Phase 4 – Implementation Report

Object-Oriented Analysis and Design – Fall 2025

Online Course Registration System

Basmala Salah Mohamed

Rana Abdelhamid Dief

GitHub Repository:
https://github.com/RanaDief/OOAD_Project

1) Introduction

1.1 Objective

The objective of Phase 4 is to implement a functional software prototype of the **Online Course Registration System**, translating the approved system design and UML diagrams into executable Java code. The system demonstrates the core registration workflow using object-oriented principles and a console-based interface.

1.2 Scope

This phase focuses on:

- Implementing core domain classes directly derived from UML diagrams
- Applying OOP principles (encapsulation, inheritance, abstraction, polymorphism)
- Simulating student, instructor, and administrator interactions
- Validating registration constraints such as prerequisites, capacity, and schedule conflicts
- Providing a menu-driven console interface

2) Development Environment & Project Setup

2.1 Technology Stack

- **Programming Language:** Java
- **Paradigm:** Object-Oriented Programming (OOP)
- **User Interface:** Console-based (menu-driven)

- **Version Control:** Git & GitHub

2.2 Project Repository

The project is hosted on GitHub and structured into clear layers separating responsibilities.

Repository link:
https://github.com/RanaDief/OOAD_Project

2.3 How to Run the Project

Prerequisites

- Java JDK installed

Steps

Clone the repository

```
git clone https://github.com/RanaDief/OOAD_Project
```

1. Open the project in your Java IDE

Navigate to:

```
Phase 4/src/app/Main.java
```

2. Run `Main.java`
3. Follow the console menus to simulate system usage

3) Overview of Implemented Features

3.1 Implemented Features Summary

- The implemented system provides a functional, console-based prototype of the **Online Course Registration System**, aligned with the approved system design and UML diagrams from earlier phases.

- The system supports three primary user roles: **Student**, **Instructor**, and **Administrator**, each with clearly defined responsibilities and access privileges.

- **Student functionalities** include:

    - Browsing all available courses and their associated sections through a menu-driven console interface.

    - Submitting course registration requests.

    - Automatic validation of registration constraints, including:

        - Verification of prerequisite completion.

        - Enforcement of course section capacity limits.

        - Detection of schedule conflicts with existing registered courses.

        - Enforcement of maximum course load and scheduling rules, where applicable.

    - Viewing registered courses along with their current registration status (Pending, Approved, or Rejected).

- **Instructor functionalities** include:

    - Viewing pending course registration requests for assigned courses.

    - Approving or rejecting student registration requests.
    - Viewing the list of enrolled students for each course section after approval.
- **Administrator functionalities** include:
    - Adding, editing, and removing courses from the system.
    - Adding, editing, and removing system users.
    - Managing registration rules such as maximum course load and scheduling constraints.
- The system implements a complete registration lifecycle, beginning with course browsing and registration requests, followed by validation and instructor decision-making, and concluding with confirmed student enrollment.
- The implementation applies core **object-oriented programming principles**, including encapsulation, inheritance, abstraction, and polymorphism, and follows a layered architectural structure separating user interface, business logic, domain models, and data storage.
- Overall, the implemented features demonstrate a minimal yet realistic academic registration system that fulfills the objectives of Phase 4 and provides a solid foundation for future system enhancements.

Feature	Status
Browse available courses	✔ Implemented
Student course registration	✔ Implemented
Prerequisite validation	✔ Implemented
Course capacity validation	✔ Implemented
Schedule conflict detection	✔ Implemented
View registered courses	✔ Implemented
Instructor approval/rejection	✔ Implemented
View enrolled students	✔ Implemented
Admin course management	✔ Implemented
Admin user management	✔ Implemented
Registration rule management	✔ Implemented
Console-based menus	✔ Implemented
In-memory persistence	✔ Implemented

4. System Design to Code Mapping

4.1 UML Class Diagram Mapping

The system classes directly reflect the design diagrams from earlier phases.

- **User (User.java)**
  Abstract base class representing common user attributes.

- **Student (Student.java)**
  Extends `User`; contains registered courses and registration actions.

- **Instructor (Instructor.java)**

Extends `User`; responsible for approving or rejecting registration requests.

- **Administrator (Administrator.java)**
  Extends `User`; manages users, courses, and registration rules.

- **Course (Course.java)**
  Represents course metadata such as code, name, and credit hours.

- **Section (Section.java)**
  Represents course sections with schedules and capacity.

- **Registration (Registration.java)**
  Stores student-course relationships and registration status.

- **RegistrationStatus (RegistrationStatus.java)**
  Enum defining states: `PENDING`, `APPROVED`, `REJECTED`.

4.2 Registration State Flow

The registration lifecycle follows the designed state machine:

1. Student submits registration → **PENDING**
2. Instructor decision:
   - Approve → **APPROVED**
   - Reject → **REJECTED**

This is implemented using `RegistrationStatus` and managed by `RegistrationService`.

5. Core Functionalities

5.1 Student Functions

5.1.1 Browse Courses

Students can view all available courses and sections through the console interface.

5.1.2 Register for Course

Before creating a registration request, the system validates:

- Prerequisite completion
- Section capacity availability
- Schedule conflicts
- Maximum course load

If valid, a registration with status **PENDING** is created.

5.1.3 View Registered Courses

Students can view all registered courses along with their registration status.

5.2 Instructor Functions

5.2.1 Approve / Reject Requests

Instructors can review pending registrations and approve or reject them.

5.2.2 View Enrolled Students

Instructors can list students enrolled in each course section.

5.3 Administrator Functions

5.3.1 Course Management

Administrators can:

- Add new courses
- Edit existing courses
- Remove courses

5.3.2 User Management

Administrators can manage:

- Students
- Instructors
- Administrators

5.3.3 Registration Rules

Administrators configure system constraints such as:

- Maximum course load
- Schedule limitations

6. Console-Based User Interaction

The system uses a menu-driven console interface.

6.1 Menu Structure

- Main Menu
    - Student Login
    - Instructor Login
    - Administrator Login
- Student Menu
    - Browse courses
    - Register for course
    - View registrations
- Instructor Menu
    - View pending requests
    - Approve/reject registrations
    - View enrolled students
- Administrator Menu
    - Manage courses
    - Manage users

- Manage registration rules

7. Screenshots of Console Output

Screenshot 1: Main Menu Interface (Student-Instructor-Admin)

```
=== Online Course Registration System (OCRS) ===

1) Login
0) Exit
> 1
Email: rana@student.edu
Password: stud123
Logged in as: STUDENT{S001, Rana, rana@student.edu}
```

```
--- Student Menu ---
Rules: Rules{open=true, maxLoad=5}
1) Browse courses
2) Register for a course (request)
3) View my registered courses (approved)
4) Drop a course
0) Logout
```

```
> 1
Email: dr.sara@uni.edu
Password: instr123
Logged in as: INSTRUCTOR{I001, Dr. Sara, dr.sara@uni.edu}

--- Instructor Menu ---
1) View my courses
2) View pending requests for a course
3) Approve request
4) Reject request
5) View enrolled students for a course
0) Logout
```

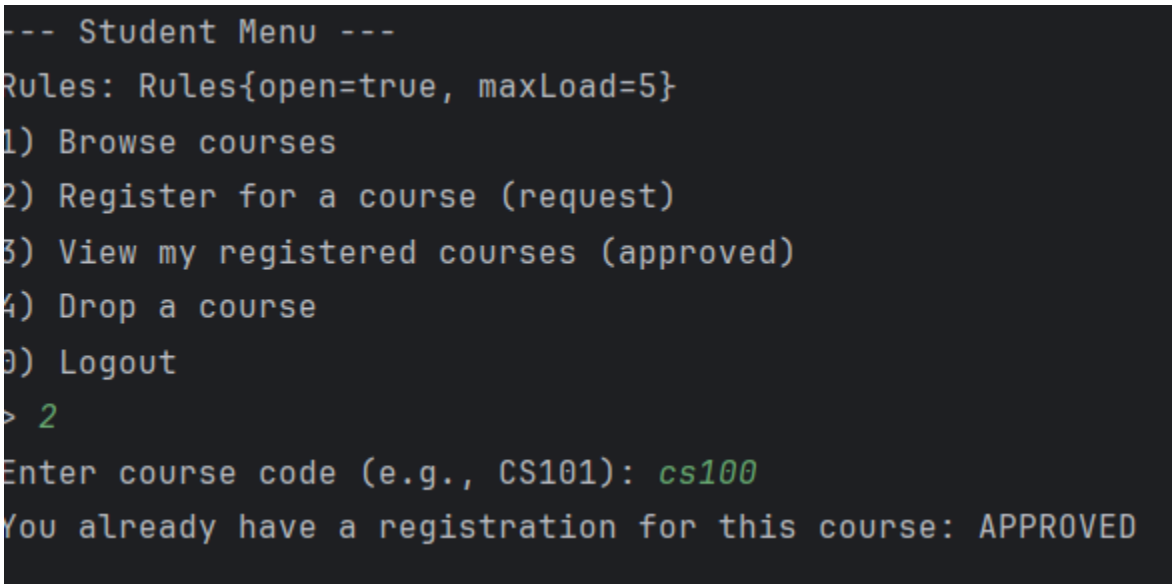
```
1) Login
0) Exit
> 1
Email: admin@uni.edu
Password: admin123
Logged in as: ADMIN{A001, System Admin, admin@uni.edu}

--- Admin Menu ---
Rules: Rules{open=true, maxLoad=5}
1) List all users
2) Add user
3) Remove user
4) List all courses
5) Add course
6) Remove course
7) Set max load
8) Toggle registration open/closed
0) Logout
```

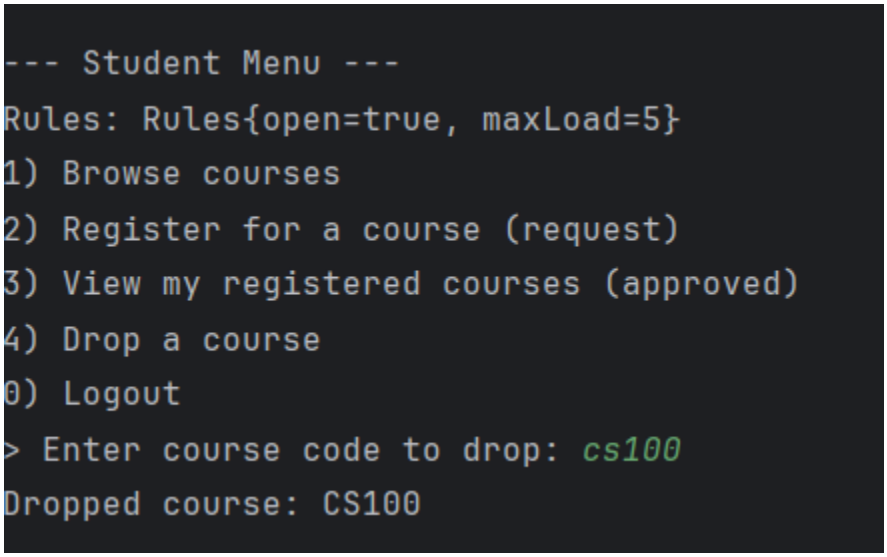
Screenshot 2: Student Browsing Available Courses

```
Available Courses:
 - CS100 - Intro to Programming (Fall 2025), Cap: 1/2, Instr: Dr. Sara
   Times: [SUNDAY 09:00-11:00]
   Prereqs: []
 - CS101 - Data Structures (Fall 2025), Cap: 0/2, Instr: Dr. Sara
   Times: [SUNDAY 11:00-13:00]
   Prereqs: [CS100]
 - CS102 - Discrete Math (Fall 2025), Cap: 0/2, Instr: Dr. Sara
   Times: [SUNDAY 12:00-14:00]
   Prereqs: []
```

Screenshot 3: Student Registration Request (Successful Validation)



Screenshot 4: Student dropped a course



Screenshot 5: Student Viewing Pending Requests

```
--- Student Menu ---
Rules: Rules{open=true, maxLoad=5}
1) Browse courses
2) Register for a course (request)
3) View my registered courses (approved)
4) Drop a course
0) Logout
> 2
Enter course code (e.g., CS101): cs101
Registration request submitted (PENDING). Wait for instructor approval.
```

Screenshot 6: Student Viewing Approved Courses

```
--- Student Menu ---
Rules: Rules{open=true, maxLoad=5}
1) Browse courses
2) Register for a course (request)
3) View my registered courses (approved)
4) Drop a course
0) Logout
> 3

My Approved Courses:
 - CS100: Intro to Programming | [SUNDAY 09:00-11:00]
```

Screenshot 7: Instructor Viewing Pending Requests

```
--- Instructor Menu ---
1) View my courses
2) View pending requests for a course
3) Approve request
4) Reject request
5) View enrolled students for a course
0) Logout
> 2
Course code: cs100

Pending Requests for CS100:
 - (none)
```

Screenshot 8: Instructor Approving Registration

```
--- Instructor Menu ---
1) View my courses
2) View pending requests for a course
3) Approve request
4) Reject request
5) View enrolled students for a course
0) Logout
> 3
Course code: cs100
Student ID to approve: s001
```

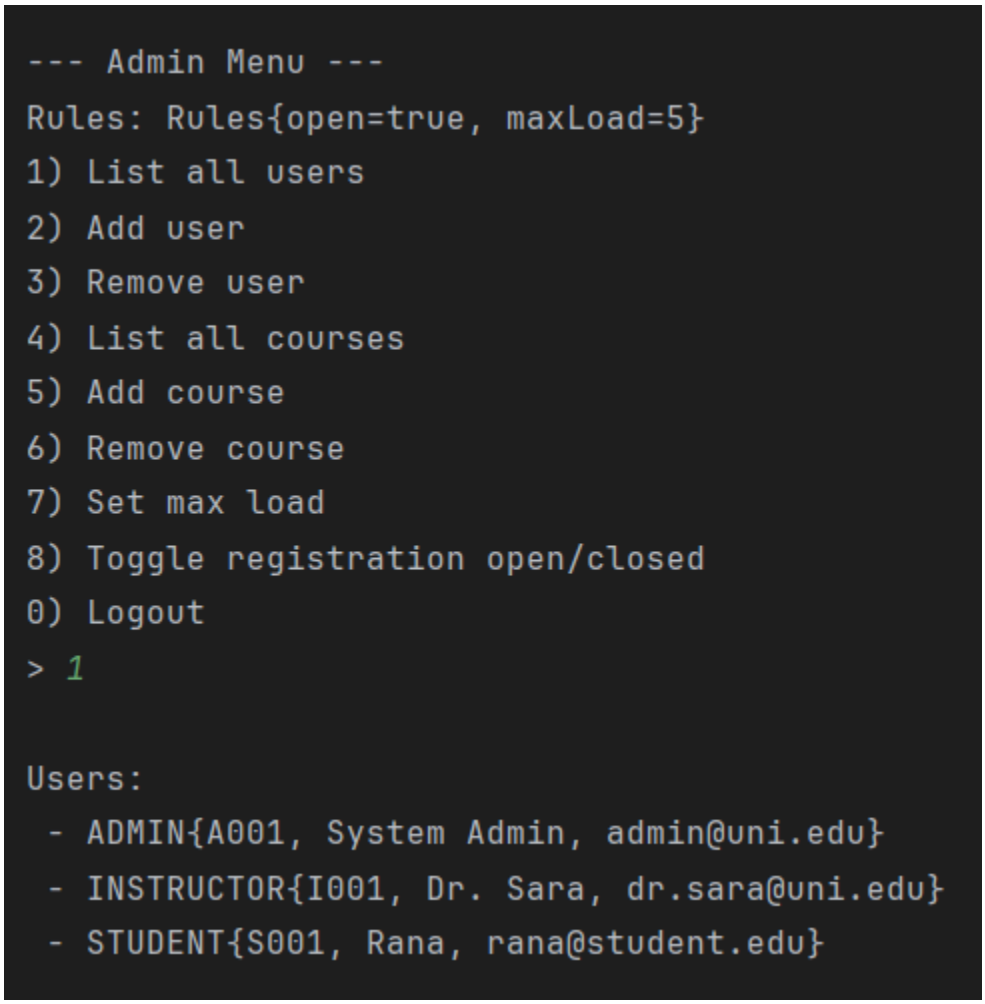
Screenshot 9: Instructor Rejecting Registration

```
--- Instructor Menu ---
1) View my courses
2) View pending requests for a course
3) Approve request
4) Reject request
5) View enrolled students for a course
0) Logout
> 4
Course code: cs101
Student ID to reject: s001
Rejected: Rana request for CS101
```

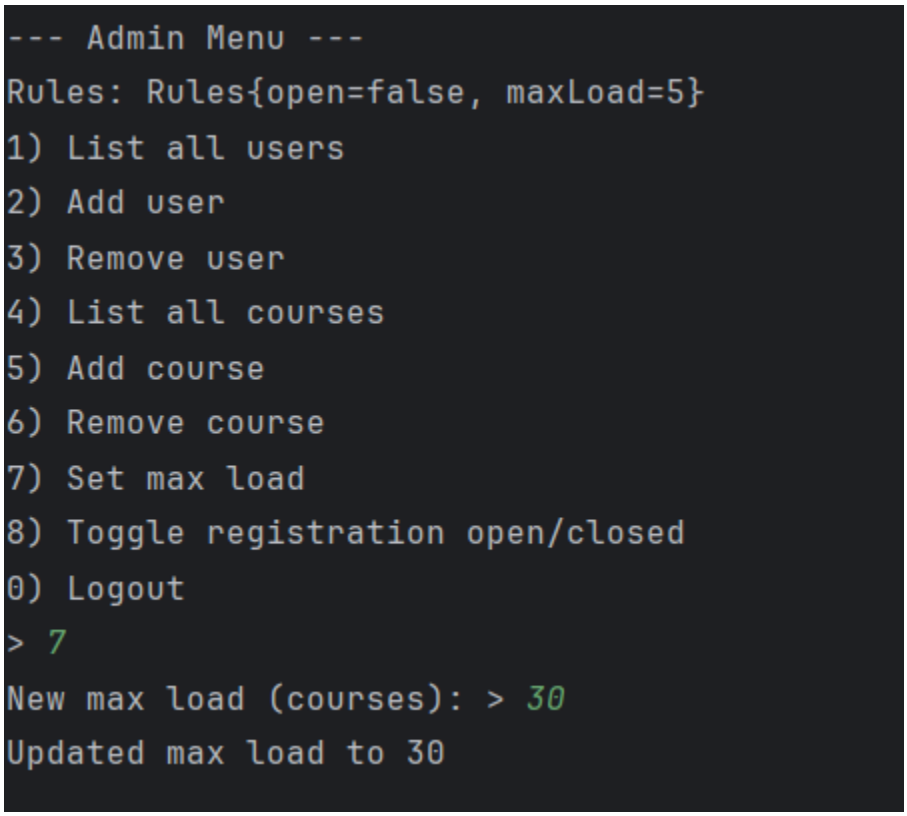
Screenshot 10: Administrator Managing Courses

```
Available Courses:
 - CS100 - Intro to Programming (Fall 2025), Cap: 0/2, Instr: Dr. Sara
   Times: [SUNDAY 09:00-11:00]
   Prereqs: []
 - CS101 - Data Structures (Fall 2025), Cap: 0/2, Instr: Dr. Sara
   Times: [SUNDAY 11:00-13:00]
   Prereqs: [CS100]
 - CS102 - Discrete Math (Fall 2025), Cap: 0/2, Instr: Dr. Sara
   Times: [SUNDAY 12:00-14:00]
   Prereqs: []
```

Screenshot 11: Administrator Managing Users



Screenshot 12: Administrator Managing Registration Rules (Max load)



Screenshot 13: Administrator Managing courses removal

```
--- Admin Menu ---
Rules: Rules{open=false, maxLoad=30}
1) List all users
2) Add user
3) Remove user
4) List all courses
5) Add course
6) Remove course
7) Set max load
8) Toggle registration open/closed
0) Logout
> 6
Course code to remove: cs100
Course removed: CS100
```

8. Class Structure Summary

The class structure of the Online Course Registration System is organized into multiple packages, each responsible for a specific part of the system. This organization helps maintain clear separation of responsibilities and improves code readability.

The `Main` class serves as the entry point of the application and is responsible for starting the system and launching the console interface. User interaction is handled through the `ConsoleUI` class, which manages menus, user input, and navigation between different system functions.

Core system logic is implemented in the service layer. Classes such as `CourseService` and `RegistrationService` handle business operations, including course management, registration processing, and validation rules. This design ensures that business logic is kept separate from user interface code.

System data is managed through the `DataStore` class, which provides in-memory storage for users, courses, and registrations. This allows the system to function without a

database while still demonstrating data handling.

The model layer contains the main domain classes that represent system entities. The `User` class defines common user attributes and is extended by `Student`, `Instructor`, and `Administrator` to support role-specific behavior. Other model classes such as `Course`, `Section`, and `Registration` represent academic structures and registration records.

Overall, the class structure follows object-oriented design principles and aligns with the system's UML diagrams, resulting in a clear and maintainable implementation

Package	Class	Responsibility
app	Main	Application entry point
ui	ConsoleUI	Menu handling and user interaction
service	CourseService	Course-related logic
service	RegistrationService	Registration workflow & validation
data	DataStore	In-memory persistence
model	User	Base user class
model	Student	Student entity
model	Instructor	Instructor entity
model	Administrator	Admin entity
model	Course	Course definition
model	Section	Course sections
model	Schedule	Time management
model	Registration	Registration record
model	RegistrationStatus	Registration state enum

9. Team Contributions

Basmala Salah

- Student workflow implementation
- Registration validation logic
- UML-to-code alignment
- Documentation support

Rana Dief

- Instructor and Administrator workflows
- Console UI integration
- System architecture & services
- GitHub repository management

10. Conclusion

Phase 4 successfully delivered a functional console-based prototype of the Online Course Registration System. The implementation adheres closely to the system design and UML diagrams, demonstrates object-oriented best practices, and fulfills all course requirements. The system effectively simulates real-world registration workflows and provides a solid foundation for future enhancements.