Project

Rollins College Course Management System

Phuc Dao
William Slowey
Alex Octuk

Fall 2025

Team Members

Phuc Dao - System Integration & Design

Responsibilities

- Imported Rollins Spring 2026 catalog and merged into the database
- Designed project concept and core features
- Created full UML class diagram
- Developed & designed the presentation

William Slowey

Responsibilities

- Designed methods for each action along with exception handling
- Utilized the data structures to store data
- Implemented login system for the GUI

Alex Octuk - GUI Design

Responsibilities

- Designed and implemented GUI base
- Adapted methods to mesh with GUI structure
- Developed project report

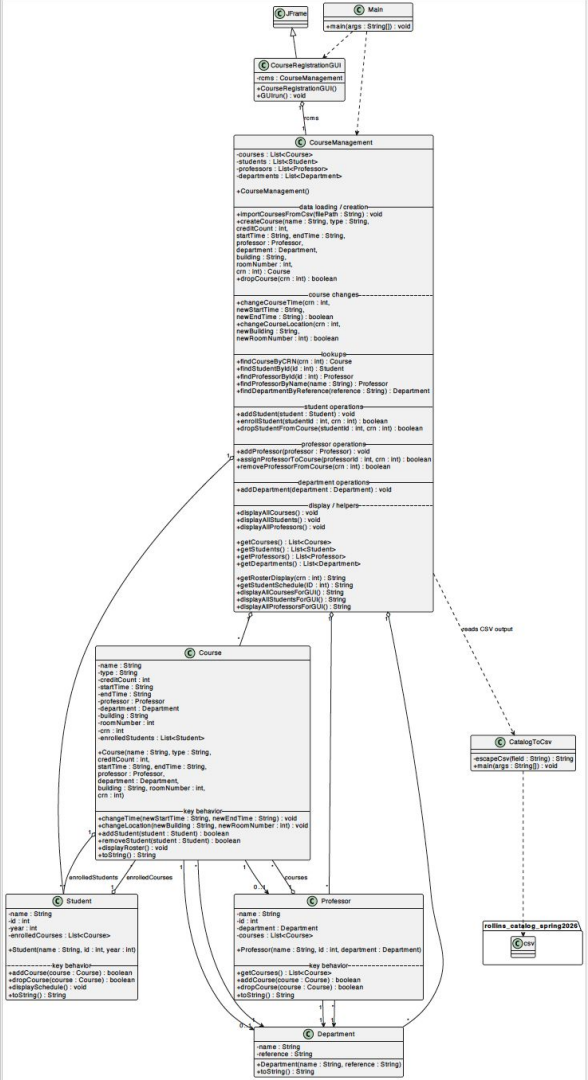
Our goal: reduce frustration, prevent crashes, and make Rollins course registration stress-free.

Concept Overview:

- A course registration system modeled after Rollins College's real enrollment structure
- Allow students to register and manage their course schedule
- Enable administrative management of users and classes
- Each system element (Students, Professors, Departments, Courses) has unique IDs for fast searching (CRN, Student ID, etc.)
- Built to be more stable and streamlined than the current Rollins system, which often struggles with performance during heavy use

DEMO

How We Used OOP Principles



How We Used OOP Principles

Encapsulation

-Each core object (Student, Course, Professor, Department) has private fields with public getters and setters

```java
public class Course {
    private String name;
    private String type;
    private int creditCount;
    private String startTime;
    private String endTime;
    private Professor professor;
    private Department department;
    private String building;
    private int roomNumber;
    private int crn; // course reference number
    private List<Student> enrolledStudents; // Track enrolled students

    public Course(String name, String type, int creditCount, String startTime, String endTime,
                  Professor professor, Department department, String building, int roomNumber, int crn) {
        this.name = name;
        this.type = type;
        this.creditCount = creditCount;
        this.startTime = startTime;
        this.endTime = endTime;
        this.professor = professor;
        this.department = department;
        this.building = building;
        this.roomNumber = roomNumber;
        this.crn = crn;
        this.enrolledStudents = new ArrayList<>();
    }

    // Getters and setters
    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
```

Abstraction

- CourseManagement.java acts as the brain of the system
    - Handles high-level actions like enrollStudent(id, crn), dropStudentFromCourse(id, crn), findCourseByCRN(crn)

```java

    //add the student to a class
    public boolean enrollStudent(int studentId, int crn) {
        Student student = findStudentById(studentId);
        Course course = findCourseByCRN(crn);

        if (student == null) {
            System.out.println("Student not found with ID: " + studentId);
            return false;
        }
        if (course == null) {
            System.out.println("Course not found with CRN: " + crn);
            return false;
        }

        if (student.addCourse(course)) {
            System.out.println(student.getName() + " enrolled in " + course.getName());
            return true;
        }
        return false;
    }

    //drop student by class
    public boolean dropStudentFromCourse(int studentId, int crn) {
        Student student = findStudentById(studentId);
        Course course = findCourseByCRN(crn);

        if (student == null) {
            System.out.println("Student not found with ID: " + studentId);
            return false;
        }
        if (course == null) {
            System.out.println("Course not found with CRN: " + crn);
            return false;
        }

        if (student.dropCourse(course)) {
            System.out.println(student.getName() + " dropped from " + course.getName());
            return true;
        }
        return false;
    }

```

How We Used OOP Principles

Inheritance

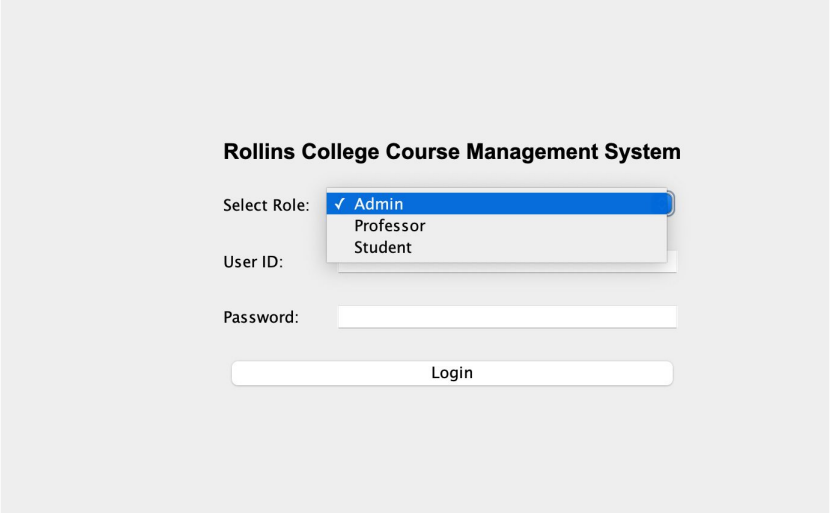
- The GUI class extends Swing components

CMS270FinalProject ▸ src ▸ (default package) ▸ CourseRegistrationGUI ▸ GUIrun() : v

```
1  import javax.swing.*;
3
4  public class CourseRegistrationGUI extends JFrame{
5
6      //create management system
7      private final CourseManagement rcms;
8
9      // Constructor
10     public CourseRegistrationGUI() {
11         this.rcms = new CourseManagement();
12         // Load the Rollins catalog into the system
13         // Make sure rollins_catalog_spring2026.csv is in your project root
14         this.rcms.importCoursesFromCsv("rollins_catalog_spring2026.csv");
15     }
16
17
18     //layout parent
19     static CardLayout layout;
20     static JPanel cards;
21
22     //card layout for changing course details
23     static CardLayout courseLayout;
24     static JPanel courseCards;
25
26     //user role tracking
27     private String userRole = ""; // "ADMIN", "PROFESSOR", "STUDENT"
28     private int userId = 0;
29
30     //Main running program
31     public void GUIrun() {
32         setTitle("Course Management - Login");
33         setSize(900, 600);
34         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
35
36         layout = new CardLayout();
37         cards = new JPanel(layout);
```

Polymorphism & Composition

- Polymorphism appears in how different user roles interact through the same system
    - The same backend (CourseManagement) supports Admin, Student, and Professor actions using different GUI screens



Challenges and Lessons Learned

Challenges

- Implementing and merging Course Catalog CSV into the system
- Creating and designing methods to mesh with data structures
- GUI Design and functionality

Lessons

- How to import and change the course catalog to match data structures used in system
  - Learning BufferedReader and IOException implementation
- How to align the needs of a method with the data structures used
  - Visibility, static vs non-static/private
- Fully implementing GUI functions and how to change screens for different functions
  - Mapping structure of GUI
  - Bookkeeping what screens go where and what is added to screens

Thank you!

Any Questions?