A Project Report On

Placement Management System

By

Sahana M

B.E

Batch : 2021-6138

Enrollment Number:EBEON0921438529

Course: JAVA FULL STACK DEVELOPMENT

Center : Bangalore S-Prayas

Under the Guidance of

Anjana Sarode

Technical Trainer

EduBridge

(School of Coding)

SpringBoot:

Spring Boot is an open-source micro framework maintained by a company called Pivotal. It provides Java developers with a platform to get started with an auto configurable production-grade Spring application. With it, developers can get started quickly without losing time on preparing and configuring their Spring application.

Features of Springboot:

- **Autoconfiguration:** Developers can automatically configure their Spring application. However, Spring Boot is also capable of changing the configuration based on the dependencies you list. For example, when you list "MySQL" as a dependency, it will configure your Spring application with the "MySQL connector" included. And if you want to add a custom configuration, you can create a class that overrides the default configuration for your "MySQL connector".
- **Standalone:** There's no need to deploy your application to a web server. You simply enter the run command to start the application.
- **Opinionated:** On the offline page we find that Spring Boot decides for you which defaults to use for the configuration. Also, it decides which packages to install for the dependencies you require. For example, if you include the Spring Boot starter "pom" for "JPA", it will autoconfigure an in-memory database, a hibernate entity manager, and a simple data source. This is an example of an opinionated default configuration that you can override. While some developers might feel this is too opinionated, Spring Boot's opinionated setup helps developers to get started quickly on their projects.

Benifits of springboot:

- A lightweight framework.
- Helps with loose coupling dependencies and testability. The modular architecture allows you to pick the parts you need and isolate them.
- Has support for both XML and annotation configuration.
- Provides abstraction on ORM software to develop the ORM persistence logic.
- Compatible with many middleware services.
- Supports the JDBC framework, which improves productivity and reduces errors.

SPRING BOOT ARCHITECTURE

Spring Boot Architecture has four layers:

- **Presentation Layer**
- **Business Layer**
- **Persistence Layer**
- **Database Layer**

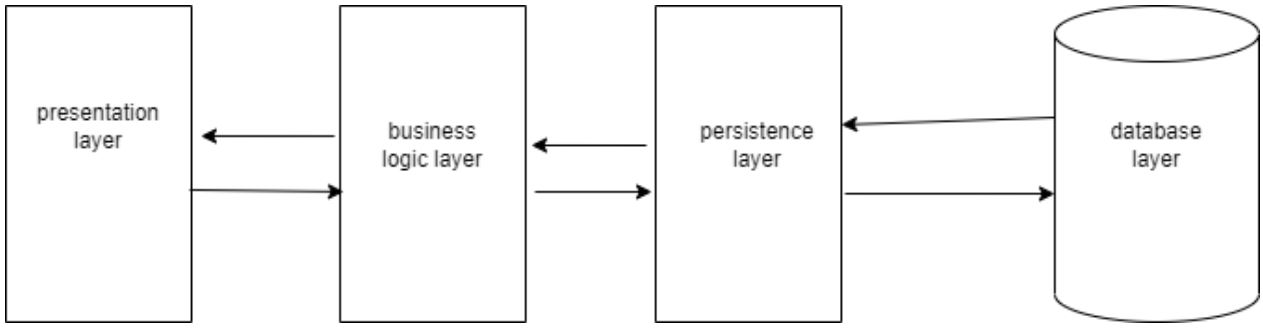


Figure 1: spring boot architecture

PRESENTATION LAYER

This layer is at the top of the architecture. This tier is responsible for:

- Performing authentication.
- Converting JSON data into an object (and vice versa).
- Handling HTTP requests.
- Transfering authentication to the business layer.

The presentation layer is the equivalent of the Controller class. The Controller class handles all the incoming REST API requests (GET, POST, PUT, DELETE, PATCH) from the Client.

BUSINESS LAYER

- The business Performing validation.
- Performing authorization.
- Handling the business logic and rules.

This layer is responsible for:

This layer is the equivalent to the Service class. It's where we handle the business logic. If you're wondering what do we mean by "business logic", I found an interesting discussion on StackExchange. In short, the business logic in software engineering is where we decide what the software needs to do. An example of this is validation. If you are ever requested to validate something, this needs to happen inside the Service class.

The Business layer communicates with both the Presentation layer and the Persistence Layer.

PERSISTENCE LAYER

This layer is responsible for:

- Containing storage logic.
- Fetching objects and translating them into database rows (and vice versa).

This layer is the equivalent of the Repository interface. We write database queries inside this interface.

The Persistence layer is the only layer that communicates with the Business layer and the Database layer.

DATABASE LAYER

This layer is responsible for:

- Performing database operations (mainly CRUD operations).
- This layer is simply the actual database that you decide to use to build your application.

Introduction :

Our Project explains about the Placement Management System.We can see different data related to Placement,College and Student.

I have developed this Application in **Java,Rest Api,Spring Boot and Mysql**.It's a web-based project so I have used **HTML,CSS,JAVASCRIPT AND BOOTSTRAP** also.

In Admin module , admin can view the Placement Details,College Details and Student Details . He can also add the new data to Placement details,College Details and Student details.

In Student module, Student can login and can view the details of Placement,College and Student.

Modules:

- Admin module

- Student module

Software Requirement:

Front end: Java,HTML, CSS, JavaScript, Bootstrap, Springboot,Thymeleaf.

Back end: MySQL workbench 8.0.23CE.

IDE: SpringToolSuite4

Browser: Best result on Google Chrome

System: Window 10

PROJECT WORKFLOW

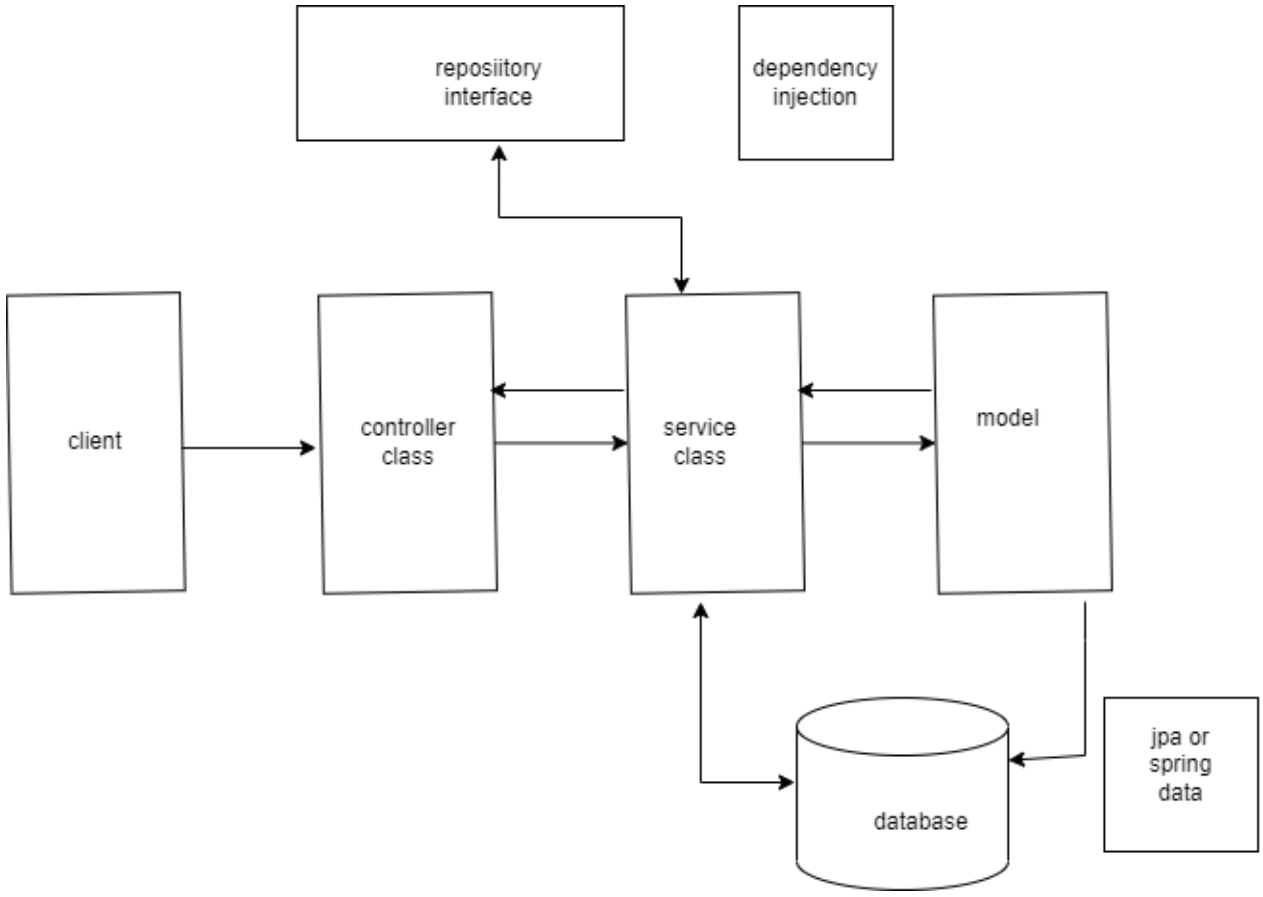


Figure 2: workflow of project

Spring Boot workflow work like follows:

- The Client makes an HTTP request.
- The Controller class receives the HTTP request.
- The Controller understands what type of request will process, and then it deals with it.
- If it is needed, it calls the service class.
- The Service Class is going to handle the business logic. It does this on the data from the database.
- If everything goes well, we return a JSP page.

Datebase script:

```
create database project;

use project;

create table Login(id bigint primary key auto_increment,username varchar(100)

,password varchar(100));


create table Student_Login(id bigint primary key

auto_increment,username varchar(100)

,password varchar(100));


select * from College;
```

```
create table College(id bigint primary key auto_increment,
collegename varchar(100),collegelocation varchar(100),collegeadmin varchar(100)
,collegeemail varchar(100));

create table Placement(id bigint primary key auto_increment,name varchar(100),
college varchar(100),date varchar(100)
,qualification varchar(100),year varchar(100));

create table Student(id bigint primary key auto_increment ,studentname varchar(100),course varchar(100),percentage int ,mailed
    varchar(100),college varchar(100),roll Long
,qualification varchar(100),hallticketno int);
INSERT INTO Login VALUES(1,"SAHANA","SAHA12");
```

select * from login;

select * from Student;

select * from Placement;

select * from College;

select * from student_login;

Output screenshots:

LOGIN PAGE

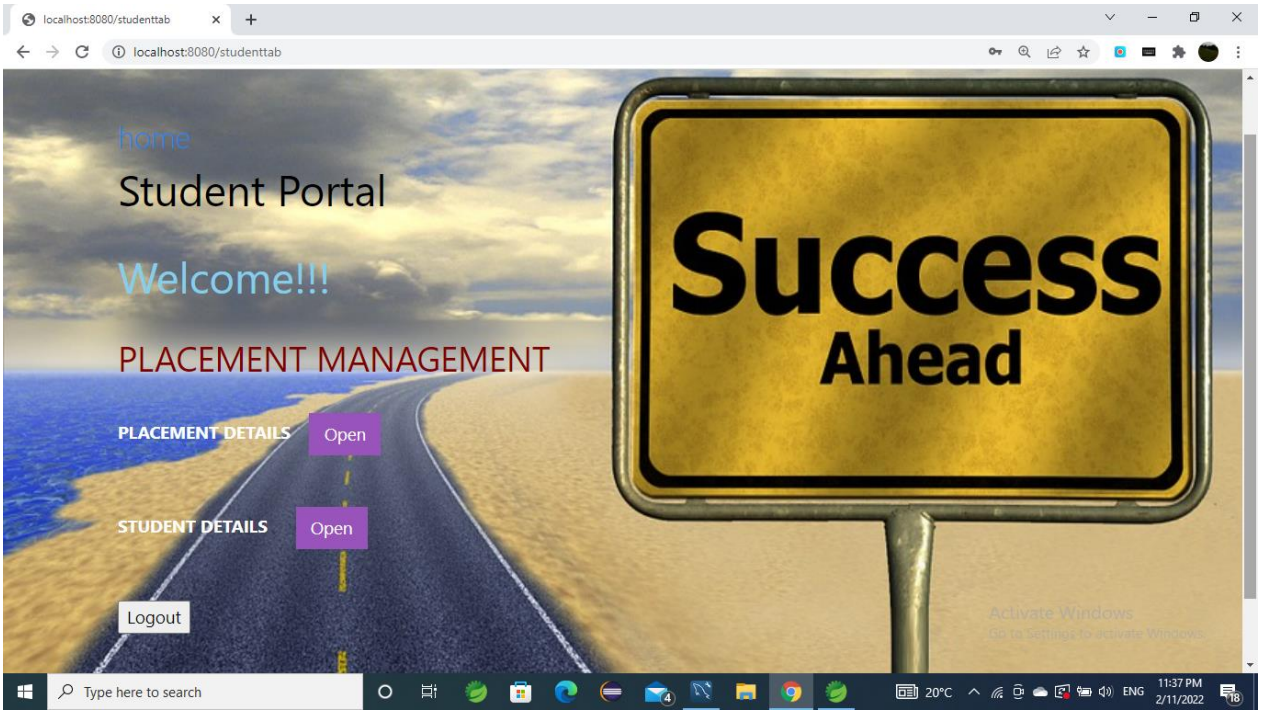


Figure 3:login_page

STUDENT_LOGIN PAGE

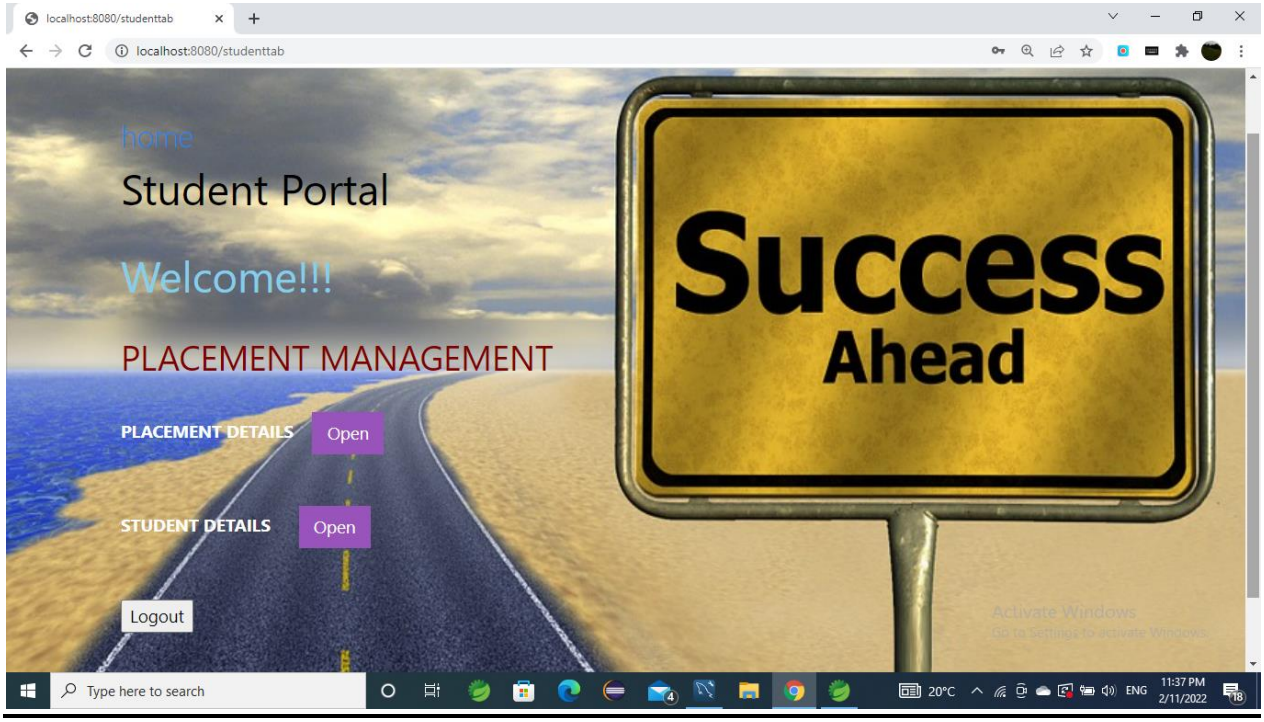


Figure 4:Student_login_page

ADMIN INDEX PAGE

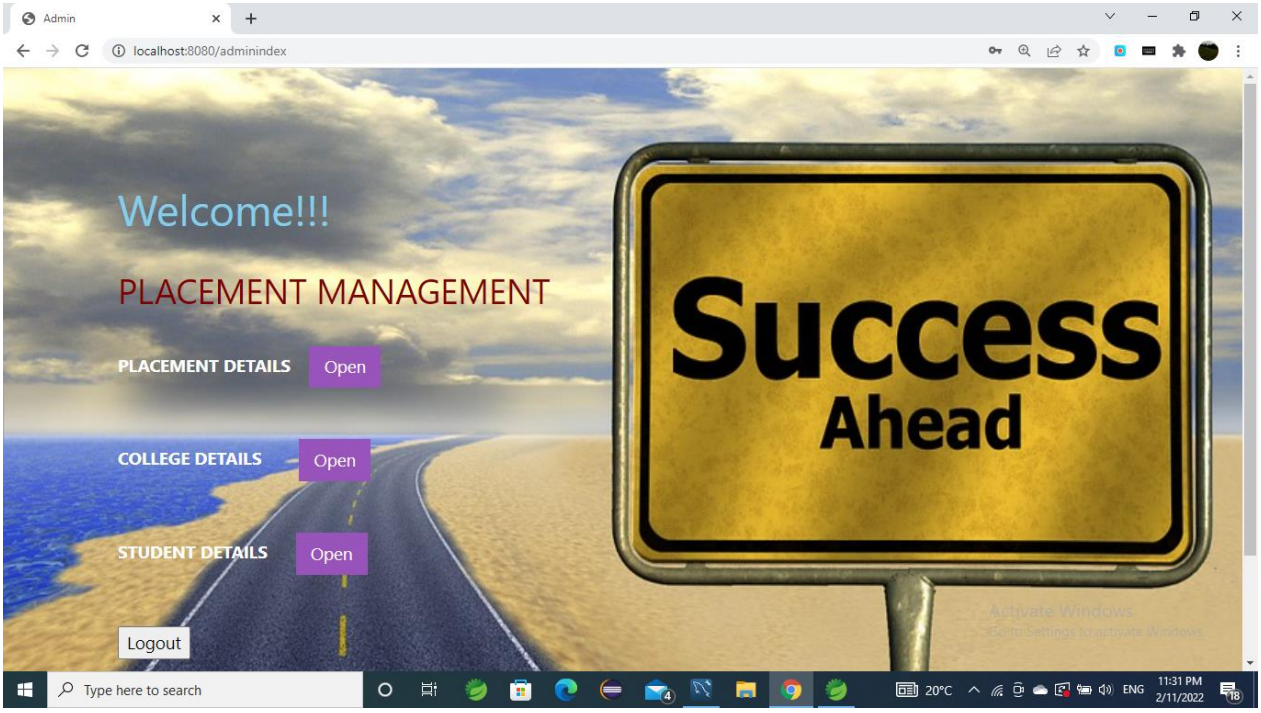


Figure 5:Admin_index_page

Student data entry

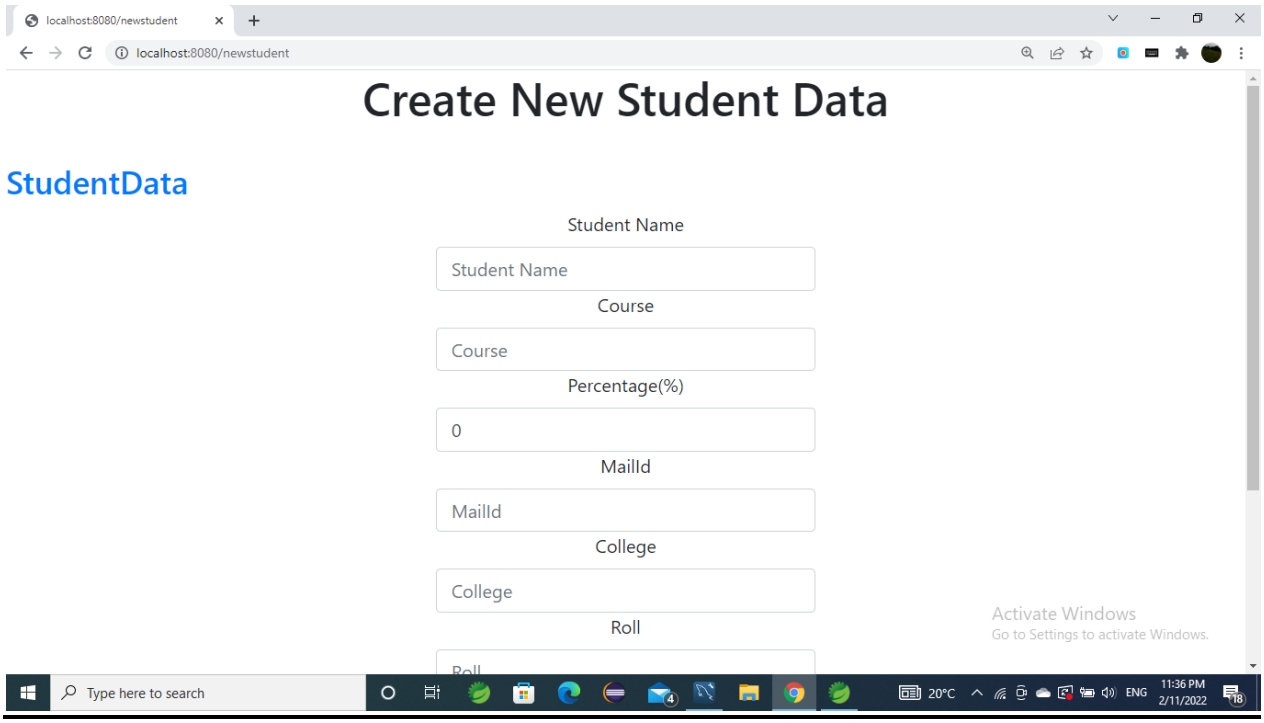


Figure 6:Student_data_form

CREATING NEW COLLEGE DATA FOR PLACEMENT

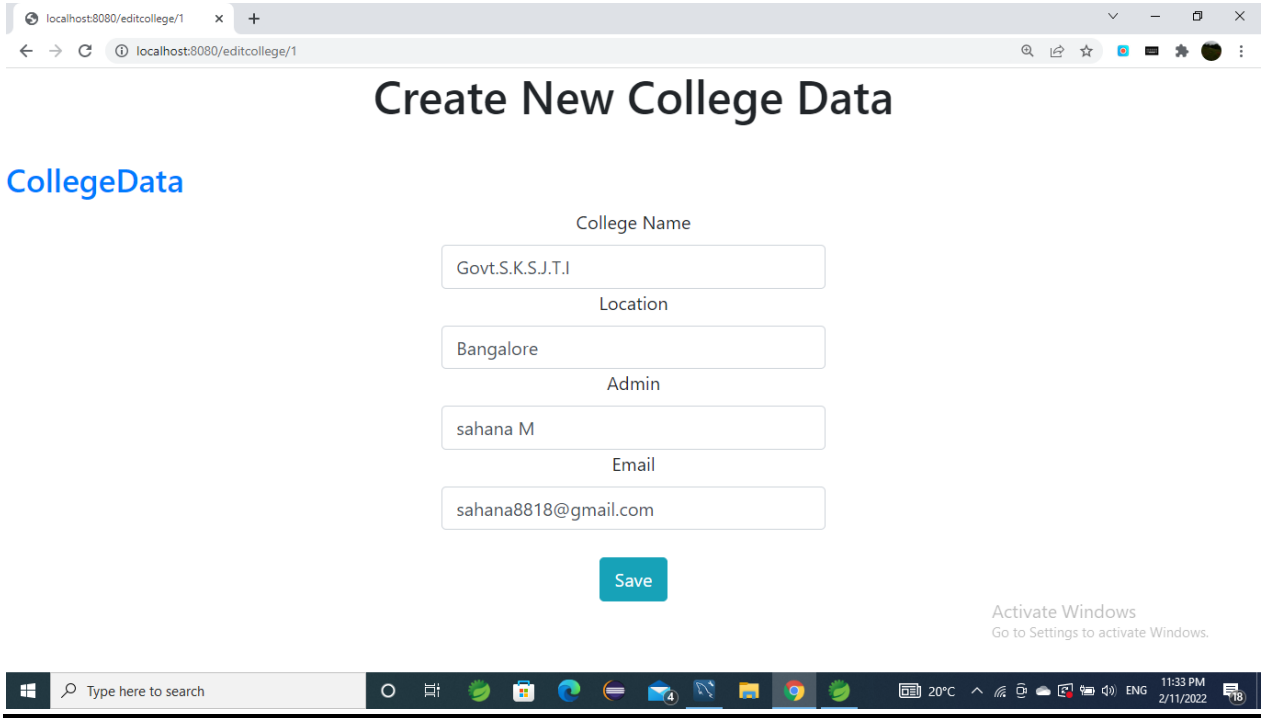


Figure 7:ADDIND NEW COLLEGE TO PLACEMENT

COLLEGE INDEX PAGE

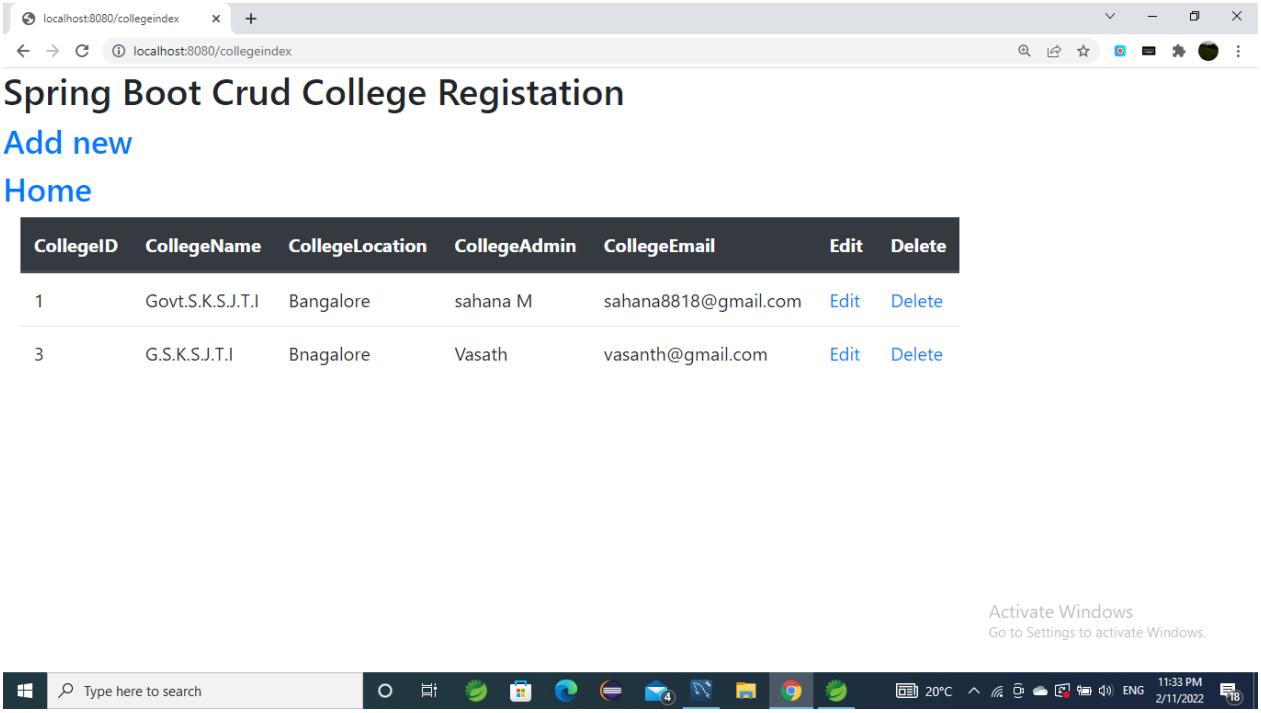


Figure 8:college_index

STUDENT INDEX PAGE

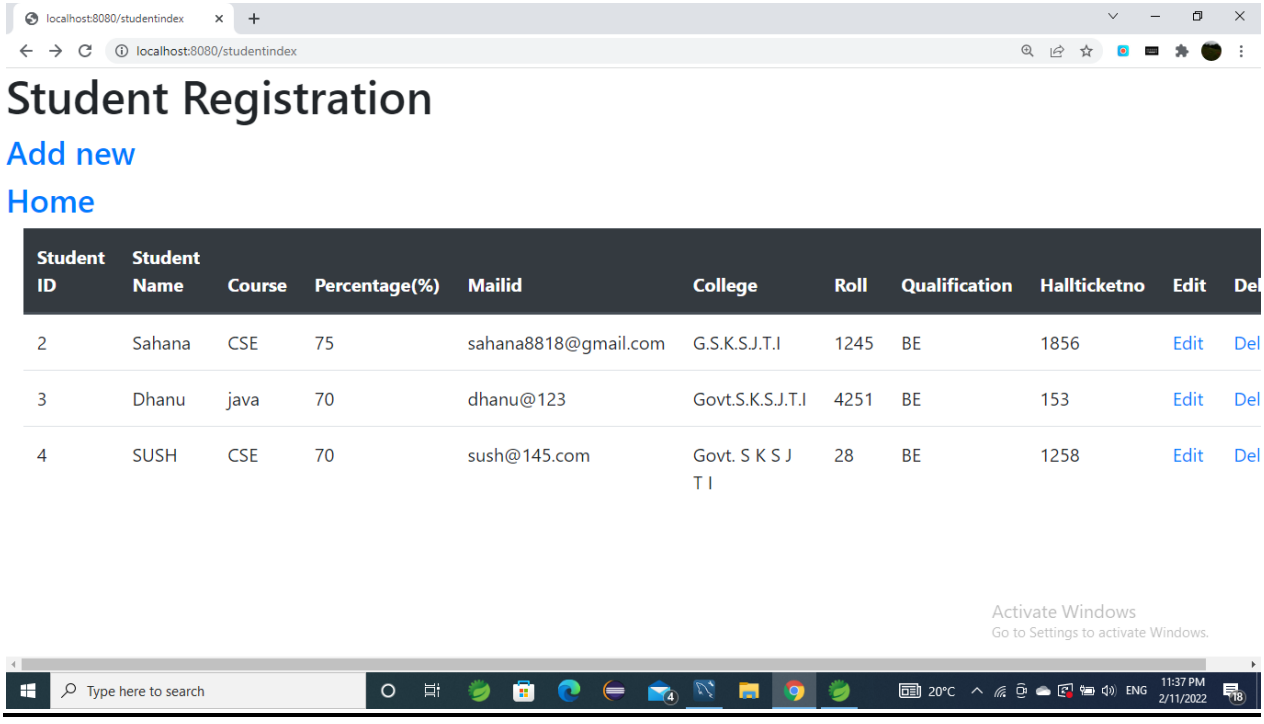


Figure 9:Student_Index_Page

NEW STUDENT Login

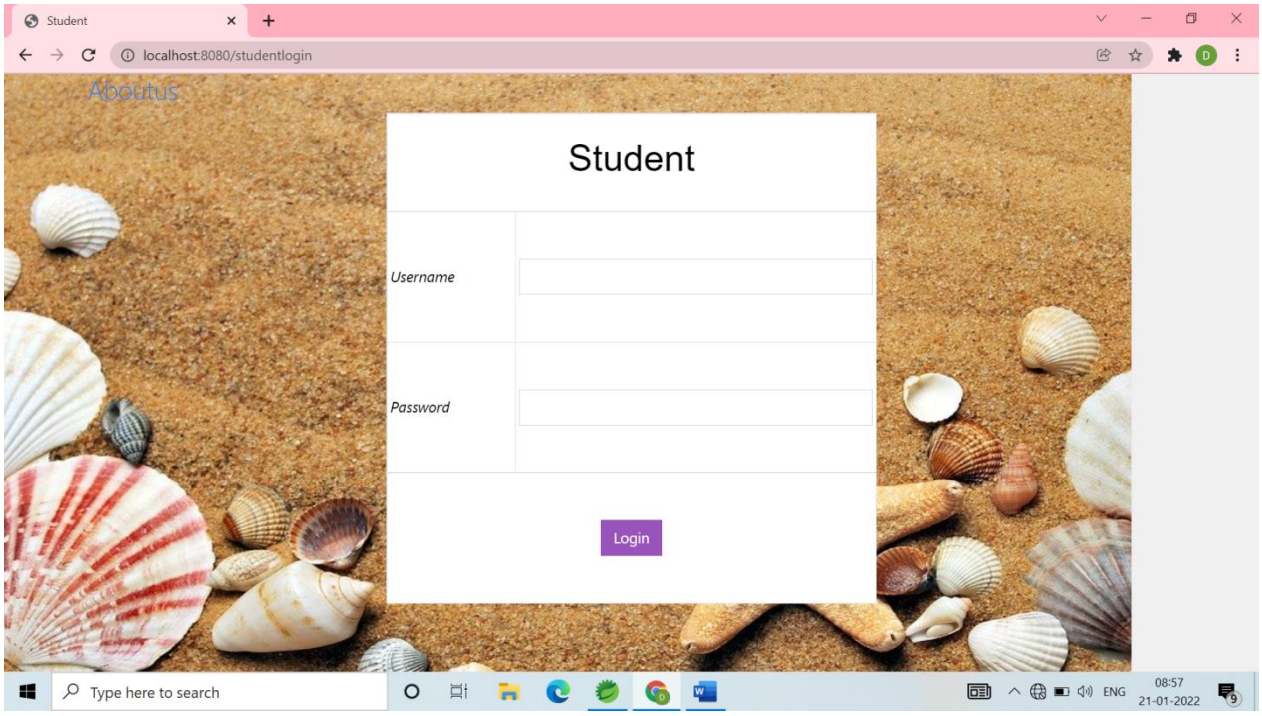
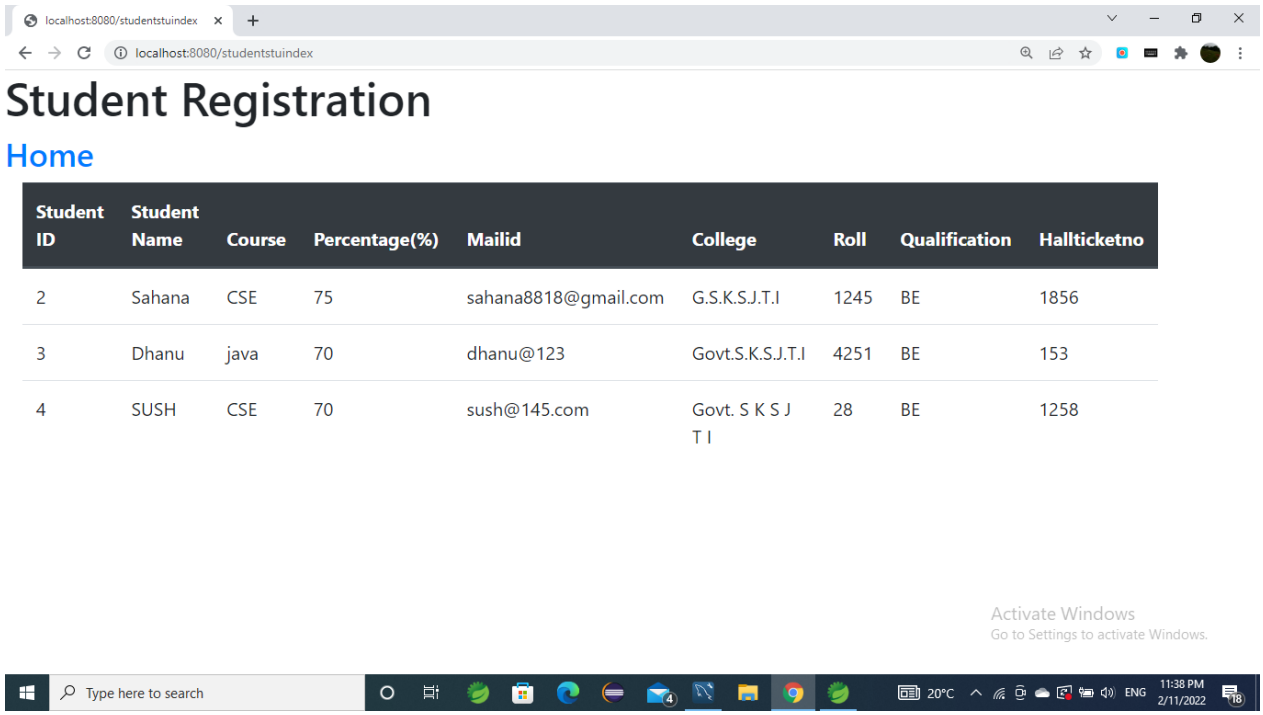


Figure 10:Student_login_Page

NEW STUDENT index for student Login



Student ID	Student Name	Course	Percentage(%)	Mailid	College	Roll	Qualification	Halltickетno
2	Sahana	CSE	75	sahana8818@gmail.com	G.S.K.S.J.T.I	1245	BE	1856
3	Dhanu	java	70	dhanu@123	Govt.S.K.S.J.T.I	4251	BE	153
4	SUSH	CSE	70	sush@145.com	Govt. S K S J T I	28	BE	1258

Figure 11:Student _details_for_student

About_US_page

Figure 12:About__US