PROJECT REPORT

Automatic Question Paper Generation Using Java(Spring MVC)

UE20CS352 – OBJECT ORIENTED ANALYSIS AND DESIGN WITH JAVA

Submitted by:

Chaitra B N **Nidhi Torvi** **Safiya Nawar**	**PES2UG20CS424** **PES2UG20CS445** **PES2UG20CS455**

Under the guidance of

Prof. Ruby Dinakar J
Professor Dept of CSE
PES University

January - May 2023

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

FACULTY OF ENGINEERING

PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)

Electronic City, Hosur Road, Bengaluru – 560 100, Karnataka, India

TABLE OF CONTENTS

1. Introduction

1.1 Problem Statement and Project Scope

A learning management system can be used for administration, documentation, tracking, delivery of educational courses, and learning and development programs. Keeping in mind that there are hundreds of students and faculty, this system allows us to manage everything from a single place.

Admin - after logging in, admin can add the courses offered and can delete the courses once they go out of demand

Students - after registering, students can view notes of all the courses added by the teacher and can attend quiz and submit quizzes as well as assignments assigned by the respected teacher.

Teacher (Faculty) - after logging in , the teacher can add notes of the course and assignments and quiz related to that course

Objective:

1. To provide an easy-to-use, bug-free application that is a one-stop shop for learning management that saves time.

2. It will be a secure and robust system that can be relied upon.

3. A safe place to store all the student and faculty information, it maintains all the records of course details, enrollment information of the students.

2. Functional Requirements

A. Admin part.

This includes adding the courses which include course id , faculty name,course name,is it certified course or not and populating the database with the courses of different topics along with the Details.

B. Teacher part.

This involves adding notes for each course. Assignments And Quiz will be added to that corresponding course.

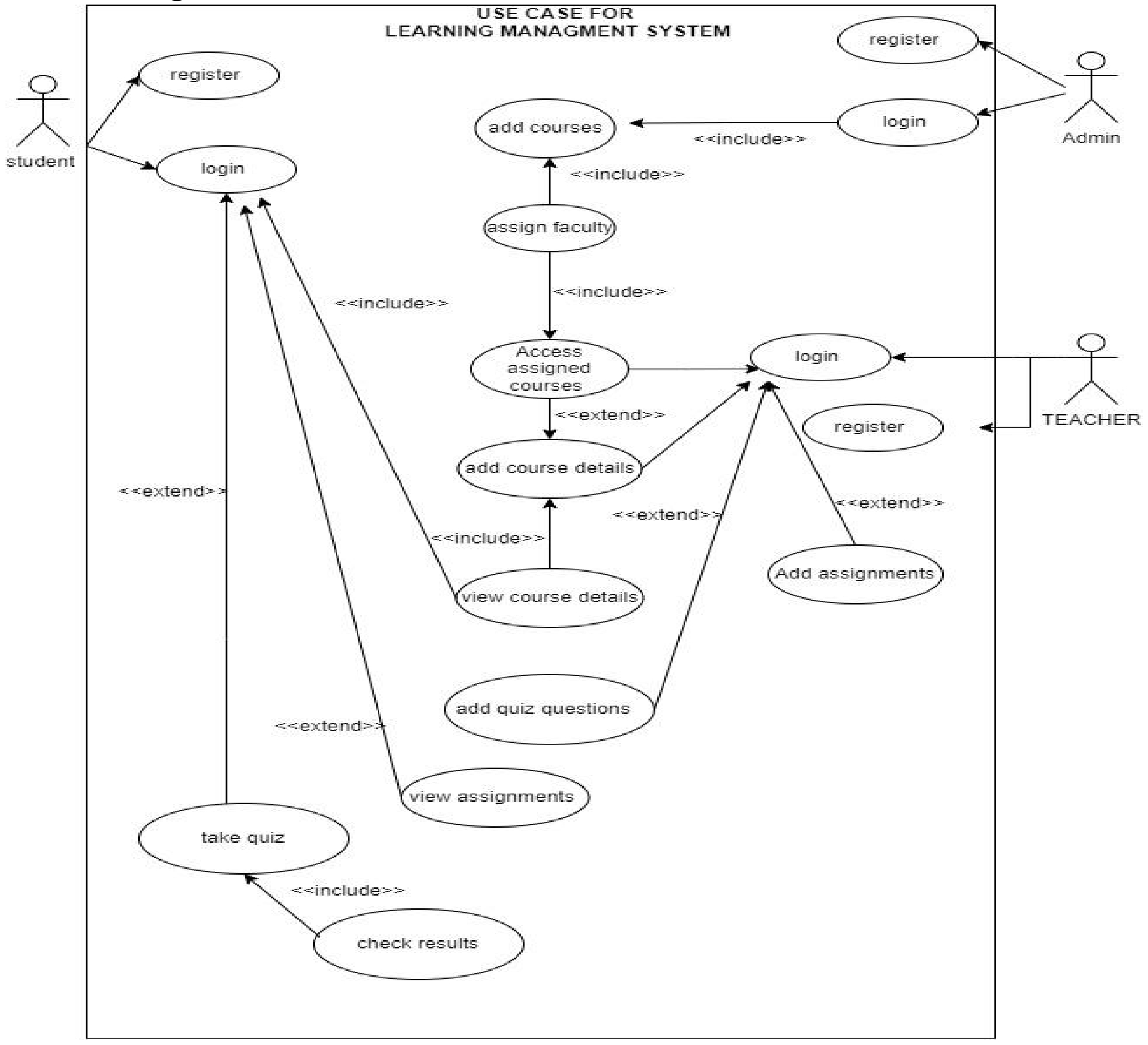
C. Student part.

This includes viewing the course and its details. Students can attempt Quiz and Assignments that are assigned by the teacher.

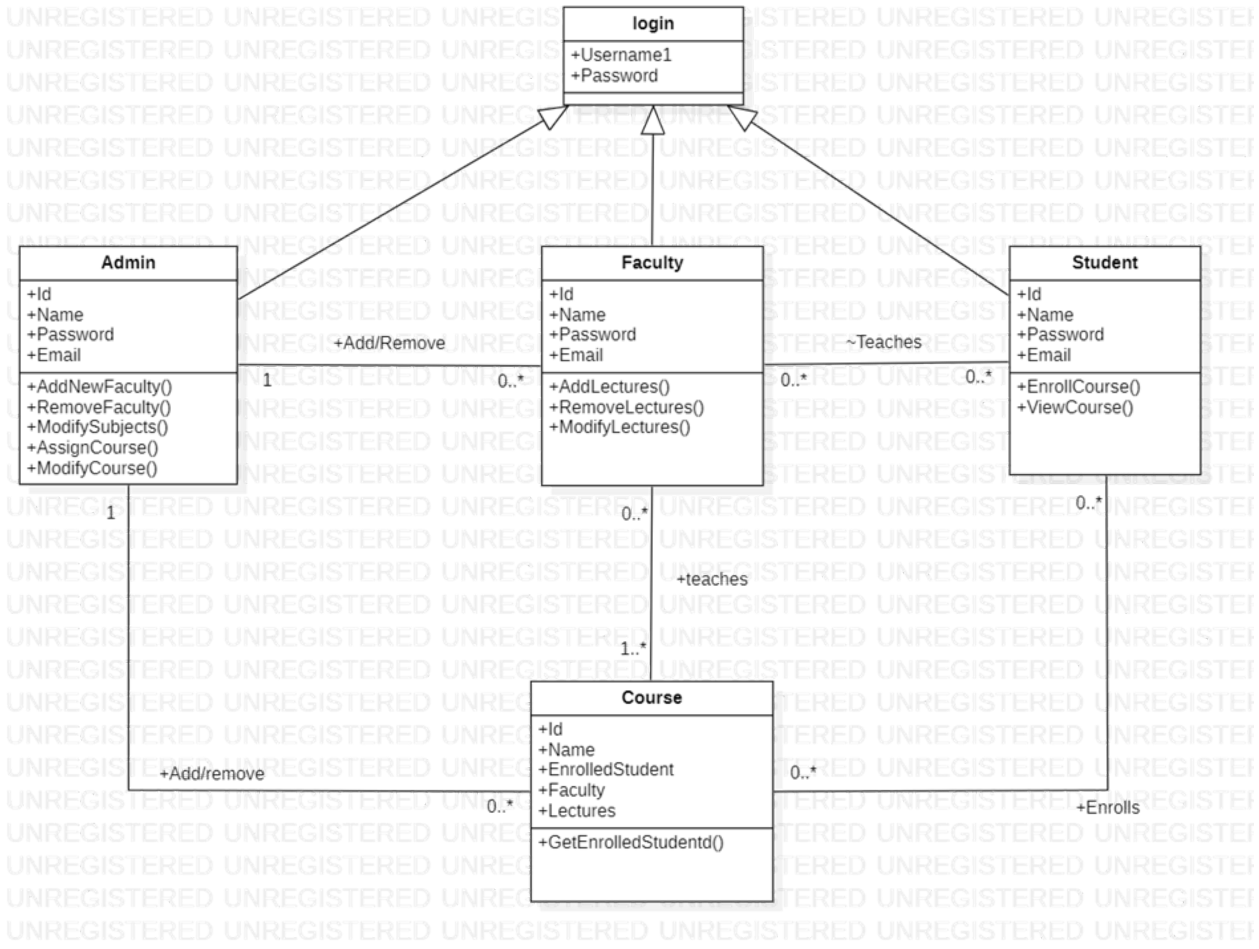
3. Project Details

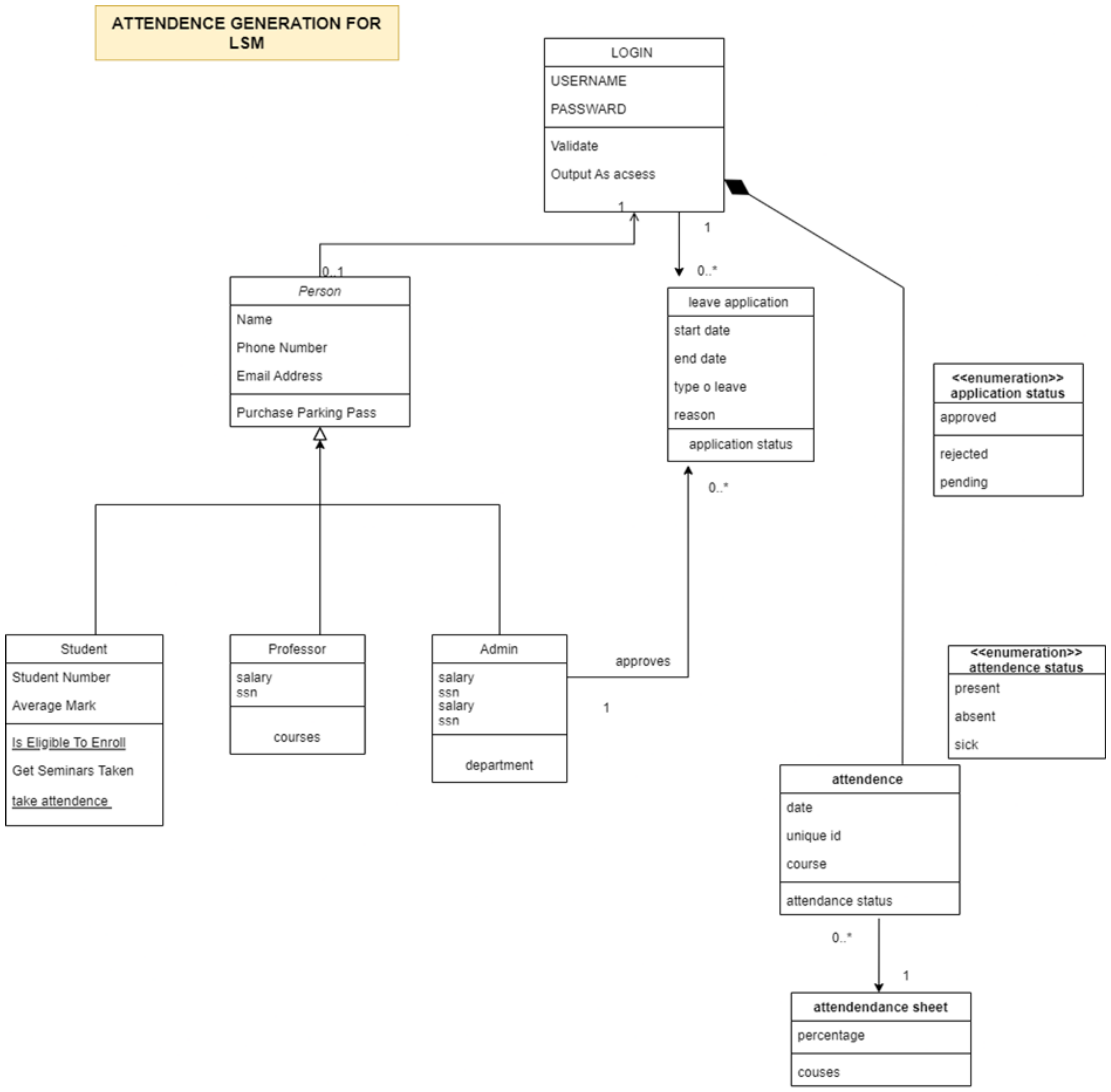
The project is a simple learning management system for learners who can all the courses. The project is developed in Java. Using Spring MVC framework and various design principles and patterns the system is highly maintainable and extensible.

3.1. Use Case Diagram



3.2. Class Diagram





4. Design Details

4.1. Architectural Patterns

The architecture used is MVC Architecture.

This project follows MVC Architecture which is used to structure code into separate, distinct components. The MVC pattern separates an application into three main components: the model, the view, and the controller. Each component has a distinct role and responsibility within the application, which helps to improve code organization, maintainability, and reusability.

Spring MVC is a popular web application framework that is built on top of the Spring Framework. It provides a powerful and flexible way to develop web applications using the Model-View-Controller (MVC) pattern. The key features of the Spring MVC framework are Handler Mapping and Adapter, Dependency Injection, Flexible Configuration, Powerful Data Binding, View Resolvers, Exception Handling, Interceptor Support, RESTful Web Services Support, Testing Support etc.

4.2. Design Patterns

Design Patterns used are Singleton Pattern, Prototype Pattern, Observer Pattern.

Singleton Pattern: Each function is handled in a separate class to ensure that only a single instance should be created and a single object can be used by all the classes.

Prototype Pattern: It is cloning of an existing object instead of creating a new one and can also be customized as per the requirement. The main reason to use this design pattern is to reduce subclassing and it lets us add or remove objects.

DAO Pattern: In Spring MVC, the DAO (Data Access Object) pattern is a common design pattern used to abstract and encapsulate the database access code. The purpose of this pattern is to provide a single point of entry for accessing the database, which can then be used by multiple Service layer components within the application. The DAO pattern consists of a set of DAO classes that encapsulate the database access code. These classes are responsible for providing methods to perform basic CRUD (Create, Read, Update, and Delete) operations on the database

4.3. Design Principles

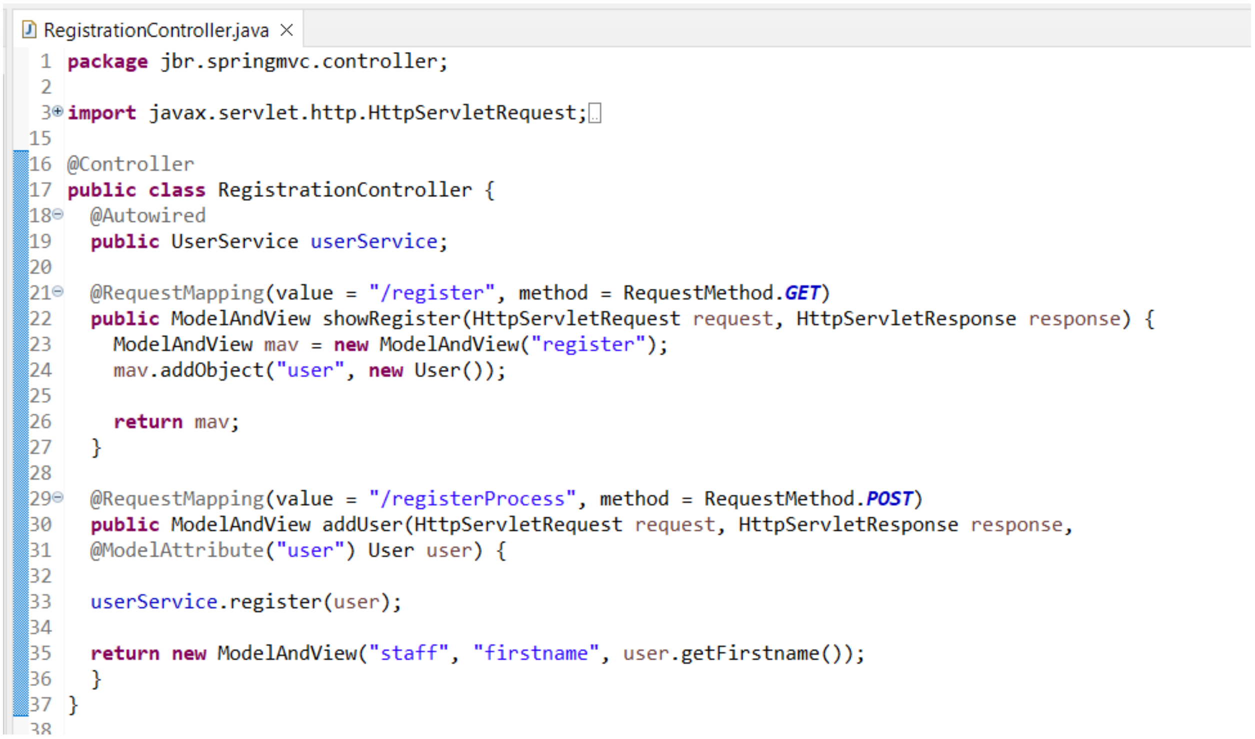
Single Responsibility Principle: The Single Responsibility Principle states that every class or module should have a single responsibility or reason to change. This means that a class should only have one job or function to perform, and that job should be well-defined and easily understandable. By adhering to this principle, we can make our code more modular, easier to test, and less prone to bugs.

Dependency Inversion Principle: The Dependency Inversion Principle states that high-level modules should not depend on low-level modules, but instead on abstractions. In other words, the code should be designed in such a way that it's easy to change low-level implementation details without affecting the high-level logic. This principle helps create more flexible and maintainable code that can adapt to changing requirements.

Open-Closed Principle (OCP): The application is open to extension but closed to modification. This means that new functionality can be added to the application without modifying the existing code, making the application more maintainable and scalable.

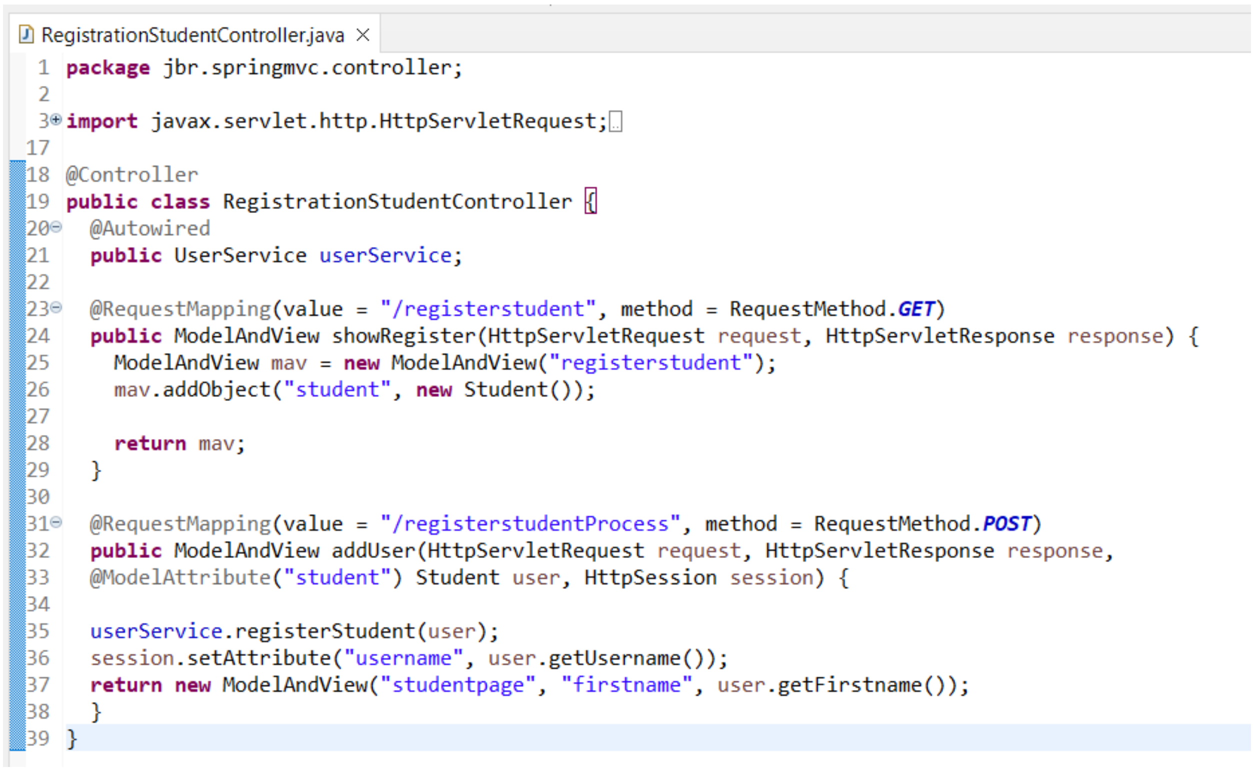
5. Code And Implementation

5.1. Use Case Implementation (Java code)



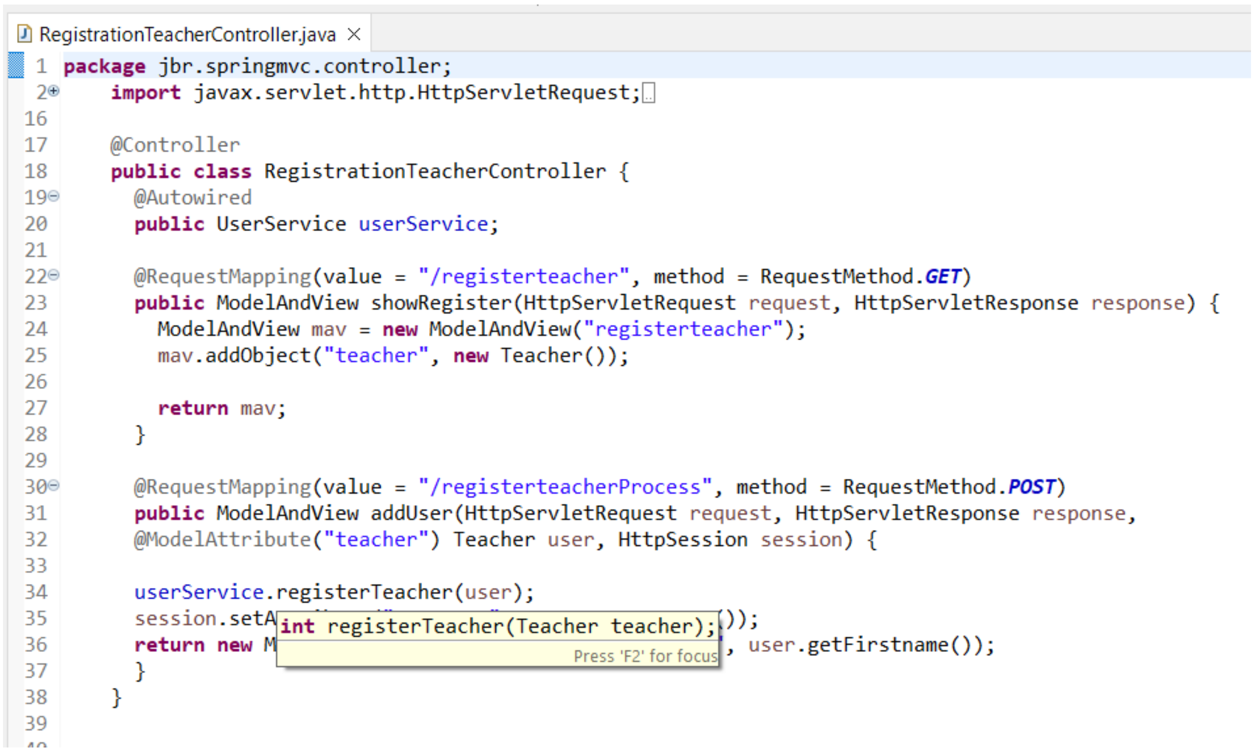
RegistrationController.java

```java
package jbr.springmvc.controller;

import javax.servlet.http.HttpServletRequest;

@Controller
public class RegistrationController {
  @Autowired
  public UserService userService;

  @RequestMapping(value = "/register", method = RequestMethod.GET)
  public ModelAndView showRegister(HttpServletRequest request, HttpServletResponse response) {
    ModelAndView mav = new ModelAndView("register");
    mav.addObject("user", new User());

    return mav;
  }

  @RequestMapping(value = "/registerProcess", method = RequestMethod.POST)
  public ModelAndView addUser(HttpServletRequest request, HttpServletResponse response,
  @ModelAttribute("user") User user) {

  userService.register(user);

  return new ModelAndView("staff", "firstname", user.getFirstname());
  }
}
```



RegistrationStudentController.java

```
package jbr.springmvc.controller;

import javax.servlet.http.HttpServletRequest;

@Controller
public class RegistrationStudentController {
  @Autowired
  public UserService userService;

  @RequestMapping(value = "/registerstudent", method = RequestMethod.GET)
  public ModelAndView showRegister(HttpServletRequest request, HttpServletResponse response) {
    ModelAndView mav = new ModelAndView("registerstudent");
    mav.addObject("student", new Student());

    return mav;
  }

  @RequestMapping(value = "/registerstudentProcess", method = RequestMethod.POST)
  public ModelAndView addUser(HttpServletRequest request, HttpServletResponse response,
  @ModelAttribute("student") Student user, HttpSession session) {

  userService.registerStudent(user);
  session.setAttribute("username", user.getUsername());
  return new ModelAndView("studentpage", "firstname", user.getFirstname());
  }
}
```



RegistrationTeacherController.java

```
package jbr.springmvc.controller;
    import javax.servlet.http.HttpServletRequest;

    @Controller
    public class RegistrationTeacherController {
      @Autowired
      public UserService userService;

      @RequestMapping(value = "/registerteacher", method = RequestMethod.GET)
      public ModelAndView showRegister(HttpServletRequest request, HttpServletResponse response) {
        ModelAndView mav = new ModelAndView("registerteacher");
        mav.addObject("teacher", new Teacher());

        return mav;
      }

      @RequestMapping(value = "/registerteacherProcess", method = RequestMethod.POST)
      public ModelAndView addUser(HttpServletRequest request, HttpServletResponse response,
      @ModelAttribute("teacher") Teacher user, HttpSession session) {

      userService.registerTeacher(user);
      session.setA                                  ));
      return new M                                  , user.getFirstname());
      }
    }
```

CourseController.java

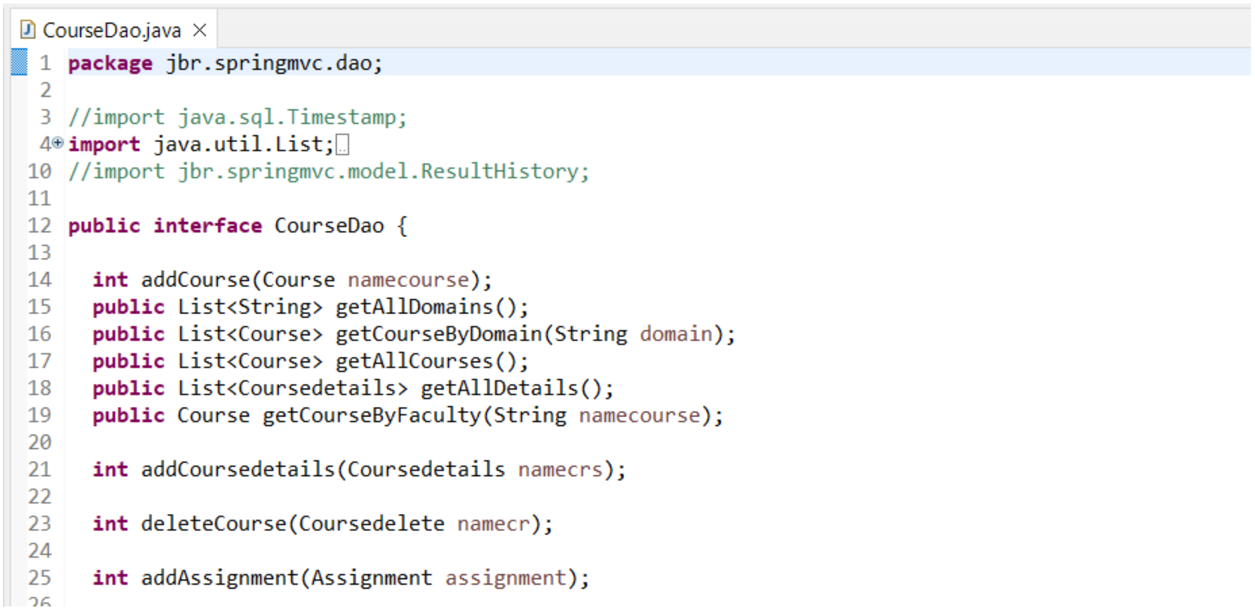
package jbr.springmvc.controller;

```
import java.util.List;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
//import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ModelAttribute;
//import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
//import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.servlet.ModelAndView;

import jbr.springmvc.model.Assignment;
import jbr.springmvc.model.Course;
import jbr.springmvc.model.Coursedelete;
import jbr.springmvc.model.Coursedetails;
//import jbr.springmvc.model.User;
import jbr.springmvc.service.CourseService;

@Controller
public class CourseController {
  @Autowired
  public CourseService courseService;

  @RequestMapping(value = "/addcourse", method = RequestMethod.GET)
    public ModelAndView showCourse(HttpServletRequest request, HttpServletResponse
response) {
    ModelAndView mav = new ModelAndView("addcourse");
    mav.addObject("namecourse", new Course());

    return mav;
  }
```

```
@RequestMapping(value = "/addCourseProcess", method = RequestMethod.POST)
   public ModelAndView addUser(HttpServletRequest request, HttpServletResponse response,
@ModelAttribute("namecourse") Course namecourse) {

courseService.addCourse(namecourse);
ModelAndView mav = new ModelAndView("addcoursesuccess");
mav.addObject("idcourse", namecourse.getIdcourse());
mav.addObject("domain", namecourse.getDomain());
mav.addObject("namecourse", namecourse.getCourse());
mav.addObject("faculty", namecourse.getFaculty());

return mav;
}

@RequestMapping(value = "/adddetails", method = RequestMethod.GET)
      public ModelAndView showCoursedetails(HttpServletRequest request,
HttpServletResponse response) {
  ModelAndView mav = new ModelAndView("adddetails");
  mav.addObject("namecrs", new Coursedetails());

  return mav;
}

@RequestMapping(value = "/addCrsProcess", method = RequestMethod.POST)
   public ModelAndView addUser1(HttpServletRequest request, HttpServletResponse response,
@ModelAttribute("namecrs") Coursedetails namecrs) {

courseService.addCoursedetails(namecrs);
ModelAndView mav = new ModelAndView("adddetailssuccess");
mav.addObject("idcrs", namecrs.getIdcrs());
mav.addObject("namecrs", namecrs.getCrs());
mav.addObject("unit1", namecrs.getUnit1());
mav.addObject("unit2", namecrs.getUnit2());
```

```
 return mav;
 }

 @RequestMapping(value = "/coursedelete", method = RequestMethod.GET)
    public ModelAndView delCourse(HttpServletRequest request, HttpServletResponse response) {
   ModelAndView mav = new ModelAndView("coursedelete");
   mav.addObject("namecr", new Coursedelete());

   return mav;
 }

 @RequestMapping(value = "/deleteCourseProcess", method = RequestMethod.POST)
    public ModelAndView addUser2(HttpServletRequest request, HttpServletResponse response,
 @ModelAttribute("namecr") Coursedelete namecr) {

 courseService.deleteCourse(namecr);
 ModelAndView mav = new ModelAndView("deletecoursesuccess");
 mav.addObject("idcr", namecr.getIdcr());
 //mav.addObject("domain", namecourse.getDomain());
 mav.addObject("namecr", namecr.getCr());
 //mav.addObject("faculty", namecourse.getFaculty());

 return mav;
 }

 @RequestMapping(value = "/assignassignment", method = RequestMethod.GET)
         public ModelAndView assignassignments(HttpServletRequest request, HttpServletResponse response) {
   ModelAndView mav = new ModelAndView("assignassignment");
   mav.addObject("assignment", new Assignment());

   return mav;
 }

 @RequestMapping(value = "/assignassignmentPrs", method = RequestMethod.POST)
```

```
    public ModelAndView addUser4(HttpServletRequest request, HttpServletResponse response,
 @ModelAttribute("assignment") Assignment assignment) {

 courseService.addAssignment(assignment);
 ModelAndView mav = new ModelAndView("addassignmentsuccess");
 mav.addObject("idcourse", assignment.getIdc());
 mav.addObject("domain", assignment.getNamec());
 mav.addObject("namecourse", assignment.getAssignnum());
 mav.addObject("faculty", assignment.getDate());

 return mav;
 }

 @RequestMapping(value = "/courselist")
 public ModelAndView displayCourses(HttpServletRequest request, Model model) {
         List<Course> coursesList = courseService.getAllCourses();
   ModelAndView mav = new ModelAndView("courselist");
   mav.addObject("courses", coursesList);
   return mav;
 }

 @RequestMapping(value = "/courselistteacher")
 public ModelAndView displayCoursesteacher(HttpServletRequest request, Model model) {
         List<Course> coursesList = courseService.getAllCourses();
   ModelAndView mav = new ModelAndView("courselistteacher");
   mav.addObject("courses", coursesList);
   return mav;
 }

 @RequestMapping(value = "/courseliststudent")
 public ModelAndView displayCoursesstudent(HttpServletRequest request, Model model) {
         List<Course> coursesList = courseService.getAllCourses();
   ModelAndView mav = new ModelAndView("courseliststudent");
   mav.addObject("courses", coursesList);
   return mav;
 }
 @RequestMapping(value = "/coursenotes")
 public ModelAndView displayCoursedetails(HttpServletRequest request, Model model) {
		List<Coursedetails> coursesList = courseService.getAllDetails();
    ModelAndView mav = new ModelAndView("coursenotes");
    mav.addObject("coursedetails", coursesList);
    return mav;
 }


 @RequestMapping(value = "/coursenotesteacher")
   public ModelAndView displayCoursedetailsteacher(HttpServletRequest request, Model
model) {
		List<Coursedetails> coursesList = courseService.getAllDetails();
    ModelAndView mav = new ModelAndView("coursenotesteacher");
    mav.addObject("coursedetails", coursesList);
    return mav;
 }
}
```



CourseDao.java

```
package jbr.springmvc.dao;

//import java.sql.Timestamp;
import java.util.List;
//import jbr.springmvc.model.ResultHistory;

public interface CourseDao {

  int addCourse(Course namecourse);
  public List<String> getAllDomains();
  public List<Course> getCourseByDomain(String domain);
  public List<Course> getAllCourses();
  public List<Coursedetails> getAllDetails();
  public Course getCourseByFaculty(String namecourse);

  int addCoursedetails(Coursedetails namecrs);

  int deleteCourse(Coursedelete namecr);

  int addAssignment(Assignment assignment);
```

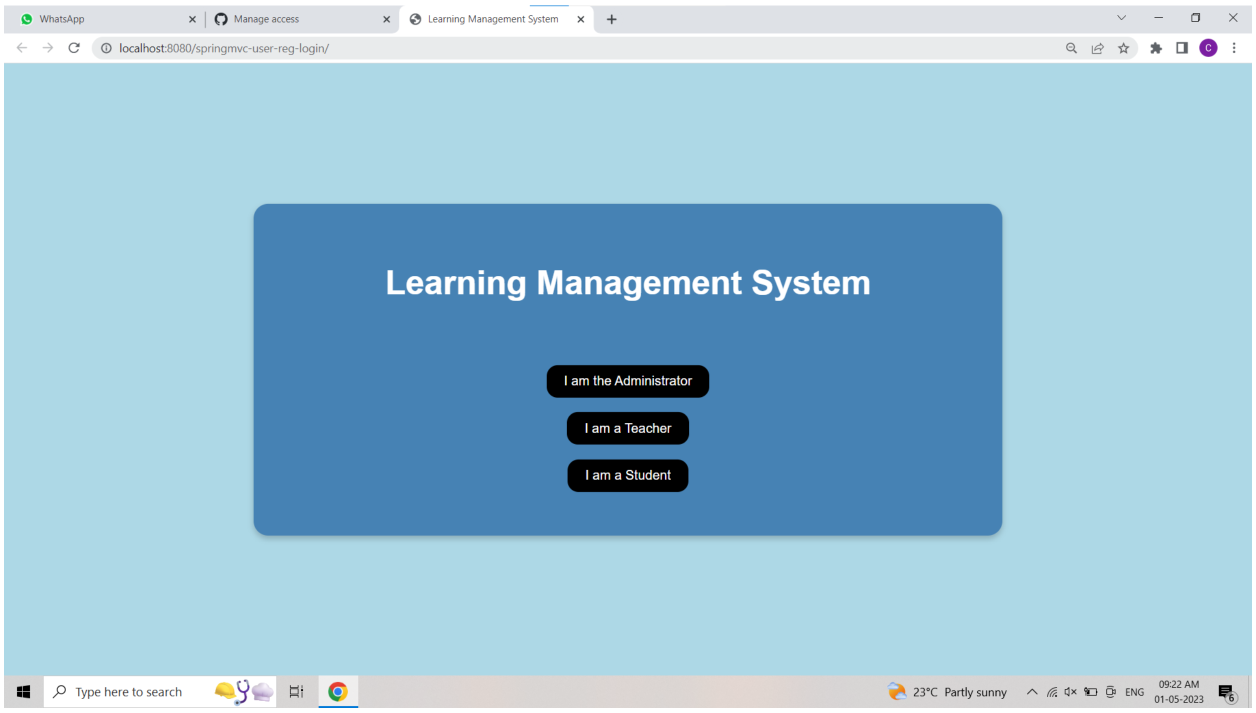
CourseDaoImpl.java

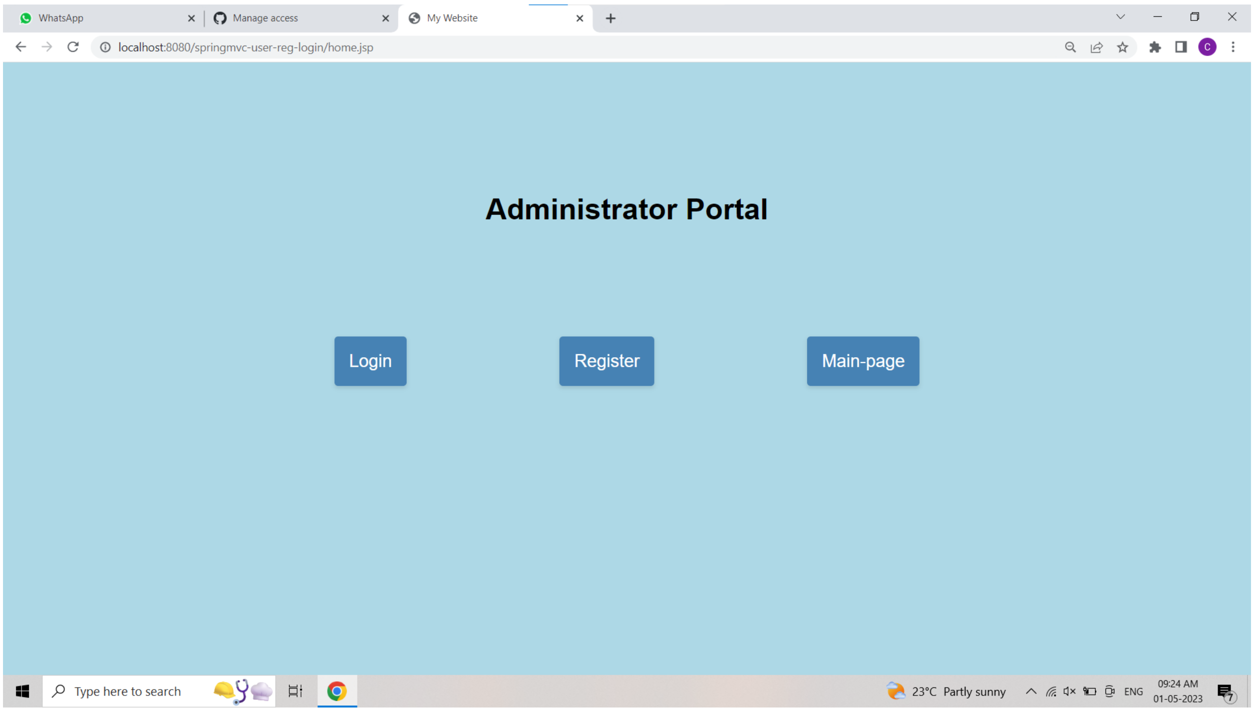
```
package jbr.springmvc.dao;

import java.sql.ResultSet;
import java.sql.SQLException;
//import java.sql.Timestamp;
import java.util.ArrayList;
import java.util.List;
import java.util.Map;

import javax.sql.DataSource;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.jdbc.core.BeanPropertyRowMapper;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.jdbc.core.RowMapper;

import jbr.springmvc.model.Assignment;
import jbr.springmvc.model.Course;
import jbr.springmvc.model.Coursedelete;
import jbr.springmvc.model.Coursedetails;
//import jbr.springmvc.model.ResultHistory;


public class CourseDaoImpl implements CourseDao
{

  @Autowired
  DataSource datasource;
  @Autowired
  JdbcTemplate jdbcTemplate;

  //addcourse
  public int addCourse(Course namecourse)
  {
        String sql = "insert into courses values(?,?,?,?,?,?,?)";
         return jdbcTemplate.update(sql, new Object[] { namecourse.getIdcourse(),
namecourse.getAddedby(),namecourse.getCourse(), namecourse.getFaculty(),
                 namecourse.getDuration(),                          namecourse.getCertificate(),
namecourse.getDomain()});
  }


  public int addCoursedetails(Coursedetails namecrs)
  {
        String sql = "insert into coursedetails values(?,?,?,?)";
   return jdbcTemplate.update(sql, new Object[] { namecrs.getIdcrs(), namecrs.getCrs(),
namecrs.getUnit1(),
                 namecrs.getUnit2()});
  }




  public int deleteCourse(Coursedelete namecr)
  {
        String sql = "delete from course WHERE namecourse= ?;";
 return jdbcTemplate.update(sql, new Object[] { namecr.getCr()});
  }


  public int addAssignment(Assignment assignment)
  {
        String sql = "insert into assignments values(?,?,?,?)";
             return    jdbcTemplate.update(sql,    new    Object[]    {    assignment.getIdc(),
assignment.getNamec(),assignment.getAssignnum(),
                 assignment.getDate()});
  }


//getalldomain
  public List<String> getAllDomains()
  {
         String sql = "SELECT DISTINCT domain FROM course";
         List<Map<String, Object>> rows = jdbcTemplate.queryForList(sql);
```

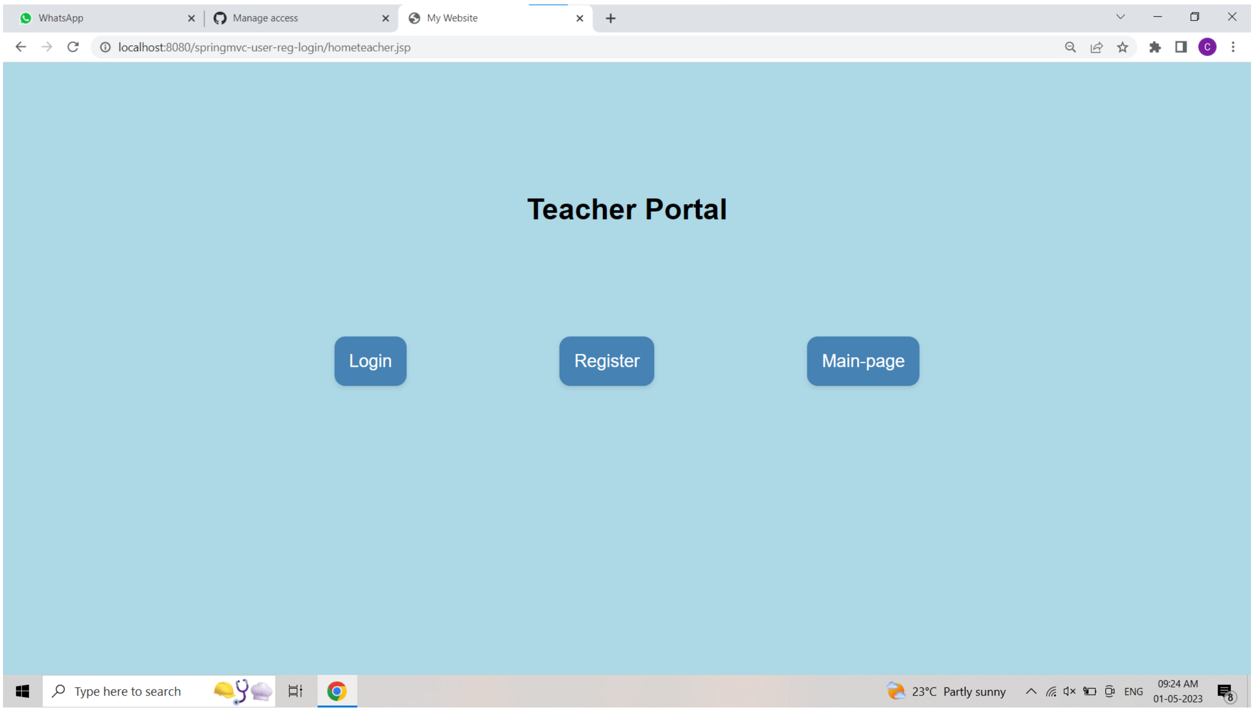
```
            List<String> domains = new ArrayList<String>();
            for (Map<String, Object> row : rows) {
               String domain = (String) row.get("domain");
               domains.add(domain);
            }
            return domains;
         }



// getcoursebydomain  public List<Course> getCourseByDomain(String namecourse);

 public List<Course> getCourseByDomain(String domain)
 {
      String sql = "SELECT DISTINCT * FROM course WHERE domain = ? ORDER BY RAND()
LIMIT 5";
     return jdbcTemplate.query(sql, new Object[]{domain}, new RowMapper<Course>() {
                    @Override
                    public Course mapRow(ResultSet rs, int rowNum) throws SQLException {
                        Course namecourse = new Course();
                        namecourse.setIdcourse(rs.getString("idcourse"));
                        namecourse.setCourse(rs.getString("namecourse"));
                        namecourse.setFaculty(rs.getString("faculty"));
                        namecourse.setDuration(rs.getInt("duration"));
                        namecourse.setCertificate(rs.getBoolean("certificate"));
                        namecourse.setDomain(rs.getString("domain"));
                        return namecourse;
                     }
         });
 }



 public List<Course> getAllCourses()

 {
     String sql = "SELECT idcourse, namecourse, faculty, duration, certificate, domain FROM
course ORDER BY domain";
                         List<Course>     courses     =     jdbcTemplate.query(sql,     new
BeanPropertyRowMapper<Course>(Course.class));
    return courses;
 }




 public List<Coursedetails> getAllDetails()
 {
         String sql = "SELECT idcrs, namecrs, unit1, unit2 FROM coursedetails ";
                  List<Coursedetails>    coursedetails    =    jdbcTemplate.query(sql,    new
BeanPropertyRowMapper<Coursedetails>(Coursedetails.class));
    return coursedetails;



 }



 public Course getCourseByFaculty(String course)
 {
         String sql = "SELECT DISTINCT * FROM course WHERE faculty = ?";
                return    jdbcTemplate.queryForObject(sql,    new    Object[]{course},    new
RowMapper<Course>() {
                 @Override
                 public Course mapRow(ResultSet rs, int rowNum) throws SQLException {
                     Course namecourse = new Course();
                     namecourse.setIdcourse(rs.getString("idcourse"));
                     namecourse.setCourse(rs.getString("namecourse"));
                     namecourse.setFaculty(rs.getString("faculty"));
                     namecourse.setDuration(rs.getInt("duration"));
                     namecourse.setCertificate(rs.getBoolean("certificate"));
                     namecourse.setDomain(rs.getString("domain"));
                     return namecourse;
                  }
        });
 }
}
```

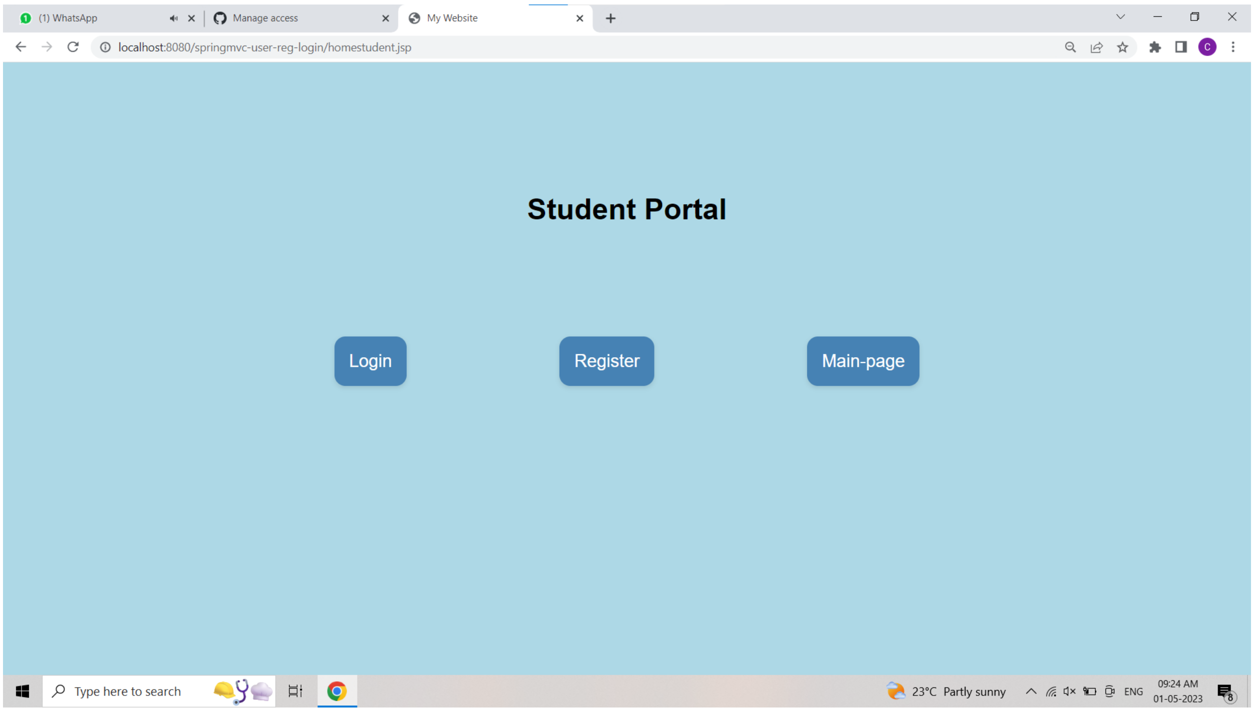
6. Results(Screenshots) And Conclusion

Main Page:

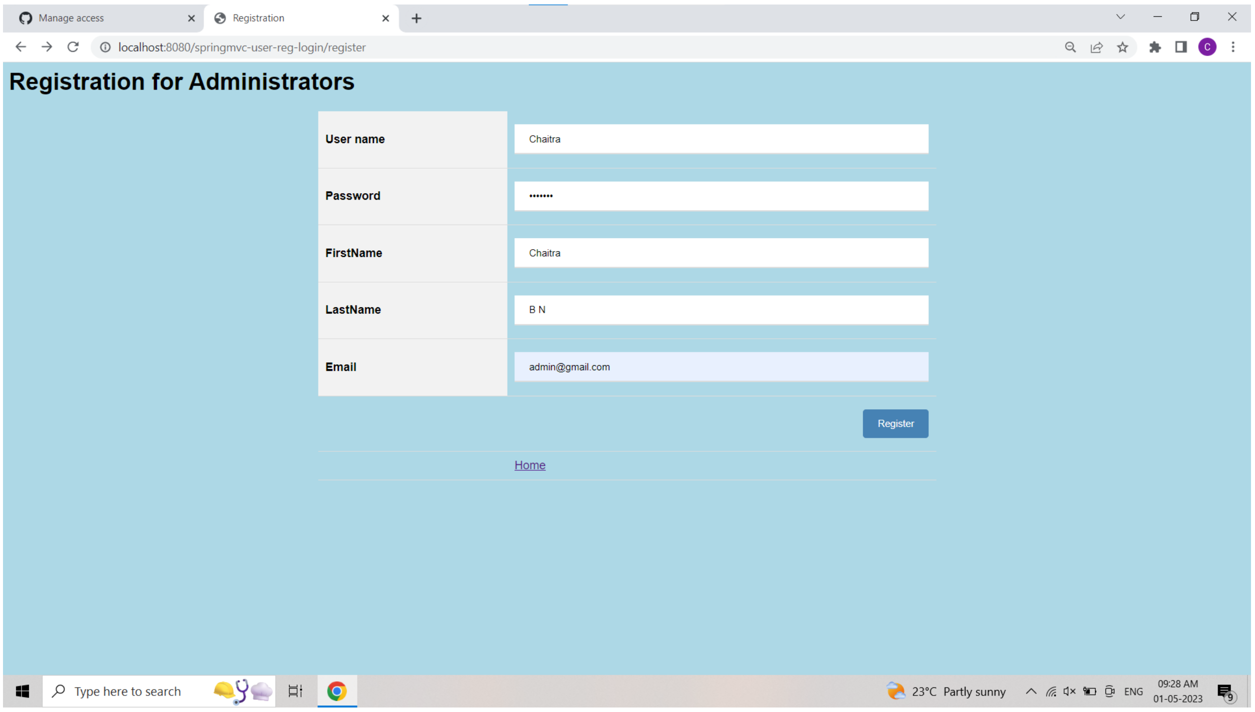



WhatsApp
Manage access
My Website
localhost:8080/springmvc-user-reg-login/home.jsp
Administrator Portal
Login
Register
Main-page
Type here to search
23°C Partly sunny
ENG
09:24 AM
01-05-2023


WhatsApp
Manage access
My Website
localhost:8080/springmvc-user-reg-login/hometeacher.jsp
Teacher Portal
Login
Register
Main-page
Type here to search
23°C Partly sunny
ENG
09:24 AM
01-05-2023



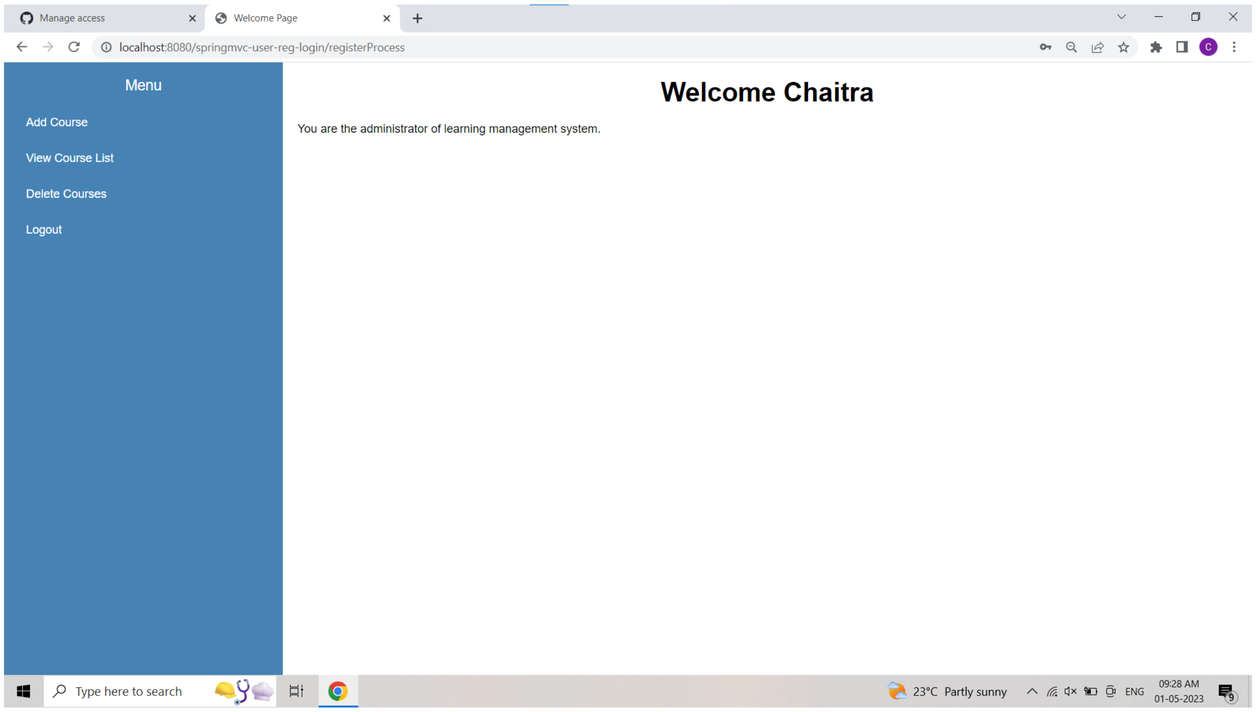
A. Registration of admin, student ,teacher.



Registration for Administrators

User name	Chaitra
Password	•••••••
FirstName	Chaitra
LastName	B N
Email	admin@gmail.com

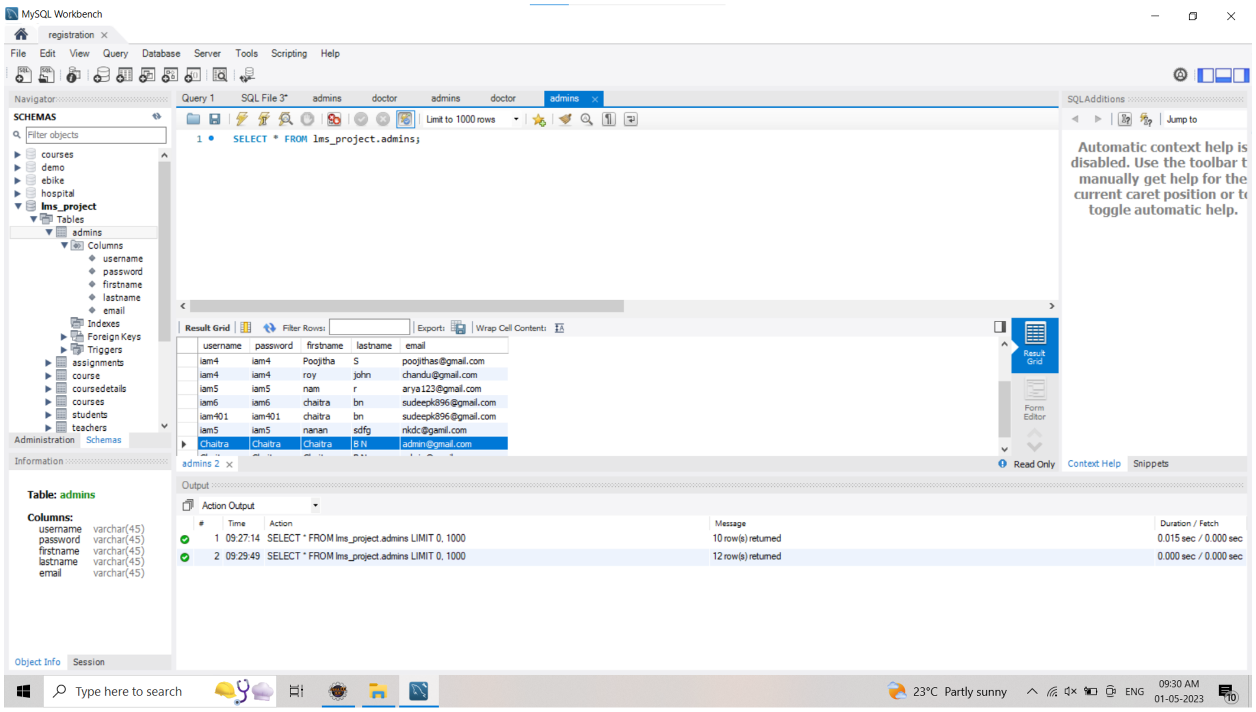
after registering



Welcome Chaitra

You are the administrator of learning management system.

added to database



```
SELECT * FROM lms_project.admins;
```

username	password	firstname	lastname	email
iam4	iam4	Poojitha	S	poojithas@gmail.com
iam4	iam4	roy	john	chandu@gmail.com
iam5	iam5	nam	r	arya123@gmail.com
iam6	iam6	chaitra	bn	sudeepk896@gmail.com
iam401	iam401	chaitra	bn	sudeepk896@gmail.com
iam5	iam5	nanan	sdfg	nkdc@gamil.com
Chaitra	Chaitra	Chaitra	B N	admin@gmail.com

Automatic context help is disabled. Use the toolbar t manually get help for the current caret position or t toggle automatic help.

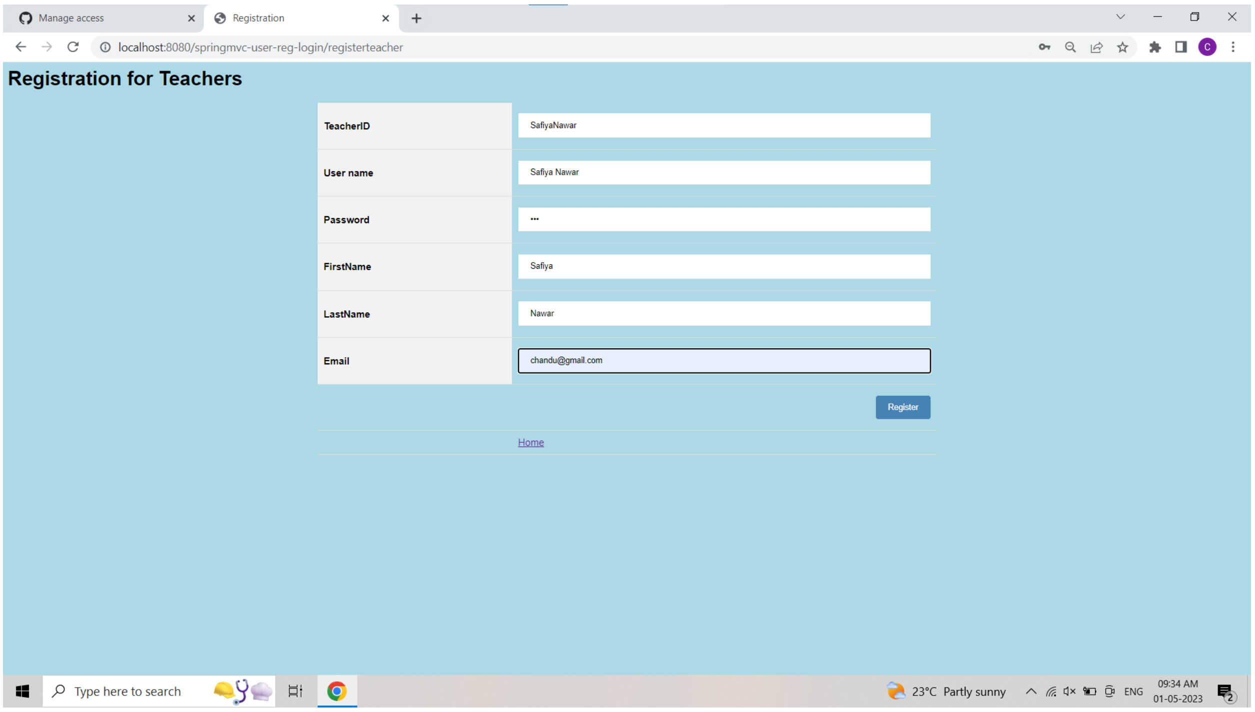
Table: admins

Columns:

username	varchar(45)
password	varchar(45)
firstname	varchar(45)
lastname	varchar(45)
email	varchar(45)

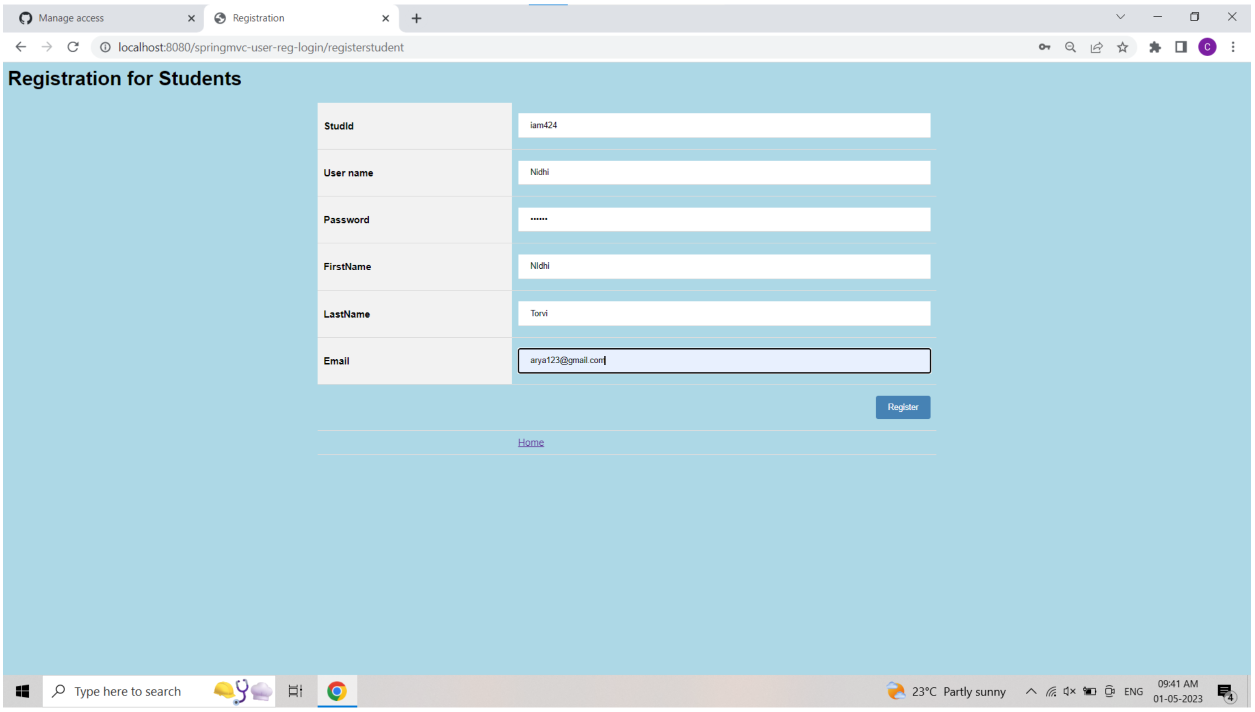
```
#  Time      Action                                        Message              Duration / Fetch
1  09:27:14  SELECT * FROM lms_project.admins LIMIT 0, 1000  10 row(s) returned  0.015 sec / 0.000 sec
2  09:29:49  SELECT * FROM lms_project.admins LIMIT 0, 1000  12 row(s) returned  0.000 sec / 0.000 sec
```

similarly student and teacher also


localhost:8080/springmvc-user-reg-login/registerteacher

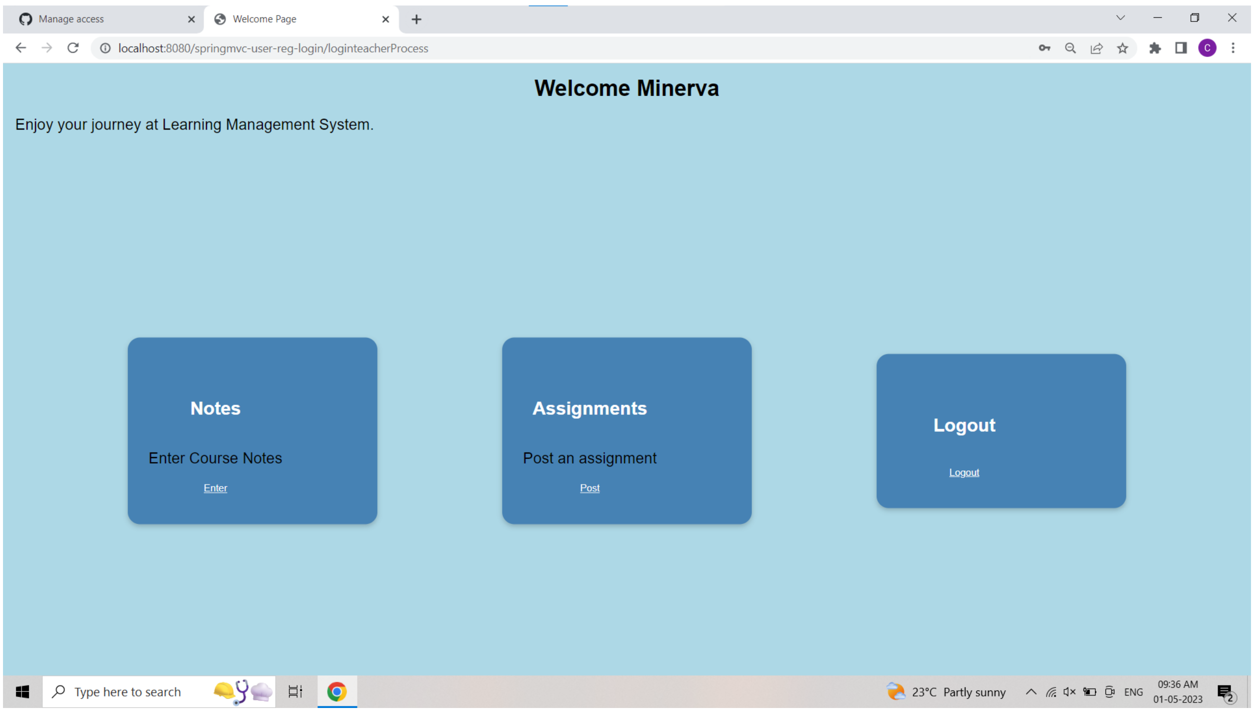
Registration for Teachers

TeacherID	SafiyaNawar
User name	Safiya Nawar
Password	•••
FirstName	Safiya
LastName	Nawar
Email	chandu@gmail.com


localhost:8080/springmvc-user-reg-login/registerstudent

Registration for Students

StudId	iam424
User name	Nidhi
Password	••••••
FirstName	Nidhi
LastName	Torvi
Email	arya123@gmail.com



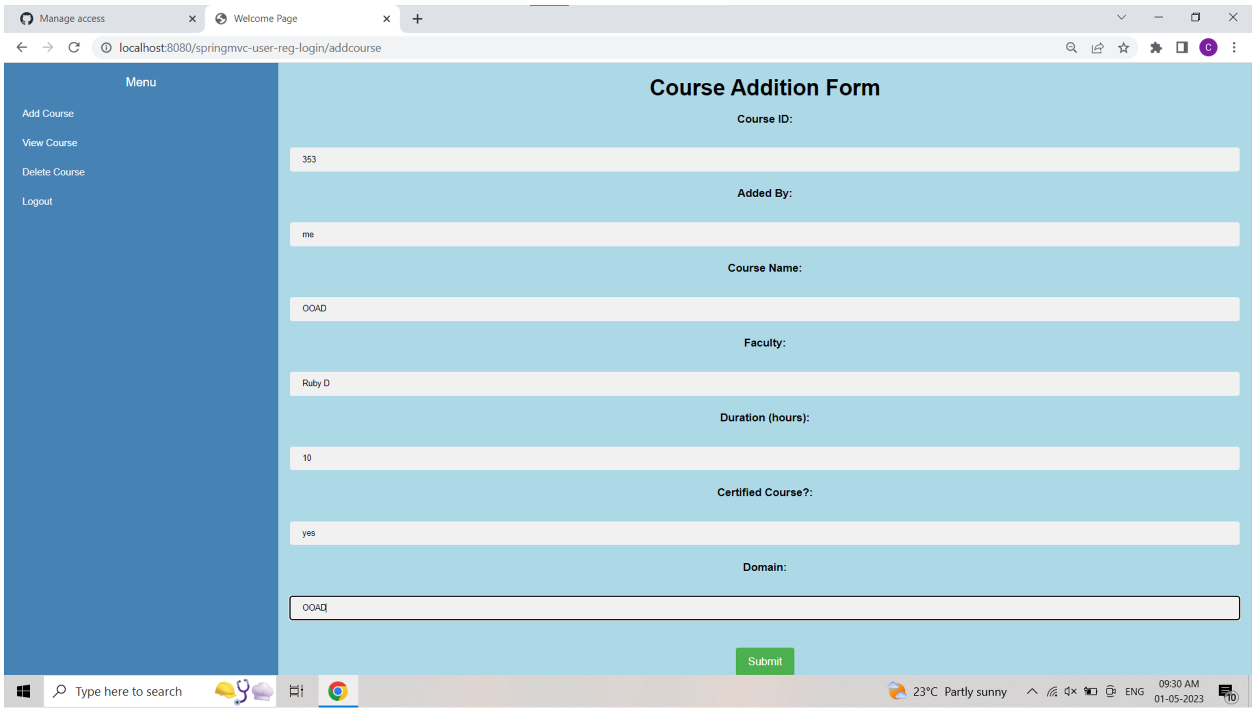
Manage access | Welcome Page

localhost:8080/springmvc-user-reg-login/loginteacherProcess

Welcome Minerva

Enjoy your journey at Learning Management System.

B. Adding courses by admin.



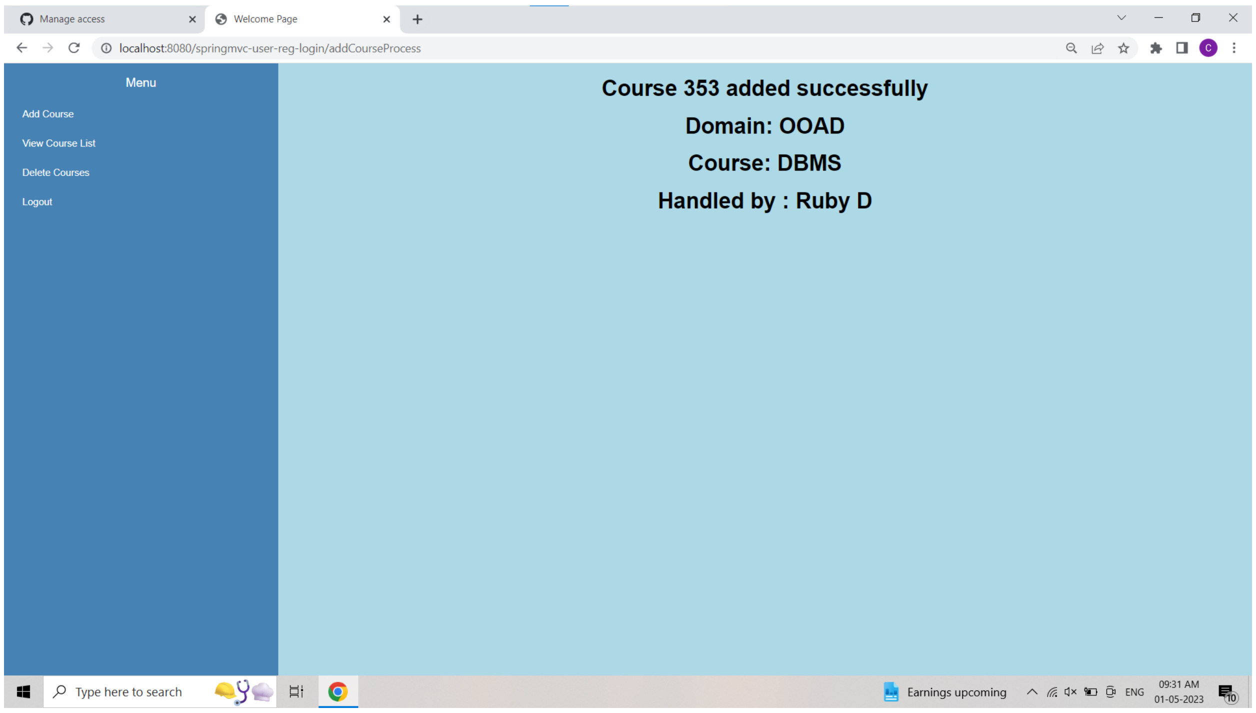
Manage access | Welcome Page

localhost:8080/springmvc-user-reg-login/addcourse

Course Addition Form

Course ID:	353
Added By:	me
Course Name:	OOAD
Faculty:	Ruby D
Duration (hours):	10
Certified Course?:	yes
Domain:	OOAD

after adding course:



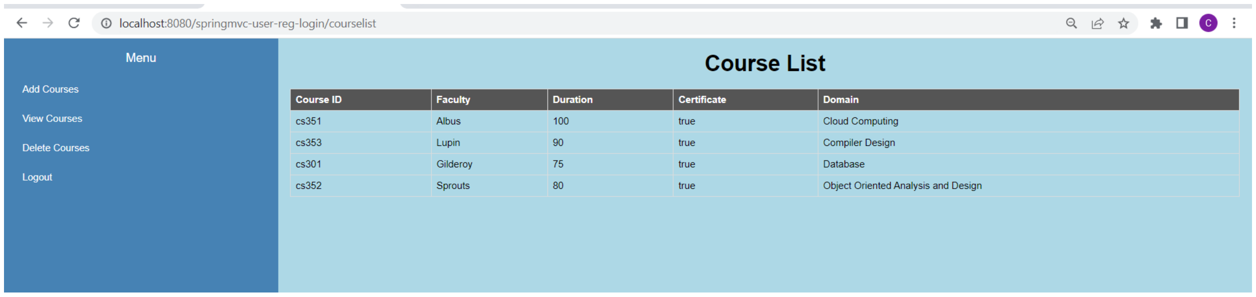
Course 353 added successfully

Domain: OOAD

Course: DBMS

Handled by : Ruby D

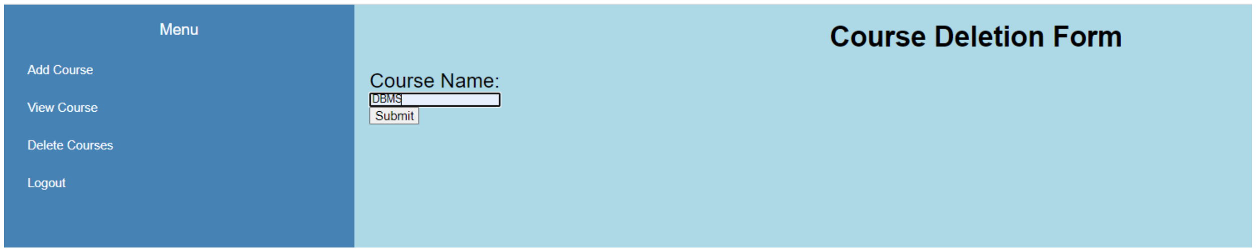
view course list by admin



Course List

Course ID	Faculty	Duration	Certificate	Domain
cs351	Albus	100	true	Cloud Computing
cs353	Lupin	90	true	Compiler Design
cs301	Gilderoy	75	true	Database
cs352	Sprouts	80	true	Object Oriented Analysis and Design

delete course list by admin

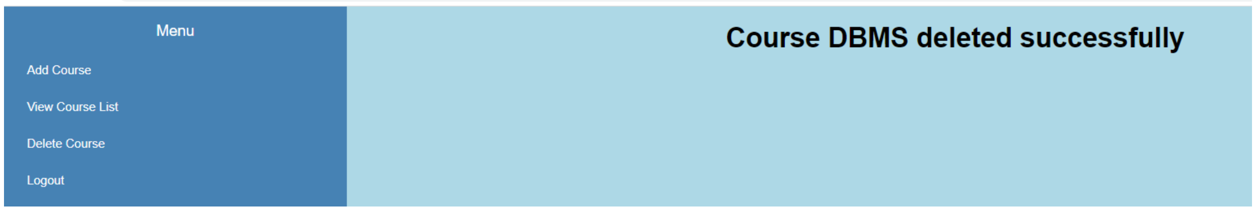


Course Deletion Form

Course Name: DBMS

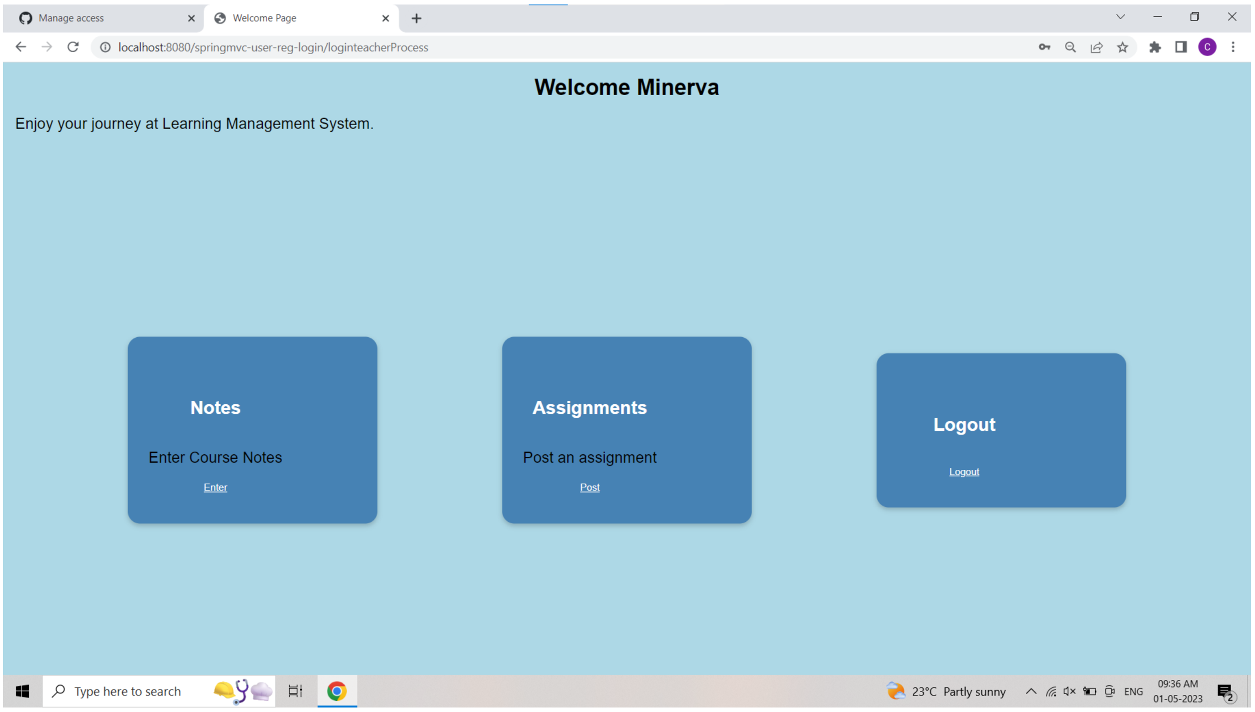
Submit

after deleting



Course DBMS deleted successfully

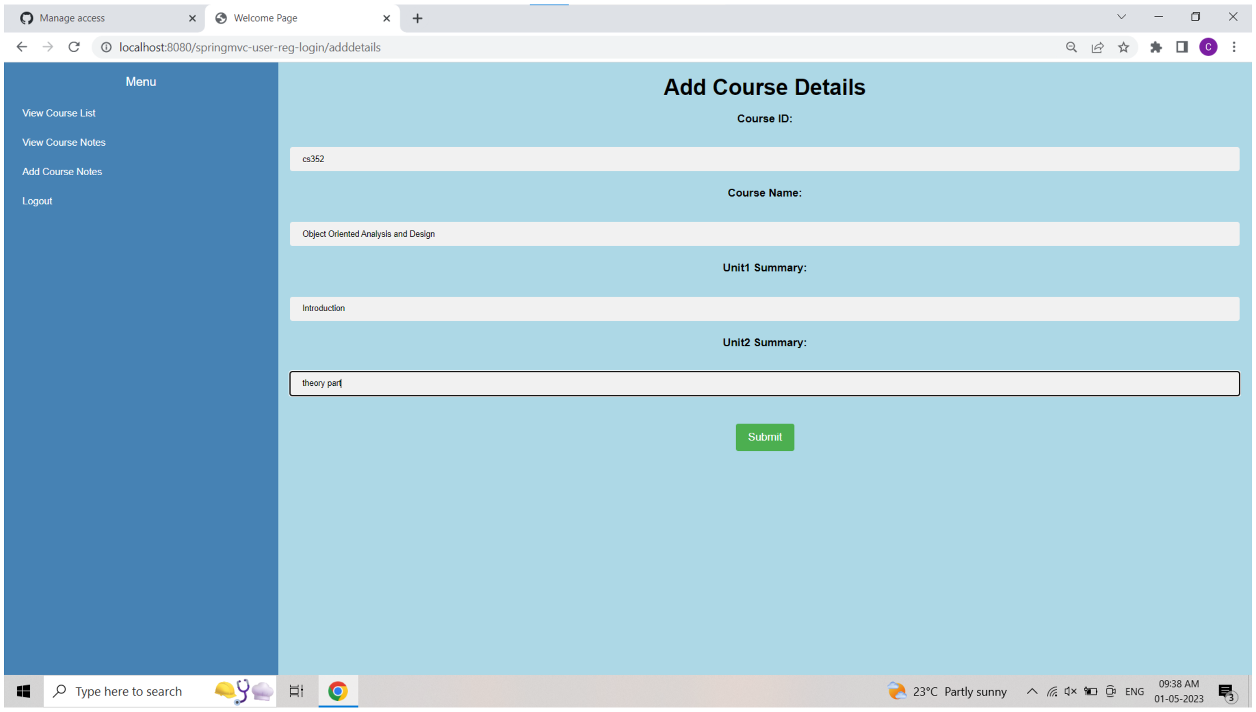
C. Adding course details(notes) by teacher.



Welcome Minerva

Enjoy your journey at Learning Management System.

select notes here



Add Course Details

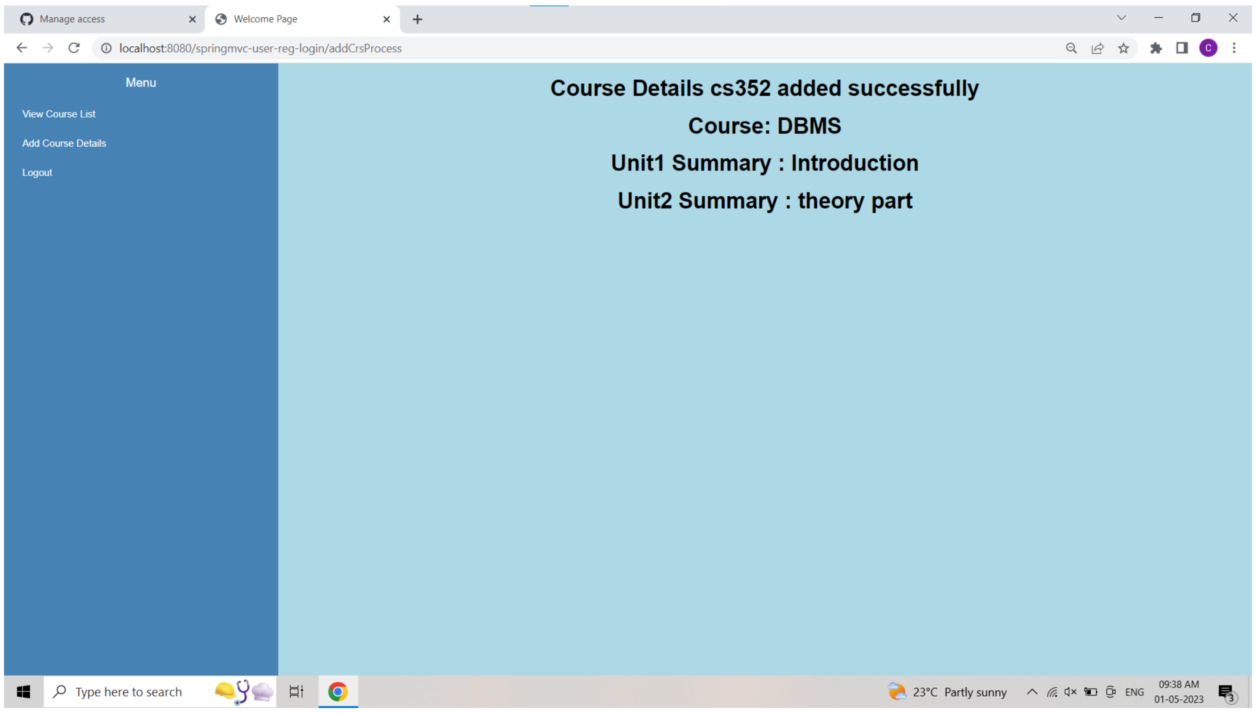
Course ID: cs352

Course Name: Object Oriented Analysis and Design

Unit1 Summary: Introduction

Unit2 Summary: theory part

after adding



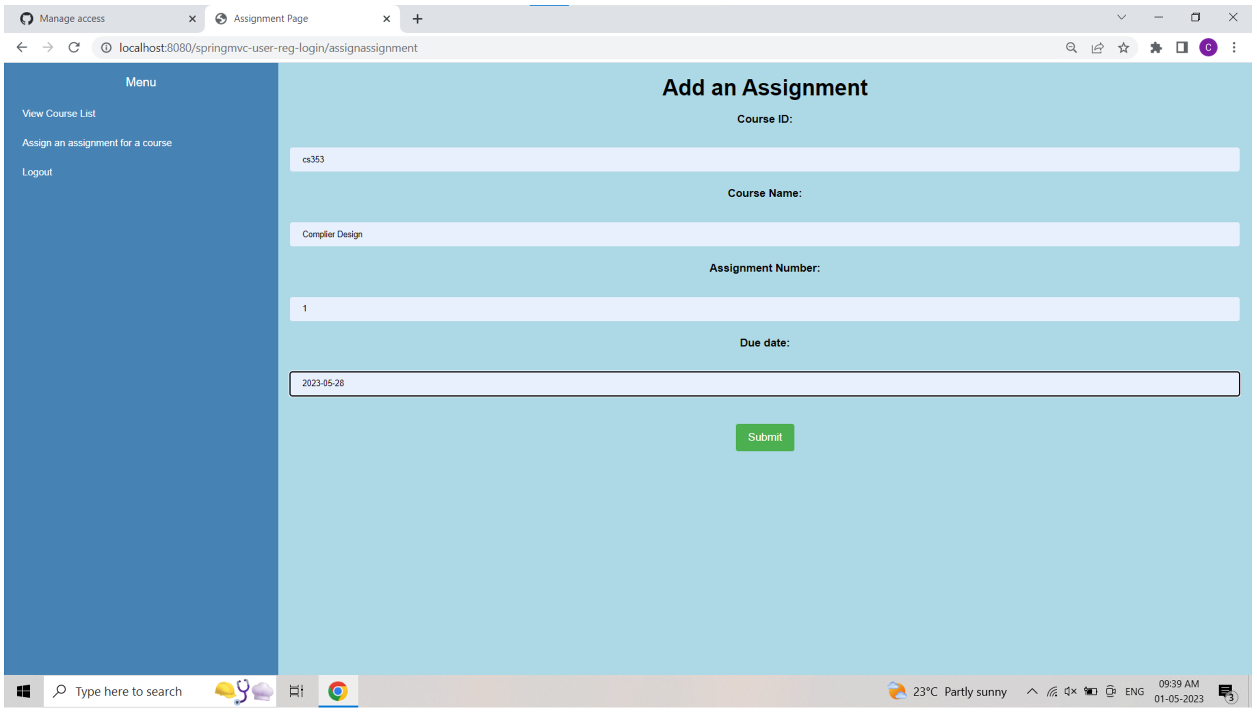
Course Details cs352 added successfully

Course: DBMS

Unit1 Summary : Introduction

Unit2 Summary : theory part

D. Adding assignments by teacher.



Add an Assignment

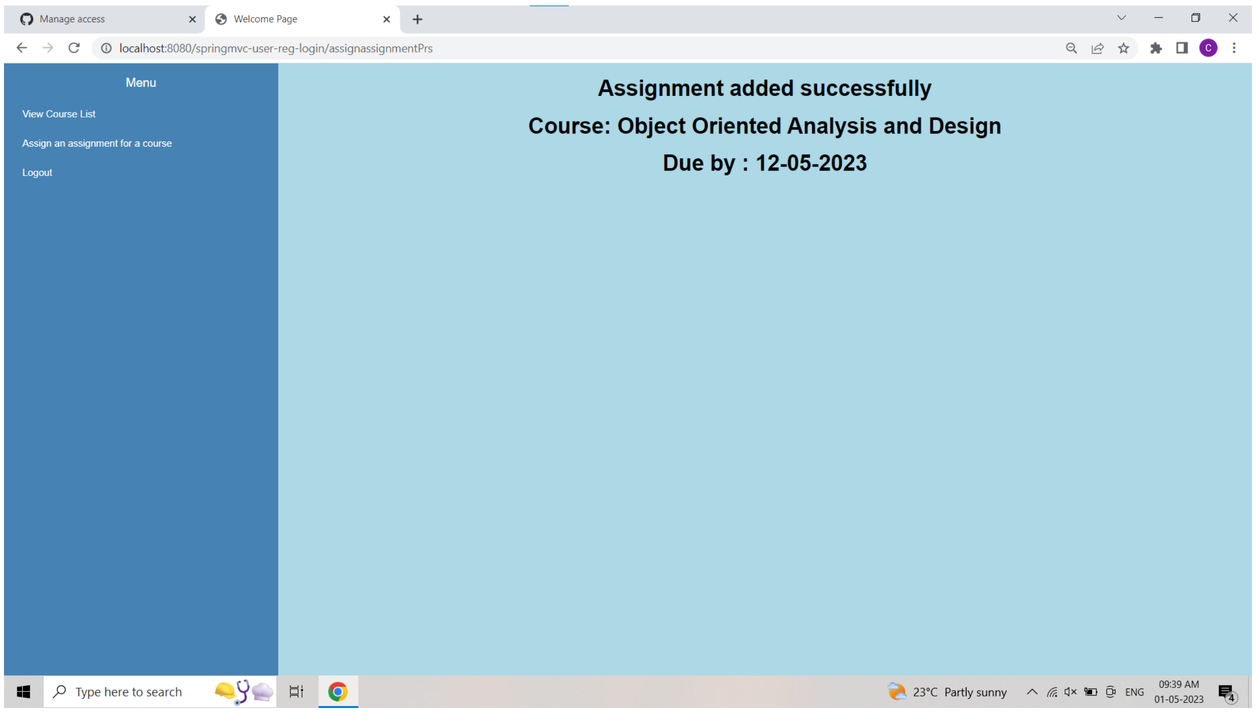
Course ID: cs353

Course Name: Complier Design

Assignment Number: 1

Due date: 2023-05-28

after adding

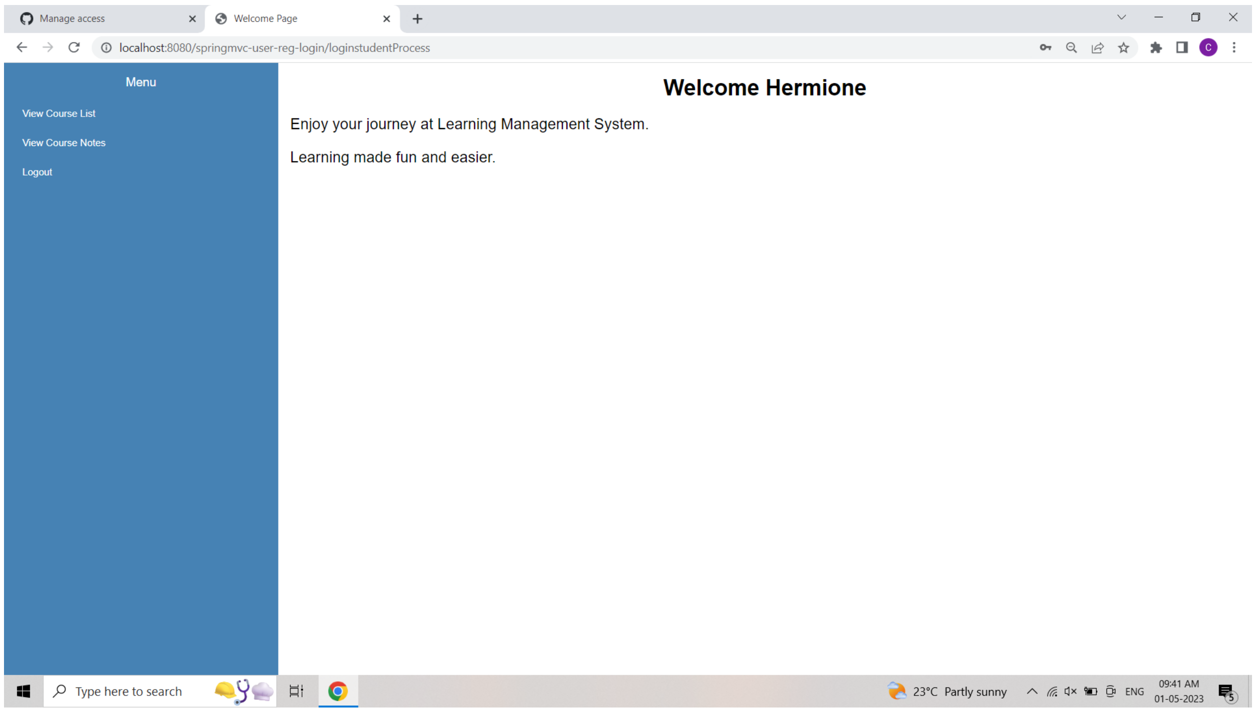


Assignment added successfully

Course: Object Oriented Analysis and Design

Due by : 12-05-2023

E. View of all notes of course and assignment by student.

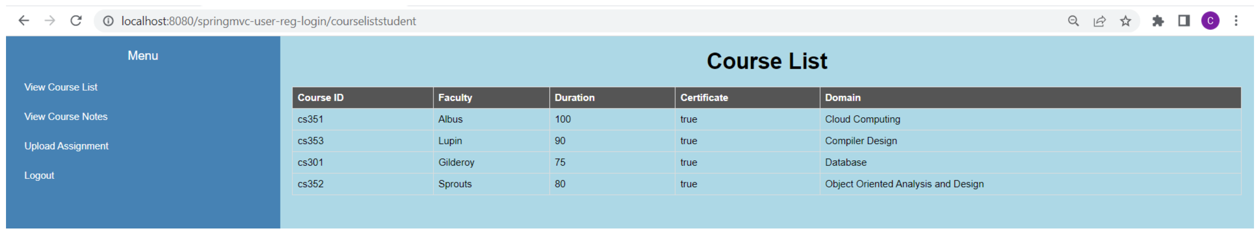


Welcome Hermione

Enjoy your journey at Learning Management System.

Learning made fun and easier.

view course details:

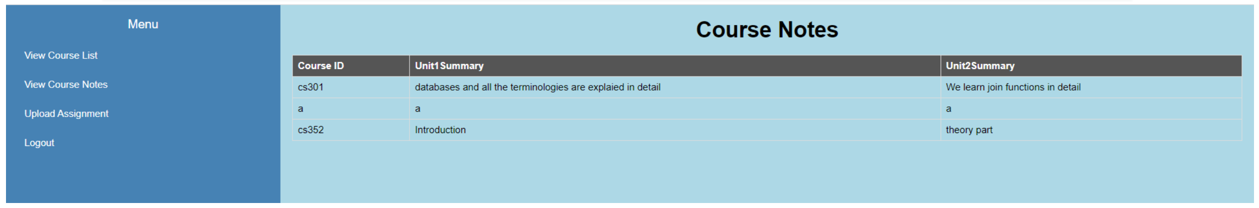


localhost:8080/springmvc-user-reg-login/courseliststudent

Course List

Course ID	Faculty	Duration	Certificate	Domain
cs351	Albus	100	true	Cloud Computing
cs353	Lupin	90	true	Compiler Design
cs301	Gilderoy	75	true	Database
cs352	Sprouts	80	true	Object Oriented Analysis and Design

view notes:



Course Notes

Course ID	Unit1Summary	Unit2Summary
cs301	databases and all the terminologies are explaied in detail	We learn join functions in detail
a	a	a
cs352	Introduction	theory part