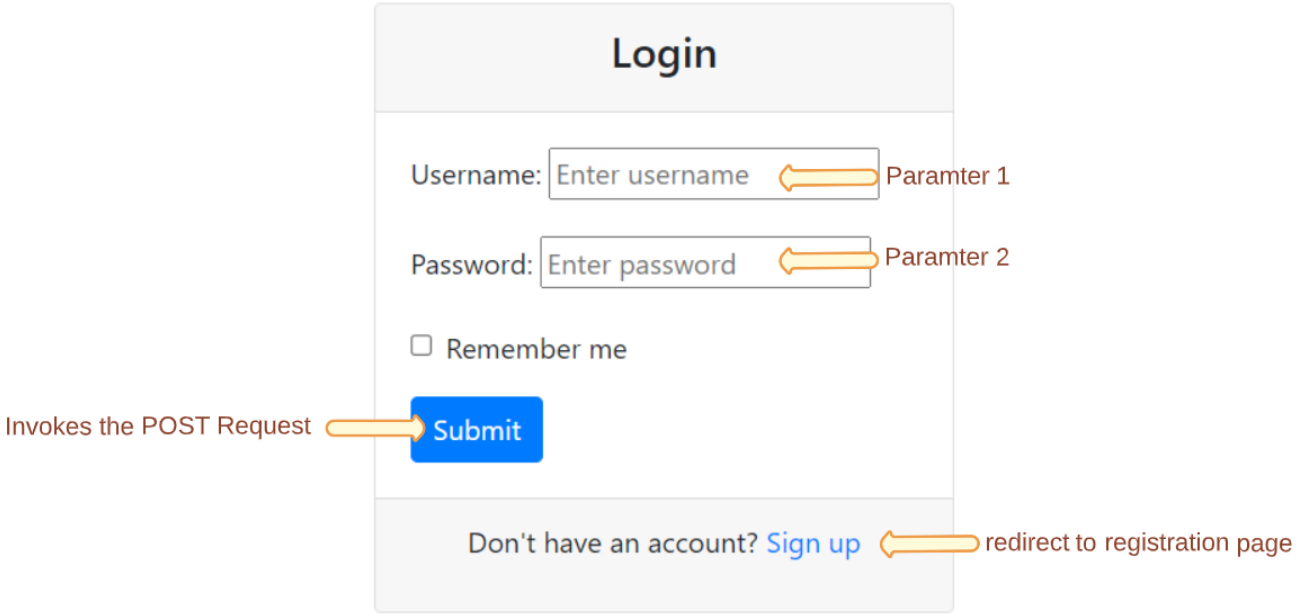
- Controller: a bridge between the view and model. When user enters data, view passes that data to the Controller. The Controller uses that data to update the database through the model.
- Model: this components dictates how I retrieve and store the data.
- View: a bunch of html and jsp pages to view content to the user.



Controllers

This web application is consisting from five controllers which constructs API gateways to and from the web application using GET and POST methods.

- **Login Controller :** when the user first makes a GET request to access the login page of the web application, this controller redirects the user to the login page. The POST method of this controller is responsible of receiving the login parameters which are username and password, then this controller will authenticate the request from the database, if correct it will create a session for this student and forwards him to next page using RequestDispatcher. If the credits are incorrect, he will be redirected again to the login screen. So no advancements will be made.

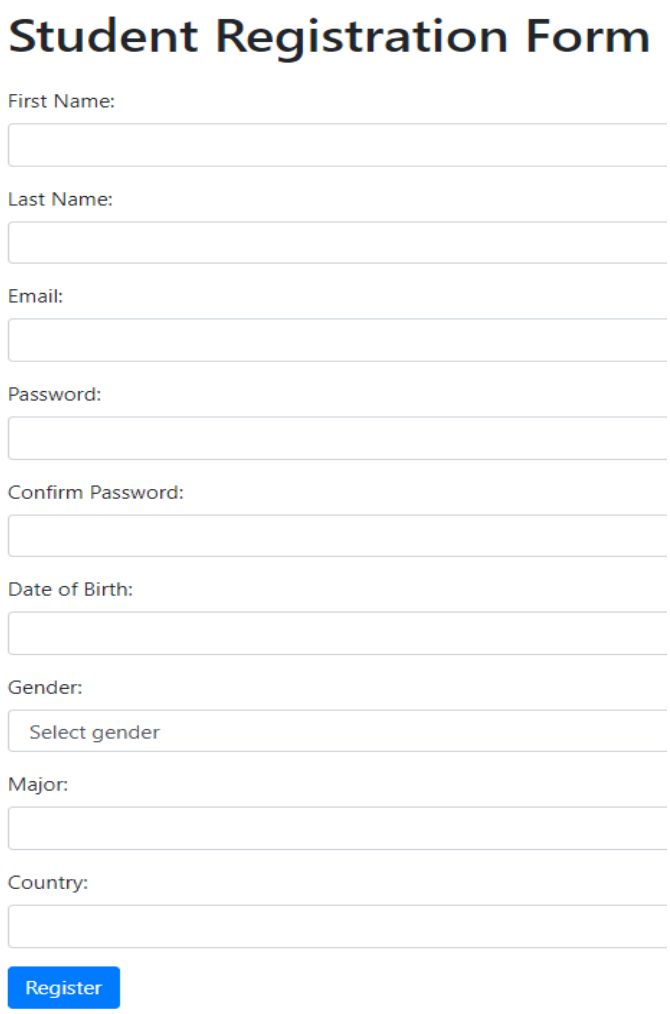


- *Logout Controller*

  a logout button will be displayed at the top of each controller, this button when clicked , it will call the logout controller, this controller will remove the user session, and redirects him to the login page. So if the user tries to access any other page without a login, his session will be empty, access wll be denied and the user is will be forwarded to login page again.

- *Registration Controller*

  when the user clicks on the sign up button or if he makes an api request to /register, this controller will be called. A registration Form will be displayed to the user to fill his data, when he's and submitted his data, this data will be saved into the database.



- *Dashboard Controller*

  firstly if you try to reach this page using get request, or if your session is empty, you will be forwarded to login page again. After confirming that the session is not empty, the student id is sent with the request as an attribute, so this controller will extract this id and fetch all classes for this student id from the data base. After that a jsp page is called to view the student classes.

My Classes

Class Name	Instructor	Time	Location	
Math	Inad	Sun 1pm-2pm	Room 202	View Details
English	khaled	thu 10am-11am	Room 122	View Details
Arabic	Sara	tue 8:30am - 10 am	Room 404	View Details

- *View Details Controller*

  after clicking on a specific view details button, a Jsp page is called to serve this request containing all course info including the student mark.

Course Information

Course Name: Math

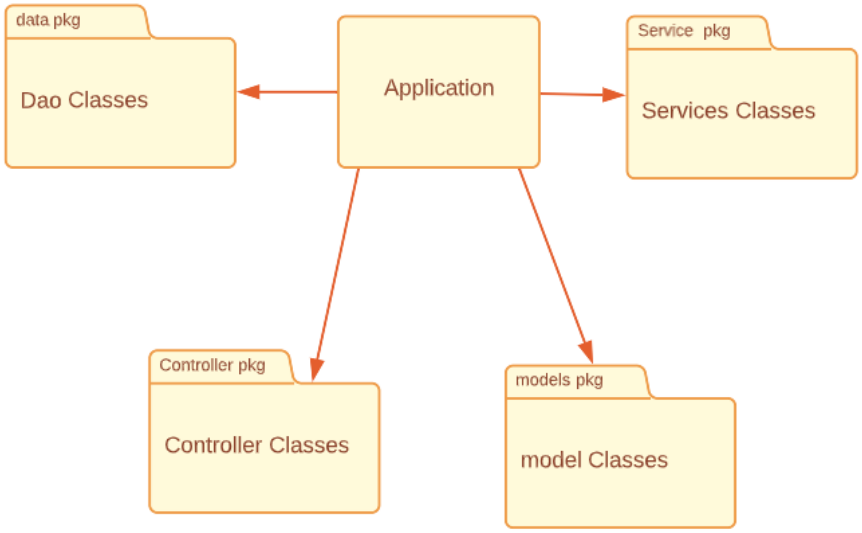
Statistic	Value
Your Grade	70.0
Highest Grade	70.0
Median Grade	70.0
Lowest Grade	20.0
Average Grade	53.333333333333336

SPRING BOOT APPLICATION

Spring Boot is a project that is built on the top of the Spring Framework. It provides an easier and faster way to set up, configure, and run both simple and web-based applications

The same system is now re-implemented using spring boot framework. In this way the application is easly and quickly developed. MVC pattern were used to build the system.

The application is consisting of multiple controllers, controllers are annotated with @Controller, each controller has a service, this service uses the dao classes to fetch data from database. Services are annotated with @Service to indicate that this class will be acting as a service in the controller. Session management is implemented to ensure no API requests are made when the user is logout out. Since browser caching is not always prevented.



URLs

- **/login**: when used with get request, it will redirect the user to the login choice page, this page is designed to add more types of people who can login to the web. But for now a only a logging in as a student is the only possible.
- **/Dashboard**: responsible of viewing the student information in addition to student classes.
- **/register:** will direct the client to a student registration form. And when submitted, it will add it to the database.
- **/course/courseId/studentId:** views the course statistics along with student mark in this course.

DATABASE DESIGN

Mysql database is used to store the student information. It consists of multiple tables, each table represents an entity, tables are linked together using foreign keys. The ER Diagram below illustrates the relationship between these tables.