Project on

Student and Teacher Portal

Department of Computer Science and Engineering
Object-Oriented Programming Lab
CSE 2112
University of Dhaka

Authors:

Papry Rahman (01)
Md. Tauseef-Ur-Rahman (24)
Anirban Roy Sourav (32)

Introduction

The Teacher-Student Portal is a web-based application designed to usable platform for teachers and students.

The Teacher-Student Portal is a web-based application for teachers and students, offering features like assignment submissions, grade tracking, forums, course management, to-do lists, and schedules. It helps students manage their academic tasks and stay organized while simplifying course management and resource distribution for teachers. This portal creates a cohesive and efficient academic environment with all essential tools in one user-friendly interface.

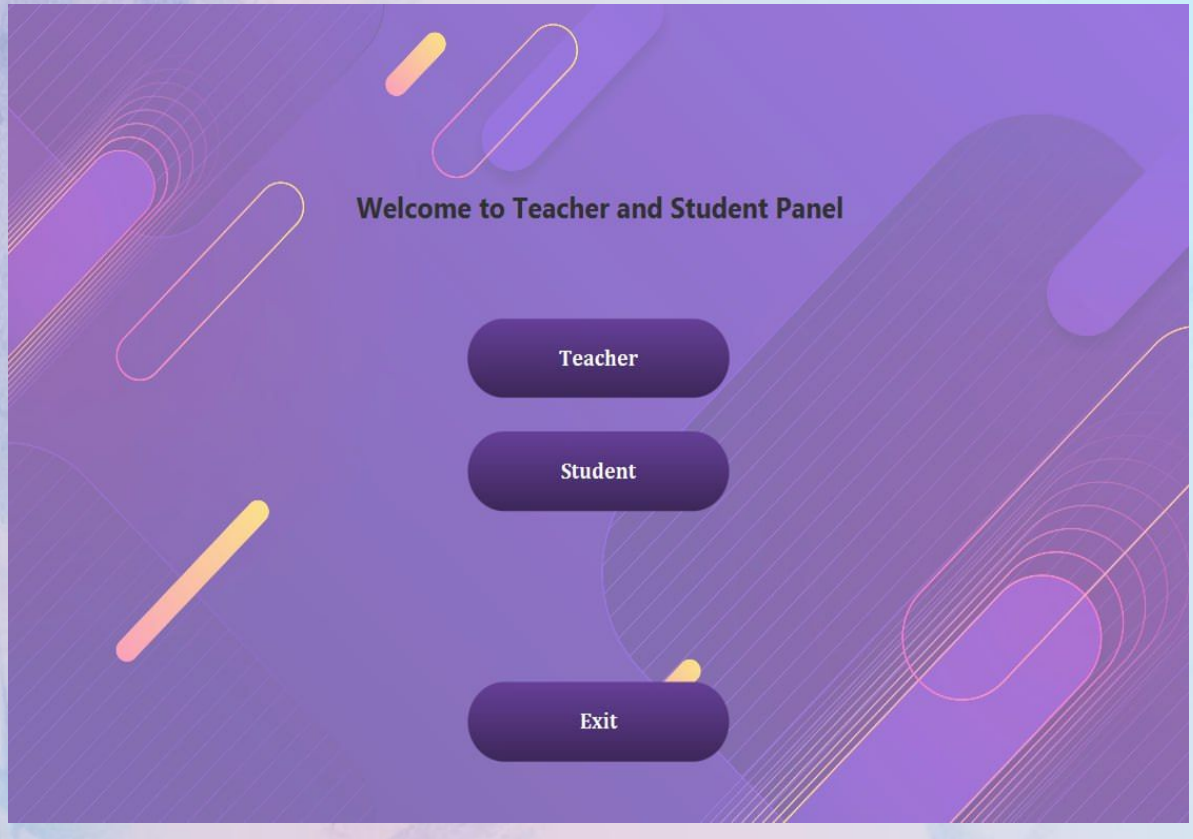
Tools

1. IDE: IntelliJ
2. Language: JAVA
3. Framework: JavaFx
4. Database: MySQL
5. Testing: Manual testing for the overall system functionality

Project Features

Roles:

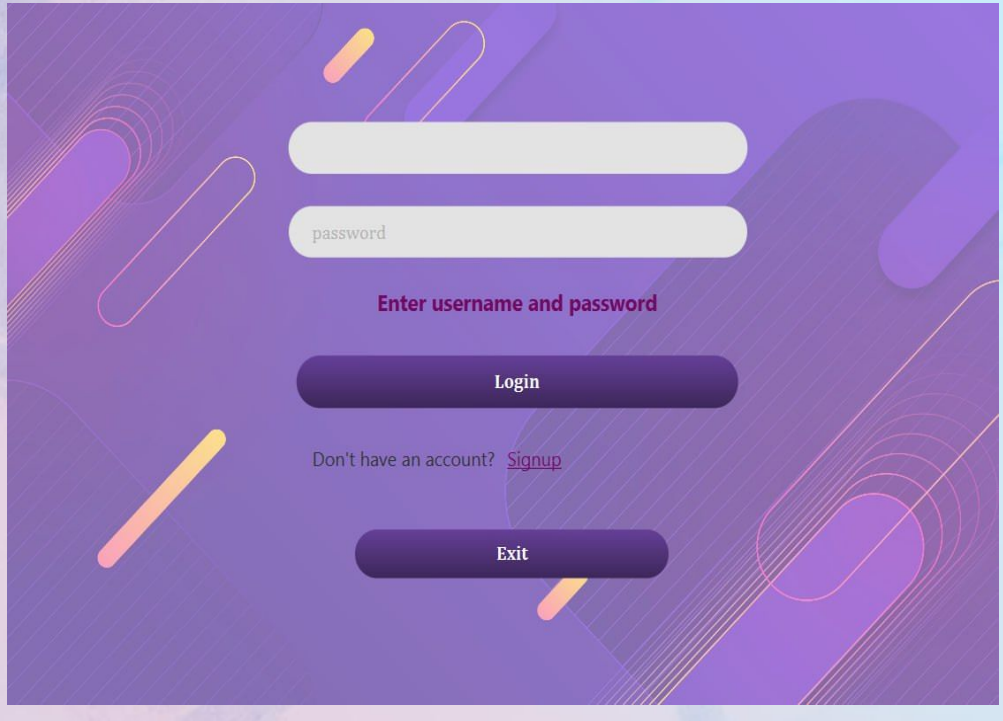
- Teacher
- Student



Student Portal

Login

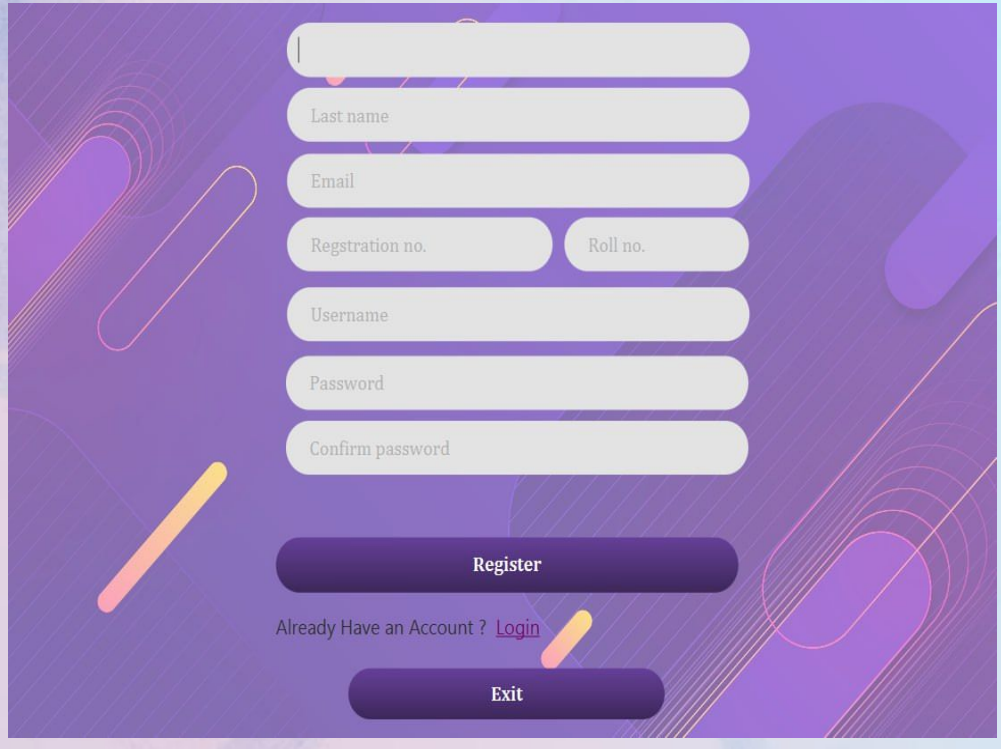
The student portal's first UI is the login page, where users enter their username and password to access the portal, authenticated by the database.



Student Portal

Registration:

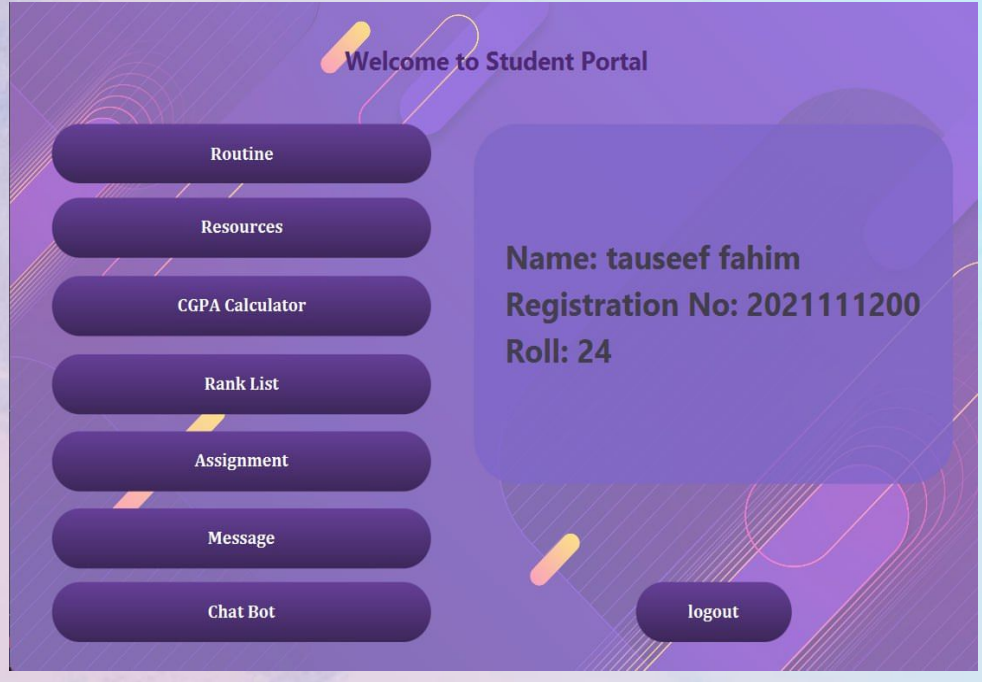
New students can register with their details on the registration page. Once registered, they can log in. The system focuses on database connection, authentication, exception handling, and directing users to the main platform.



Student Portal

Dashboard:

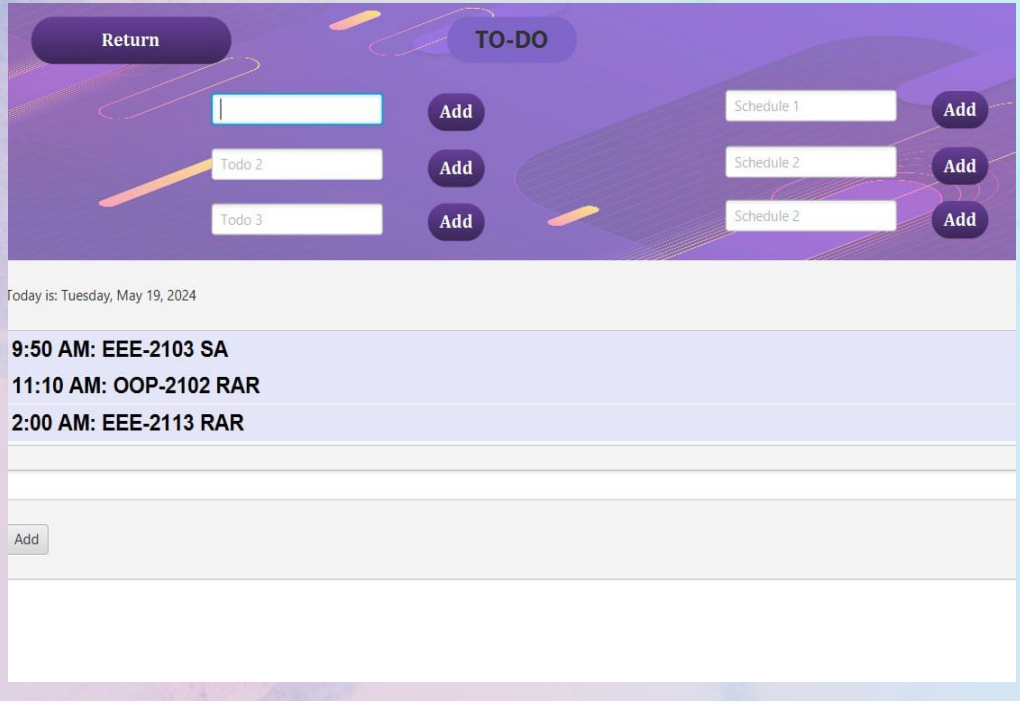
The dashboard integrates the main platform for students. After logging in, information is displayed on the right, with all other options on the left.



Student Portal

Routine:

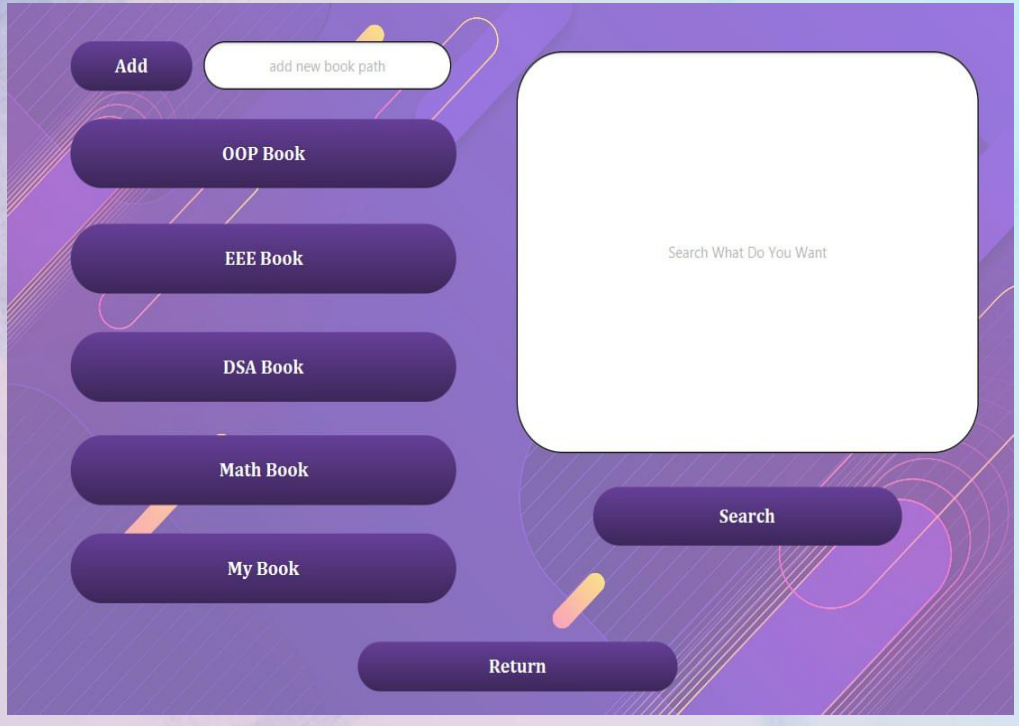
The Routine section displays the class schedule, a dynamic to-do list, and a short text editor. Class schedules are created from course details and dates, while to-do lists are linked to the database for dynamic updates.



Student Portal

Resources:

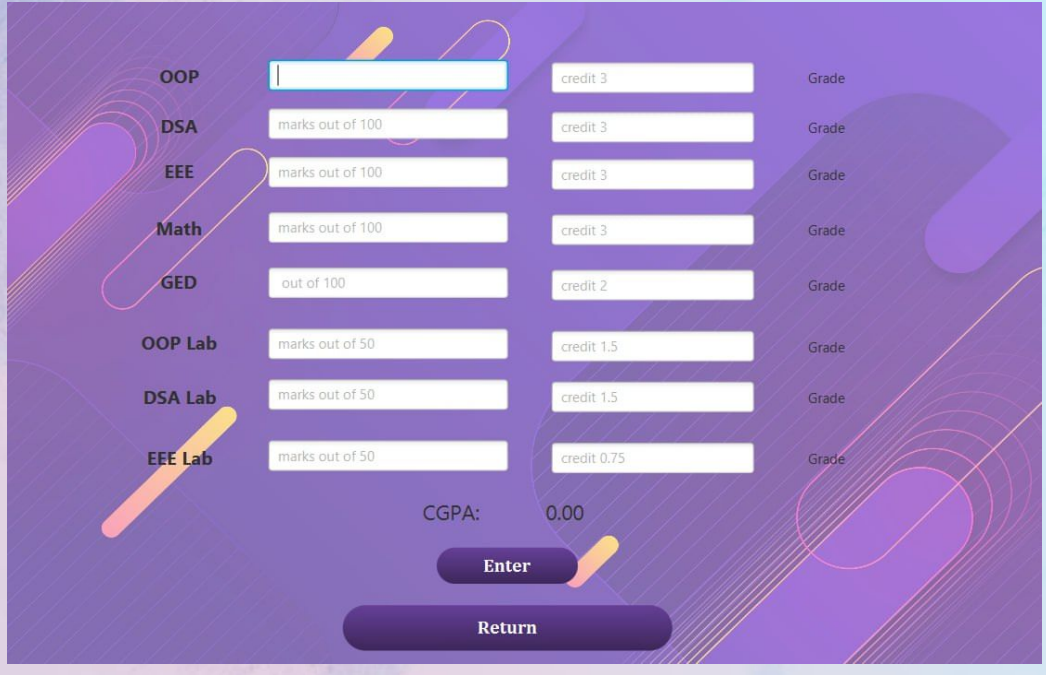
We extract our resources from local memory by creating objects for each course and dynamically adding new resources via a database. A connected search engine links to the web portal. We use a thread to open, read multiple files, and stay on the search engine.



Student Portal

CGPA calculator:

Calculates CGPA by entering course marks, showing grades and final CGPA, factoring in credits.



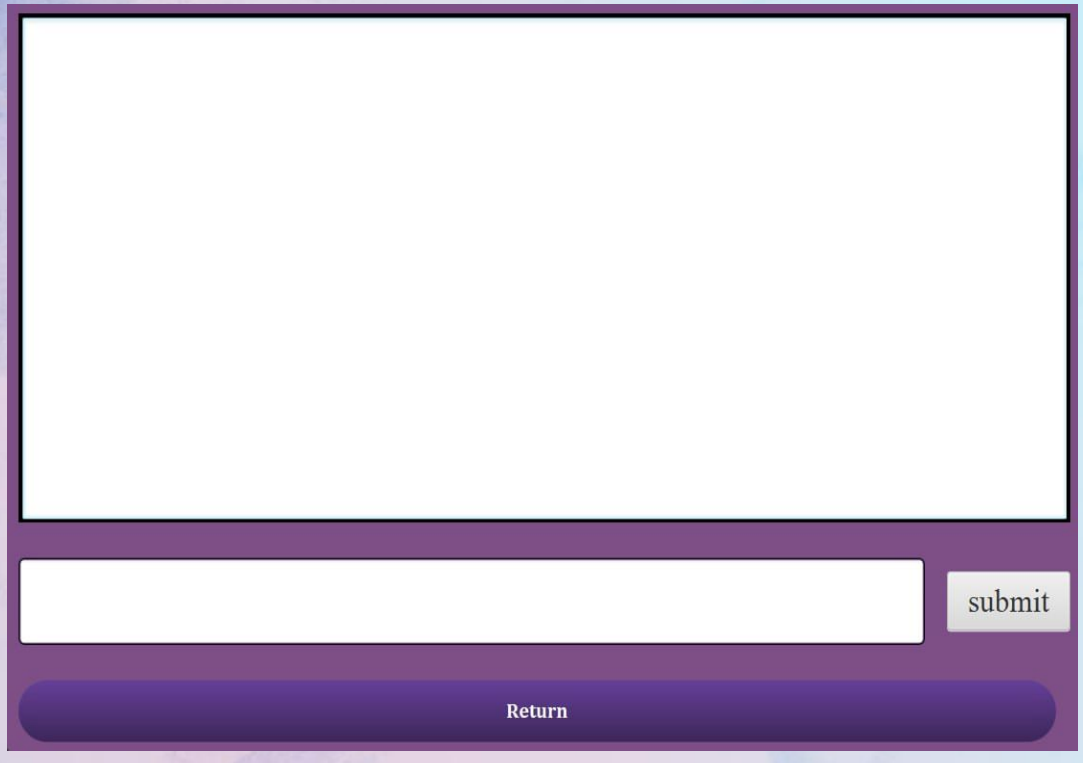
Student Portal

CGPA Graph:

Extracts CGPA from the database to create a CGPA vs. roll number graph, displaying student ranks and competitiveness

Student Portal

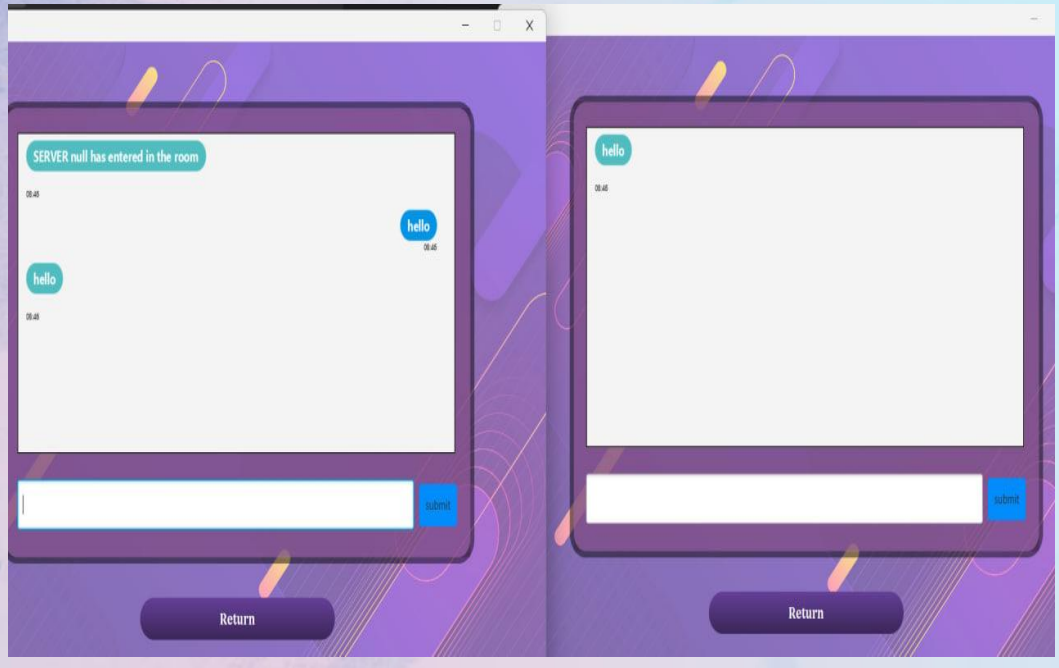
ChatBot:
Designed to assist students through conversation. Redirects to the website for queries beyond its capabilities.



Student Portal

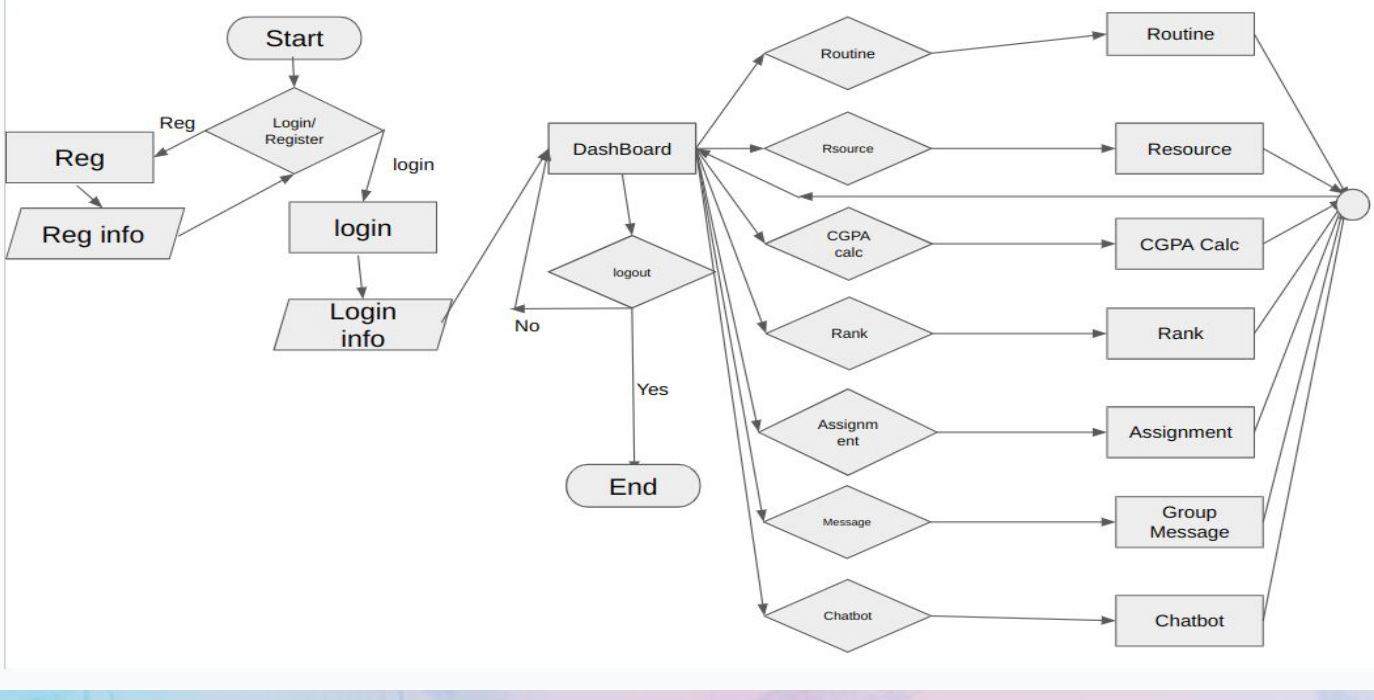
Message:

Facilitates communication and idea sharing among active students using thread and socket programming (server-client setup).



Student Portal

Workflow

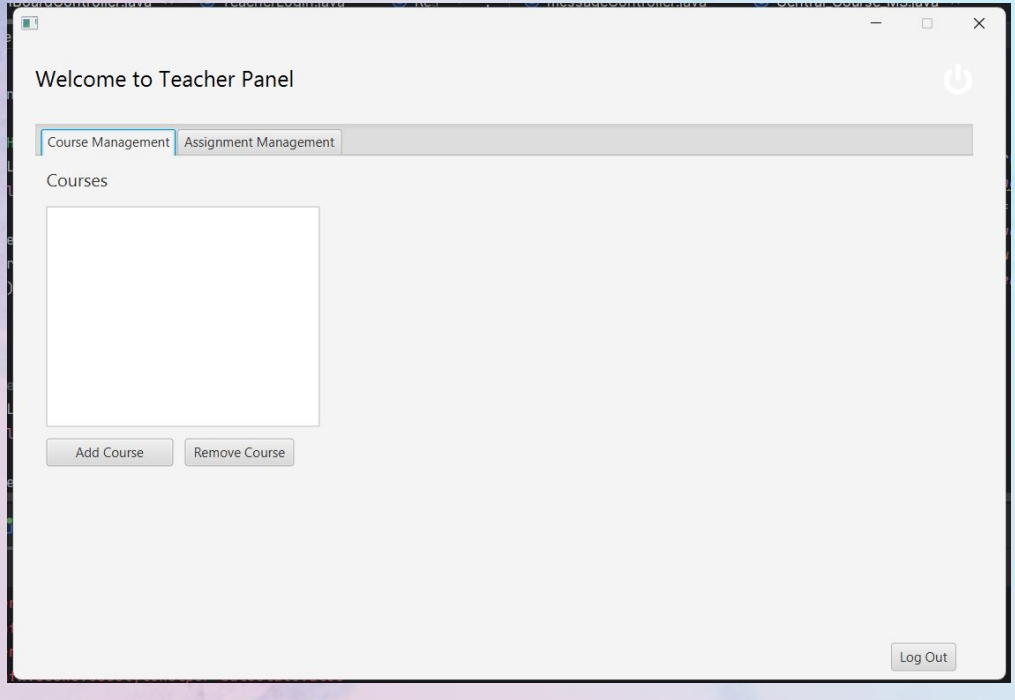


The student portal operates through interconnected pages, each modular and linked to a central database

Teacher's Portal

Dashboard:

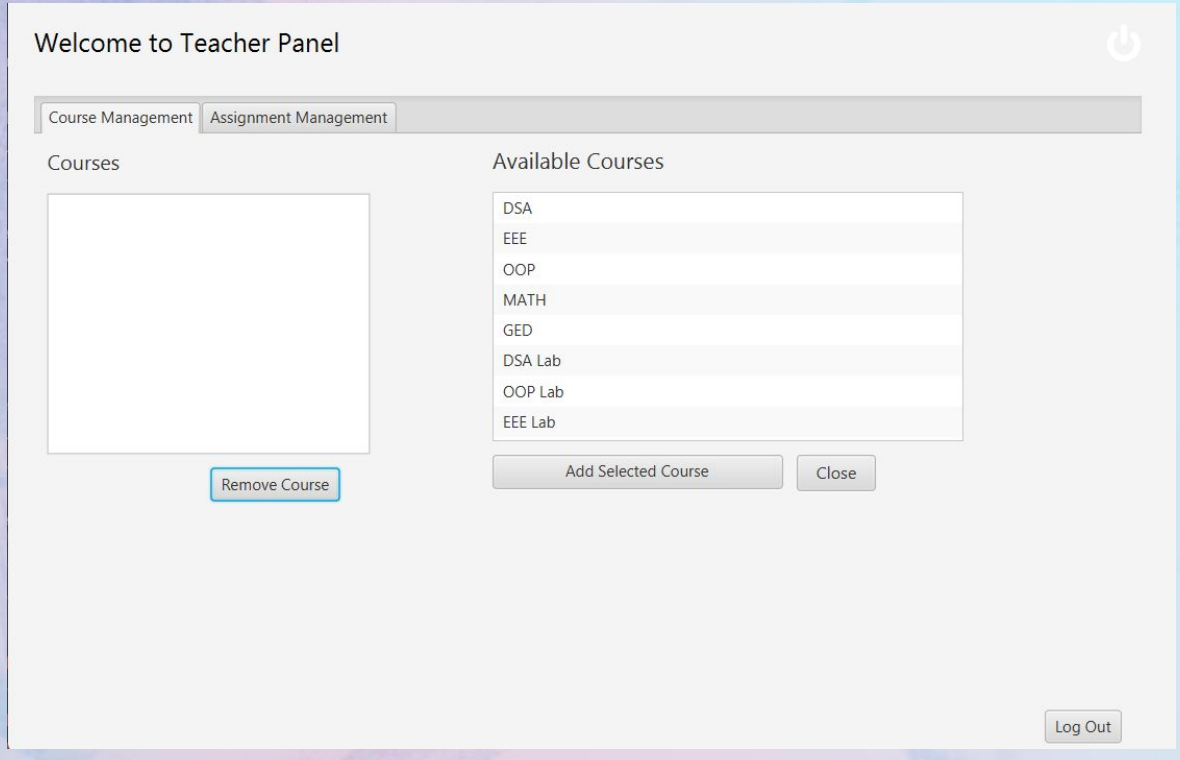
In a teacher's dashboard there are two sections. One for course management and another for assignments management for each course.



Teacher's Portal

Available courses:

Courses can be added to teacher's assigned course by clicking "**Add Course**".

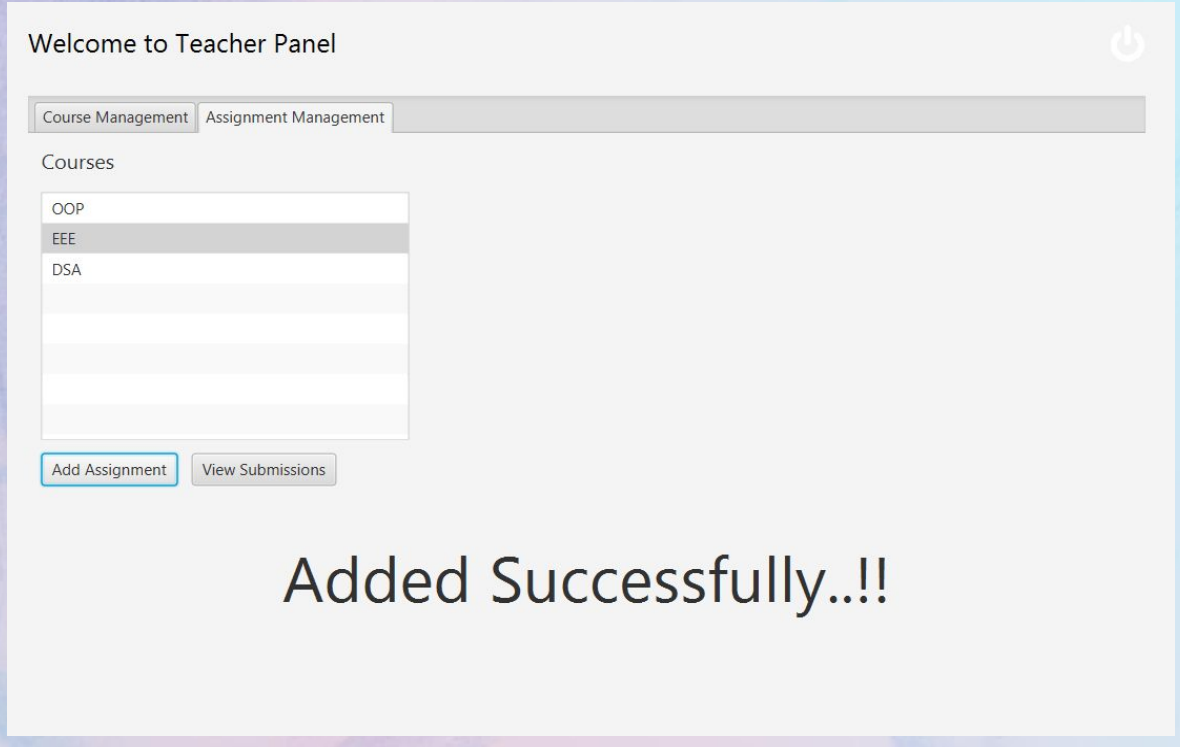


Teacher's Portal

ssigning tasks:

ssignments can be given to tudents by logging in as a eacher.

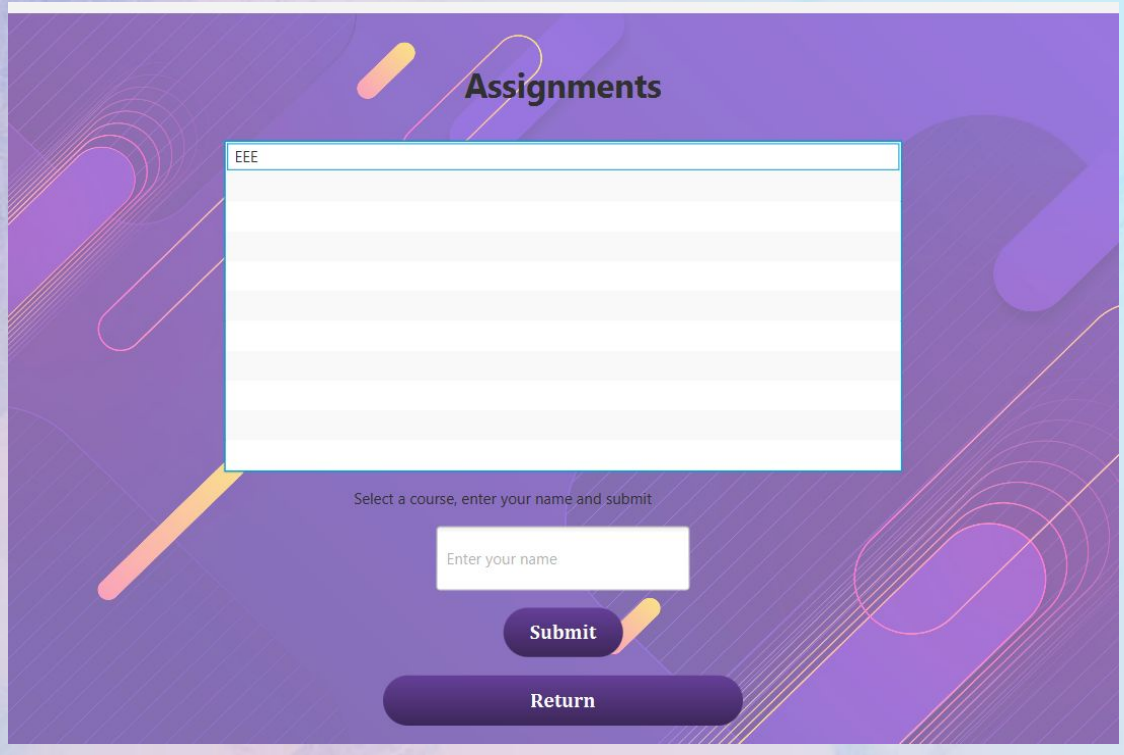
or example, here we are dding EEE assignment to all he students.



Teacher's Portal

Assignment added to Student's panel:

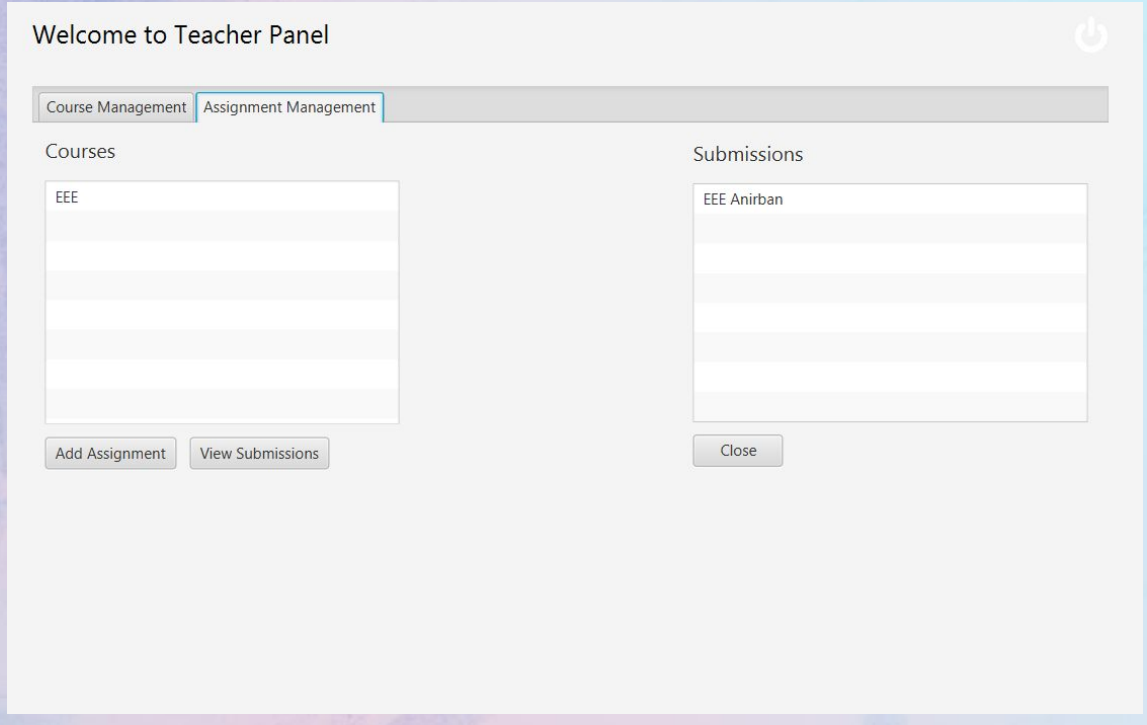
After logging out from teacher's panel and logging in into a students profile, we can see that the assignment is uploaded successfully.



Teacher's Portal

Option for viewing Submissions:

A teacher can see who have submitted the given assignments.



Implementation of OOP and Design Principles

Our project emphasizes key object-oriented principles: polymorphism, abstraction, encapsulation, and inheritance. For CGPA calculation, polymorphism handles varying evaluation systems per course. Encapsulation securely manages student data, while inheritance facilitates feature inheritance as needed.

We follow the Open-Closed Principle, enabling extension without code modification. Adding semesters or students is seamless, maintaining existing code integrity. Our modular design fosters independent high-level modules, easing extension and maintenance. Subclassing enhances program flexibility and correctness.

Conclusion

In the early stages, our optimism for the student and teacher portal was high, envisioning a comprehensive platform for resources, course management, and results, fostering connectivity. Although currently limited to one semester, we aspire to expand it to encompass our entire undergraduate journey. Despite this, we've applied our OOP knowledge and honed our software design skills, tackling real-world challenges and devising solutions along the way.

Thank You