



UE21CS352B – Object Oriented Analysis & Design using Java

Mini Project Report

“Course Management System”

Submitted by:

KIRAN MADEV SUDHAM

PES1UG21CS280

KALLES ACHAR KG

PES1UG21CS266

MD IBRAHIM

PES1UG22CS817

MOHAMMED FAISAL YENKTAPURI

PES1UG22CS818

6TH Semester E Section

Prof. BHARGAVI MOKASHI

January - May 2024

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

FACULTY OF ENGINEERING

PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)

100ft Ring Road, Bengaluru – 560 085, Karnataka, India

INTRODUCTION

Introduction: In schools and colleges, keeping track of classes, assignments, grades, and student details can be a bit of a hassle. The Course Management System project aims to make things easier by creating a computer program that handles all these tasks.

It has 3 major modules- Admin, Student, and Instructor. Admin can create accounts for students and instructors and also assign instructors to a course and assign students to a course. Students and Instructors can log in to their respective accounts and have different features like submit homework, view grades and post homework, and grade homework. Functionalities provided by different modules: Admin -Create accounts for student and instructor -Add course -Assign instructor to the course and course to student -Update, View Details Instructor -Post notes, assignment -Grade assignment Student -Submit assignment -View notes -View Grades -View Personal Info

The goals of this project are pretty straightforward to make a program that lets administrators create accounts for students and teachers, add classes, and assign teachers to those classes. Allow teachers to upload class materials like notes and assignments, grade assignments, and give feedback. Let students submit assignments, access class materials, see their grades, and update their personal info. Make sure the program is easy to use for everyone, with a simple layout and clear instructions. Keep everything secure, so only the right people can access student info and grades. Make sure the program can handle lots of users and assignments without slowing down.

With the Course Management System, our goal is to streamline academic processes, simplify study routines, and propel students forward on their educational journey.

PROBLEM STATEMENT

In educational institutions, managing courses, assignments, grades, and user accounts efficiently is essential for ensuring a smooth learning experience. However, traditional methods of course management often involve manual processes that are time-consuming, error-prone, and lack scalability. To address these challenges, there is a need for a comprehensive software solution that automates course management tasks and provides a user-friendly interface for administrators, instructors, and students.

The Course Management System project aims to develop a robust and user-friendly platform for managing courses, assignments, grades, and user accounts within an educational institution. The system should cater to the diverse needs of administrators, instructors, and students, providing functionalities such as account creation, course enrollment, assignment submission, grading, and performance tracking.

Key objectives of the project include:

Efficient Course Management: Develop functionalities for creating and managing courses, assigning instructors to courses, and enrolling students in courses. The system should support the addition, modification, and deletion of courses as per institutional requirements.

User Account Management: Implement features for creating user accounts for administrators, instructors, and students. Administrators should have the authority to manage user accounts, reset passwords, and update user information.

Assignment Management: Enable instructors to post course materials such as notes and assignments, grade student assignments, and provide feedback. Students should be able to submit assignments, view course materials, and check their grades.

User-Friendly Interface: Design an intuitive and user-friendly interface for administrators, instructors, and students, ensuring ease of navigation and accessibility of features. The interface should be responsive and compatible with various devices and screen sizes.

Security and Authentication: Implement robust security measures to safeguard user data and prevent unauthorized access. Use encryption techniques for sensitive information such as passwords and ensure secure authentication mechanisms for user logins.

Scalability and Performance: Design the system to handle a large volume of users, courses, and assignments efficiently. Optimize performance by implementing caching mechanisms, database indexing, and load balancing strategies.

SYSTEM OVERVIEW

The system consists of three major modules: Admin, Instructor, and Student. Each module is designed to cater to specific user roles and responsibilities, offering a range of functionalities to facilitate efficient course management.

Admin Module: This module empowers administrators with the ability to manage user accounts, courses, and assignments. Admins can create accounts for students and instructors, add new courses, assign instructors to courses, and enroll students in courses.

Instructor Module: Instructors have access to features such as posting course materials (notes, assignments), grading student assignments, and viewing course-related information. This module aims to streamline the teaching and assessment process for instructors.

Student Module: Students can utilize this module to submit assignments, access course materials (notes), check grades, and view their personal information. The module enhances the learning experience for students by providing easy access to course-related resources and performance metrics.

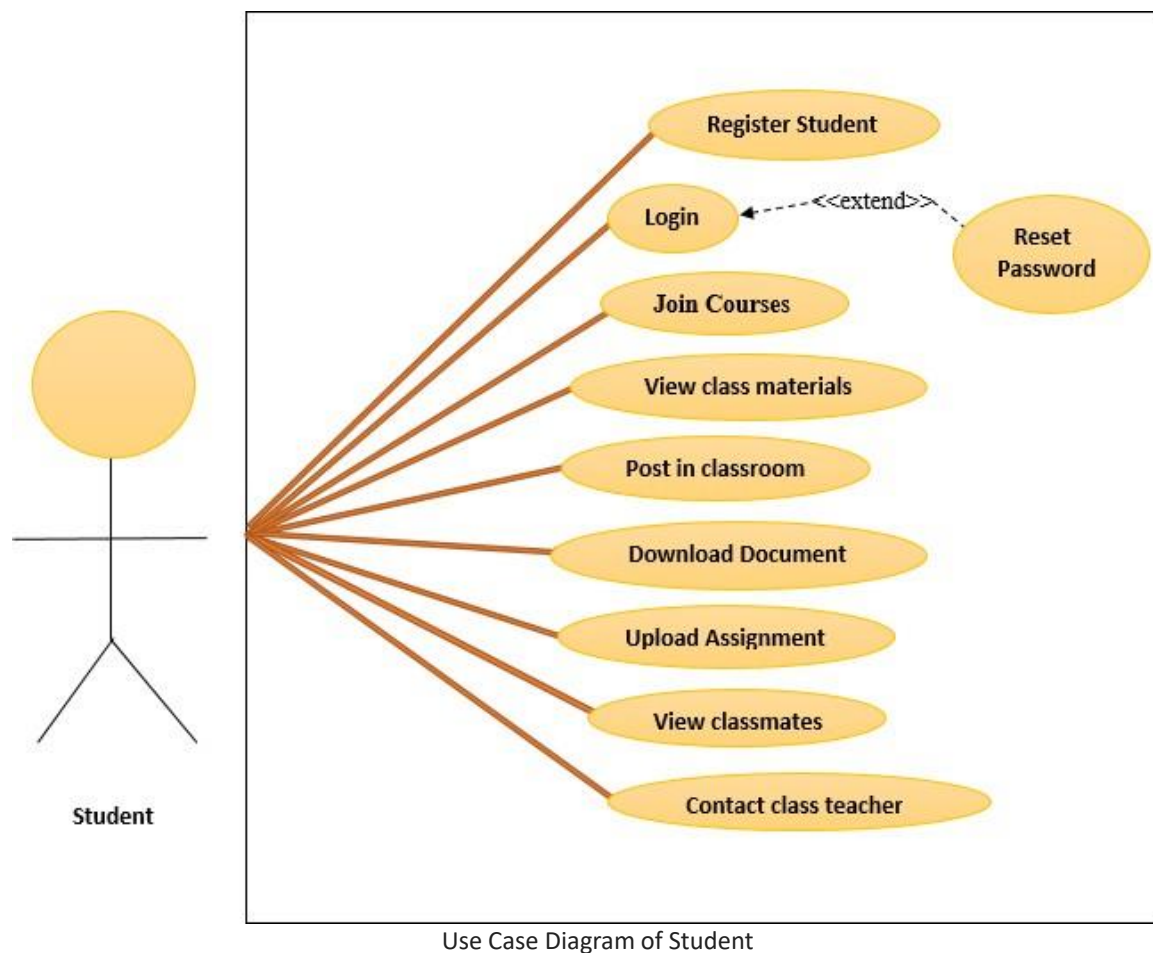
SYSTEM DESIGN

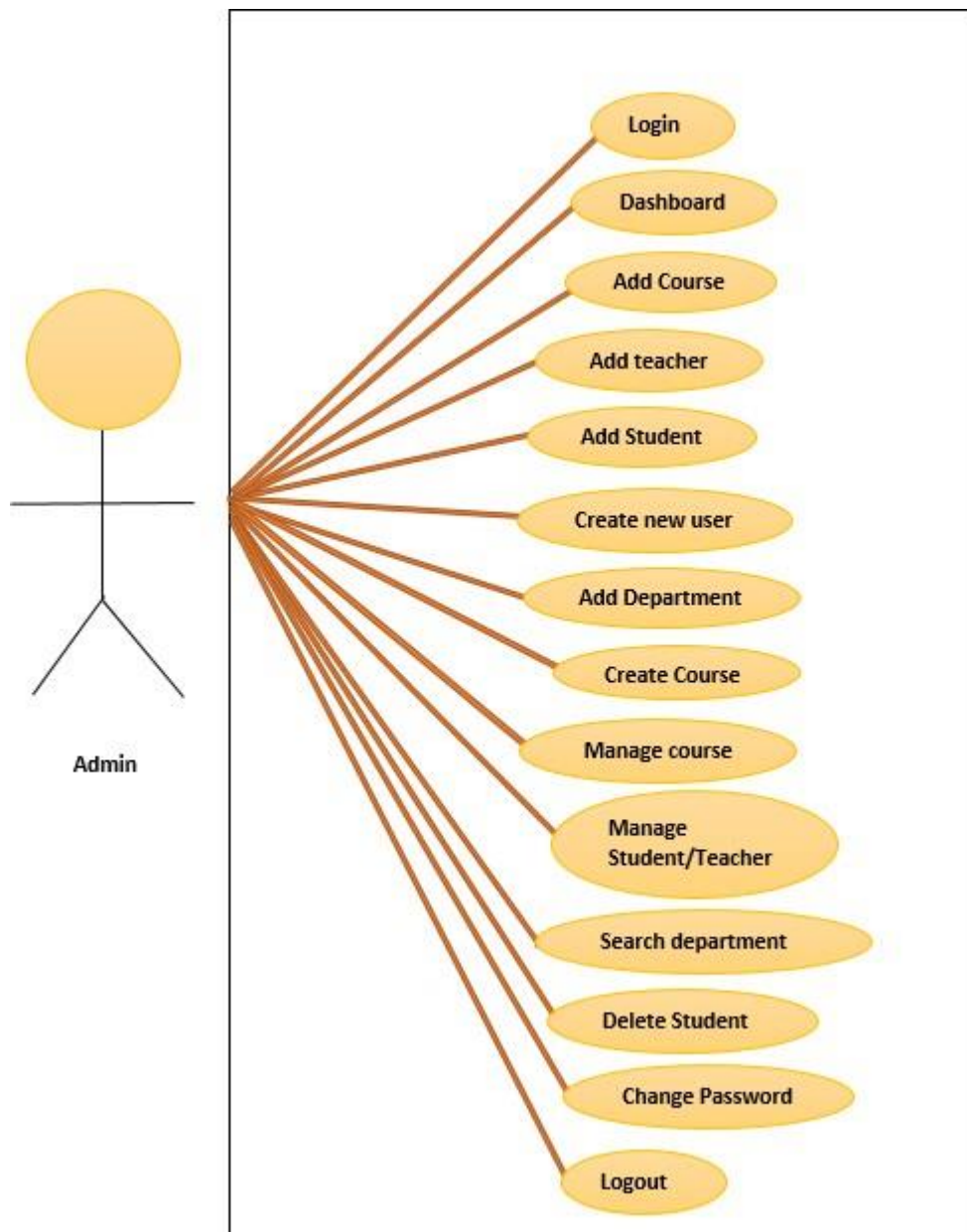
Use Case Diagram

The use case diagram provides a visual representation of the interactions between different actors (Admin, Instructor, Student) and the system. It outlines the various actions that each user role can perform within the system, helping to identify the system's functional requirements.

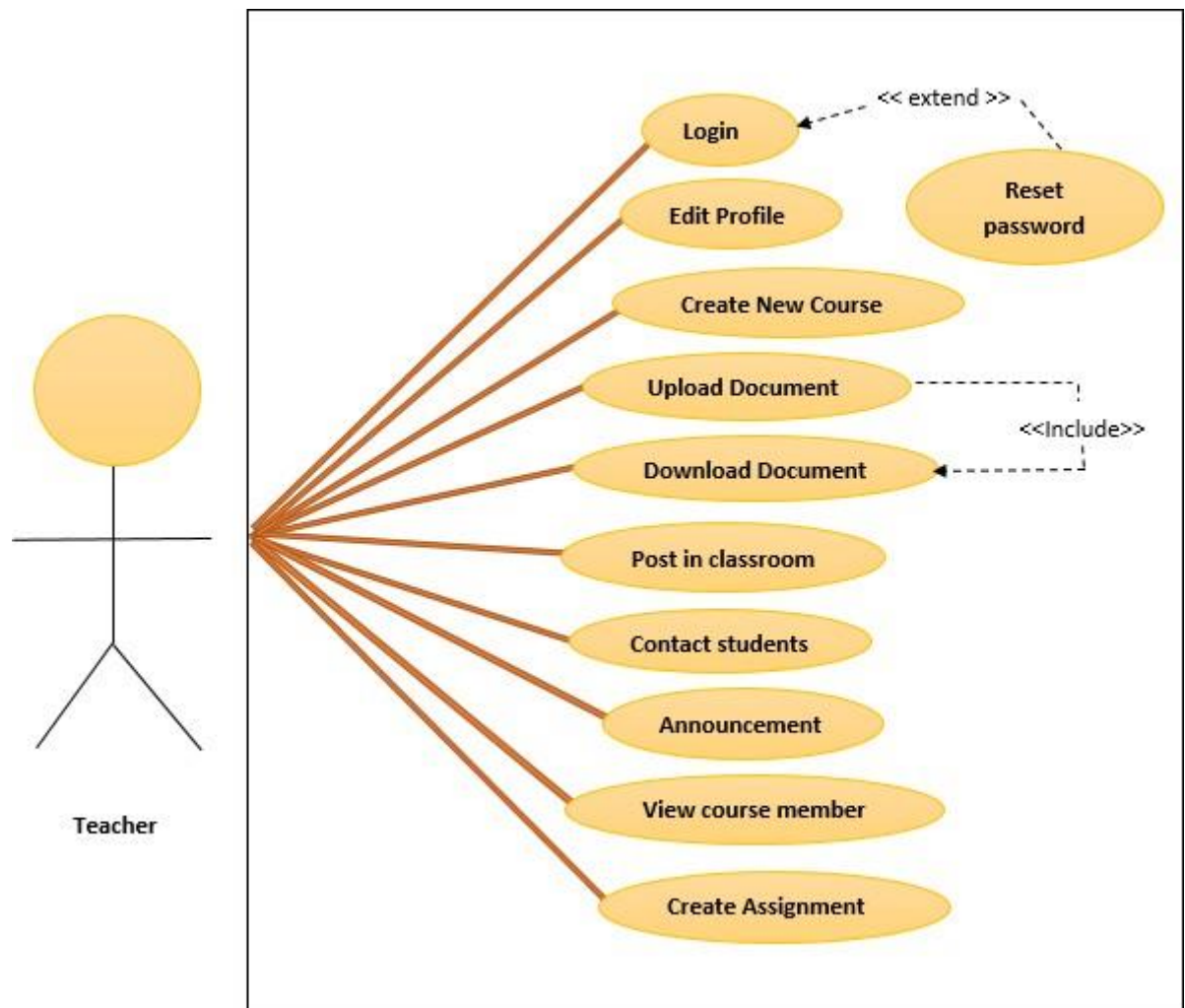
We have three actors in my projects, they are:

- Admin
- Student/User
- Teacher





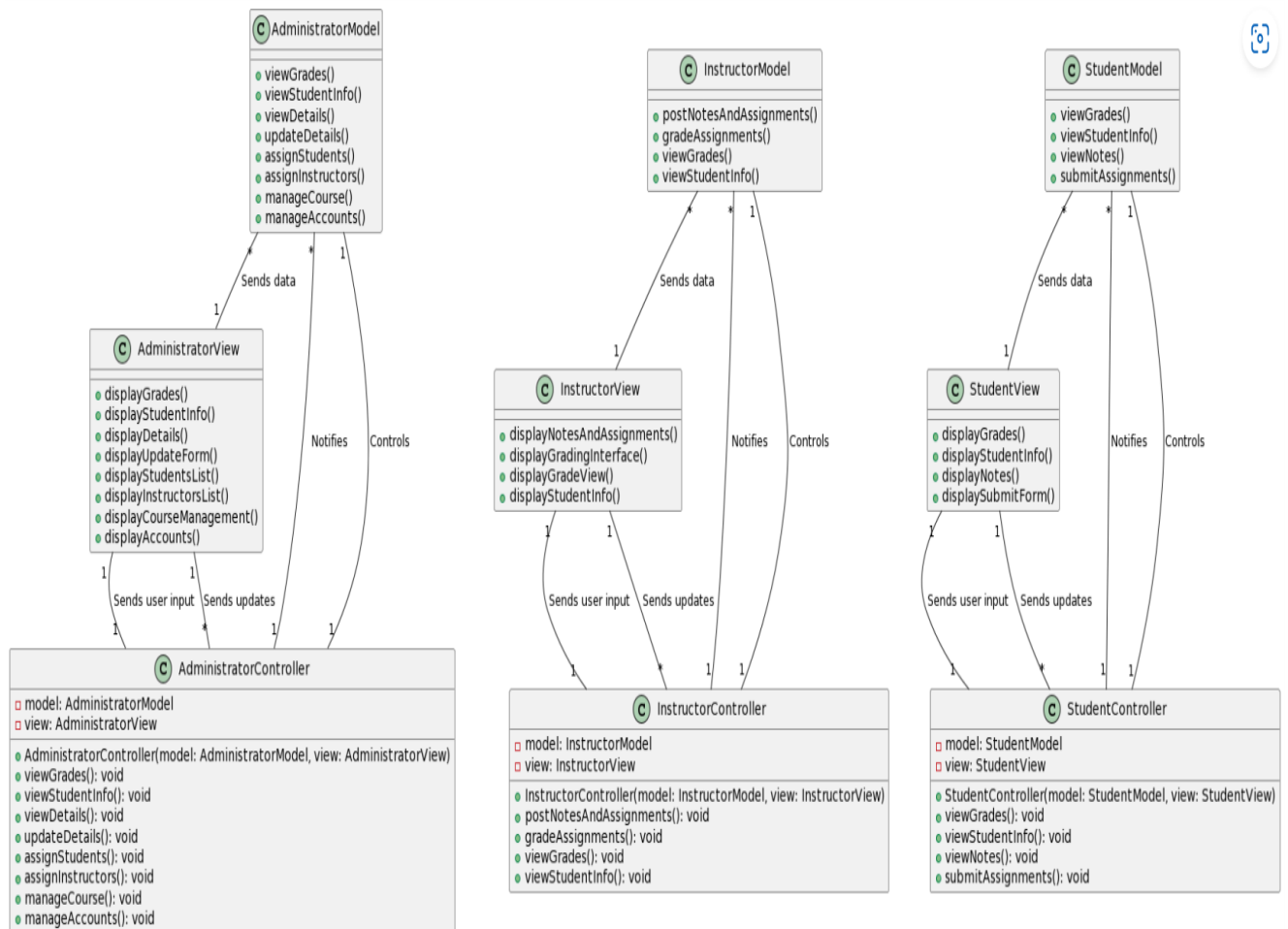
Use Case Diagram of Admin



Use Case Diagram of Teacher

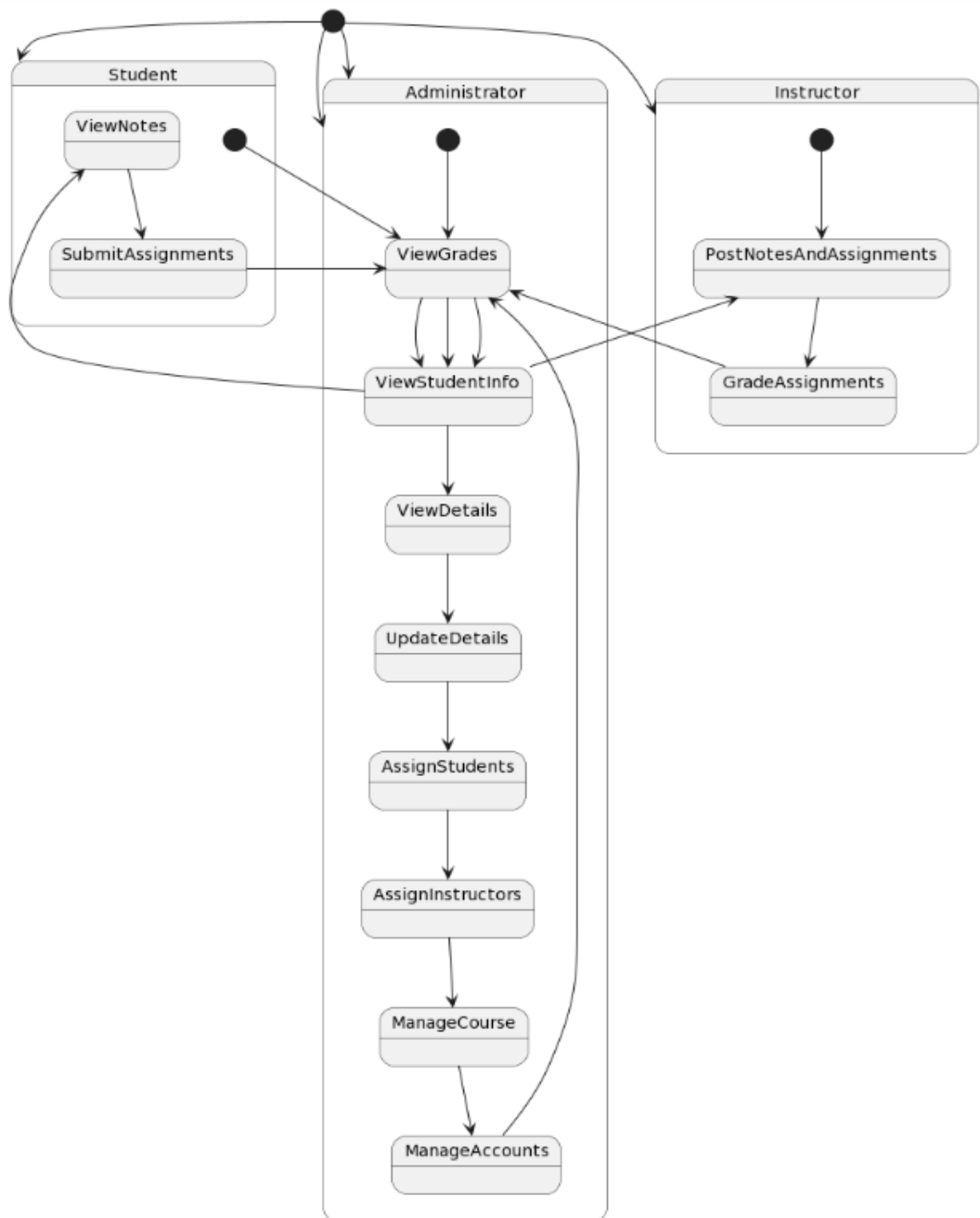
Class diagram

The class diagram illustrates the structural relationships between classes in the system, including inheritance, associations, and dependencies. It identifies the essential entities within the system, such as User, Course, Assignment, and their attributes and methods, laying the foundation for system implementation.



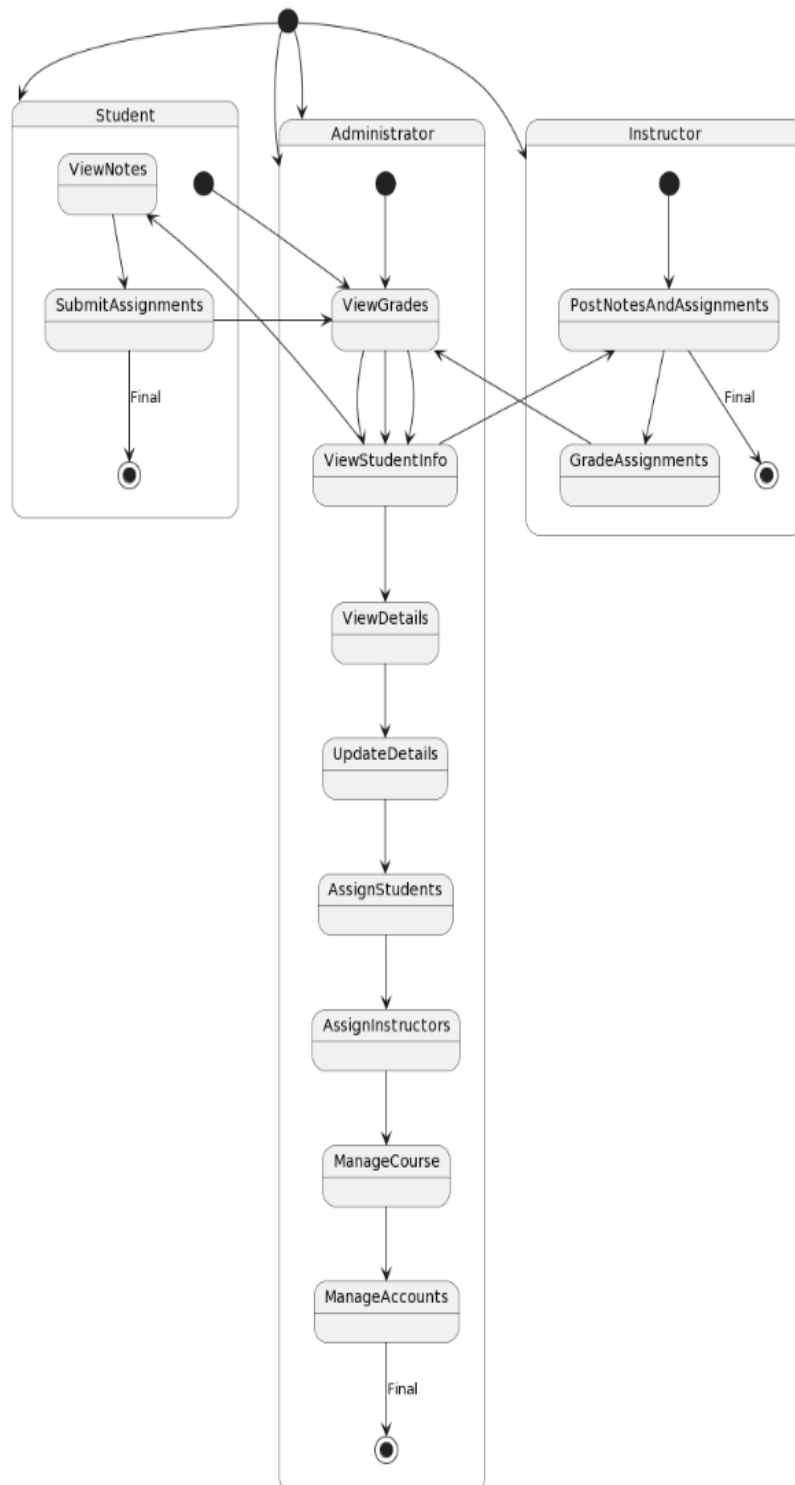
State Diagram

The state diagram, also known as a state machine diagram, represents the transitions of an object through different states in response to events. This diagram helps to visualize the lifecycle of objects within the system and the events that trigger state transitions, aiding in the understanding of system behavior.



Activity Diagram

The activity diagram illustrates the flow of activities or processes within the system. It represents the sequential and parallel activities that occur in response to various events or actions. This diagram provides a high-level view of the system's functionality and helps in identifying potential bottlenecks or inefficiencies in the workflow.



Architecture Patterns

1. Three-Tier Architecture:

Description:

The Course Management System follows a three-tier architecture comprising Presentation Layer, Business Logic Layer, and Data Access Layer.

Application:

Presentation Layer: Handles user interactions, such as logging in, navigating courses, and submitting assignments, through a user-friendly interface.

Business Logic Layer: Implements core functionalities like user authentication, course management, and assignment grading, ensuring separation of concerns and modularity.

Data Access Layer: Facilitates interaction with the underlying database, managing data retrieval, storage, and manipulation operations.

Design Principles

1. Single Responsibility Principle (SRP):

Description:

Each class or module in the system should have only one reason to change, focusing on a single responsibility.

Application:

User class: Responsible for user authentication and authorization, adhering to the principle of separation of concerns.

CourseManager class: Handles course-related functionalities such as adding courses, assigning instructors, and enrolling students, ensuring a clear and focused responsibility.

2. Open/Closed Principle (OCP):

Description:

Software entities should be open for extension but closed for modification, allowing for new functionalities to be added without altering existing code.

Application:

Assignment class: Designed to be open for extension by allowing subclasses to implement different types of assignments (e.g., written, programming), without modifying the base class.

UI class: Utilizes interfaces to define contracts for user interactions, allowing for the implementation of new user interface components without modifying existing code.

Design Patterns

1. Factory Method Pattern:

Description:

Defines an interface for creating objects, but allows subclasses to alter the type of objects that will be created.

Application:

UserFactory class: Provides a method for creating user objects based on the type of user (Admin, Instructor, Student), allowing for flexible user creation without exposing the creation logic.

2. Observer Pattern:

Description:

Defines a one-to-many dependency between objects, where changes in one object trigger updates in dependent objects.

Application:

AssignmentNotifier class: Observes changes in assignments (e.g., new assignment posted) and notifies students who are enrolled in the corresponding course, ensuring timely updates on course activities.

3. Strategy Pattern:**Description:**

Defines a family of algorithms, encapsulates each algorithm, and makes them interchangeable.

Application:

GradingStrategy interface: Defines a common interface for grading strategies (e.g., numeric grading, letter grading), allowing for interchangeable implementations based on instructor preferences or institutional grading policies.

4. Composite Pattern:**Description:**

Composes objects into tree structures to represent part-whole hierarchies. Clients can treat individual objects and compositions of objects uniformly.

Application:

CourseComponent interface: Represents both individual courses and composite courses (courses comprising multiple sub-courses), allowing for uniform treatment of courses regardless of their complexity.

Git hub link to the Codebase

<https://github.com/Kallesh07k/OOAJ>

Individual contributions of the team members

MOHAMMED FAISAL YENKTAPURI (PES1UG22CS818):

Responsible for creating accounts for students and instructors, and implementing the assignment submission feature for students.

MD IBRAHIM (PES1UG22CS817):

Tasked with adding courses and implementing functionality for students to view course materials such as notes.

KIRAN MADEV SUDHAM (PES1UG21CS280):

Assigned to implement posting notes and assignments for instructors, and enabling students to view their grades.

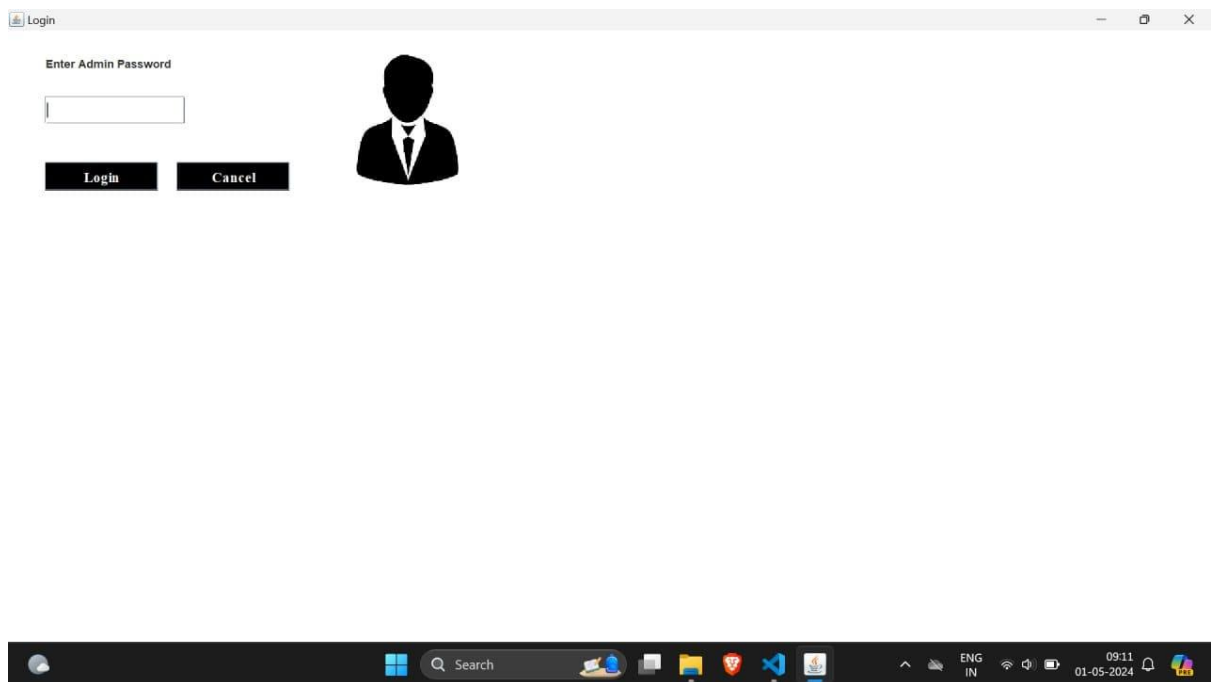
KALLES ACHAR KG (PES1UG21CS266):

Responsible for grading assignments and implementing functionality for students to view their personal information.

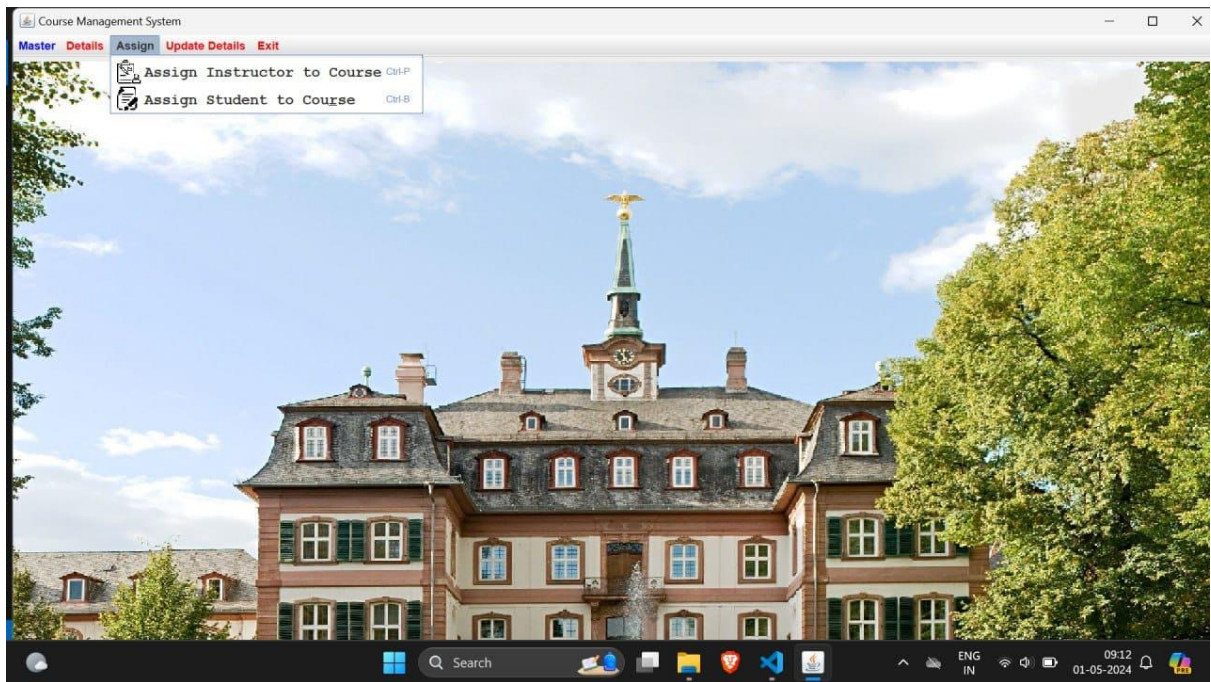
DESIGN SPECIFICATION

Front-End Design:

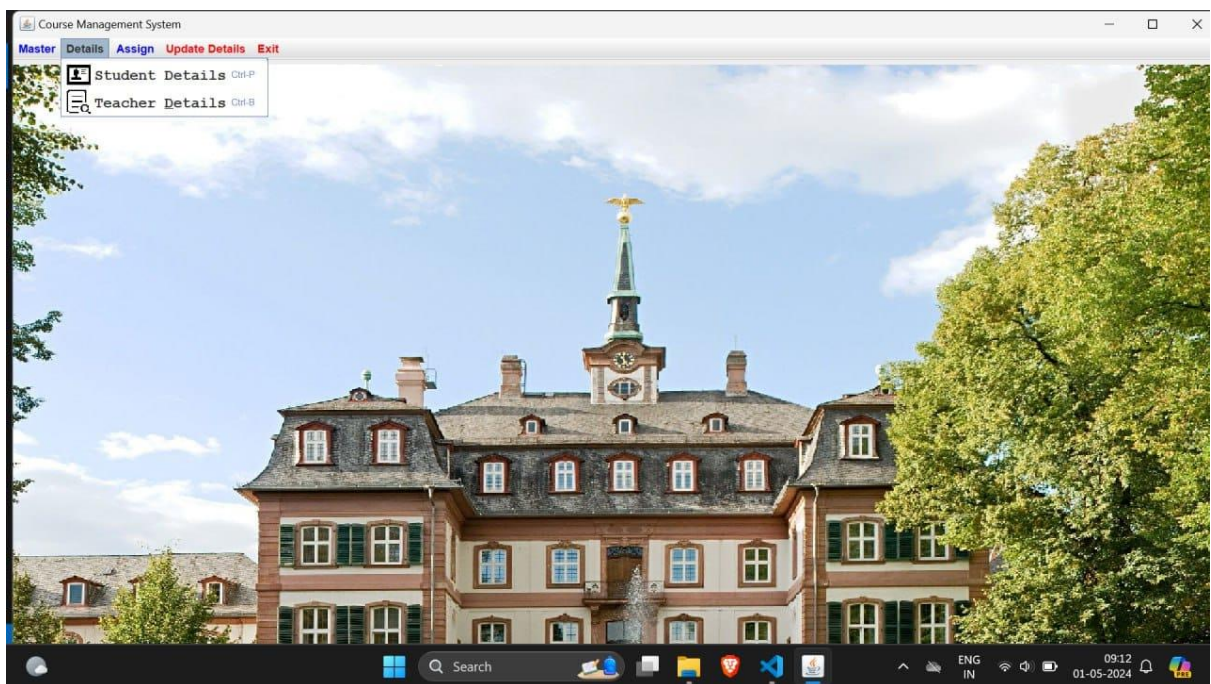
Front-end design is a very important part of a website that converts data into graphical interface. This front-end design is created using languages like HTML, CSS, and presentational JavaScript. There are several tools and platform like Notepad++, Sublime Text that can be used to develop the frontend design of a website and understanding which tools are best fit for inelastic task which marks the difference between developing a hacked site or scalable site.



ENTER ADMIN PASSWORD



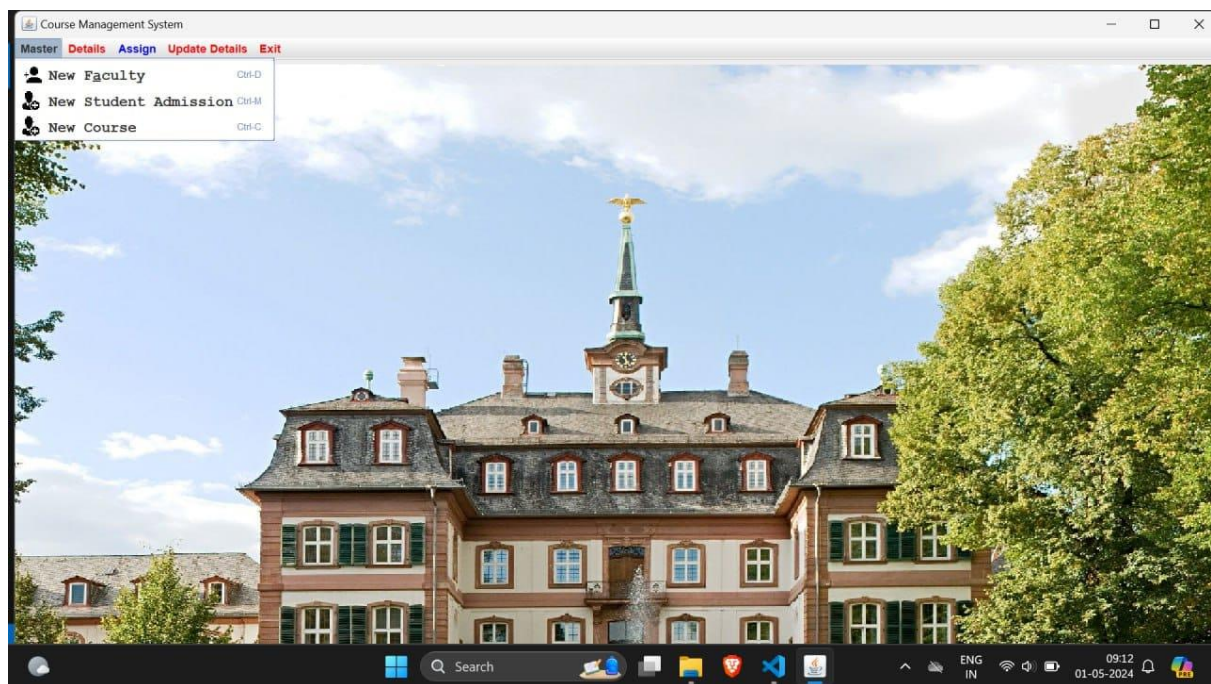
ASSIGN



DETAILS



LOGIN PAGE



ADDING NEW FACULTY / COURSE / STUDENT

Course Management System

Master Details Assign Update

Update Student details

Update Student Details:

Enter srn number to update the data of student

Name	<input type="text"/>	Email	<input type="text"/>
SRN	<input type="text"/>	Phone	<input type="text"/>
Course	<input type="text"/>	Branch	<input type="text"/>

28°C High winds soon 09:14 01-05-2024

UPDATE STUDENT DETAILS

Course Management System

Me Teacher Details

Enter Empid to delete Teacher :

Add New Teacher

Update Teacher Details

28°C Sunny 09:13 01-05-2024

TEACHERS DETAILS

Course Management System

Master Details Assign Update Del Add Course

New course details

Course - Id

Course Name

28°C Sunny 09:13 01-05-2024

COURSE DETAILS

Course Management System

Student Details

Enter ssn to delete Student :

Add New Student

Update Student Details

28°C Sunny 09:13 01-05-2024

STUDENT DETAILS

Course Management System

Master Details Assign Update D Add Student

New Student Details

Name	<input type="text"/>	Email	<input type="text"/>
SRN	<input type="text"/>	Phone	<input type="text"/>
Course	B.Tech	Branch	Computer Science

Submit Cancel

28°C Sunny 09:13 01-05-2024

NEW STUDENT DETAILS

Course Management System

Master Details Assign Update D Add Teacher

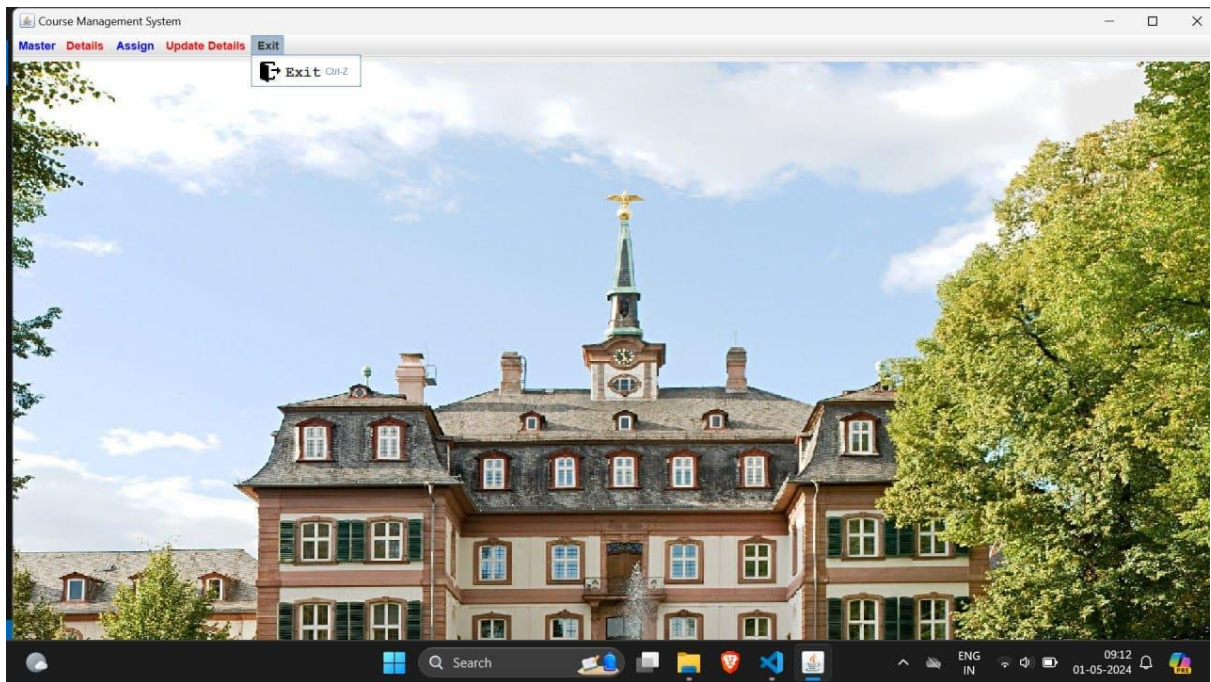
New Teacher Details

Name	<input type="text"/>	empid	<input type="text"/>
email	<input type="text"/>	phone	<input type="text"/>
Course	B.Tech	Branch	Computer Science

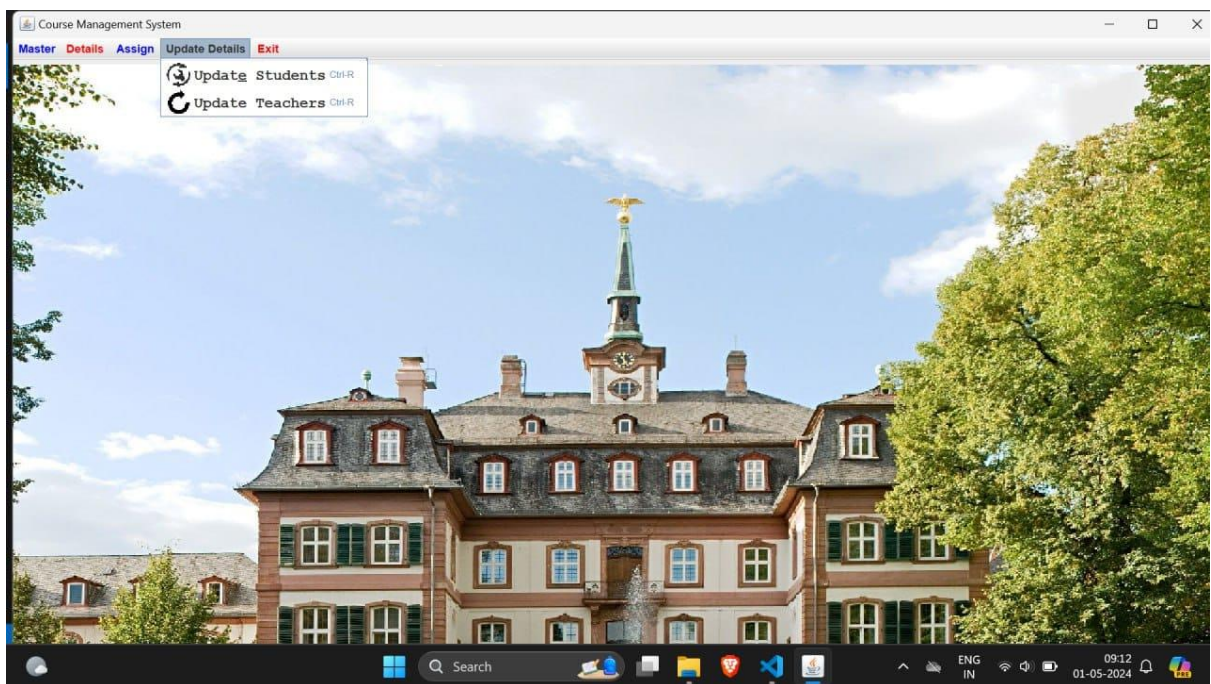
Submit Cancel

28°C Sunny 09:13 01-05-2024

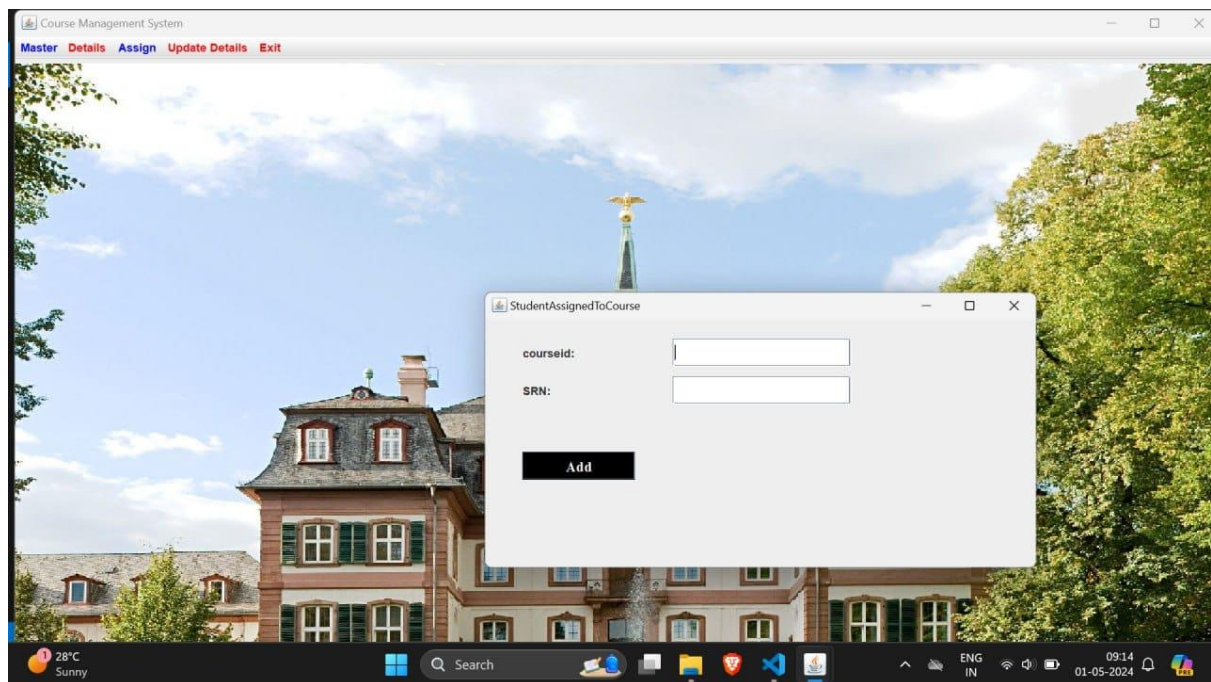
NEW TEACHER DETAILS



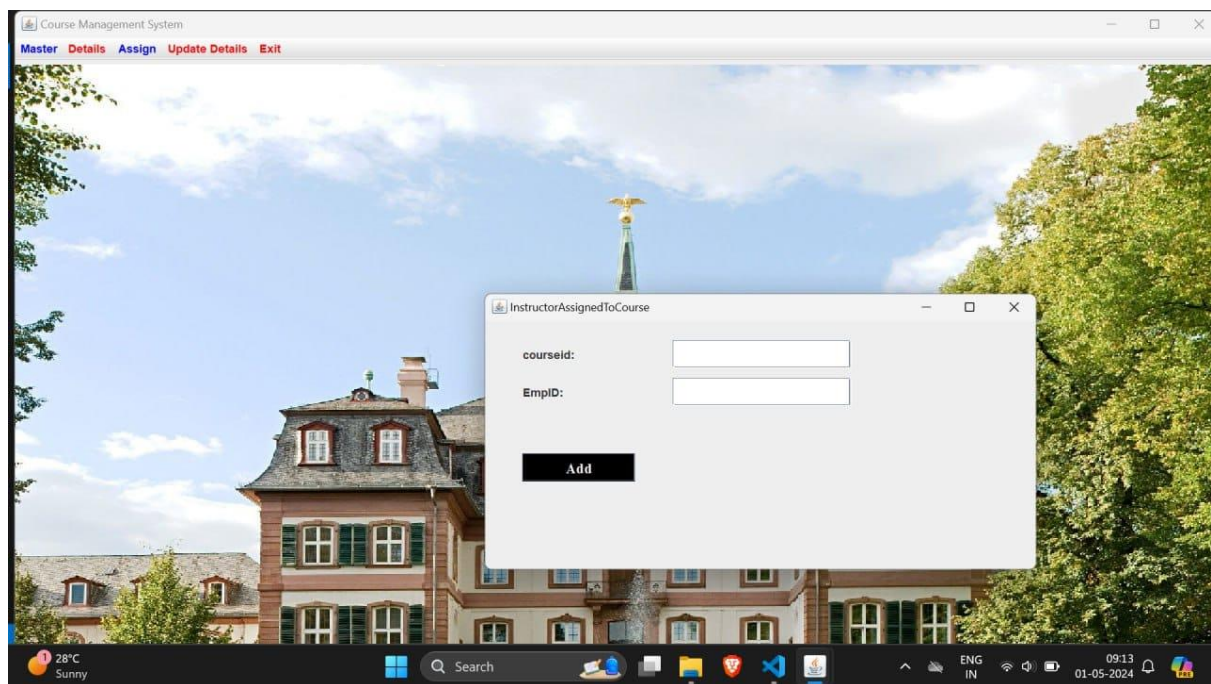
EXIT



UPDATE DETAILS



COURSE ASSIGNED TO STUDENTS



COURSE ASSIGNED TO TEACHER

Course Management System

Master Details Assign Update Details Exit

Update Teacher details

Update Teacher Details:

Enter emp id to update the data of teacher **Update**

Name	<input type="text"/>	Empid	<input type="text"/>
Email	<input type="text"/>	Phone	<input type="text"/>
Course	<input type="text"/>	Branch	<input type="text"/>

Submit **Cancel**

28°C High winds soon

Search

ENG IN

09:14 01-05-2024


UPDATE TEACHER DETAILS

StudentLogin

Username

Password

Login **Cancel**



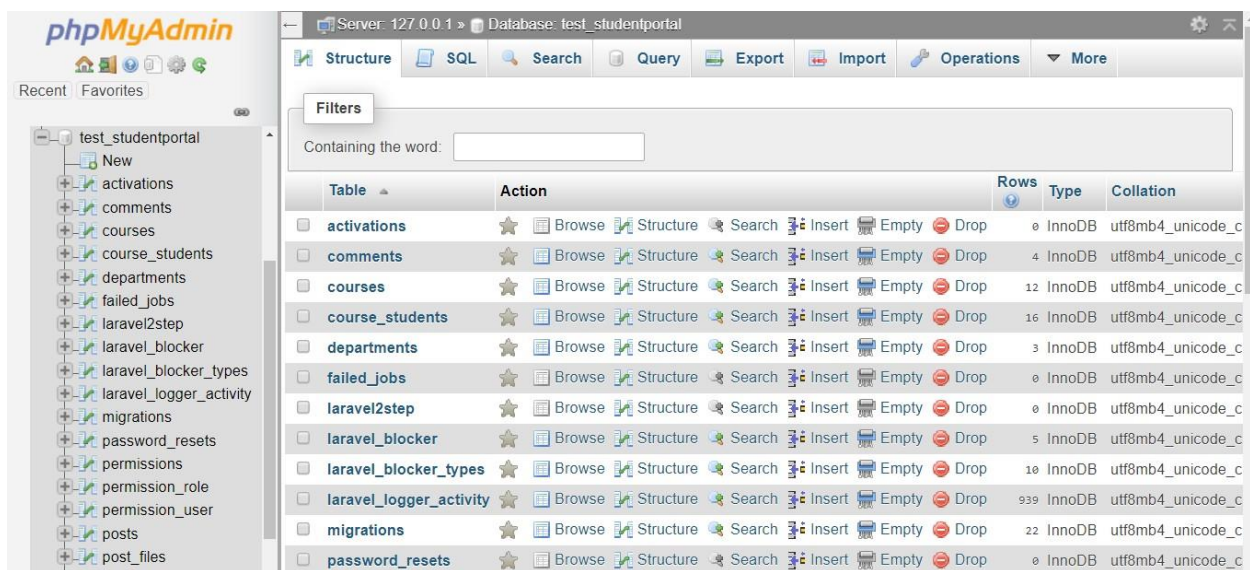
09:12 01-05-2024

STUDENT LOGIN

Back-end Design

For the back-end design and development we used Laravel Framework. Laravel is open- source PHP web framework that is server-side programming language. It is used for the backend side of web development. This framework following the model-view-controller architectural pattern. We used this framework because it's easy to creating web application with an easy to use syntax, routing, authentication, sessions and catching. The back-end design is a terminology that receives the client's request and holds the logic to send a response back to the client.

This database contains the collection of information which helps to access the data, manage and update. This information is organized logically so that it can be recuperated easily



The screenshot displays the phpMyAdmin web interface. On the left, a sidebar shows the database 'test_studentportal' with a tree view of its tables. The main panel shows the 'Structure' tab for the 'test_studentportal' database. A table with 12 columns is visible, listing the tables and their properties.

Table	Action	Rows	Type	Collation
activations	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_c
comments	Browse Structure Search Insert Empty Drop	4	InnoDB	utf8mb4_unicode_c
courses	Browse Structure Search Insert Empty Drop	12	InnoDB	utf8mb4_unicode_c
course_students	Browse Structure Search Insert Empty Drop	16	InnoDB	utf8mb4_unicode_c
departments	Browse Structure Search Insert Empty Drop	3	InnoDB	utf8mb4_unicode_c
failed_jobs	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_c
laravel2step	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_c
laravel_blocker	Browse Structure Search Insert Empty Drop	5	InnoDB	utf8mb4_unicode_c
laravel_blocker_types	Browse Structure Search Insert Empty Drop	10	InnoDB	utf8mb4_unicode_c
laravel_logger_activity	Browse Structure Search Insert Empty Drop	939	InnoDB	utf8mb4_unicode_c
migrations	Browse Structure Search Insert Empty Drop	22	InnoDB	utf8mb4_unicode_c
password_resets	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_c

Database View