SOFTWARE REQUIREMENTS SPECIFICATION

for

Academia-Course Registration Portal

Version 1.0

Prepared by : 1. Akshat Batra (IMT2023025)
2. Harshita Bansal (IMT2023035)

Submitted to : Prof. SUjit Kumar Chakrabrati
Professor IIITB

November 12, 2025

Contents

1 Introduction

1.1 Purpose

It becomes very difficult to manage course registration related data manually as the number of students and courses increases every semester. Any course related information can be modified at any time such as assigning a new faculty to a course, changing the faculty later, updating the number of seats, or enrolling new students. Handling these activities through paper records or informal systems often leads to data inconsistency, confusion and delay in the registration process.

So, "Academia – Course Registration Portal" is the solution. "Academia – Course Registration Portal" is a centralized system that stores and manages information related to students, faculty and courses. The main concept of this system is to simplify and streamline course enrollment and course allocation processes, ensuring that course seat availability, user access permissions, and enrollment records are handled efficiently, consistently, and securely.

1.2 Intended Audience and Reading Suggestions

This SRS is intended for the following groups of users, each having a different purpose for referring to this document:

Audience	**Purpose of Reading**
Project Evaluators / Instructor	To review the scope, completeness, and correctness of the system requirements and verify whether they align with institutional expectations.
Developers / Programmers	To understand the system workflow, role-based functionalities, and required behaviors before implementing or modifying the system.
Testers / QA Team	To derive functional and non-functional test cases and ensure that the implemented system meets the stated requirements.
End Users (Admin / Faculty / Students)	To understand available features, allowed operations, and usage boundaries according to their respective roles.

Readers who are unfamiliar with the system are advised to begin with **Section 2 (Overall Description)** to gain a high-level understanding before referring to detailed functional and technical specifications.

1.3 Project Scope

"Academia – Course Registration Portal" creates a structured environment for Administrator, Faculty members and Students to manage course registration activities.

After logging into the system, a Student can view the list of available courses and enroll or unenroll based on seat availability. The Student can also view the list of courses in which he/she is currently enrolled.

Faculty members can create new courses, modify existing courses and view the list of students enrolled in their courses. Each course is associated with a specific faculty member responsible for maintaining that course information.

The Administrator is responsible for creating and modifying user accounts. The Administrator can also activate or block student accounts to control access based on institutional guidelines.

Figure 1.1: Entire Work Flow of Academia Course Registration Portal

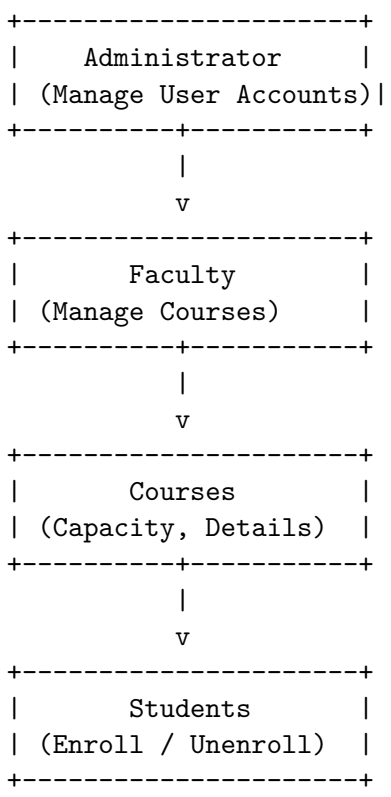


Figure 1.1 illustrates the overall workflow of the system. The relationships among entities are structured and dependent:

- Students select and enroll in Courses.
- Courses are created and managed by Faculty members.

- Faculty and Student accounts are created and controlled by the Administrator.

Thus, the system establishes a clear and interactive flow between Administrator, Faculty and Student roles.

1.4 Problem Statement

Managing course registration manually in educational institutes often becomes difficult as the number of students and courses increases every semester. Manual or partially automated systems can lead to several issues such as:

- Confusion during peak registration periods.
- Seat conflicts and inconsistencies in course capacity.
- Unauthorized or unintended course access.
- Difficulty in verifying or updating student enrollments.

"Academia – Course Registration Portal" provides a structured solution to these issues by maintaining course information, user access and enrollment data in a centralized, controlled and verifiable system. This ensures that course registration workflows are performed securely, accurately and efficiently.

1.5 Definitions, Acronyms, and Abbreviations

- **Admin**: System operator responsible for creating, modifying and managing user accounts.
- **Faculty**: Instructor responsible for creating and maintaining courses and viewing enrolled student lists.
- **Student**: End-user who enrolls into available courses.
- **Enrollment Capacity**: Maximum number of seats allowed for a course. The *seats_left* value must not become negative.
- **SRS**: Software Requirements Specification.
- **NFR**: Non-Functional Requirement.
- **CLI**: Command Line Interface used to interact with the system.

1.6 Stakeholder Analysis

Primary Stakeholders:

- **Students (End-Users):** Students use the system to view available courses, enroll or unenroll from courses and view their academic plan. Their main interest is smooth and error-free registration.
- **Faculty Members (Course Managers):** Faculty create and update courses and view the list of enrolled students. Their goal is to manage course offerings efficiently.
- **Administrator (System Controller):** The Administrator creates and modifies user accounts and controls access to the system. The Administrator ensures that the portal runs correctly and policies are maintained.

Secondary Stakeholders:

- **Instructors / Evaluators:** They review the system for academic assessment, system completeness and correct implementation.
- **Future Maintainers / Developers:** They may add more modules or migrate the system to a web/database-based environment. Their interest is clean structure and maintainability of the system.

2 Overall Description

2.1 Product Perspective

"Academia – Course Registration Portal" is a standalone client–server based application. It does not depend on any external database or third-party services. All data such as user accounts, course details and enrollment records are stored in local data files on the server machine.

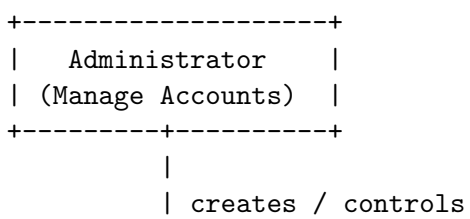
The system follows a role-based access structure where each user type (Administrator, Faculty and Student) interacts with the system through a Command Line Interface (CLI). The client sends user requests to the server through TCP socket communication, and the server processes the request and returns the output. The server ensures that the data remains consistent and that no invalid or unauthorized operations take place.

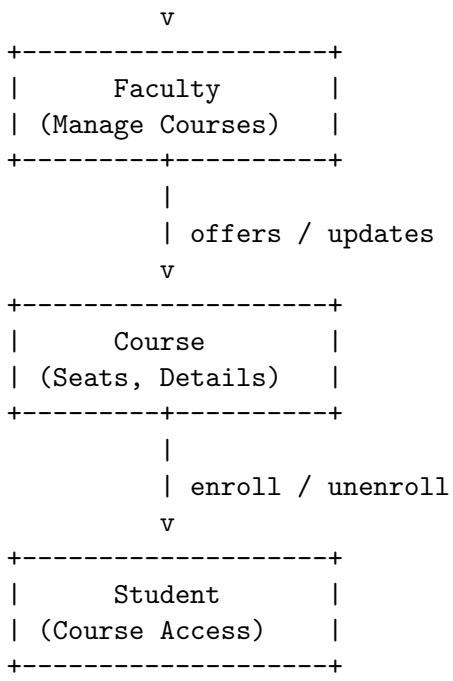
2.2 User Classes and Characteristics

The system contains three primary user classes:

- **Administrator (Admin)**
  Responsible for creating user accounts, modifying user information and activating or blocking student accounts. This user has the highest control privileges in the system.
- **Faculty**
  Responsible for creating and updating course details. Faculty can also view the list of students enrolled in their courses. Faculty members have limited access, restricted only to the courses they manage.
- **Student**
  Can view the available list of courses and enroll/unenroll based on seat availability. Students only interact with the system to manage their enrolled courses.

User Roles Diagram (UML - Text Representation)





2.3 Product Functions

The major functions of the system include:

- **Login and Authentication** using unique user credentials.
- **User Management (Admin)**: Add/Modify users and control student access.
- **Course Management (Faculty)**: Create/Update courses and view enrolled student lists.
- **Course Catalog Viewing (All Roles)**: Display all available courses.
- **Enrollment Operations (Student)**: Enroll or unenroll from courses based on availability.

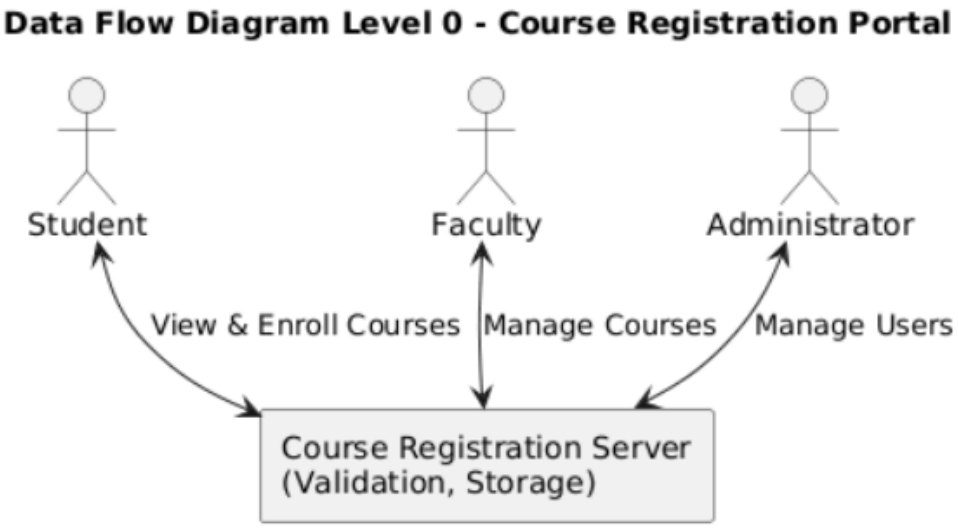


Figure 2.1: Data Flow Diagram – Level 0: Course Registration System (Context Diagram)

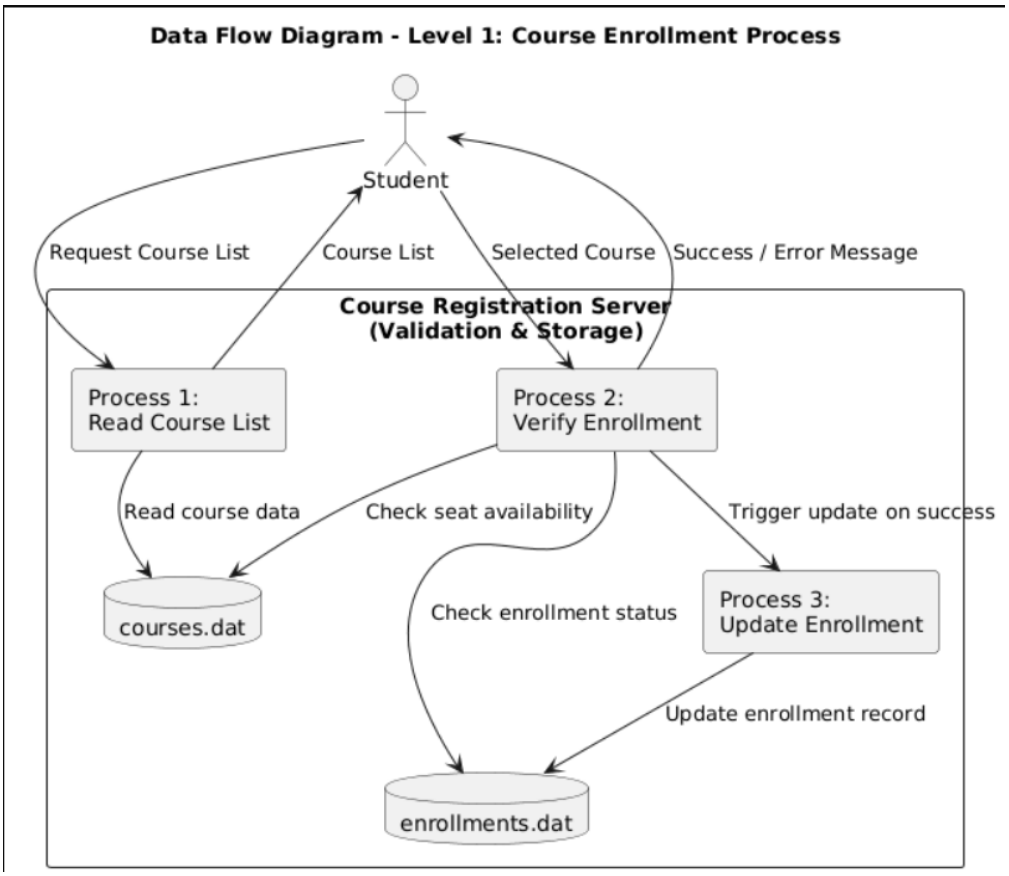


Figure 2.2: Data Flow Diagram – Level 1: Course Enrollment Process

2.4 Operating Environment

- Operating System: Linux (Ubuntu recommended)
- Programming Language: C (GCC Compiler)
- Interface: Command Line Interface (CLI)
- Communication: TCP/IP Socket (Server runs on Port 8080)
- Storage: Local binary data files (`users.dat`, `courses.dat`, `enrollments.dat`)

2.5 Design

The system design is role-based, where each user interacts through menu-driven screens.

Student Activity Diagram (UML - Text Representation)

```
Login -> View Courses -> Select Enroll/Unenroll -> Server Updates Records -> End
```

Faculty Activity Diagram

```
Login -> Create/Update Course -> View Enrolled Students -> Logout
```

Administrator Activity Diagram

```
Login -> Add/Modify Users -> Activate/Block Accounts -> Logout
```

2.6 Design and Implementation Constraints

- The system uses a file-based data storage model instead of a relational database.
- The server must always be running before any client can connect.
- Only one server instance should modify the data files at a time to avoid data corruption.
- The user interface is limited to a Command Line Interface (CLI).
- Network communication is restricted to TCP/IP over a fixed port (8080).

2.7 Assumptions and Dependencies

- Users are assumed to have basic familiarity with command-line operations.
- The server machine must have read/write permissions to store data files.
- Students will have valid institutional credentials to access the system.

- The system depends on stable network connectivity between client and server.

2.8 Requirement Elicitation Summary

The requirements of this system were gathered through:

- Observation of the existing manual course registration workflow.
- Understanding of role responsibilities (Admin, Faculty, Student).
- Iterative review of system behavior during OS laboratory development stages.
- Informal discussion and feedback sessions with peers and teaching staff.

These requirements were refined into functional and non-functional requirements described in the following sections.

3 System Features

"Academia – Course Registration Portal" is a course enrollment management system. The main purpose of this product is to allow students to enroll in courses, faculty to manage courses, and the administrator to control system access.

3.1 Description and Priority

"Academia – Course Registration Portal" contains multiple role-based features. All features are necessary to ensure smooth and controlled course registration. The features listed below are arranged in priority order (highest to lowest):

1. **User Authentication and Login:** This is the primary entry point. Every user must log in with valid credentials. (High Priority)
2. **Student Enrollment Management:** Students can view available courses and enroll or unenroll based on seat availability. This is the central functional feature of the software. (High Priority)
3. **Course Management (Faculty):** Faculty members create new courses, update course information and view list of enrolled students. (Medium Priority)
4. **User Account Management (Administrator):** Administrator creates and modifies user accounts and activates/blocks student access if required. (Medium Priority)
5. **Course Catalog Viewing:** All users can view the complete list of available courses along with seat availability. (Supporting Feature)

3.2 System Architecture / Implementation Overview

The "Academia – Course Registration Portal" is implemented using C programming language with file-based storage and client–server communication using TCP sockets.

- **Back-End (Server):** Written in C, responsible for handling authentication, file read/write, and all business logic.
- **Front-End (Client):** Command Line Interface (CLI) where users interact through text menus.

- **Data Storage:** Data is stored in structured binary files such as `users.dat`, `courses.dat`, and `enrollments.dat`.
- **Communication:** The client and server exchange information over TCP/IP using a predefined message protocol.

3.3 Functional Requirements

The functional requirements are categorized based on the role of the user: Administrator, Faculty and Student. Each requirement is uniquely numbered and assigned a priority where:

- **High Priority**: Core system functionality; must be implemented.
- **Medium Priority**: Important functionality that supports main features.
- **Low Priority**: Enhancement or additional convenience.

Administrator Functional Requirements

- **FR-01 (High)**: The system shall allow Administrator to log in using a valid username and password.
- **FR-02 (High)**: The system shall allow Administrator to create new user accounts (Student or Faculty).
- **FR-03 (Medium)**: The system shall allow Administrator to modify user information.
- **FR-04 (High)**: The system shall allow Administrator to activate or block student accounts.
- **FR-05 (Medium)**: The system shall prevent Administrator from deleting system-critical accounts (e.g., own account).

Faculty Functional Requirements

- **FR-06 (High)**: The system shall allow Faculty to log in using valid credentials.
- **FR-07 (High)**: The system shall allow Faculty to create and register new courses.
- **FR-08 (Medium)**: The system shall allow Faculty to update course details such as course name and seat capacity.
- **FR-09 (High)**: The system shall allow Faculty to view the list of students enrolled in their courses.
- **FR-10 (Medium)**: The system shall prevent Faculty from modifying users or other faculty-owned courses.

Student Functional Requirements

- **FR-11 (High)**: The system shall allow Students to log in using valid credentials.
- **FR-12 (High)**: The system shall allow Students to view the complete list of available courses with seat availability.
- **FR-13 (High)**: The system shall allow Students to enroll in a course if:
  - seats are available, and
  - the student is not already enrolled in that course.
- **FR-14 (High)**: The system shall allow Students to unenroll from a course they are currently enrolled in.
- **FR-15 (Medium)**: The system shall allow Students to view a list of all courses they are currently enrolled in.

General System Functional Requirements

- **FR-16 (High)**: The system shall authenticate all users before providing access to any feature.
- **FR-17 (High)**: The system shall maintain data consistency when multiple clients access the server at the same time.
- **FR-18 (High)**: The system shall ensure that course seat count never becomes negative.
- **FR-19 (Medium)**: The system shall store updated data immediately after every successful operation.
- **FR-20 (Low)**: The system may display informative error messages for invalid operations.

4 Other Nonfunctional Requirements

4.1 Usability Requirements

The system provides a clear, menu-driven Command Line Interface (CLI) so that users with minimal technical skill can operate it. All options are numbered and displayed in a structured format to ensure ease of navigation. Error and confirmation messages are simple and informative to guide the user.

4.2 Performance Requirements

- The system should respond to standard operations (login, view courses, enroll) in real time on a local network.
- The server must be able to support multiple clients accessing it concurrently.
- File updates (such as enrollment changes) must occur immediately after the operation.

4.3 Reliability Requirements

- The system shall maintain data consistency during simultaneous access.
- The system shall prevent data corruption by controlling write access to data files.
- The system shall perform validity checks before every update to ensure error-free execution.

4.4 Security Requirements

- The system must validate user identity using username and password.
- Students cannot perform faculty or administrator tasks.
- Unauthorized access or data manipulation must be strictly restricted.

4.5 Software Quality Attributes

Maintainability: The codebase is modular (authentication, user management, course management), making it easier to update or extend.

Portability: The system runs on any Linux machine with GCC and POSIX sockets.

Scalability: New users and courses can be added without structural changes.

4.6 Business Rules

- A student cannot enroll in the same course more than once.
- A course cannot exceed its maximum seat capacity.
- Only the faculty assigned to a course may view or manage its roster.
- Only the administrator may create or update user accounts.
- Every system action must be validated before updating shared files.

5 Other Requirements

5.1 Hardware Requirements

- A standard computer capable of running a Linux operating system.
- Minimum 2 GB RAM and 50 MB of free storage for program files and data.
- Network connectivity for client–server communication.

5.2 Software Requirements

- Operating System: Linux (Ubuntu recommended)
- Compiler: GCC (GNU Compiler Collection)
- Libraries: Standard C Libraries and POSIX socket support
- Terminal environment for executing client and server programs

5.3 Communication Requirements

- The server and client must communicate over TCP/IP.
- The default port used by the server is Port 8080.
- Both server and client programs must run on the same network or properly routed remote network.

5.4 Backup and Recovery Requirements

- The system data is stored in local files (`users.dat`, `courses.dat`, `enrollments.dat`).
- Periodic manual backup is recommended to prevent data loss.
- In case of corruption, files may be restored from the latest backup without reinitializing the system.

5.5 Safety Requirements

- System should prevent incomplete file writes by ensuring that updates are atomic.
- Any unexpected termination should not leave data in an inconsistent state.

6 Use Case Description

Use Case	Actor	Steps
Login to System	All Users	Enter username and password → System verifies credentials → Display role-based menu
View Course Catalog	Student / Faculty / Admin	Request list of courses → Server reads courses.dat → Display course name, faculty, seats available
Enroll in Course	Student	Select course → System checks seat availability and existing enrollment → If valid, enroll student and reduce seat count → Confirmation displayed
Unenroll from Course	Student	Select course from enrolled list → System validates enrollment → Remove enrollment and increase seat count → Confirmation displayed
View Enrolled Students	Faculty	Faculty selects a course → Server retrieves list of enrolled students → Display roster
Create Course	Faculty	Faculty enters course details (course name, capacity) → Server stores new course in courses.dat
Modify Course	Faculty	Faculty selects course → Update seat capacity or course name → Server writes changes to file
Add User Account	Administrator	Start user creation → Enter role, name, credentials → Server stores new user in users.dat
Activate/Block User	Administrator	Select user → Change account status → Server updates user flag in users.dat
Logout	All Users	User selects logout option → Session terminated and connection safely closed

Table 6.1: Use Case Descriptions

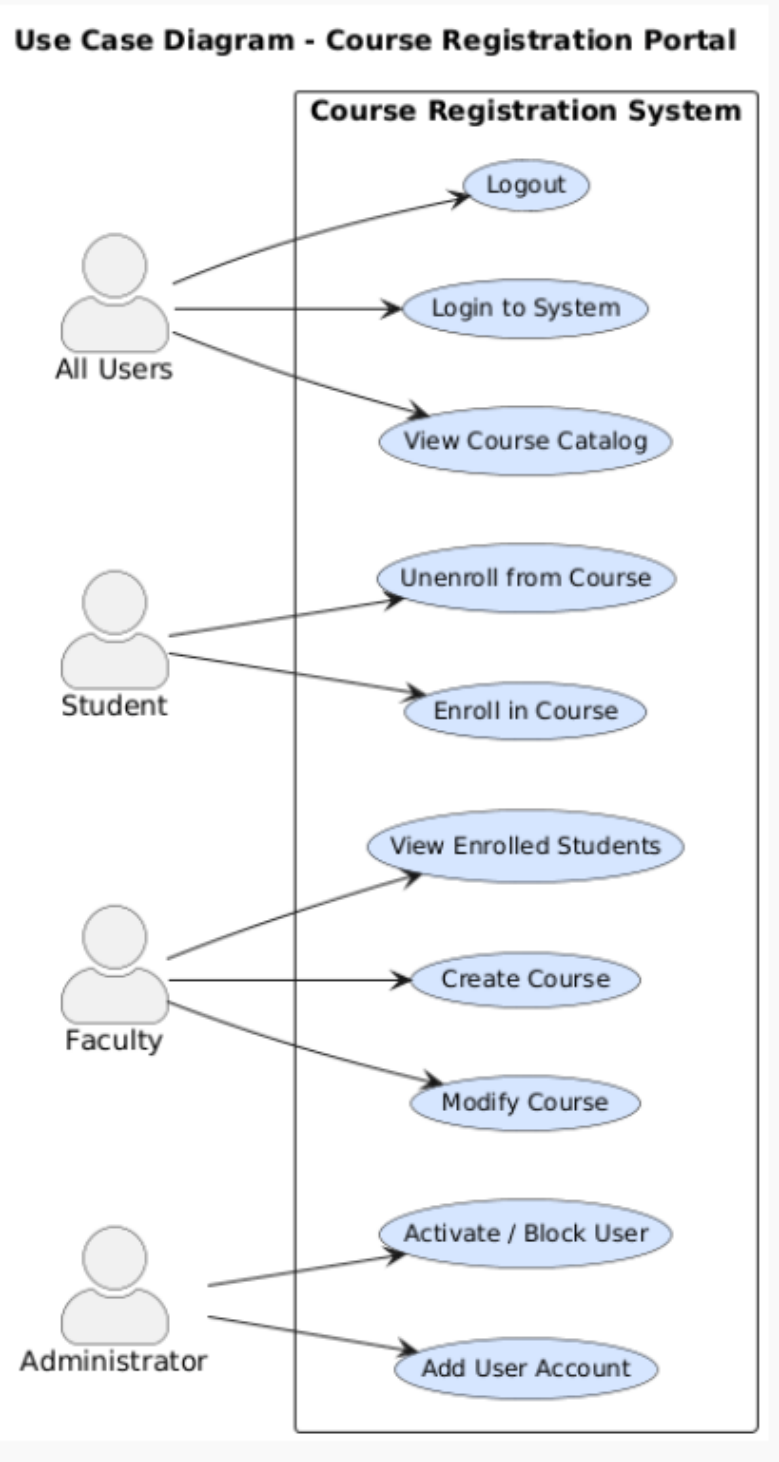


Figure 6.1: Use Case Diagram – Course Registration Portal

7 Diagrams

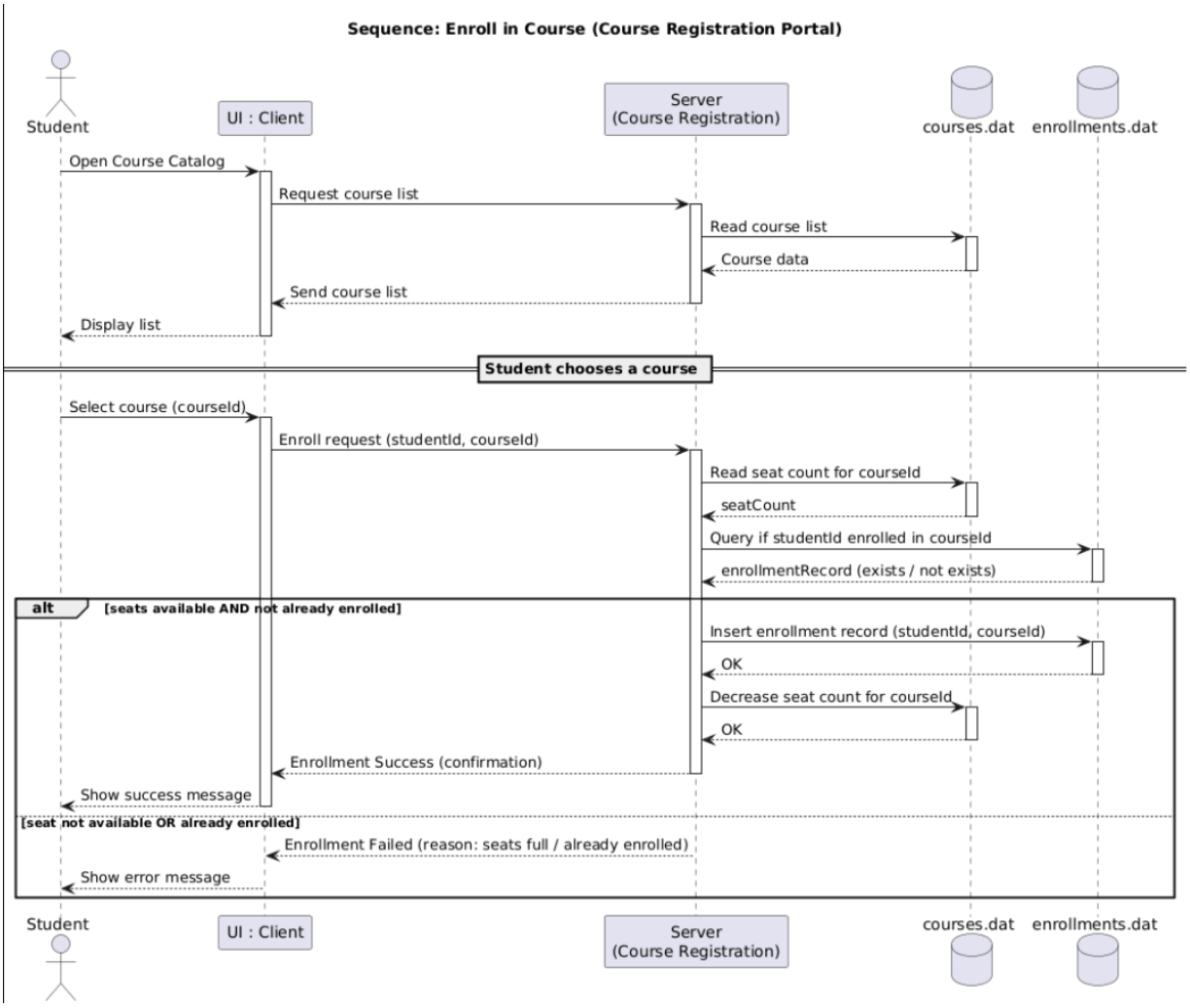


Figure 7.1: Sequence Diagram – Course Registration Portal

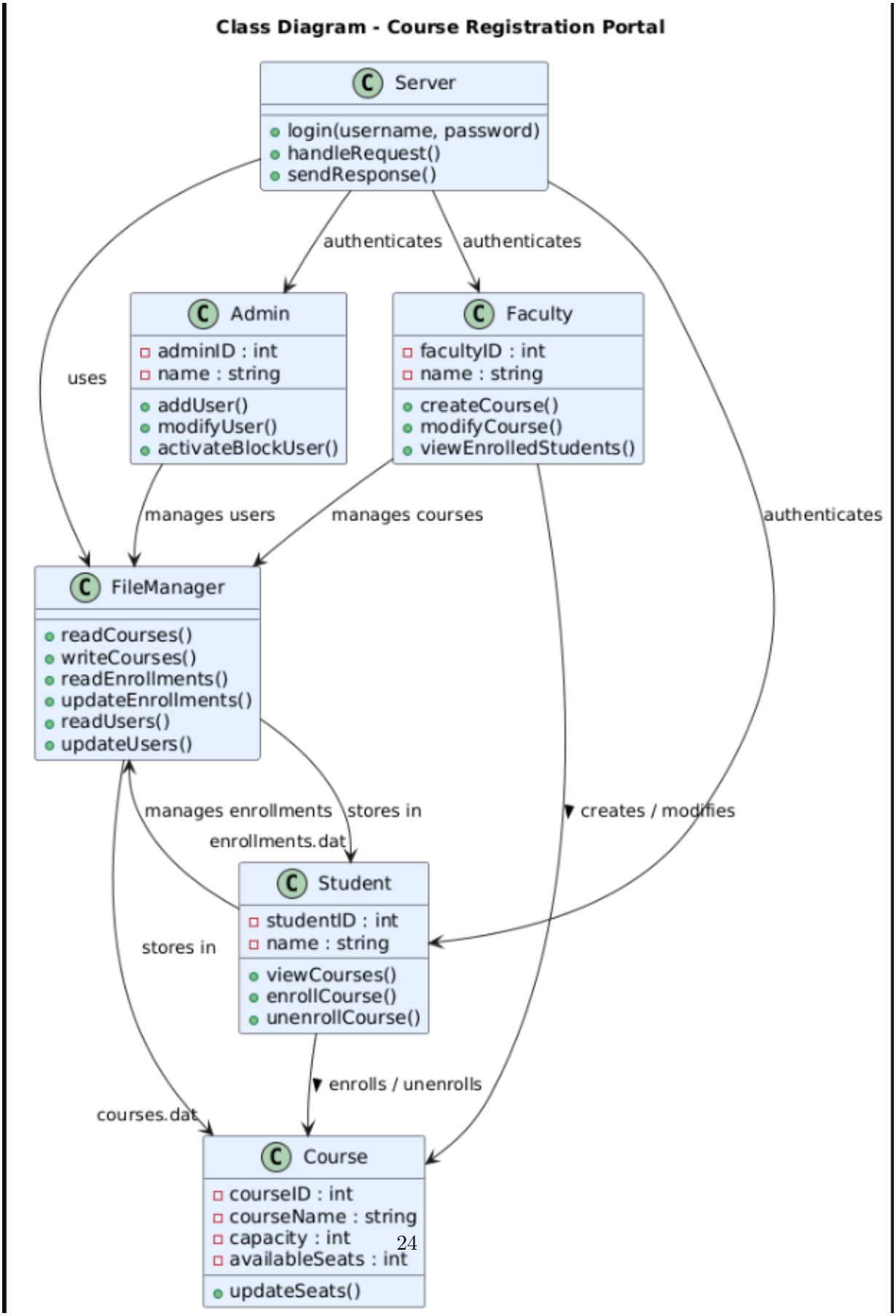


Figure 7.2: Class Diagram – Course Registration Portal

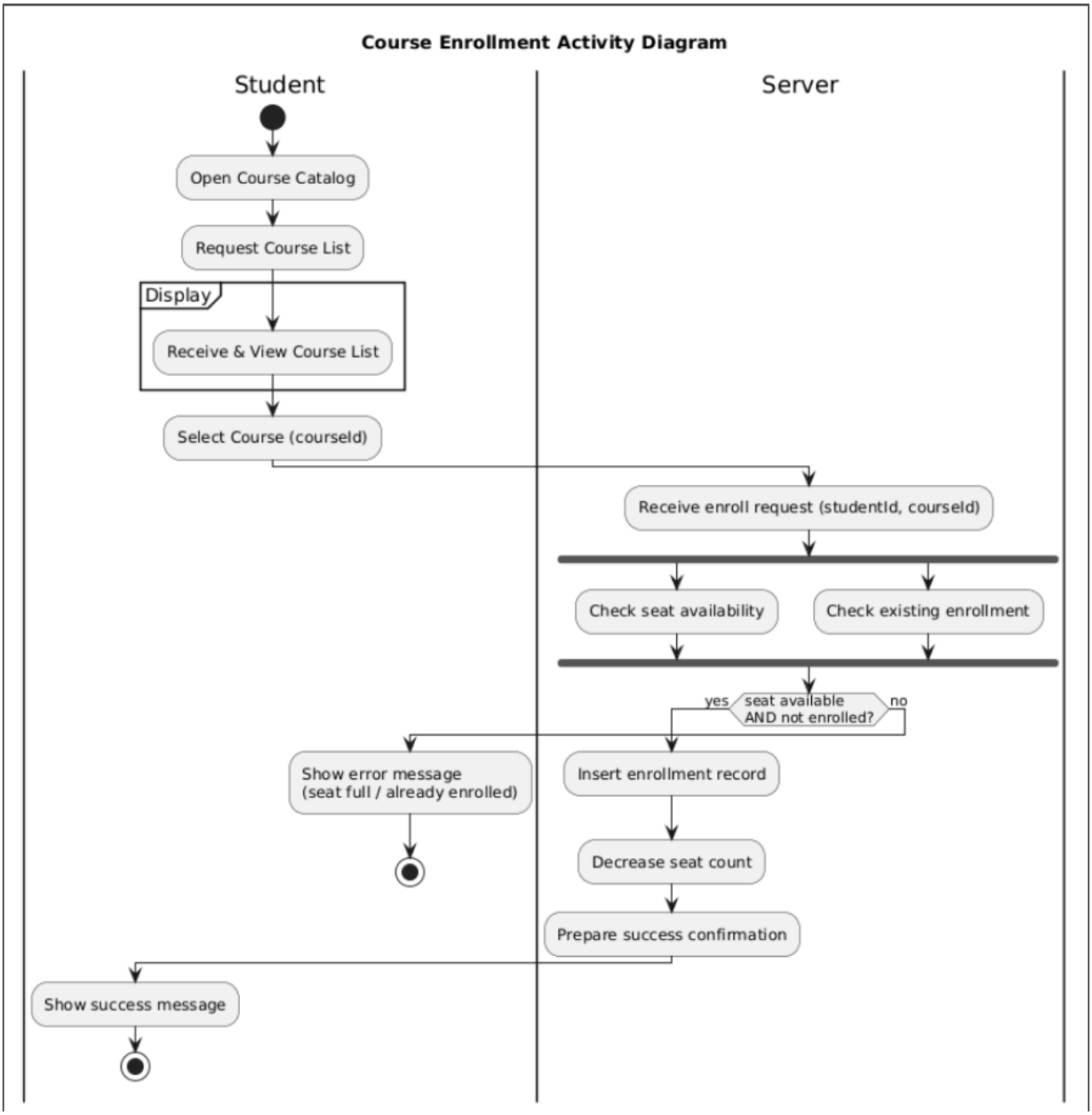


Figure 7.3: Activity Diagram – Course Registration Portal

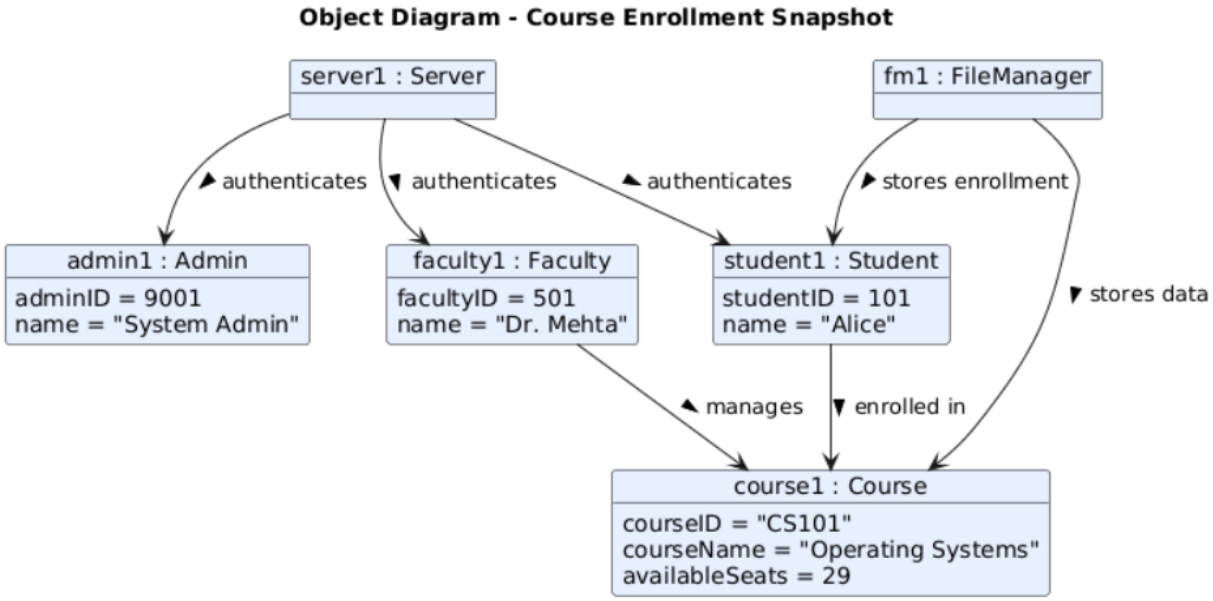


Figure 7.4: Object Diagram – Course Registration Portal

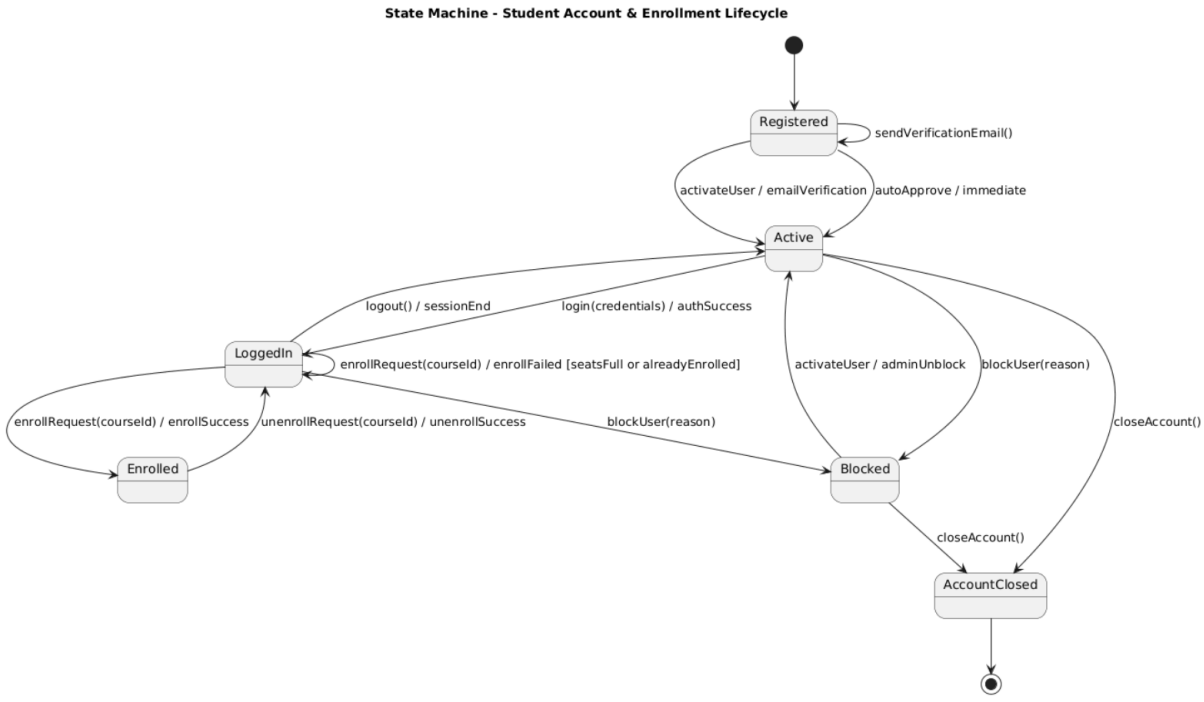


Figure 7.5: State Diagram – Course Registration Portal