Swarnandhra Institute of Engineering and Engineering

**Ctrl Freaks**

PLACEMENT MANAGEMENT SYSTEM

* Ch.TANUSHA
* V.Vijaya
* B.Nokesh
* Y.Surya Teja
* N.Dileep

INTRODUCTION:

* The project aims to automate the process of placement management system and make the process of recruitment easier.
* It provides a platform where it allows the students to view the companies that are currently recruiting and also provides information on the companies and the packages they offer so that the students may view and assess their opportunities.
* It helps the company to find the eligible candidates according to their criteria.
* The project is aimed at developing an application for the “PLACEMENT MANAGEMENT SYSTEM” for the college. This application can be accessed and effectively used throughout the organization with proper login credentials.
* In this application the placement officers in the college is to manage the student information with regard to placement.
* The key features of this project is that it is one time registration enabled. Our project provides the facility of maintaining the details of the students. It reduces the manual work and consumes less paper work to reduce the time.
* There are mainly 2 Modules in this system:

1. Student Module.

2. Admin Module

* **Admin Module** :The Placement officer is the administrator of the system. Admin plays a very crucial role in the system. Admin can log in through a username and password. He/she can maintain manage the placement activities via the system. Admin can add departments, create new batches, add /delete drives. Students can directly join in the placement drive if interested. Admin can manage the training programs conducted in the college. Admin is also able to analyze the placement activities of each student.

**Student Module:**  Every student is given a default username and password, using this he/she can enter the system. Students can fill the necessary details like ,EmailId,10 th and 12th CPGA and Passing YEARetc. if interested, students can register for the upcoming drives. The student is also able to attend the online aptitude test being conducted in the system. Based on this aptitude test and other criteria the placement probabilities of a particular student is been predicted.

**SOFTWARE REQUIREMENTS:**

OPERATING SYSTEM : WINDOWS 11

BROWSER : CHROME

FRONT-END : ReactJs

DATABASE : MySQL

BACK-END : Spring Boot

**Initializing the Spring Boot Application :**

1.Go to the <https://start.spring.io/> this website.

2.Then choose lang, and Add Dependencies as shown below;

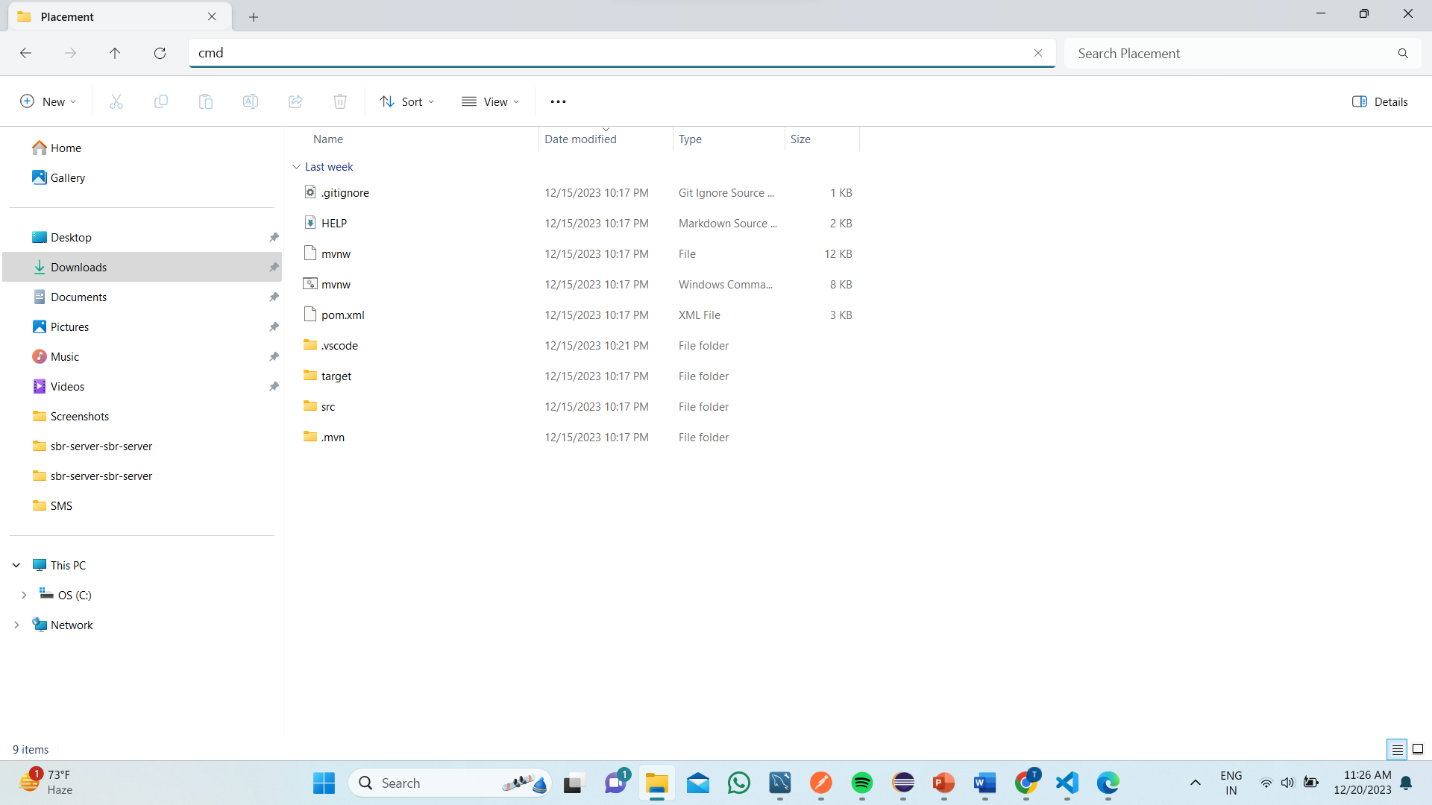


3. After Generate this ,and open the downloaded zip file and right click on it,,,,,

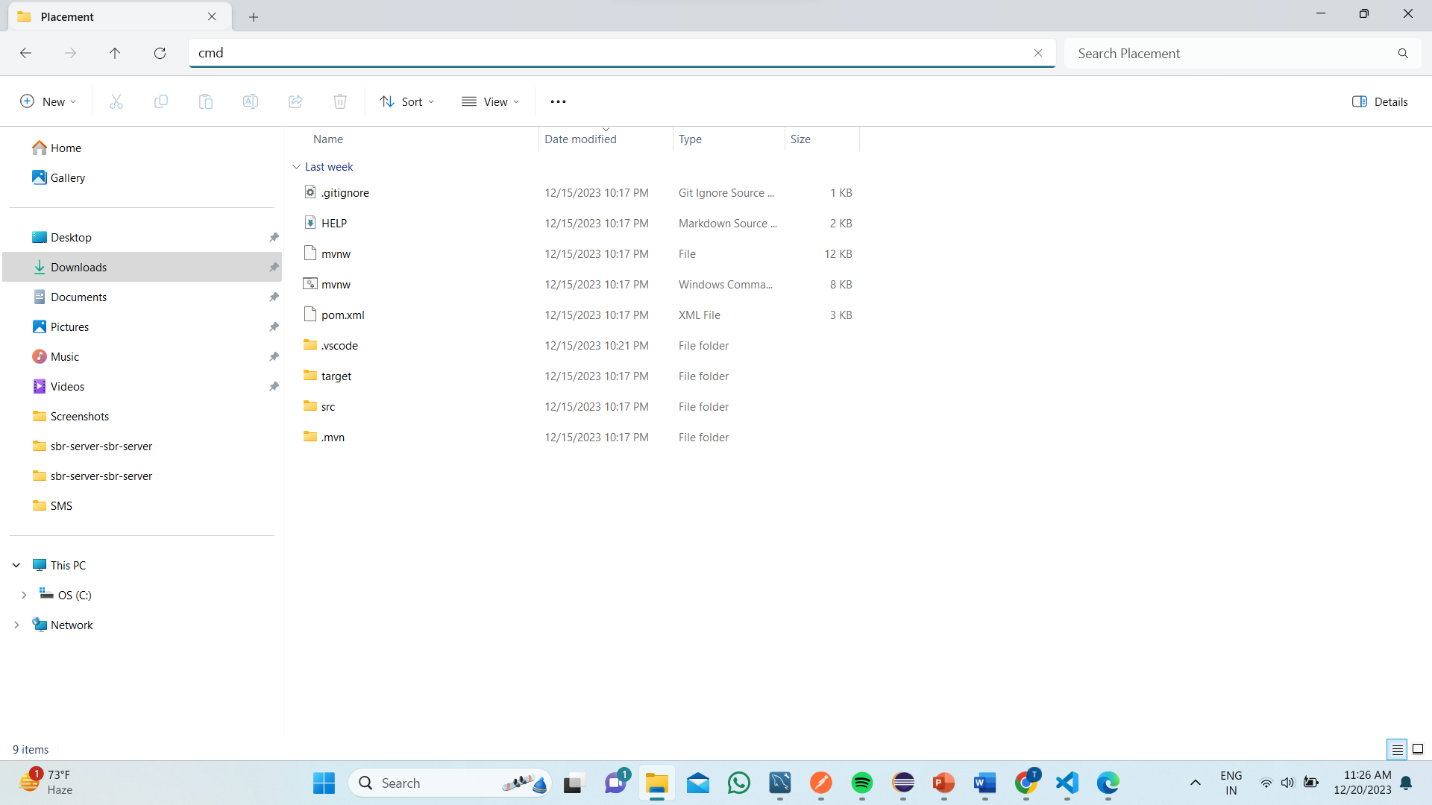
Then tap Extract all and click “**EXTRACT**”.

4. And go to search bar, then select the COPY ADD and REMOVE IT

And type “*CMD”.*

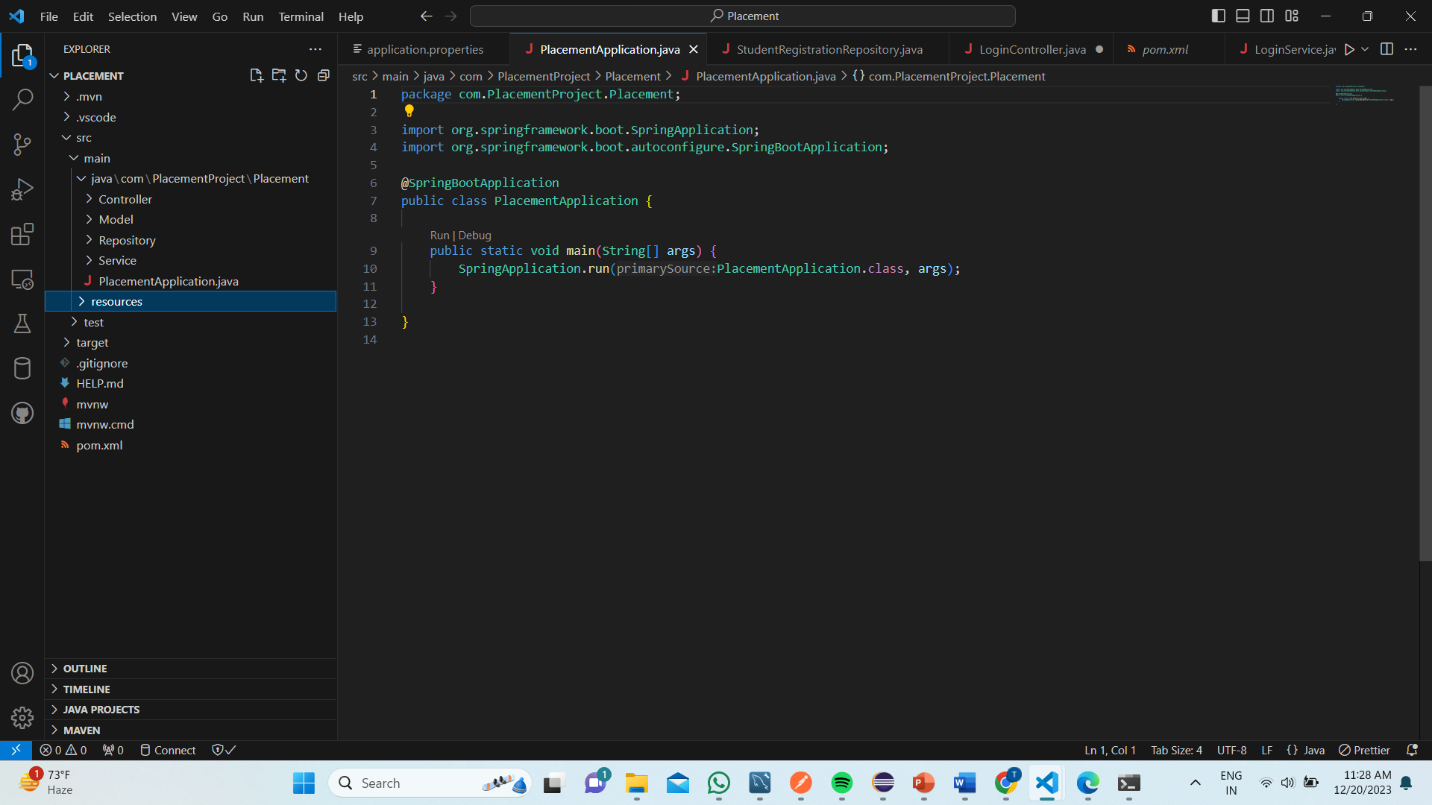
**

5. and COMMAND PROMPT app will open, then type **<<<** code .



6. It directly goes to **VSCODE APP.**

7.And the selected folder will directly Opened… and we create the Folders and named as Model, Controller, Repository ,Service Respectively.



7. After this ,,, In admin module I created one sub-Module named as **COMPANY**

After that I created a Model ,Controller , Service and Repository folders inside the java/com/PlacementProject /Placement and inside that Model folder I created CompanyDetails.java and inside that Controller folder I created CompanyController.java file and Inside Service folder I created CompanyService.java and inside the CompanyRepository files respectively..

**Program for CompanyDetails.java:**

package com.PlacementProject.Placement.Model;

import jakarta.persistence.Column;

import jakarta.persistence.Entity;

import jakarta.persistence.Id;

import jakarta.persistence.Table;

import jakarta.validation.constraints.NotBlank;

import lombok.AllArgsConstructor;

import lombok.Builder;

import lombok.Data;

import lombok.NoArgsConstructor;

@Entity

@Data

@AllArgsConstructor

@NoArgsConstructor

@Builder

@Table(name="company")

public class CompanyDetails {

    @Id

    private Long id;

   @NotBlank(message = "PLEASE ENTER THE COMPANY NAME")

    private String companyName;

    private String location;

    @Column(length = 1000)

    private String description;

    private String salary;

    private String date;

    private String studentsRequired;

}

**Model class:**

In Spring MVC, the model works a container that contains the data of the application. Here, a data can be in any form such as objects, strings, information from the database, etc.

**Program for CompanyRepository.java :**

package com.PlacementProject.Placement.Repository;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.stereotype.Repository;

import com.PlacementProject.Placement.Model.CompanyDetails;

@Repository

public interface CompanyRepository extends JpaRepository<CompanyDetails,Long>{

}

**Program for CompanyRepository.java :**

package com.PlacementProject.Placement.Repository;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.stereotype.Repository;

import com.PlacementProject.Placement.Model.CompanyDetails;

@Repository

public interface CompanyRepository extends JpaRepository<CompanyDetails,Long>{

}

**Program for CompanyService.java :**

package com.PlacementProject.Placement.Service;

import java.util.List;

import java.util.Optional;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import com.PlacementProject.Placement.Model.CompanyDetails;

import com.PlacementProject.Placement.Repository.CompanyRepository;

@Service

public class CompanyService {

    @Autowired

    private  CompanyRepository companyRepository;

    public String addCompanyDetails( CompanyDetails companyDetails) {

        companyRepository.save(companyDetails);

        return "DETAILS\_ARE\_INSERTED";

    }

    public CompanyDetails getCompanyDetailsById(Long id) {

        return companyRepository.findById(id).get();

    }

    public List<CompanyDetails> getAllCompanyDetails() {

        List<CompanyDetails> details=companyRepository.findAll();

         return details;

    }

    public String updateCompanyDetails(Long id, CompanyDetails companyDetails) {

        Optional<CompanyDetails> companyDetails2=companyRepository.findById(id);

        CompanyDetails comw=companyDetails2.get();

 comw.setCompanyName(companyDetails.getCompanyName());

          comw.setLocation(companyDetails.getLocation());

           comw.setDescription(companyDetails.getDescription());

           comw.setSalary(companyDetails.getSalary());

           comw.setStudentsRequired(companyDetails.getStudentsRequired());

        companyRepository.save(companyDetails);

        return "Updated";

    }

    public String deletecompanyDetails(Long id) {

        companyRepository.deleteById(id);

        return "deleted";

    }

}

**Program for CompanyController.java :**

package com.PlacementProject.Placement.Controller;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.DeleteMapping;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.PutMapping;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RestController;

import com.PlacementProject.Placement.Model.CompanyDetails;

import com.PlacementProject.Placement.Service.CompanyService;

import jakarta.validation.Valid;

@RestController

@RequestMapping("/Company")

public class CompanyController {

    @Autowired

    private CompanyService companyService;

    @PostMapping("/addCompany")

    public String addComapanyDetails(@Valid @RequestBody CompanyDetails companyDetails)

    {

        return companyService.addCompanyDetails(companyDetails);

    }

     @GetMapping("/getDetails/{CompanyId}")

    public CompanyDetails getCompanyDetailsById(@PathVariable("CompanyId") Long id)

    {

        return companyService.getCompanyDetailsById(id);

    }

    @GetMapping("/getAllCompanyDetails")

    public List<CompanyDetails> getAllCompanyDetails()

    {

        return companyService.getAllCompanyDetails();

    }

    @PutMapping("/updateCompanyDetails/{id}")

    public String updateCompanyDetails( @PathVariable("id")Long id ,@RequestBody CompanyDetails companyDetails)

    {

        return companyService.updateCompanyDetails(id,companyDetails);

    }

    @DeleteMapping("/deleteComoanyDetails/{id}")

    public String deletecompanyDetails(@PathVariable("id")Long id)

    {

        return companyService.deletecompanyDetails(id);

    }

}

**Controller class:**

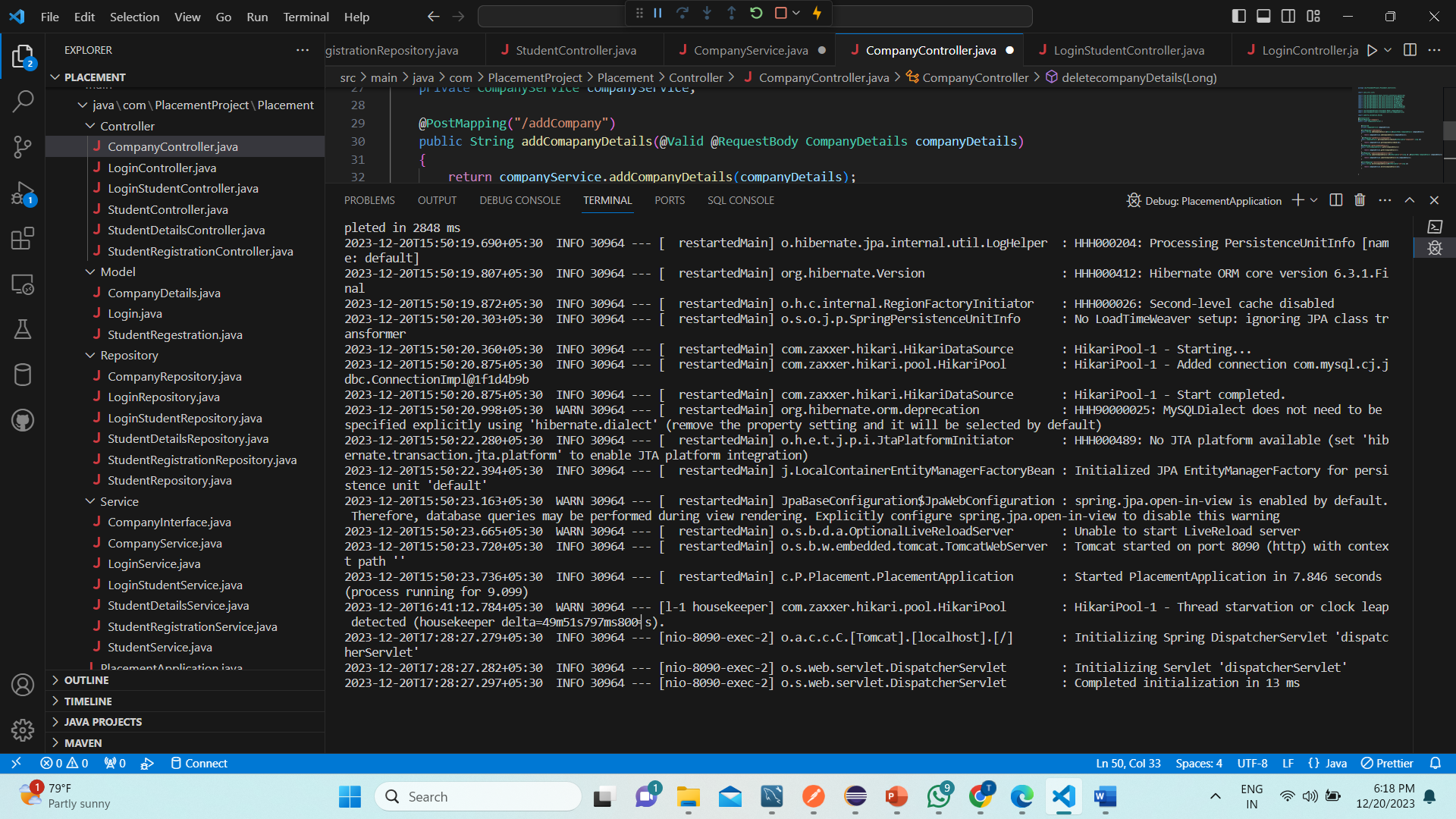
In Spring Boot, the controller class is responsible for processing incoming REST API requests, preparing a model, and returning the view to be rendered as a response. The controller classes in Spring are annotated either by the @Controller or the @RestController annotation.

**@RestController:**

* The @RestController annotation is a specialized version of the @Controller annotation. It is used to indicate that the class defines a RESTful API endpoint.
* When you annotate a class with @RestController, it implies that every method inside the class is treated as a controller method and returns the response in a format suitable for RESTful services (typically JSON).

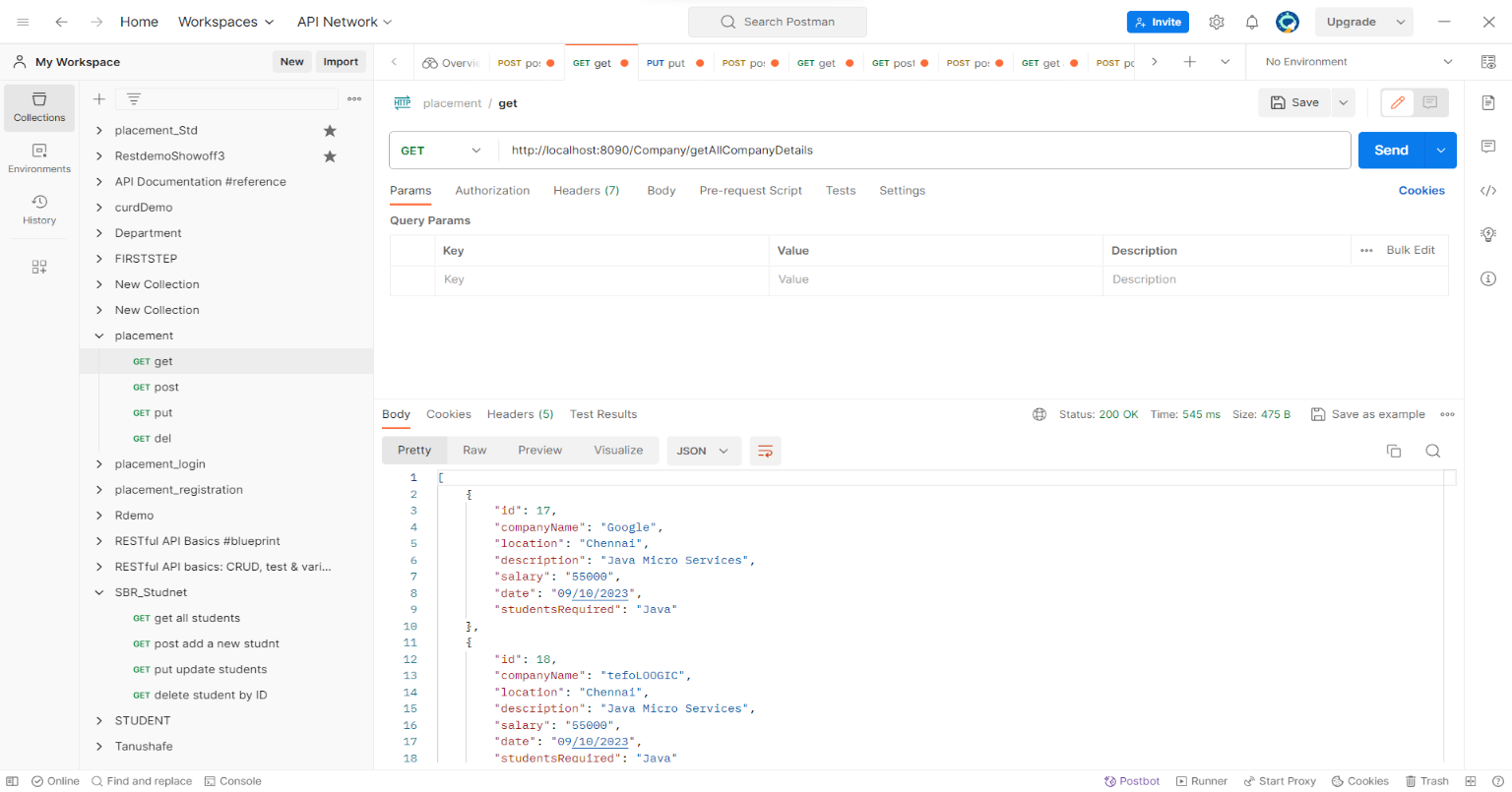
**@RequestMapping:**

* The @RequestMapping annotation is used to map web requests to specific methods in a controller class. It can be applied at the class level and/or method level.
* It allows you to define the base URI for all the methods in the class and then further refine the URI for each method.
* It can specify the HTTP method (GET, POST, PUT, DELETE) and other request parameters
* Then after run the application. As default the application will run on port 8080 in Apache Tomcat server.But, here I can change my Port in Application properties by server.port:8090

****

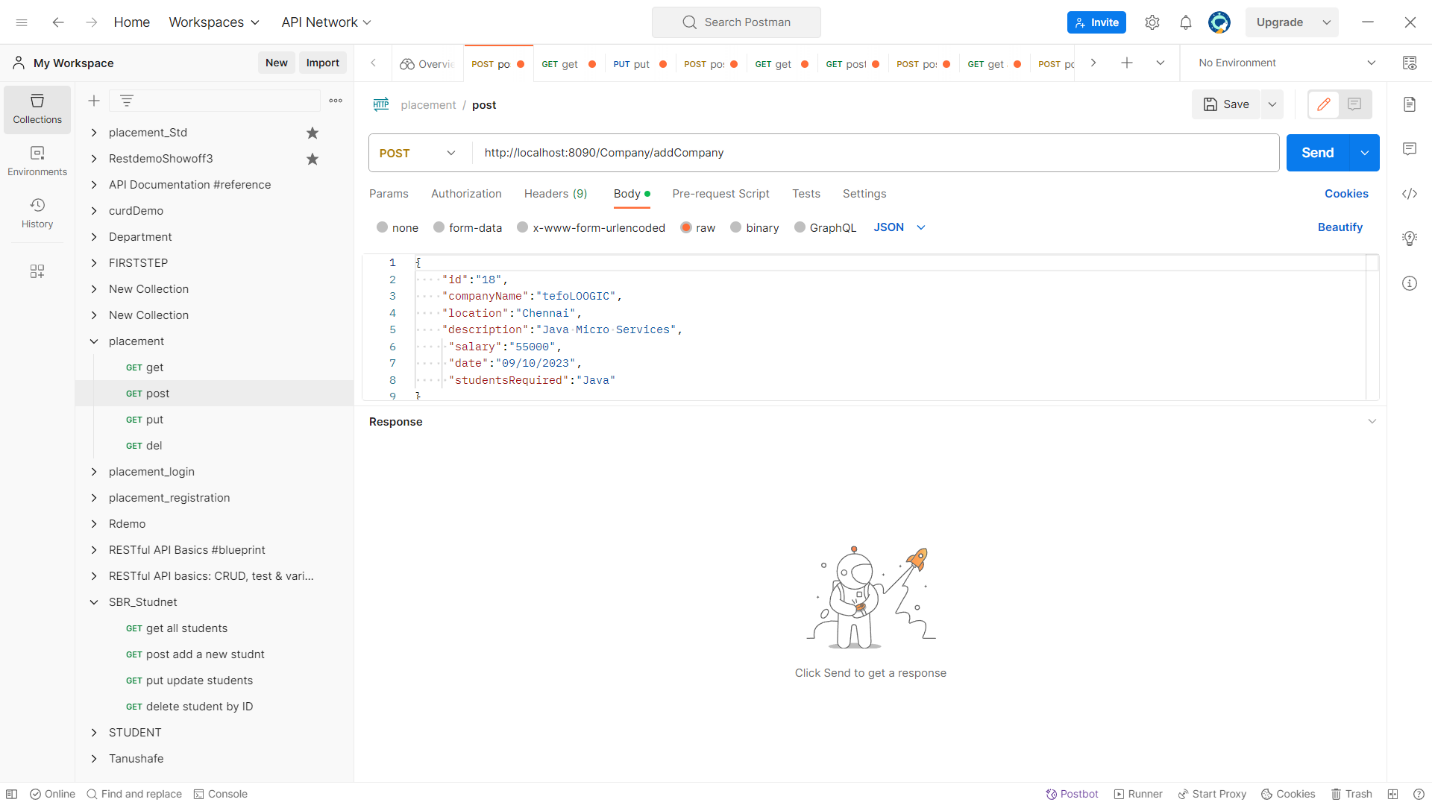
**@GetMapping:**

* This annotation maps the HTTP GET request to the specified URI pattern, in this case, "/Company //getDetails/{CompanyId}



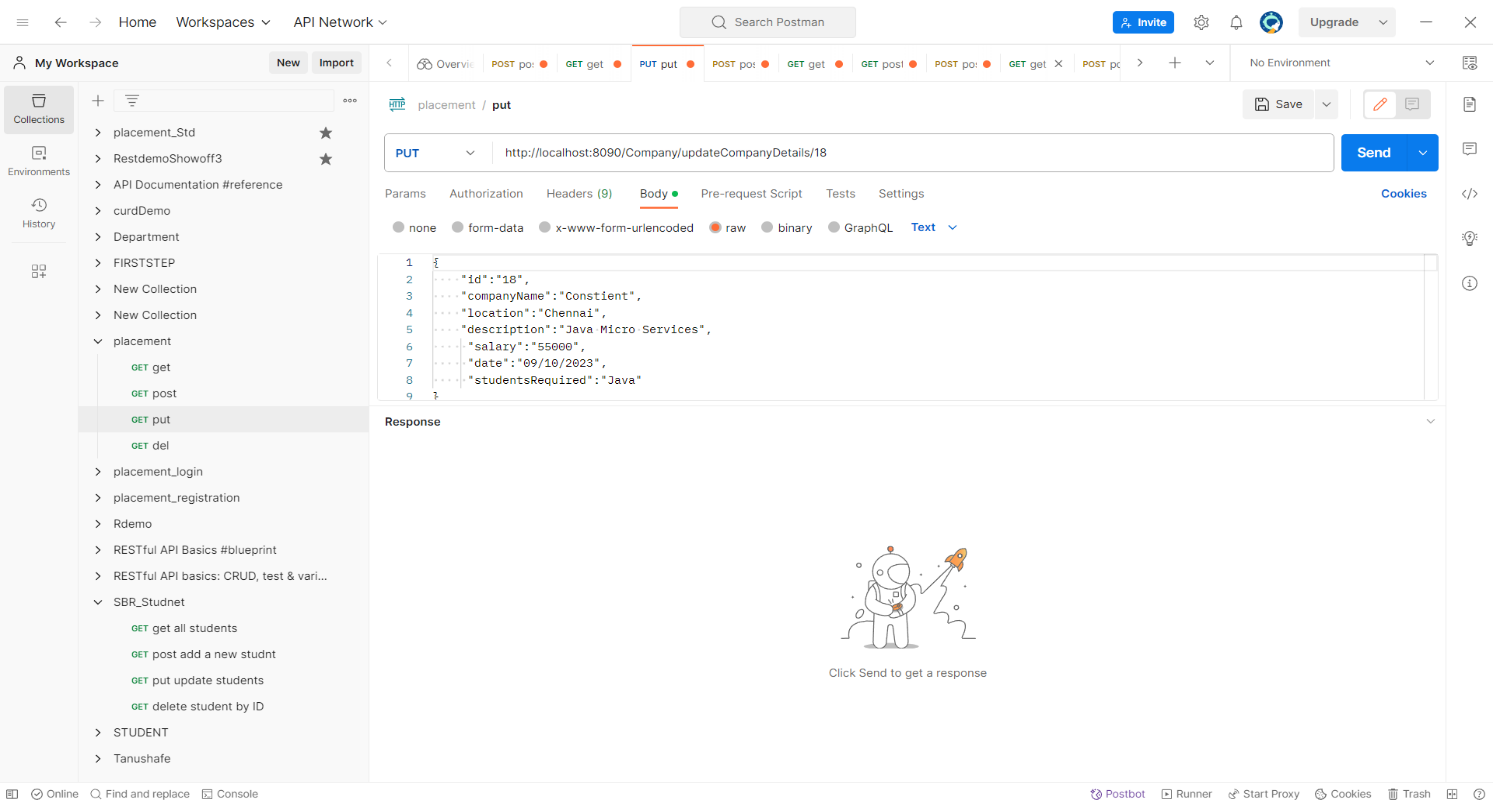
**@PostMapping:**

* This annotation maps the HTTP POST request to the *"http://localhost:8090/Company/addCompany"* URI.

****

**@PutMapping:**

* This annotation maps the HTTP PUT request to the "*http://localhost:8090/Company/updateCompanyDetails/18*" URI. It is used for updating *CompanyDetails* details by consuming the request body, which is expected to contain JSON representing a modified Company\_Details object.



* This annotation maps the HTTP DELETE request to the “” URI. It is used to delete the Company\_details associated with the provided path Id.
* After this ,,, In admin module I created one sub-Module named as **Student\_Registration…..,,,,,**After that I created a Model ,Controller , Service and Repository folders inside the java/com/PlacementProject /Placement and inside that Model folder I created StudentRegistration.java and inside that Controller folder I created StudentRegistrationController.java file and Inside Service folder I created StudentRegistrationService.java and inside the StudentRegistrationRepository .java files respectively..

**Program for CompanyDetails.java:**

package com.PlacementProject.Placement.Model;

import jakarta.persistence.Entity;

import jakarta.persistence.GeneratedValue;

import jakarta.persistence.GenerationType;

import jakarta.persistence.Id;

import jakarta.persistence.Table;

import jakarta.validation.constraints.Email;

import jakarta.validation.constraints.NotBlank;

import lombok.AllArgsConstructor;

import lombok.Builder;

import lombok.Data;

import lombok.NoArgsConstructor;

@Entity

@Data

@AllArgsConstructor

@NoArgsConstructor

@Builder

@Table(name = "StudentRegestration")

public class StudentRegestration {

    @Id

    @GeneratedValue(strategy = GenerationType.IDENTITY)

    private Long id;

    private String firstName;

    private String lastName;

     @NotBlank(message = "Regetration Number is required")

    private String regestrationNumber;

    @Email(message = "Email should be valid")

    private String emailId;

    private String phoneNumber;

    private String college;

    private String branch;

    private String tenthpercentage;

    private String interpercentage;

    private String btechpercentage;

   }

**Program for StudentRegistrationController.java:**

package com.PlacementProject.Placement.Controller;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RestController;

import com.PlacementProject.Placement.Model.StudentRegestration;

import com.PlacementProject.Placement.Service.StudentRegistrationService;

import jakarta.validation.Valid;

@RestController

@RequestMapping("/Student")

public class StudentRegistrationController {

  @Autowired

    private StudentRegistrationService studentRegistrationService;

   @PostMapping("/StudentRegistrations")

    public String studentRegistration(@Valid @RequestBody StudentRegestration studentRegestration)

   {

      studentRegistrationService.studentRegistration(studentRegestration);

      return "Registered Successfull";

   }

}

**Program for StudentRegistrationRepository.java:**

package com.PlacementProject.Placement.Repository;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.stereotype.Repository;

import com.PlacementProject.Placement.Model.StudentRegestration;

@Repository

public interface StudentRegistrationRepository extends JpaRepository<StudentRegestration,Long>{

}

**Program for StudentRegistrationService.java:**

package com.PlacementProject.Placement.Service;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import com.PlacementProject.Placement.Model.StudentRegestration;

import com.PlacementProject.Placement.Repository.StudentRegistrationRepository;

@Service

public class StudentRegistrationService {

   @Autowired

    private StudentRegistrationRepository studentRegistrationRepository;

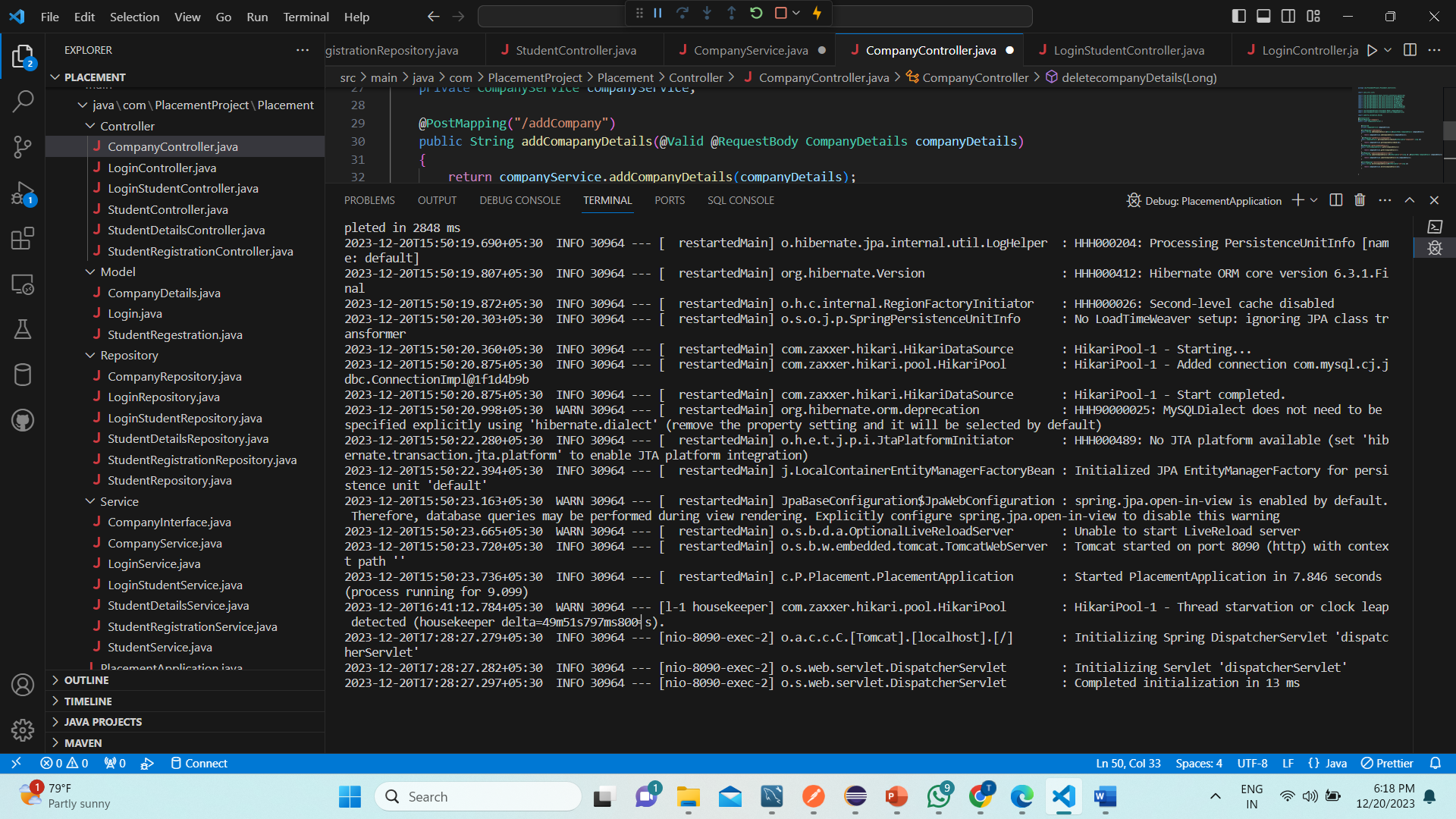
    public String studentRegistration(StudentRegestration studentRegestration) {

        studentRegistrationRepository.save(studentRegestration);

        return null;

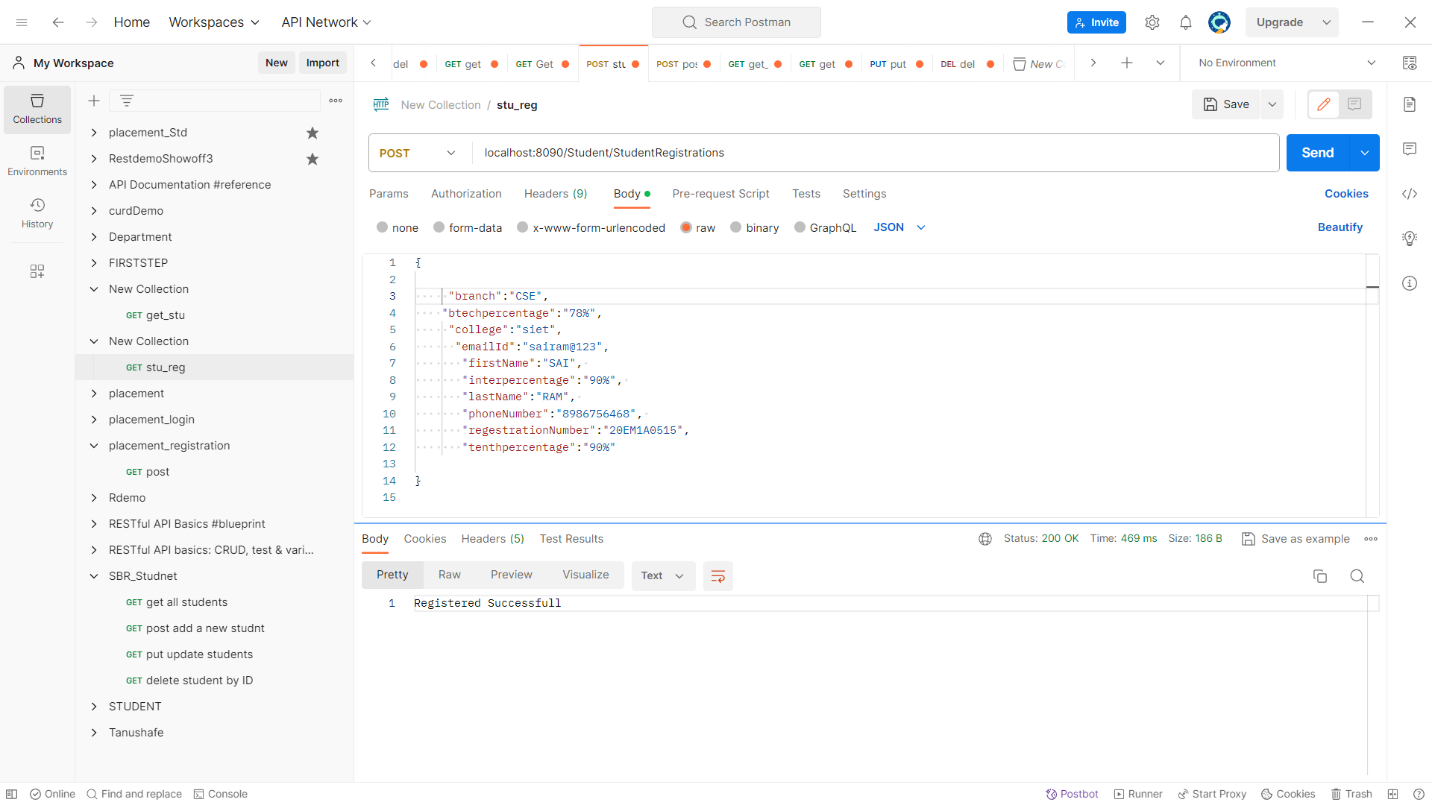
    }

}



**@PostMapping:**

* This annotation maps the HTTP POST request to the *"localhost:8090/Student/StudentRegistrations*”URI.body, which is expected to contain JSON representing StudentRegistration object**.**



* After Adding the body and send request it will gives return a “Registered Successful”.

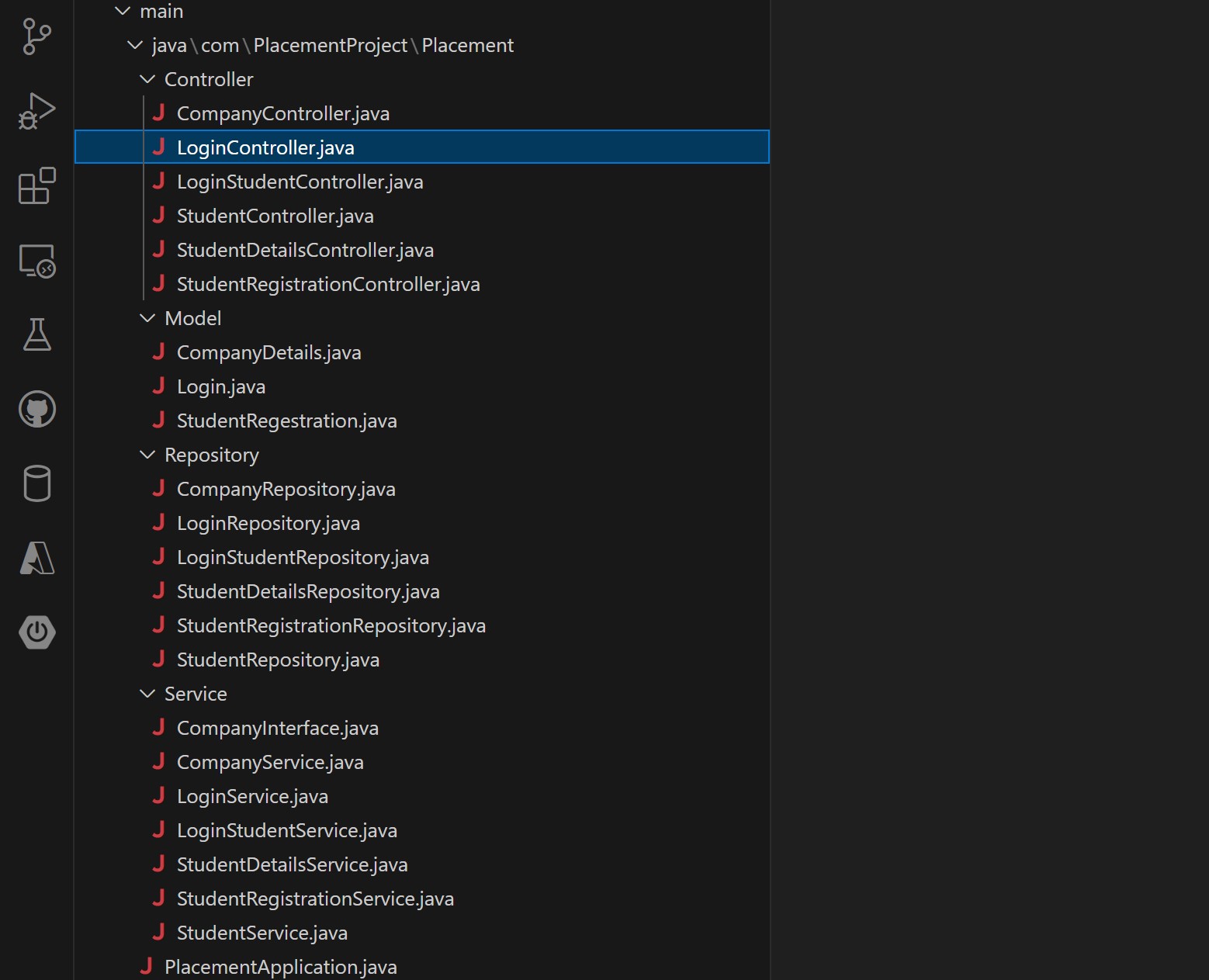
**Login Module:**

1.Created a new file in Controller Folder with the name “LoginController.java”.

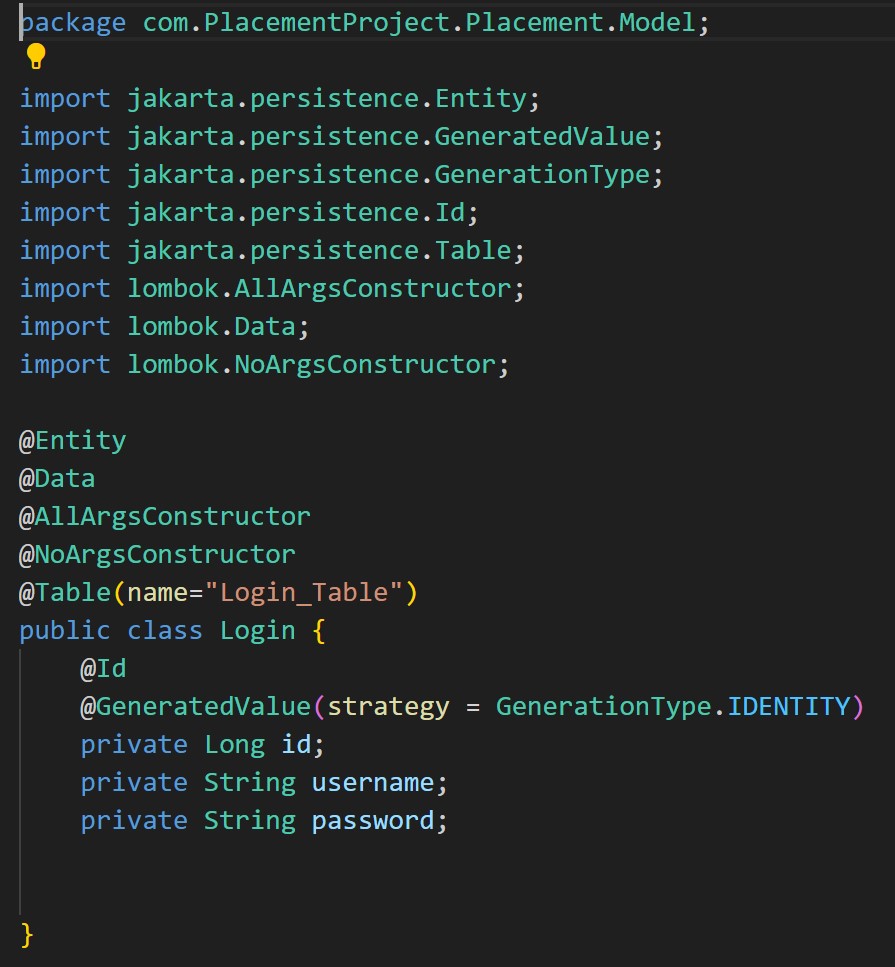
2. Next Created another file in the Model Folder with the name “Login.java”.

3.Created an another file in the Repository Folder with name “LoginRepository.java.

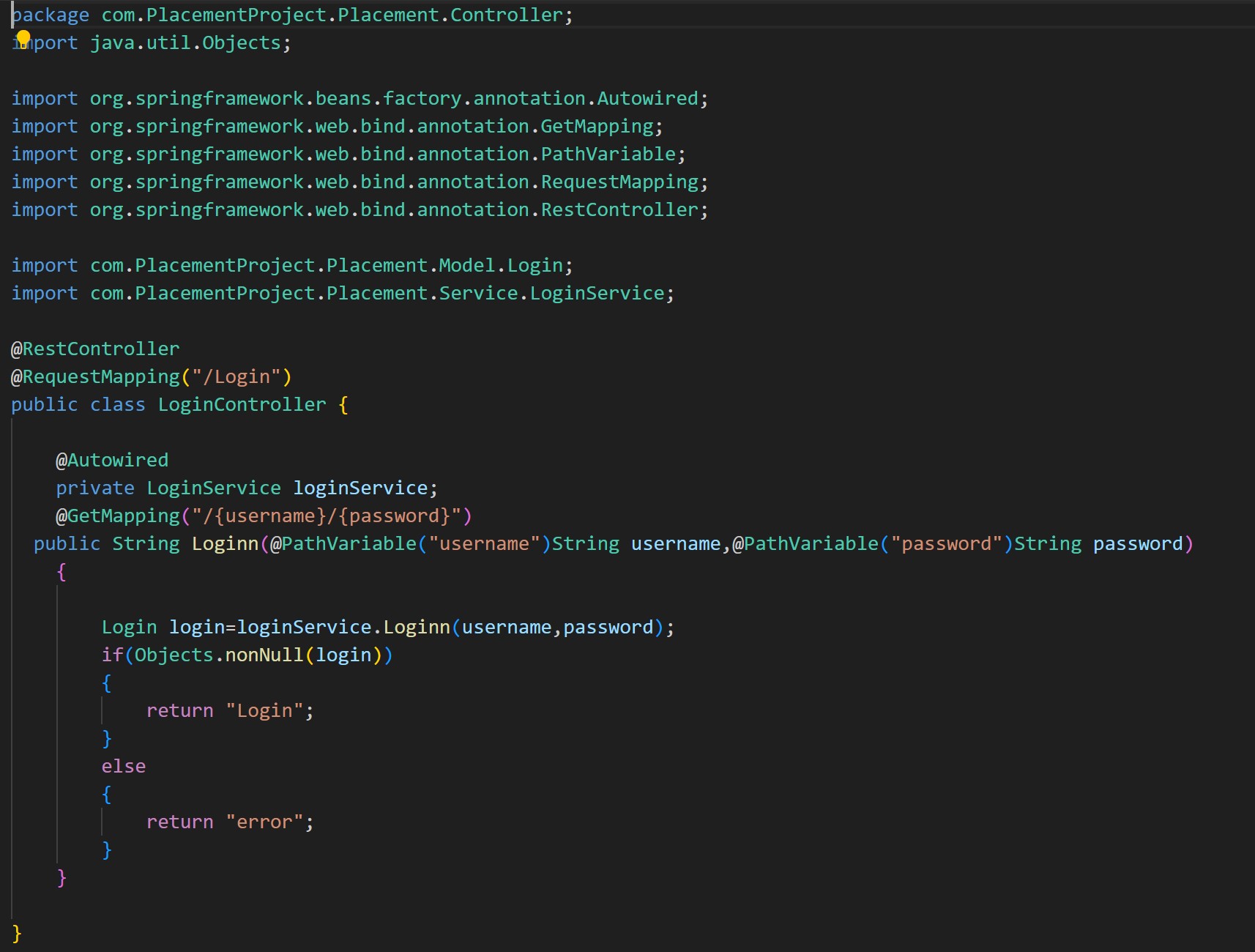
4. At last we created a file in Service Folder with name “LoginService.java”.



Program for Login.java:



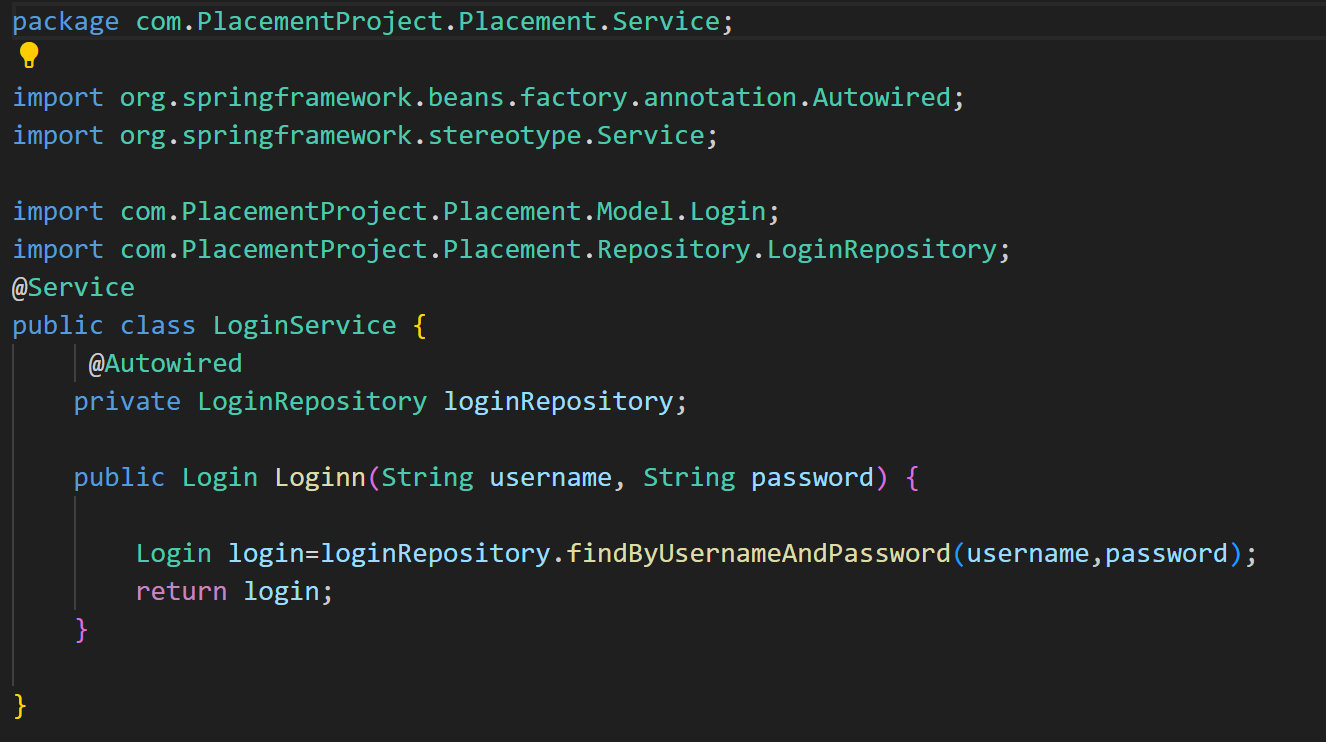
Program for LoginController.java:



Program for LoginRepository.java:



Program for LoginService.java:



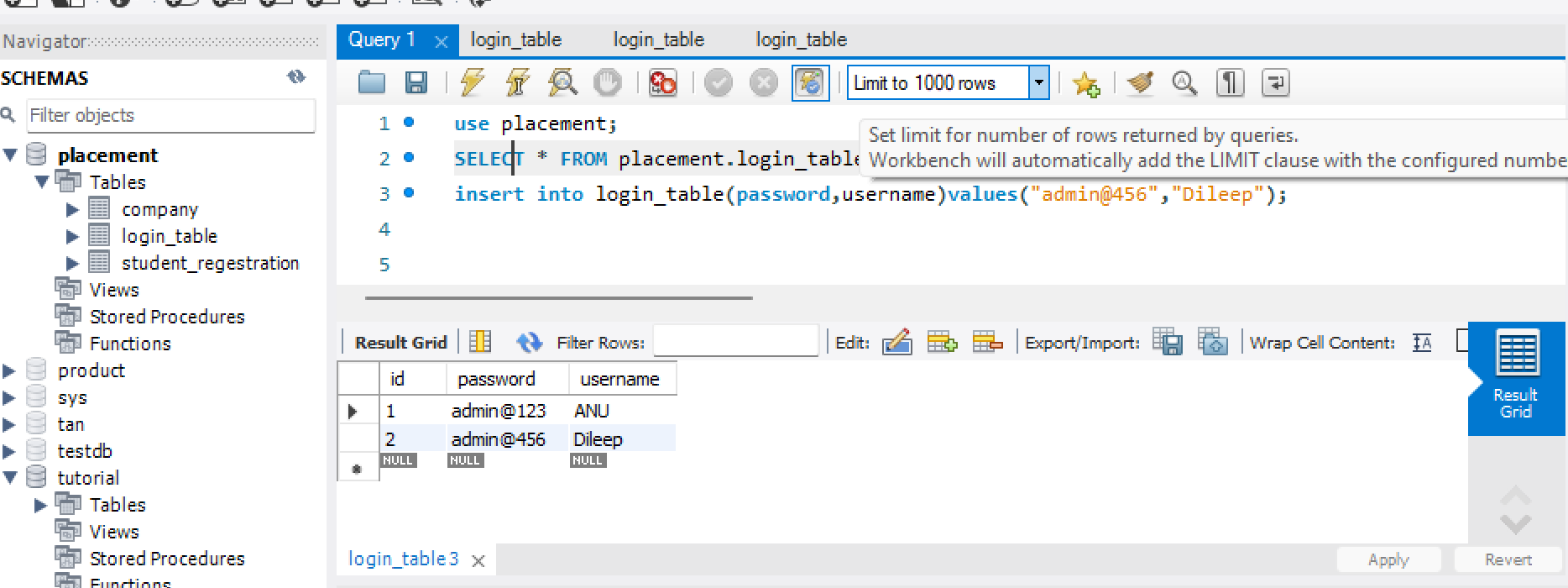
Creating a Database in Mysql workbench:

**1.**Create a database with the name “Placement” in Mysql workbench.

**2.** And Create a Table with name “login\_table” using the Login.java module.

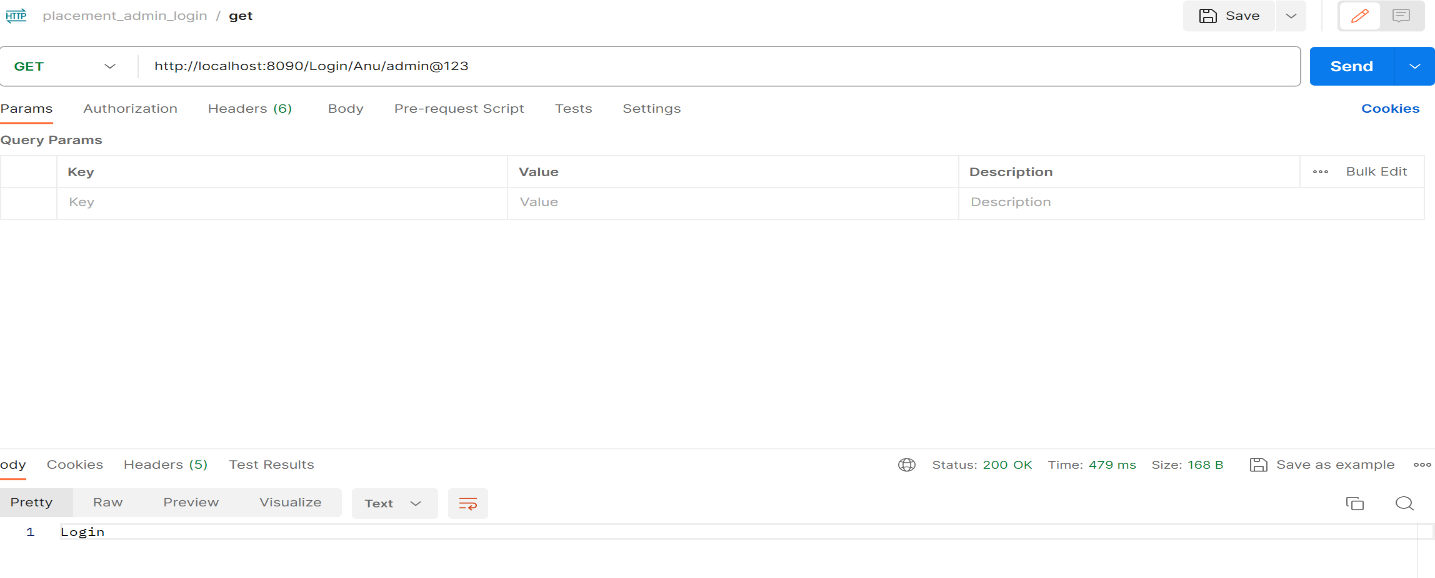
**3.** Insert the values in the table with specified columns names and give values (username , password).

**4.** Run the database to see the inserted values..

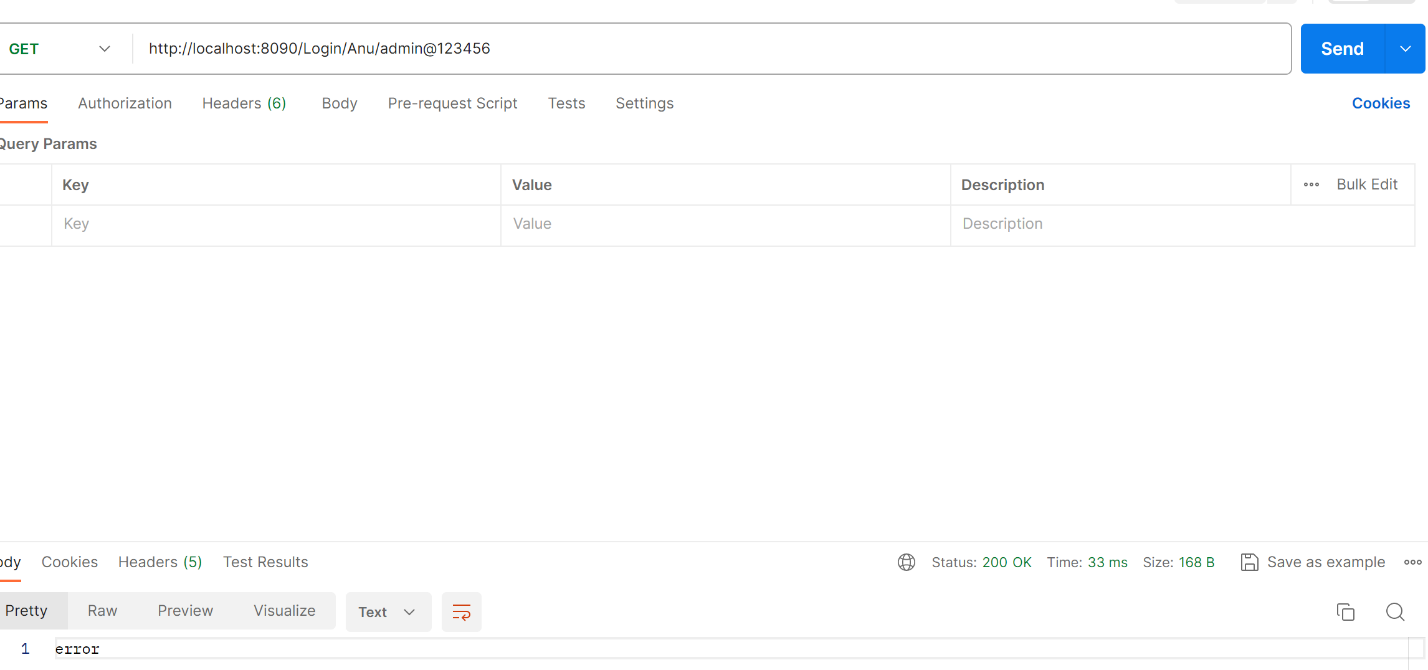


Check the process in the Postman app:

1. Open Postman App and create a new collection with the name “placement\_admin\_login”.
2. And Add the get request with name “Get”.
3. Enter the url as “ <http://localhost:8090/Login/Anu/admin@123> **”** to check if the admin details are present in the Database or Not .
4. If the admin details are present in the Database it will display as “Login”.



1. If We enter wrong Details or Invalid Details , it will shows an Error.



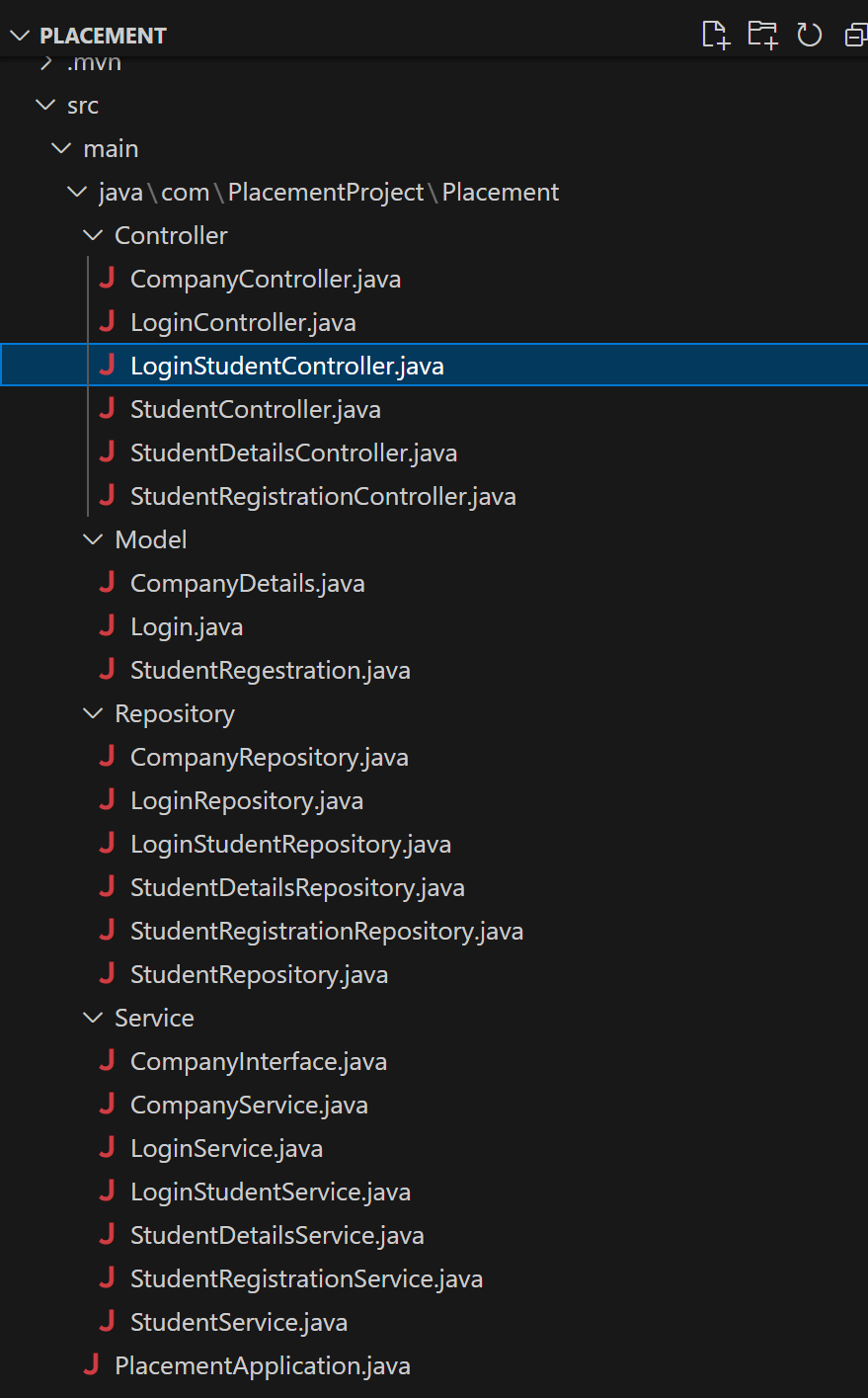
**Student Login Module:**

1.Created a new file in Controller Folder with the name “LoginStudentController.java”.

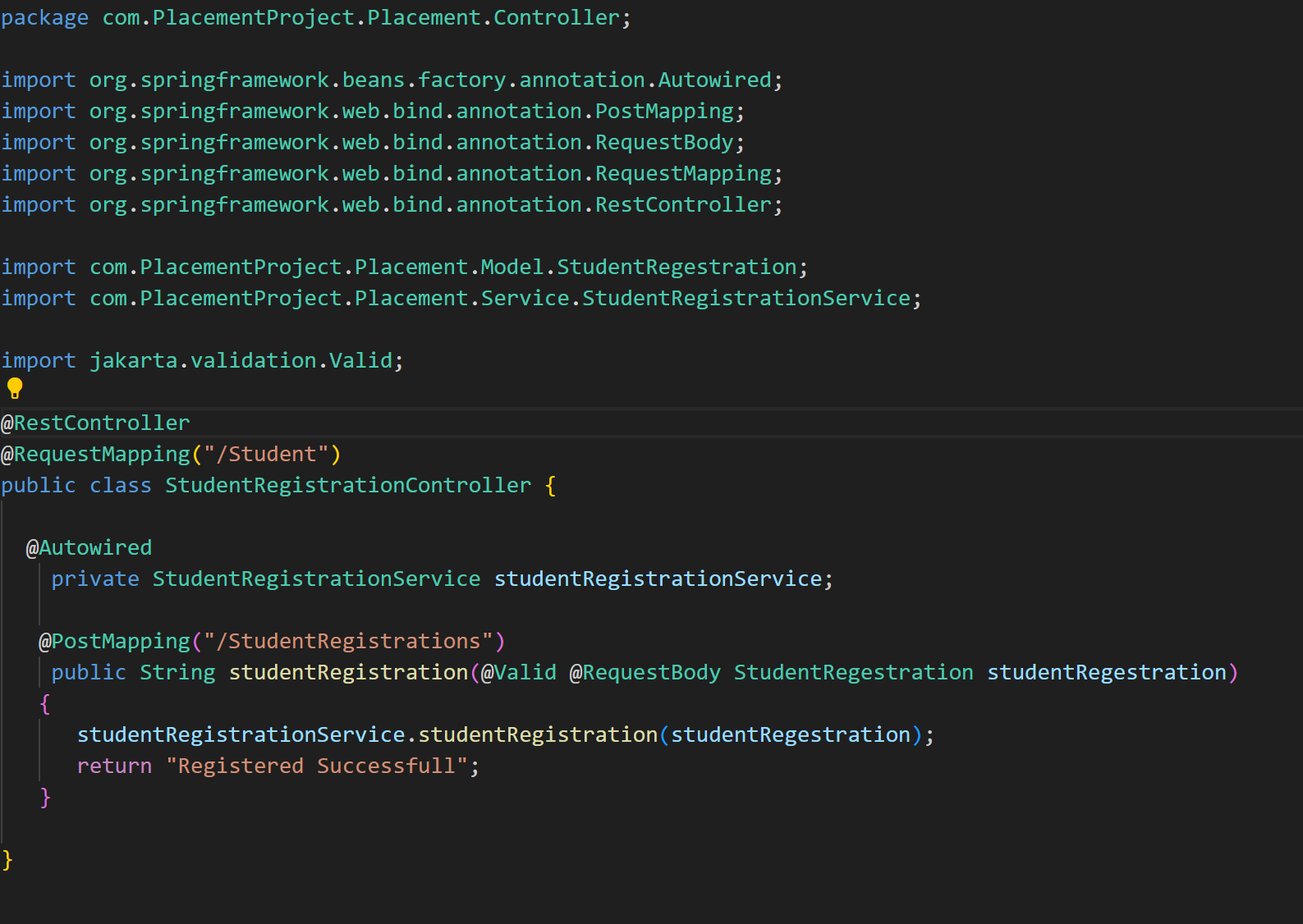
2. Next Created another file in the Model Folder with the name “Login.java”.

3.Created an another file in the Repository Folder with name “LoginStudentRepository.java.

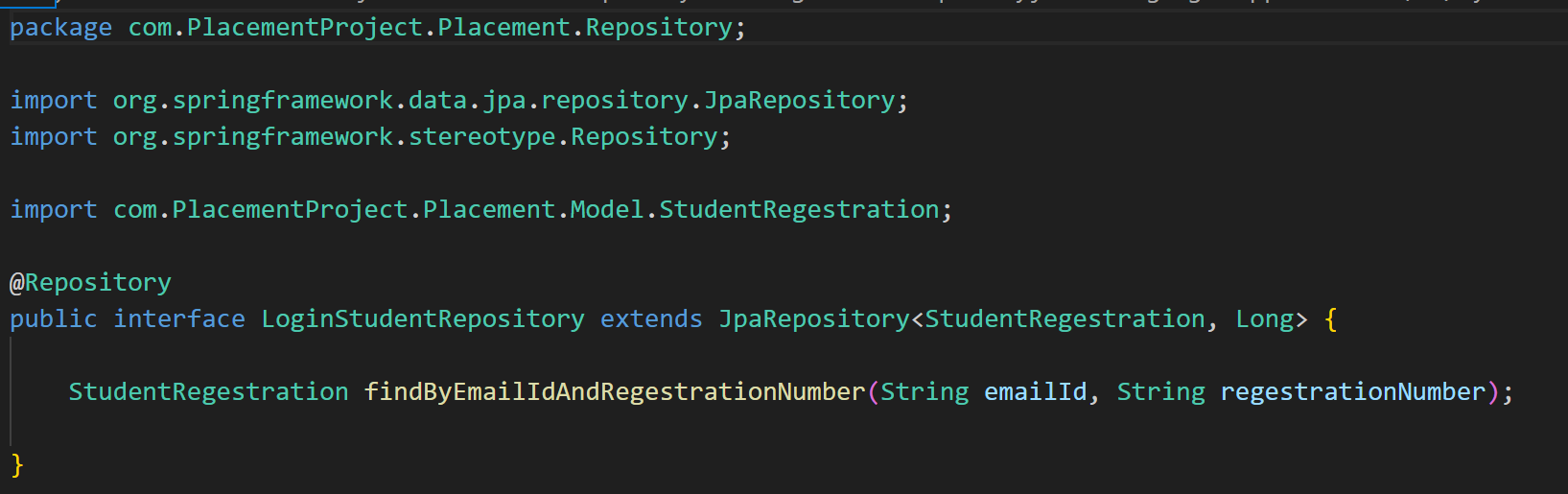
4. At last we created a file in Service Folder with name “LoginStudentService.java”.



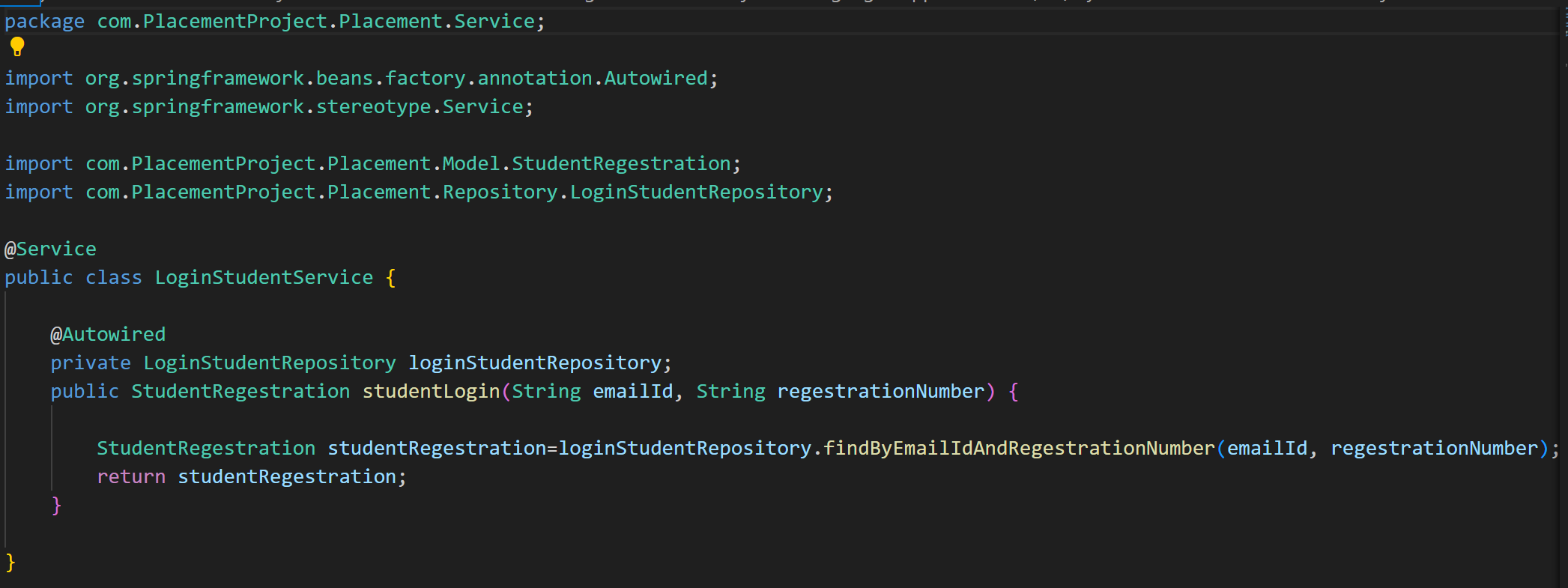
Program for StudentController.java:



Program for LoginStudentRepository.java:

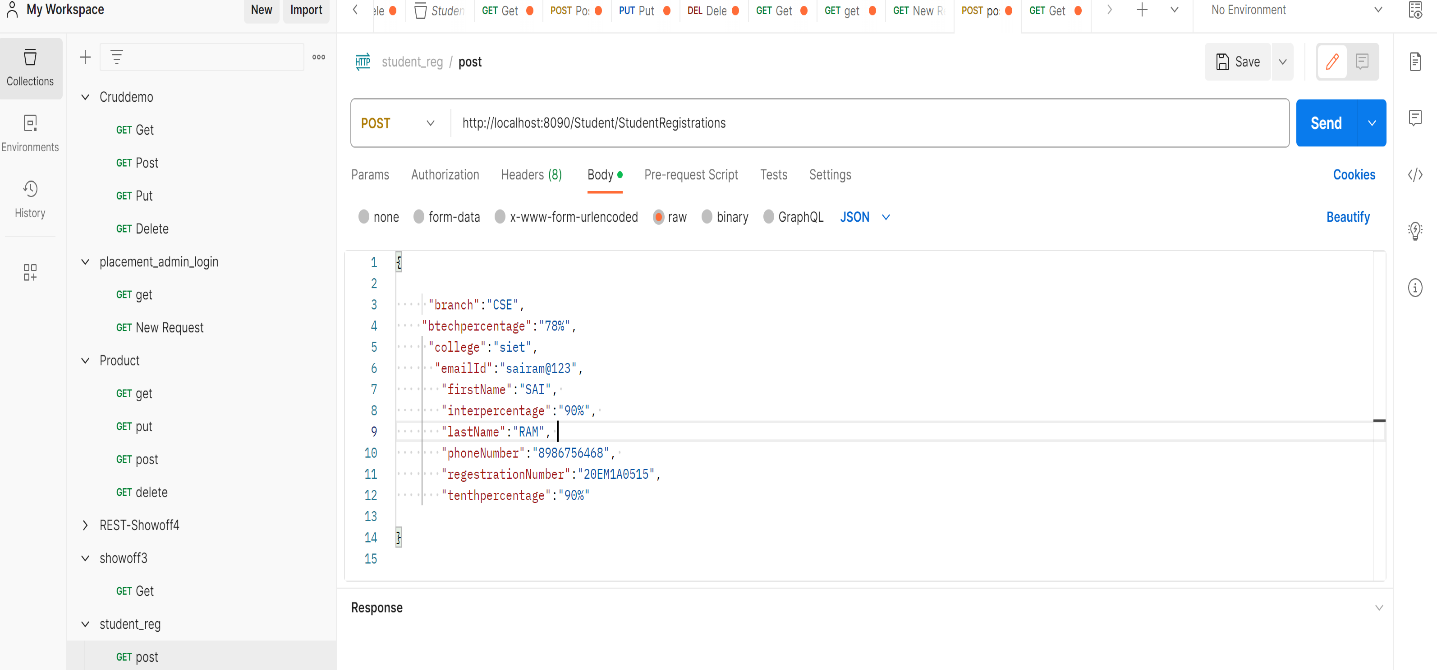


Program for LoginStudentService.java:



Check the process in the Postman app:

1. Open Postman App and create a new collection with the name “Student\_reg”.
2. And Add the get request with name “Post”.
3. Give the details as specified in Model module.
4. Enter the url as “ http://localhost:8090/Student/StudentRegistrations **”** and click Send Button.

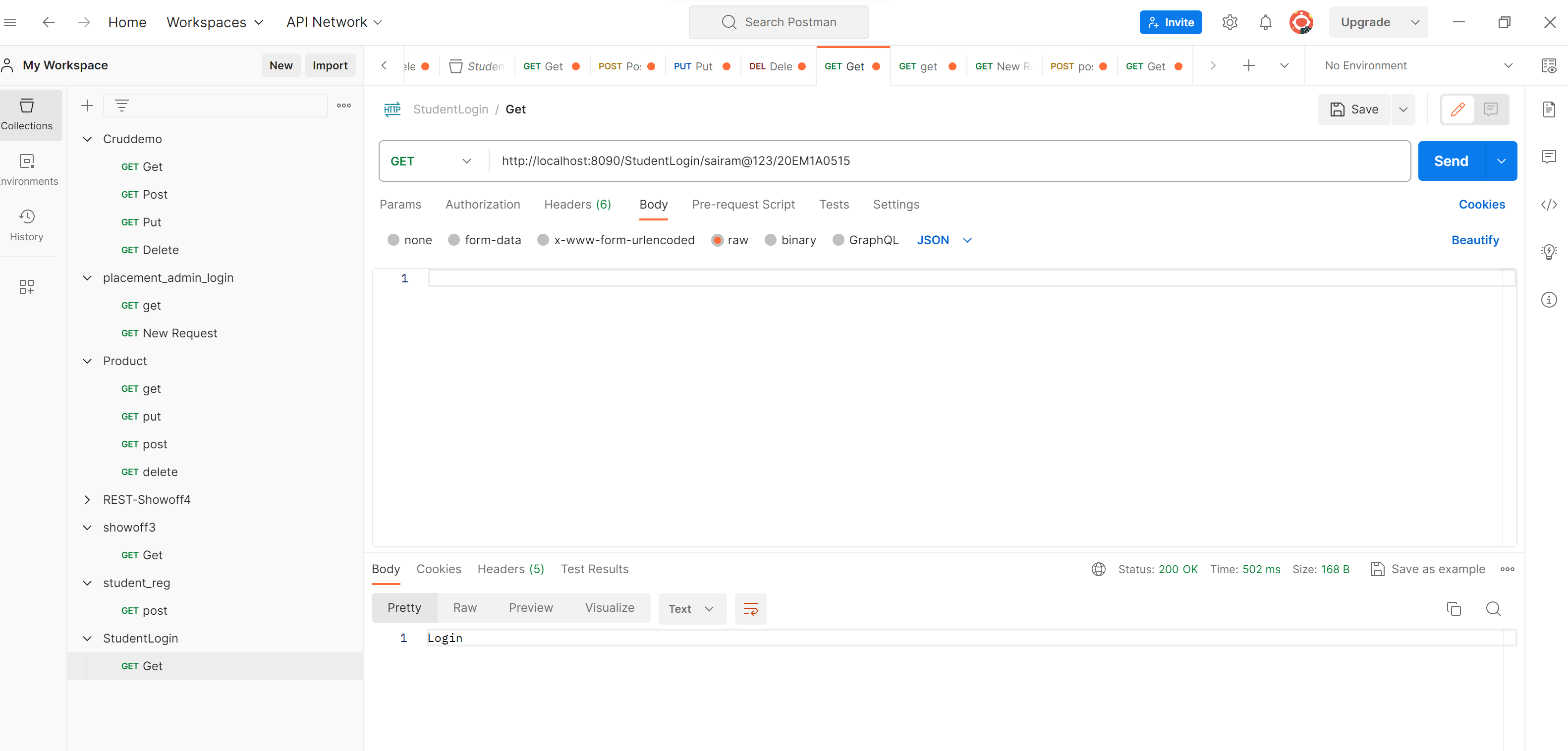


5.Check whether the details are present in the database or not.

6.Again create a new collection with the name “StudentLogin”.

And add the Get request.

7.Enter the url [http://localhost:8090/StudentLogin/sairam@123/20EM1A0515](http://localhost:8090/StudentLogin/sairam@123/20EM1A0515%20)  and click Send Button.



8. If the details are present in Database , it will show “Login”.

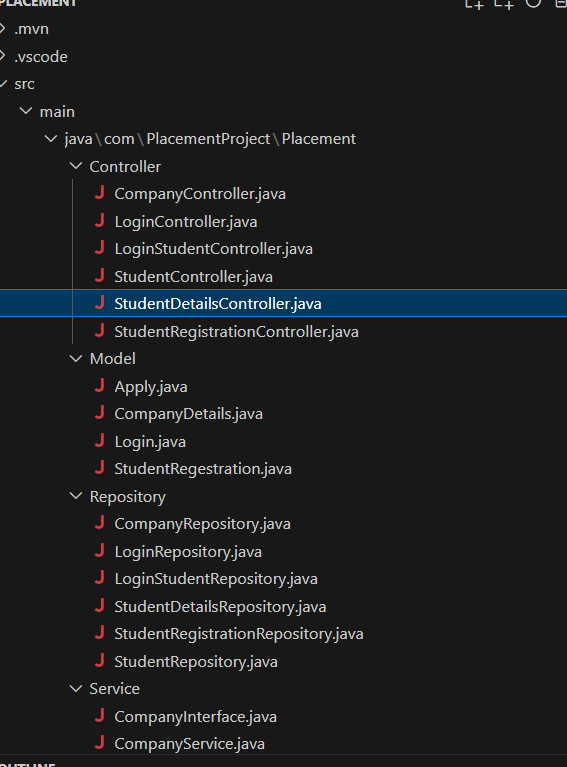
**Student View Module:**

1.Created a new file in Controller Folder with the name “studentController.java”.

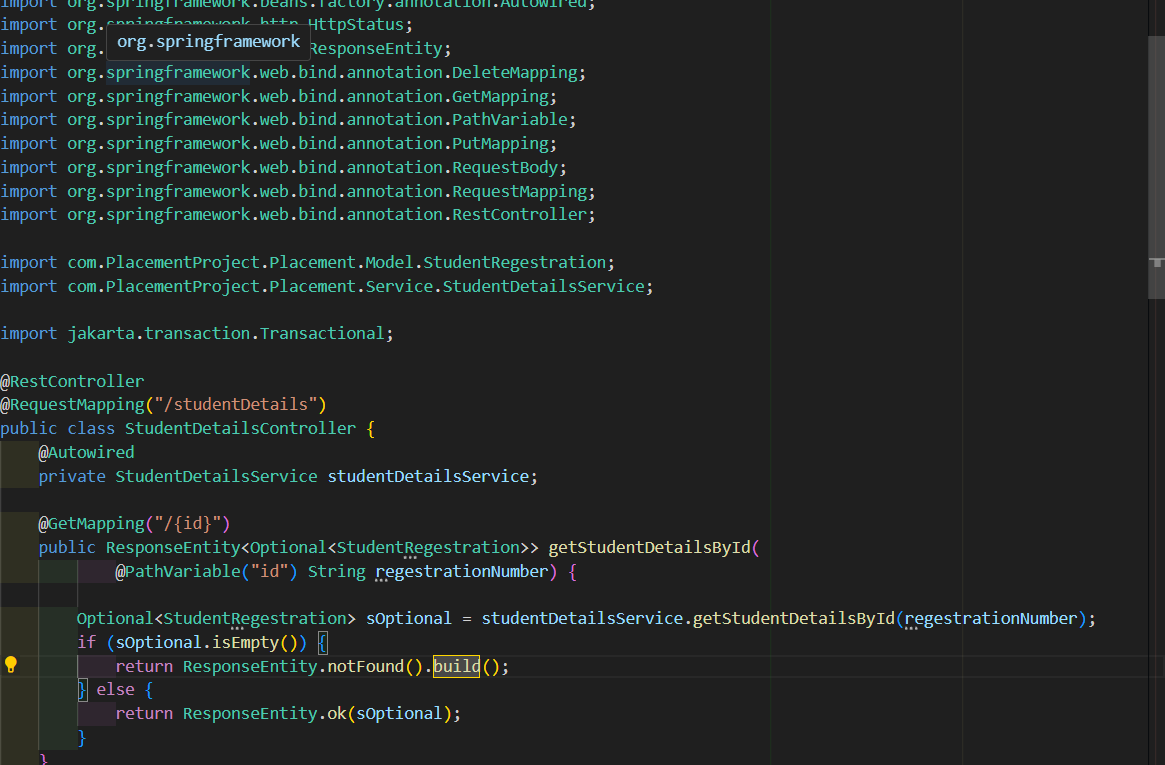
2. Use the same class “studentRegistartion.java” in the model folder.

3 Created an another file in the Repository Folder with name “StudentDeatilsRepository.java.

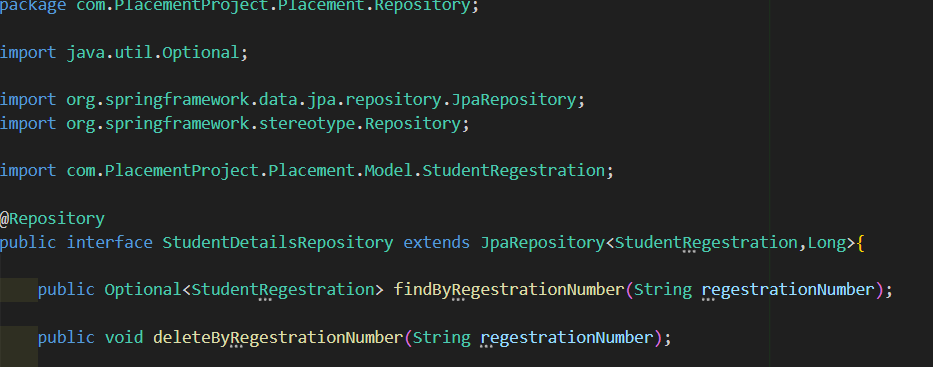
4. At last we created a file in Service Folder with name “StudentDeatilsService.java”.

****

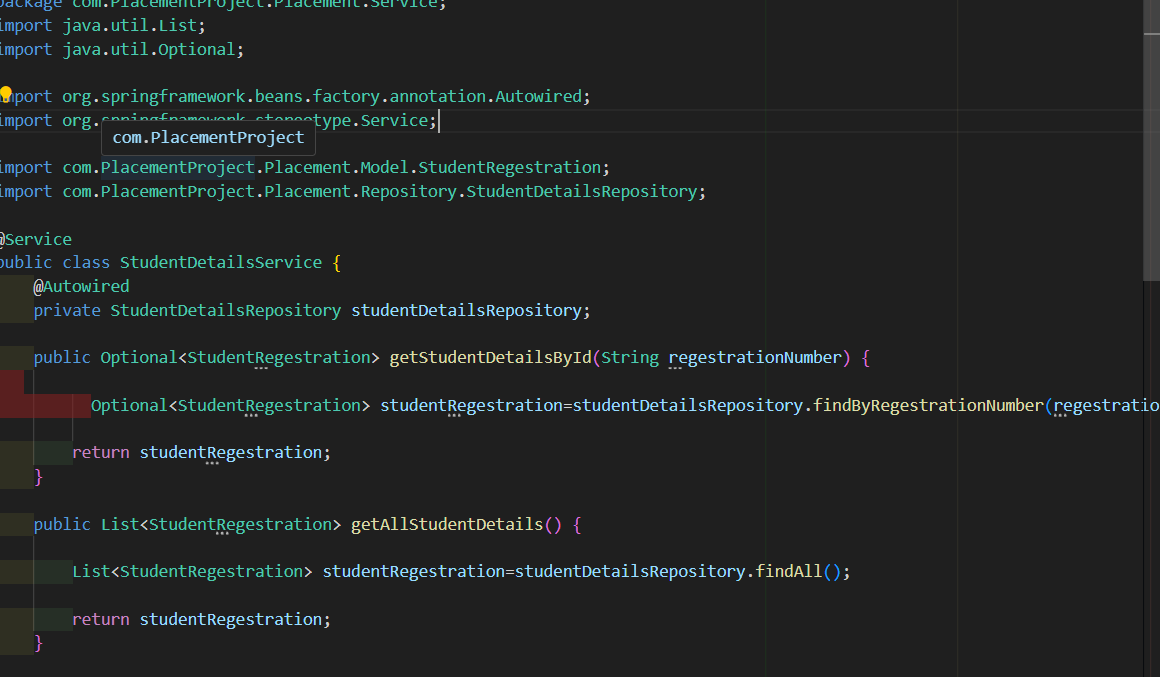
Program for StudentDetailsController.java:

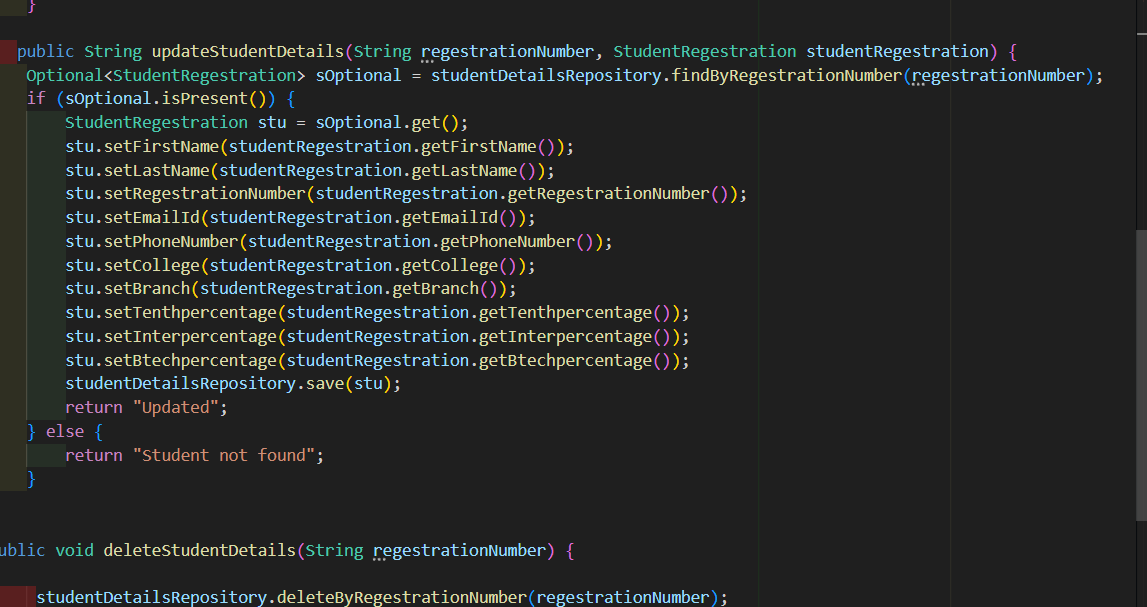


Program for StudentDeatilsRepository.java:



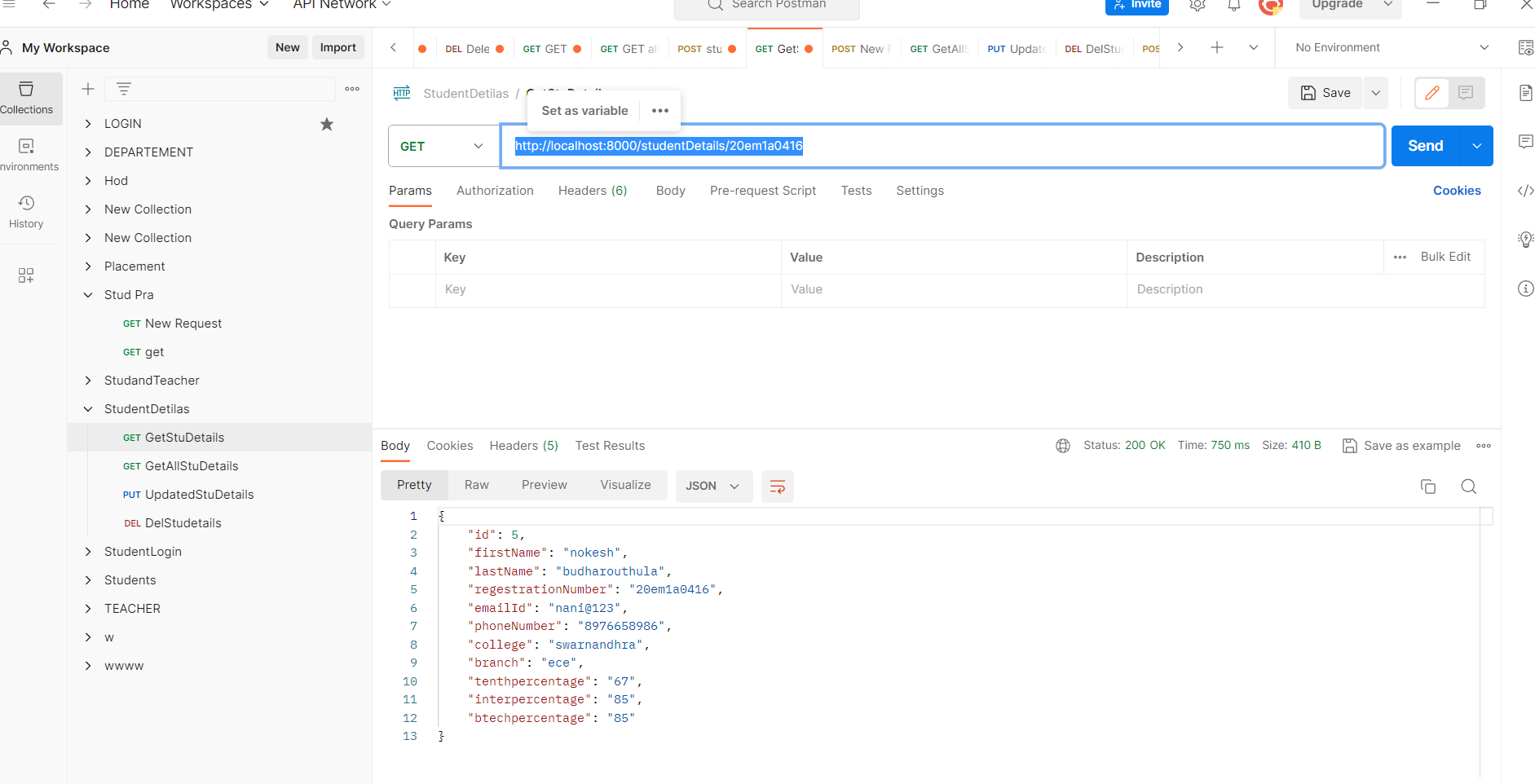
Program for StudentDeatilsService.java:





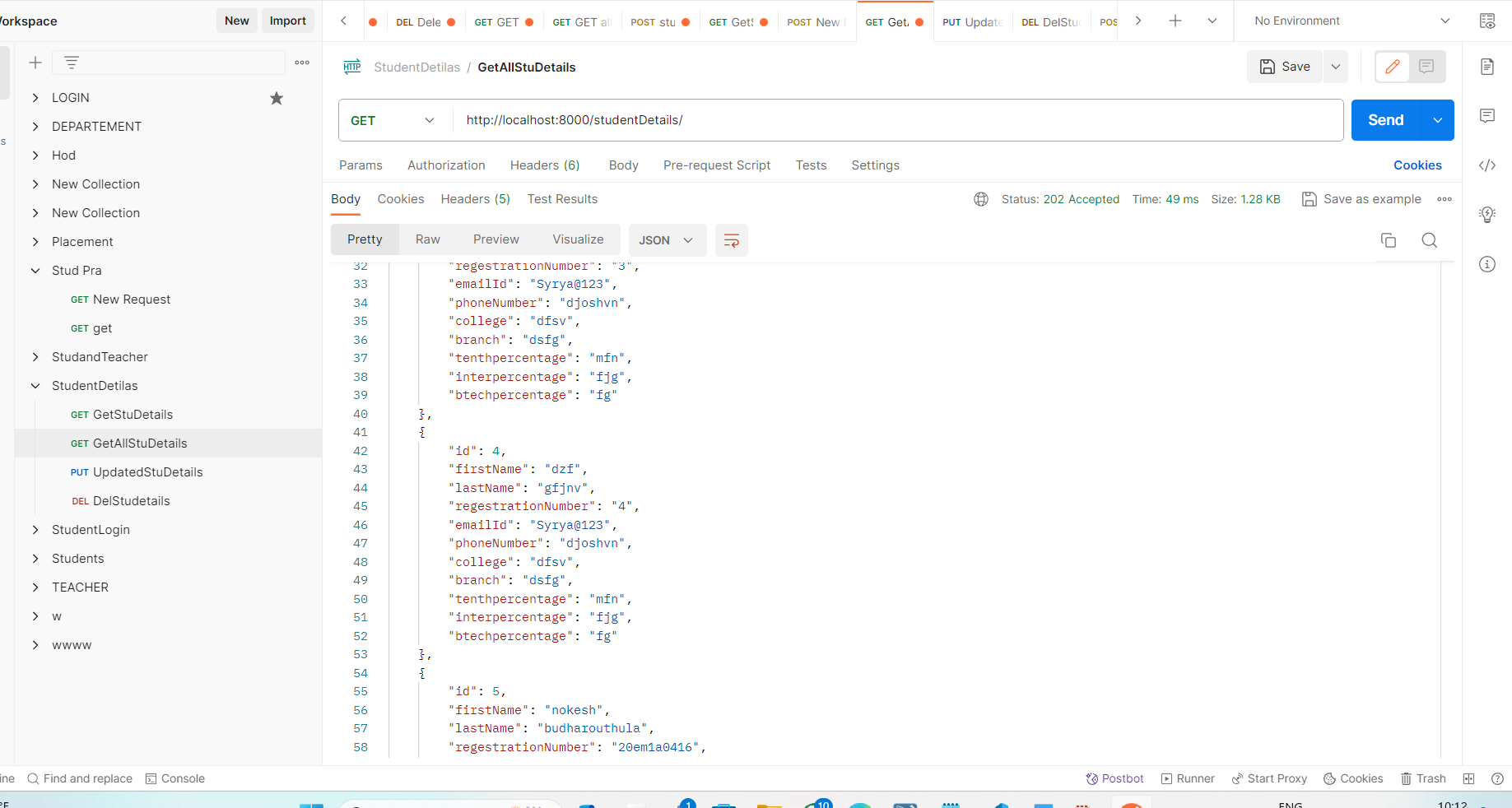
Check the process in the Postman app:

1. Open Postman App and create a new collection with the name “Student\_Deatils”.
2. And Add the get request with name “GetStuDetails”.
3. Get the details as specifinf the registeration number.
4. Enter the url as “ http://localhost:8000/studentDetails/20em1a0416**”** and click Send Button.



5 .If you want to get all the student details that are registered enter the

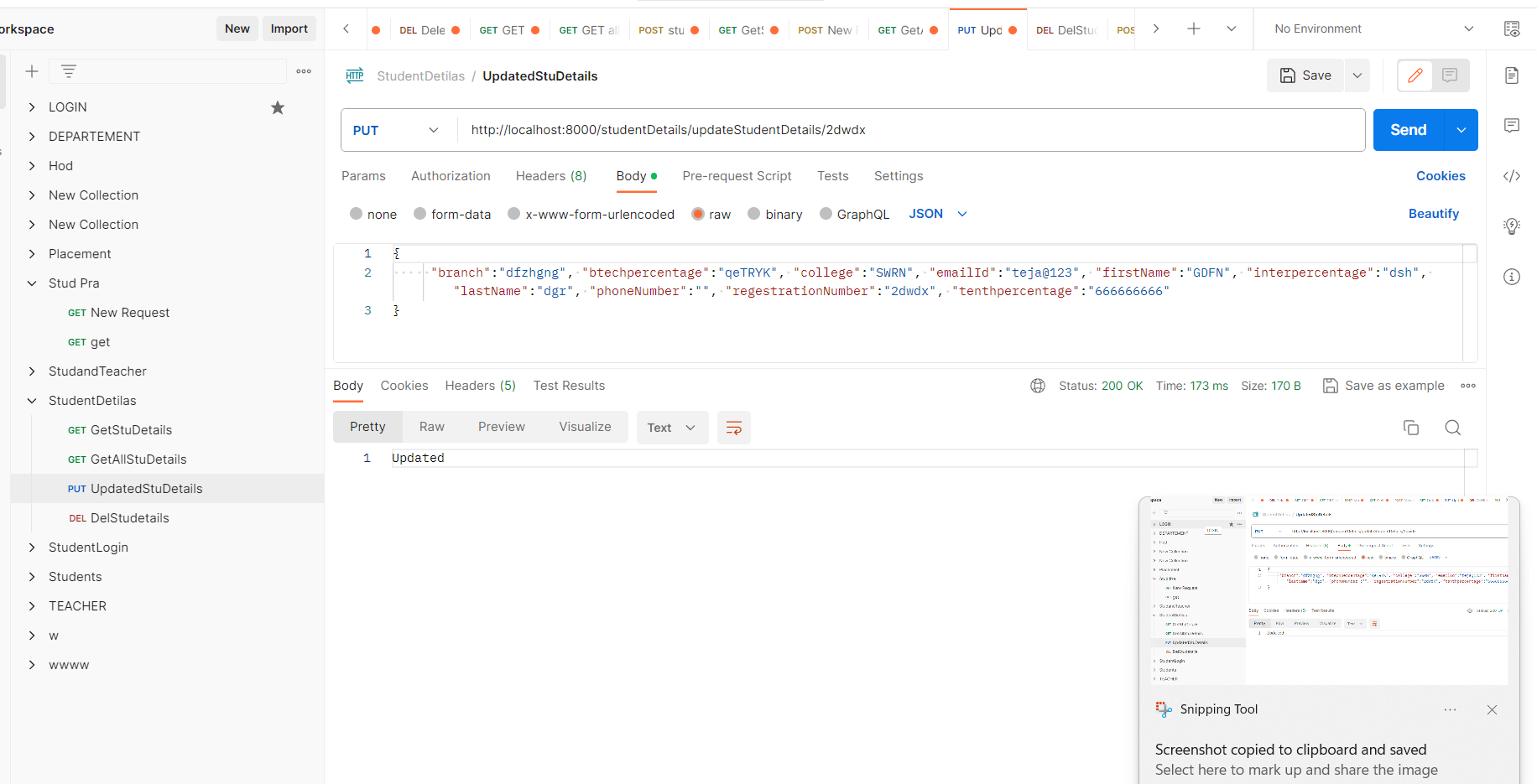
url=” <http://localhost:8000/studentDetails/>” in the post man using get operation



5 .If you want to update the student details then enter the registered number with

the put operation

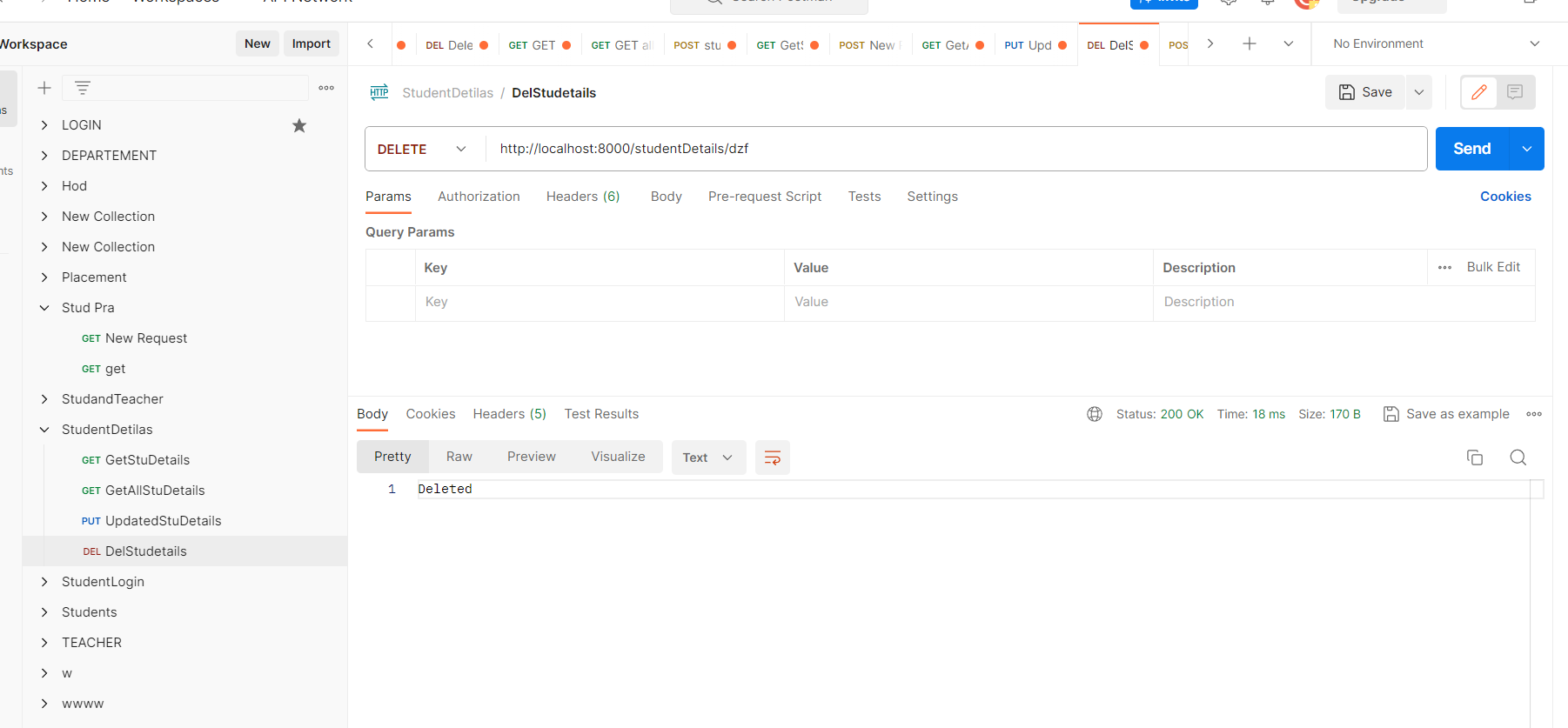
url=”http://localhost:8000/studentDetails/updateStudentDetails/2dwdx” in the post man



6 .If you want to delete any student details then enter the registered number with

the delete operation

url=”http://localhost:8000/studentDetails/dzf” in the post man



**Thank You**