28TH APRIL 2025

FINAL REPORT

ONLINE COURSE REGISTRATION SYSTEM

Prepared by:
Zainab Fatima (22i-1064)
Shayane Zainab (22i-1049)

INTRODUCTION

EnrollMate is a course enrollment and registration system designed to streamline the process of enrolling and dropping courses for students. It ensures that students can only enroll in classes with available seats and provides a user-friendly interface for course management.

PURPOSE OF THIS PROJECT

EnrollMate aims to automate and simplify the course enrollment process for students, faculty, and administrators. The system ensures efficient course registration, seat allocation, and real-time updates, reducing manual effort and administrative burden while improving accessibility and ease of use.

SCOPE OF THIS PROJECT

The system enables students to view, enroll in, and drop courses effortlessly while ensuring that faculty members can manage course enrollments and communicate important announcements to students. Administrators have the ability to add, update, and remove courses while monitoring seat availability in real time.

Additionally, the system prevents over-enrollment by displaying seat limits and sending notifications for enrollment deadlines. The project is designed as a web-based platform with role-based access to ensure secure and efficient course management.

OVERVIEW

EnrollMate consists of three primary user roles:

- **Students:** Can search, filter, enroll in, and drop courses.
- **Administrators:** Can manage course offerings and generate reports.

INTERFACE REQUIREMENTS

The system will feature a responsive web-based UI that is accessible from desktops, tablets, and mobile devices. It will include a student dashboard where users can view their enrolled courses and upcoming deadlines. Faculty members will have access to a dedicated portal for managing their courses and student rosters. Administrators will utilize a panel for adding, updating, and removing courses efficiently. Additionally, the system will provide real-time notifications and status updates to keep users informed of critical information.

04

FUNCTIONAL REQUIREMENTS

1. **Course Viewing:** The system shall allow students to view available courses, including seat availability, faculty details, and schedules.
2. **Course Enrollment:** The system shall enable students to enroll in a course if seats are available, ensuring proper validation of prerequisites.
3. **Course Drop:** The system shall allow students to drop a course before the deadline and update available seat counts accordingly.
4. **Faculty Course Management:** The system shall enable faculty members to manage course rosters, view student enrollments, and send announcements.
5. **Administrator Course Control:** Administrators shall have the ability to add, update, and remove courses while monitoring enrollment trends in real time.
6. **Seat Availability Notifications:** The system shall display a "Seats Full" message when a course reaches capacity and prevent additional enrollments.
7. **Enrollment Notifications:** The system shall send automated notifications to students regarding enrollment deadlines, seat availability updates, and course-related announcements.

PERFORMANCE REQUIREMENTS

1. The system should process enrollment requests within 2 seconds.
2. Must support up to 10,000 concurrent users.
3. Course availability should update in real time.
4. Data retrieval should not exceed 3 seconds for any query.
5. The system should maintain 99.9% uptime.

05

USER STORIES

Students:

1. As a student, I want to view available courses so that I can choose which ones to enroll in.
2. As a student, I want to enroll in a course so that I can attend classes and earn credits.
3. As a student, I want to drop a course so that I can adjust my schedule as needed.
4. As a student, I want to receive notifications and reminders about enrollment deadlines so that I don’t miss important dates.
5. As a student, I want to view my enrolled courses in a dashboard so that I can easily track my schedule.

Admininstrator:

1. As an administrator, I want to manage course seat limits so that classes do not exceed capacity.
2. As an administrator, I want to generate reports on student enrollments so that I can analyze trends and make decisions.
3. As an administrator, I want to add, update, or remove courses from the system so that the course catalog remains current.
4. As an administrator, I want to oversee student enrollment trends so that I can predict demand for future semesters.
5. As an administrator, I want to set enrollment deadlines and rules so that students follow structured registration processes.

USE CASES

1. **View Available Courses**
   - **Actor:** Student
   - **Description:** Allows students to browse available courses with details like schedule and seat availability.
   - **Preconditions:** The student must be logged in.
   - **Postconditions:** The system displays the list of available courses.
2. **Enroll in a Course**
   - **Actor:** Student
   - **Description:** Enables students to enroll in a course with available seats.
   - **Preconditions:** The student must be logged in and meet course prerequisites.
   - **Postconditions:** The student is enrolled in the selected course.
3. **Drop a Course**
   - **Actor:** Student
   - **Description:** Allows students to drop an enrolled course before the deadline.
   - **Preconditions:** The student must be enrolled in the course.
   - **Postconditions:** The course is removed from the student's schedule, and the seat count is updated.
4. **View Enrolled Courses Dashboard**
   - **Actor:** Student
   - **Description:** Displays a student’s enrolled courses in a dashboard.
   - **Preconditions:** The student must be logged in.
   - **Postconditions:** The system presents the enrolled courses.

USE CASES

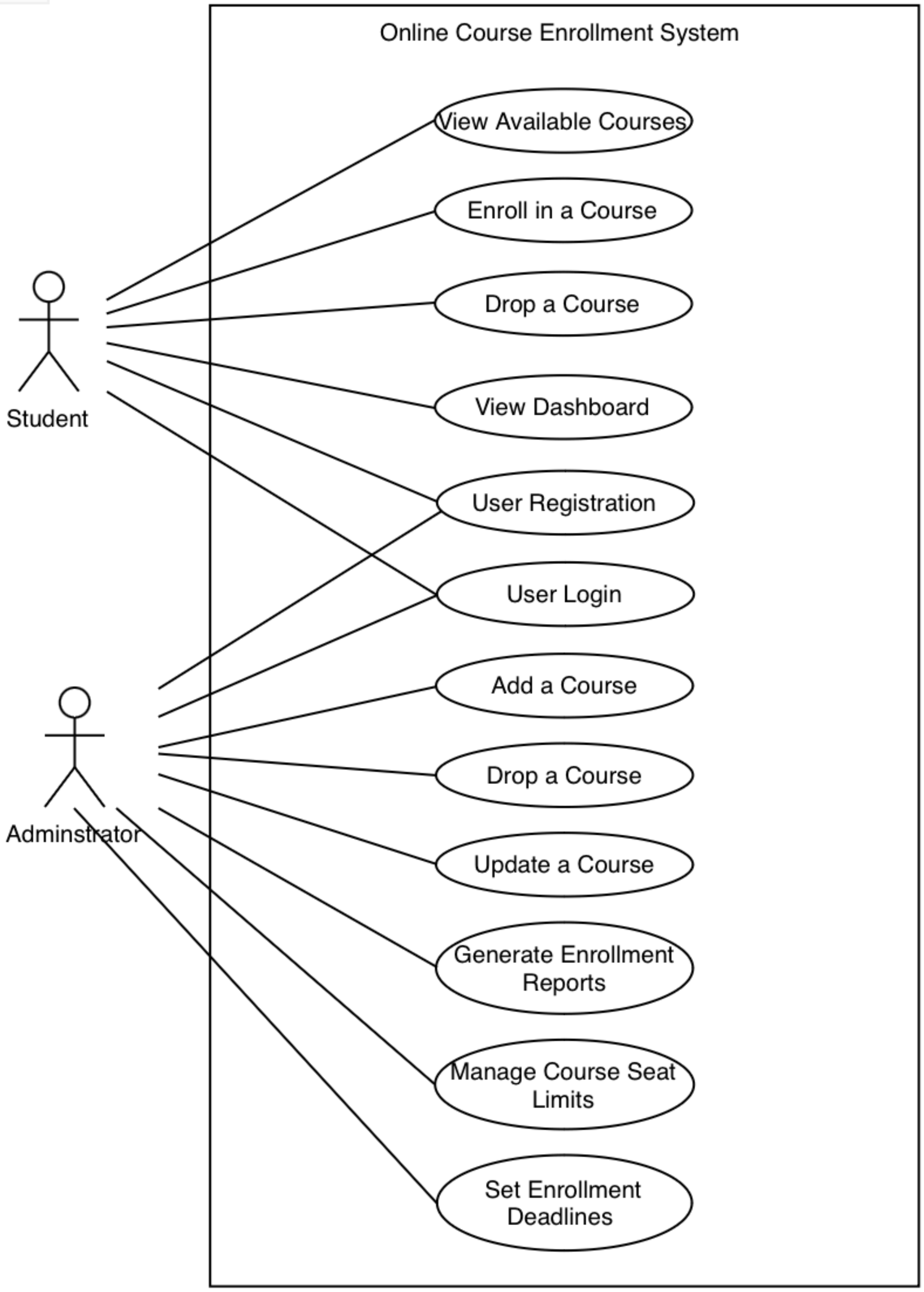
5. **User Registration**
    - **Actor:** Student, Administrator
    - **Description:** Allows new users to register an account in the system.
    - **Preconditions:** User must provide valid credentials and required information.
    - **Postconditions:** A new account is created, and the user can log in.
6. **User Login**
    - **Actor:** Student, Administrator
    - **Description:** Enables users to log into the system securely.
    - **Preconditions:** The user must be registered in the system.
    - **Postconditions:** The user gains access to their respective dashboard.
7. **Add a Course**
    - **Actor:** Administrator
    - **Description:** Allows administrators to add a new course to the catalog.
    - **Preconditions:** Administrator must be logged in.
    - **Postconditions:** A new course is added and made available for enrollment.
8. **Update a Course**
    - **Actor:** Administrator
    - **Description:** Allows administrators to modify existing course details.
    - **Preconditions:** The course must exist in the system.
    - **Postconditions:** The course details are updated and saved.

USE CASES

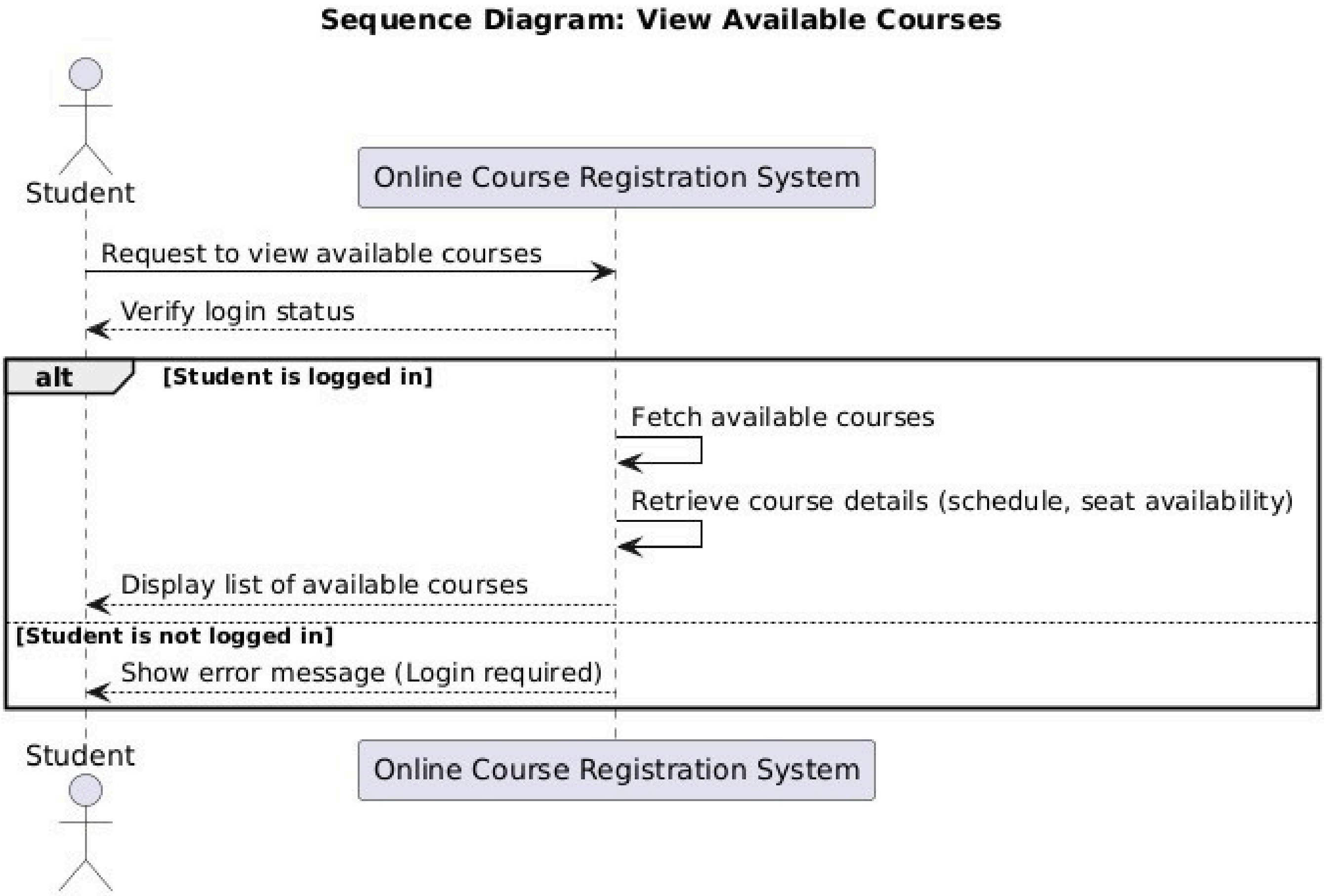
9. **Manage Course Seat Limits**
    - **Actor:** Administrator
    - **Description:** Allows administrators to set and update course seat limits.
    - **Preconditions:** The course must exist in the system.
    - **Postconditions:** Seat limits are updated and enforced.
10. **Generate Enrollment Reports**
    - **Actor:** Administrator
    - **Description:** Provides reports on student enrollments and trends.
    - **Preconditions:** The system must have stored enrollment data.
    - **Postconditions:** Reports are generated and displayed.
11. **Remove a Course**
    - **Actor:** Administrator
    - **Description:** Allows administrators to remove a course from the catalog.
    - **Preconditions:** The course must exist in the system, and no students should be enrolled in it.
    - **Postconditions:** The course is deleted from the catalog.
12. **Set Enrollment Deadlines and Rules**
    - **Actor:** Administrator
    - **Description:** Allows administrators to configure registration deadlines and policies.
    - **Preconditions:** The administrator must have system access.
    - **Postconditions:** The system enforces registration deadlines for students.

09

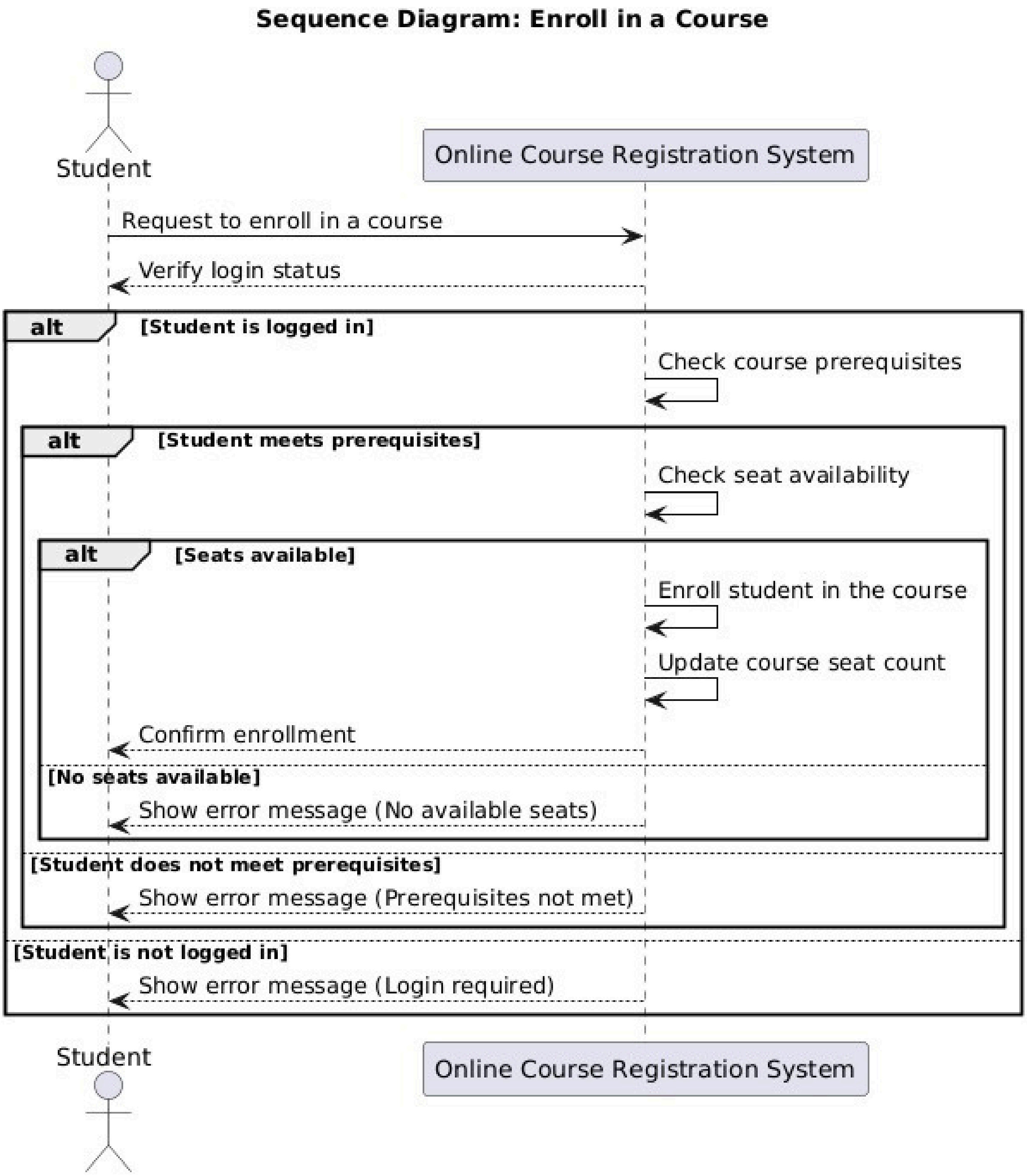
USE CASE DIAGRAM



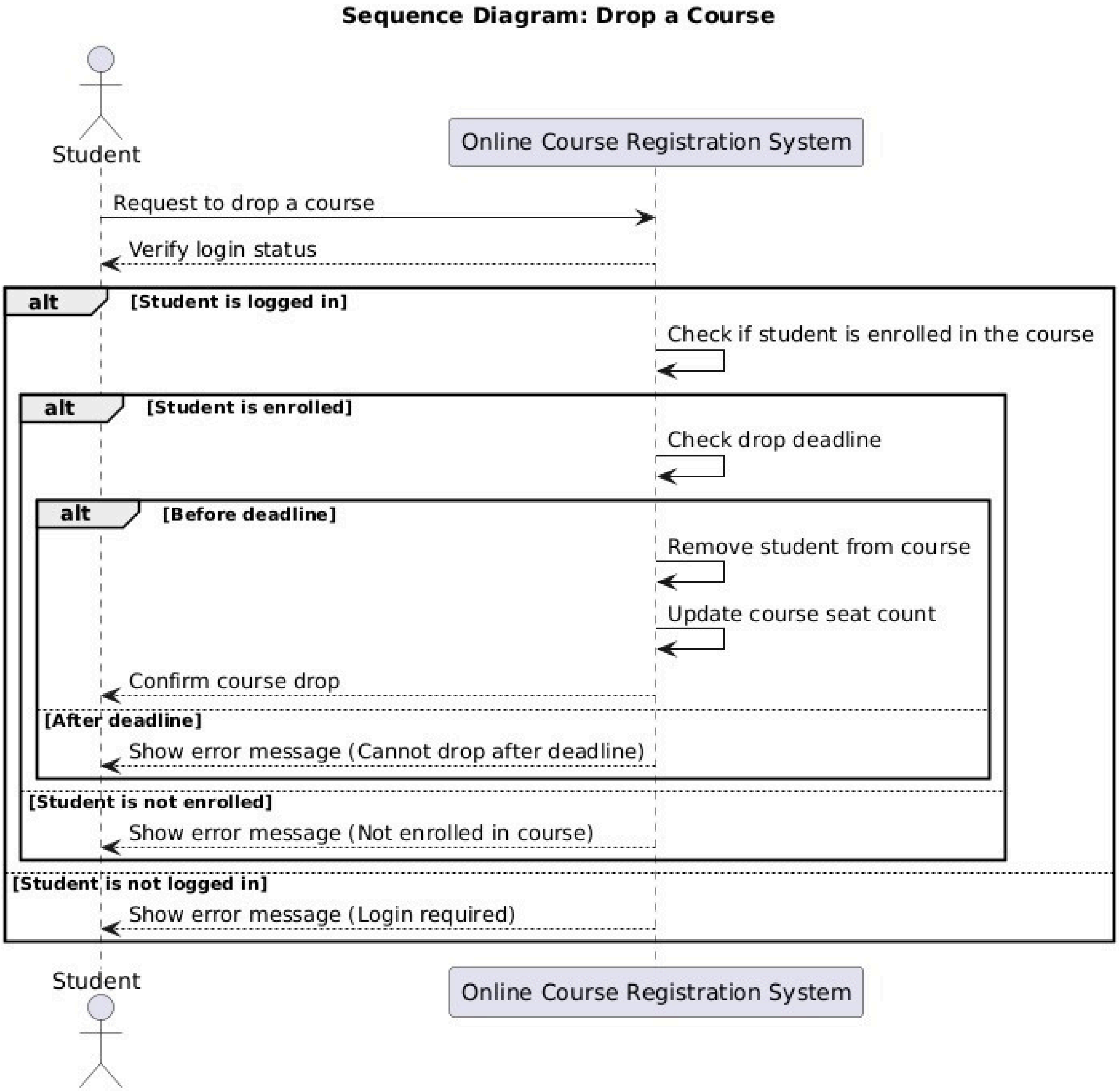
SEQUENCE DIAGRAM: USE CASE 1


Sequence Diagram: View Available Courses
Student
Online Course Registration System
Request to view available courses
Verify login status
alt
[Student is logged in]
Fetch available courses
Retrieve course details (schedule, seat availability)
Display list of available courses
[Student is not logged in]
Show error message (Login required)
Student
Online Course Registration System

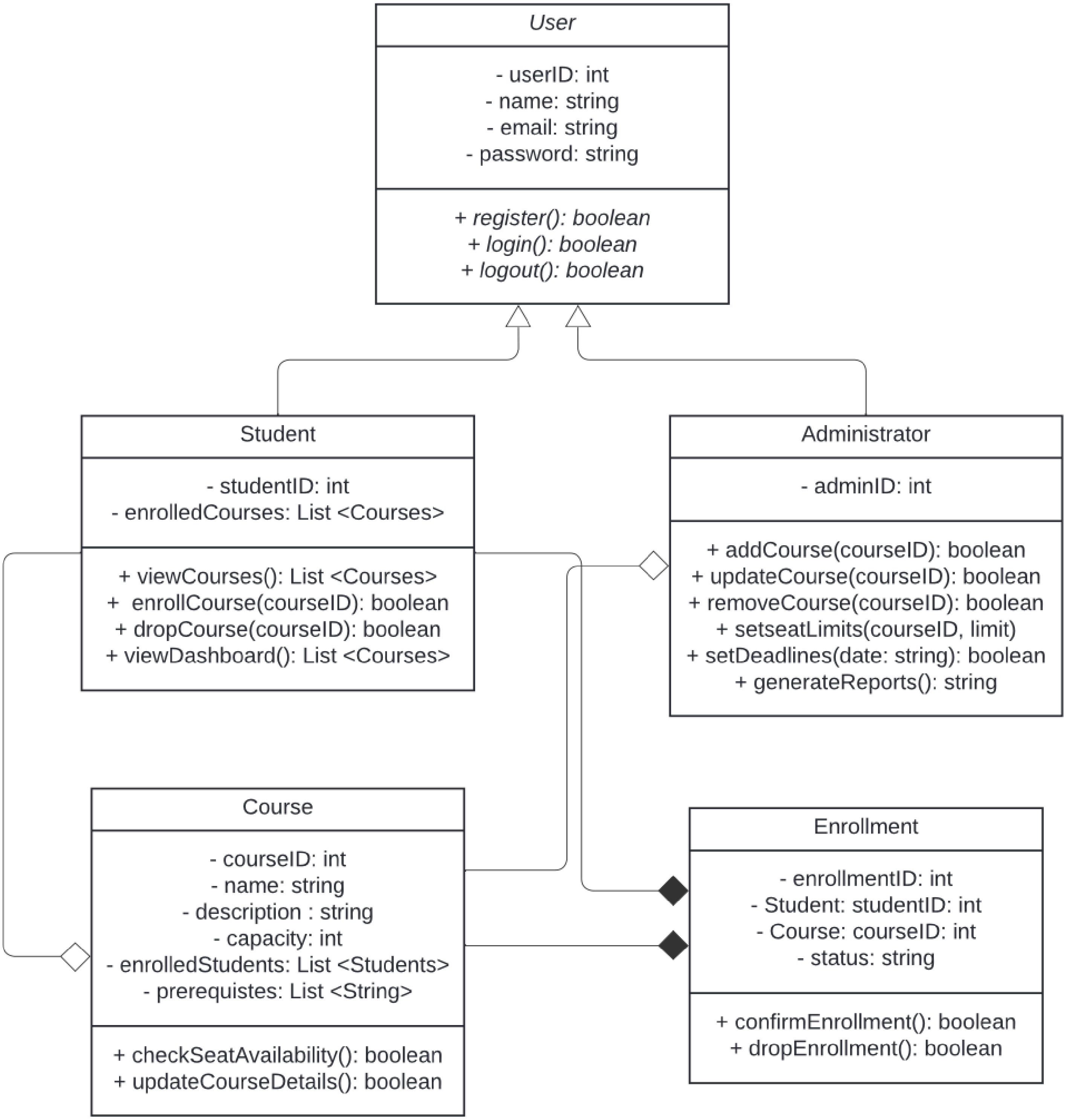
SEQUENCE DIAGRAM: USE CASE 2


Sequence Diagram: Enroll in a Course
Student
Online Course Registration System
Request to enroll in a course
Verify login status
alt
[Student is logged in]
Check course prerequisites
alt
[Student meets prerequisites]
Check seat availability
alt
[Seats available]
Enroll student in the course
Update course seat count
Confirm enrollment
[No seats available]
Show error message (No available seats)
[Student does not meet prerequisites]
Show error message (Prerequisites not met)
[Student is not logged in]
Show error message (Login required)
Student
Online Course Registration System

SEQUENCE DIAGRAM: USE CASE 3


Sequence Diagram: Drop a Course
Student
Online Course Registration System
Request to drop a course
Verify login status
alt
[Student is logged in]
Check if student is enrolled in the course
alt
[Student is enrolled]
Check drop deadline
alt
[Before deadline]
Remove student from course
Update course seat count
Confirm course drop
[After deadline]
Show error message (Cannot drop after deadline)
[Student is not enrolled]
Show error message (Not enrolled in course)
[Student is not logged in]
Show error message (Login required)
Student
Online Course Registration System

UML CLASS DIAGRAM


User
- userID: int
- name: string
- email: string
- password: string
+ register(): boolean
+ login(): boolean
+ logout(): boolean
Student
- studentID: int
- enrolledCourses: List <Courses>
+ viewCourses(): List <Courses>
+ enrollCourse(courseID): boolean
+ dropCourse(courseID): boolean
+ viewDashboard(): List <Courses>
Administrator
- adminID: int
+ addCourse(courseID): boolean
+ updateCourse(courseID): boolean
+ removeCourse(courseID): boolean
+ setseatLimits(courseID, limit)
+ setDeadlines(date: string): boolean
+ generateReports(): string
Course
- courseID: int
- name: string
- description : string
- capacity: int
- enrolledStudents: List <Students>
- prerequistes: List <String>
+ checkSeatAvailability(): boolean
+ updateCourseDetails(): boolean
Enrollment
- enrollmentID: int
- Student: studentID: int
- Course: courseID: int
- status: string
+ confirmEnrollment(): boolean
+ dropEnrollment(): boolean

SPRINT 1 BACKLOG

The goal is to develop the fundamental authentication, course management functionalities.

User Stories	Priority	Estimated Effort
User Registration	H	3
User Login	H	2
Add a Course	H	3
Update a Course	H	3
Remove a Course	H	3
Manage Seat Availabilty	H	4
Set Deadlines and Rules	M	5

SPRINT 2 BACKLOG

the goal is to develop the fundamental course enrollment functionalities and enhance user experience.

User Stories	Priority	Estimated Effort
View Available Courses	H	3
Enroll in a Course	H	5
Drop a Course	H	4
View Enrolled Courses Dashboard	M	3
Receive Enrollment Notifications	L	2
Generate Enrollment Reports	L	2

SPRINT 3 BACKLOG

Final testing, optimizations, and UI/UX improvements.

User Stories	Priority	Estimated Effort
UI/UX Enhancements	L	3
Security Enhancements	H	5
Final Testing and Bug Fixes	H	5

WORK BREAKDOWN STRUCTURE

Level 1: Project Management

- Project planning, scheduling, and monitoring.

Level 2: Requirements Gathering

- Identify Functional Requirements (Registration, Login, Course Management)
- Identify Non-Functional Requirements (Performance, Security, Scalability)

Level 3: Design Phase

- Database Design (Users, Courses, Enrollments)
- System Architecture Design (Class Diagrams, Use Case Diagrams)

Level 4: Development Phase

- Authentication Module (Register, Login)
- Course Management Module (Add, Update, Remove Courses)
- Enrollment Management Module (Enroll/Drop Courses)

Level 5: Testing Phase

- Unit Testing (Test individual modules)
- Integration Testing (Test system as a whole)

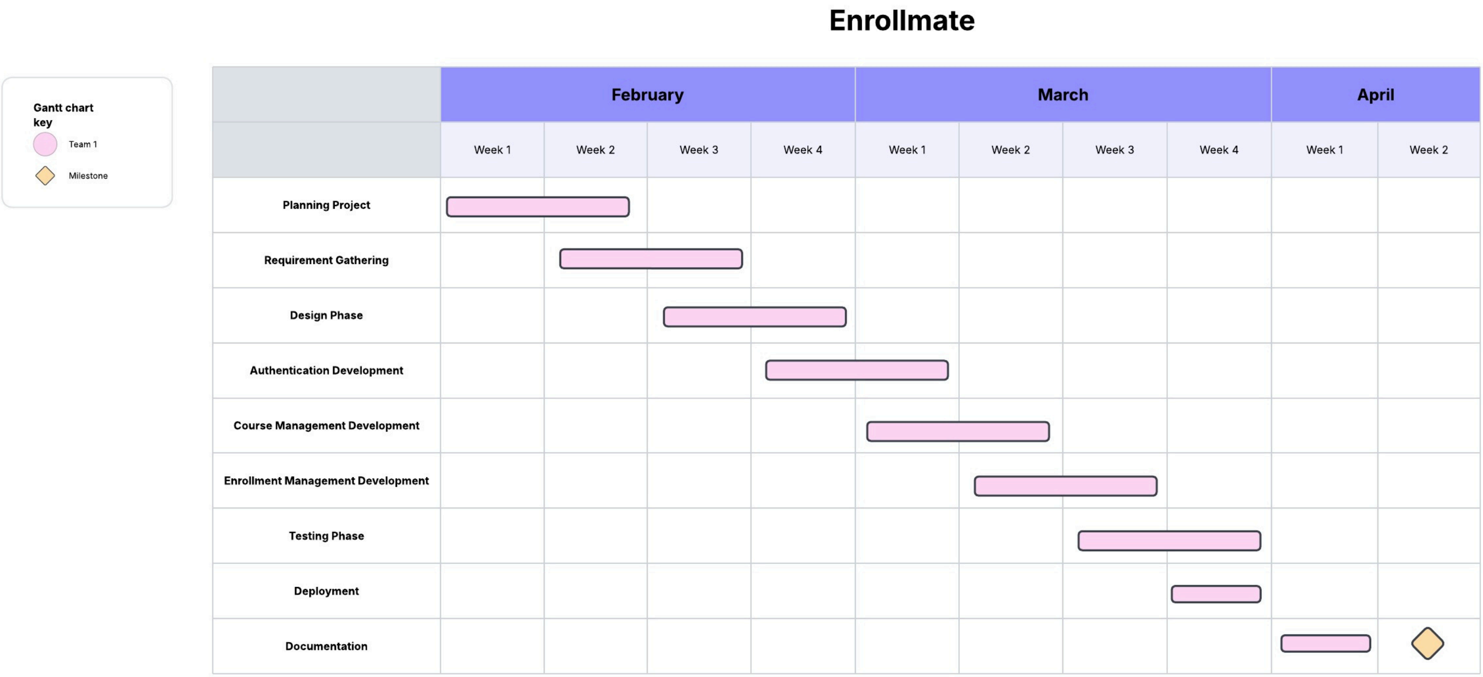
Level 6: Deployment

- Deploy system on a server.

Level 7: Documentation

- Final Reports, User Manuals, Technical Documentation.

GANTT CHART



Enrollmate

Gantt chart key

● Team 1

◆ Milestone

	February				March				April	
	Week 1	Week 2	Week 3	Week 4	Week 1	Week 2	Week 3	Week 4	Week 1	Week 2
Planning Project	●	●								
Requirement Gathering		●	●							
Design Phase			●	●						
Authentication Development				●	●					
Course Management Development					●	●				
Enrollment Management Development						●	●			
Testing Phase							●	●		
Deployment								●		
Documentation									●	◆

IDENTIFYING SUBSYSTEMS

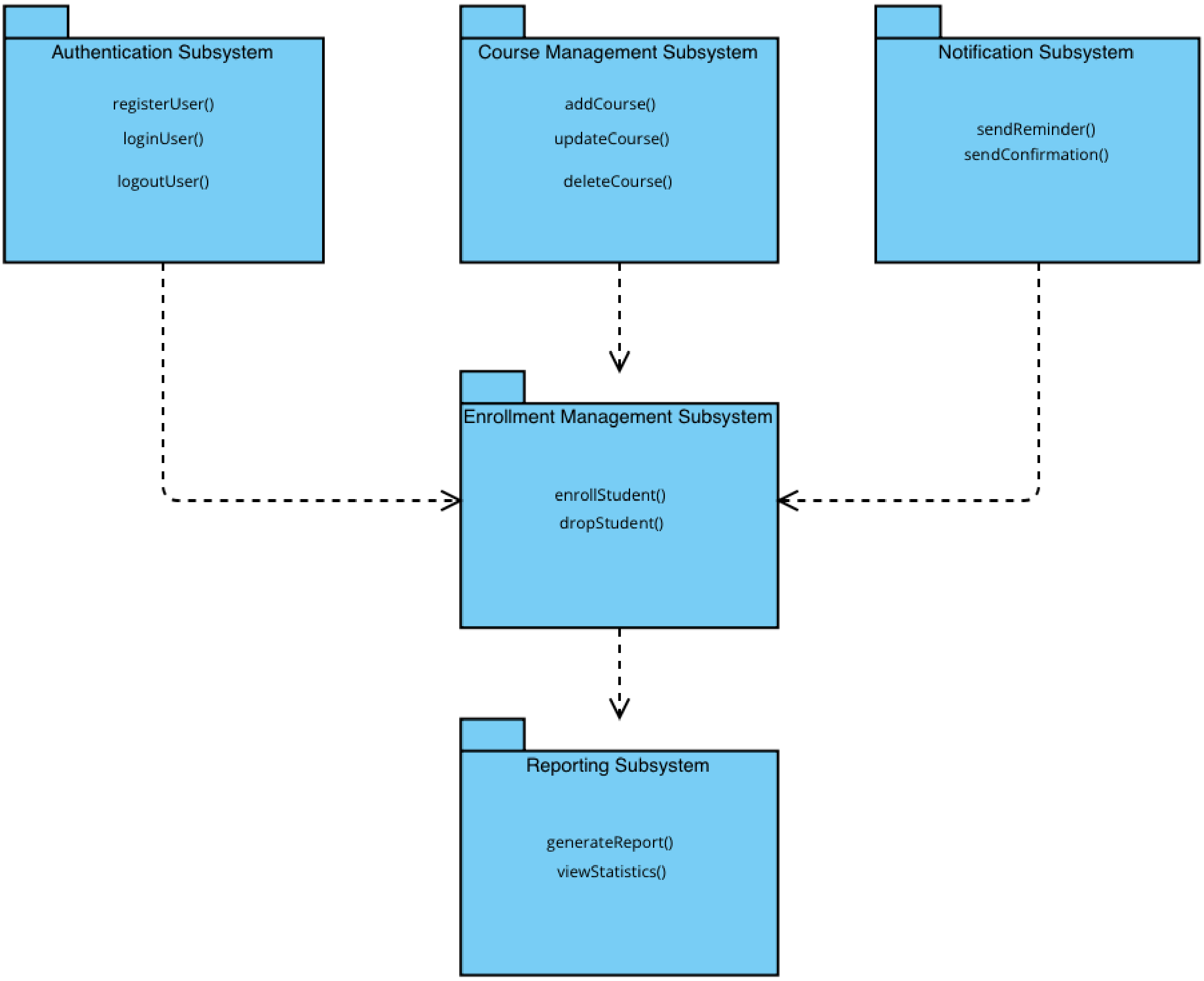
Subsystem	Responsibility
Authetication Subsystem	User Registration, Login, Logout
Course Management Subsystem	Add, Update, Delete Courses
Enrollment Management Subsystem	Enroll, and Drop Courses
Notification Subsystem	Send Enrollment and Deadline Alerts
Reporting Subsystem	Generate and View Enrollment Reports

ARCHITECTURE STYLE

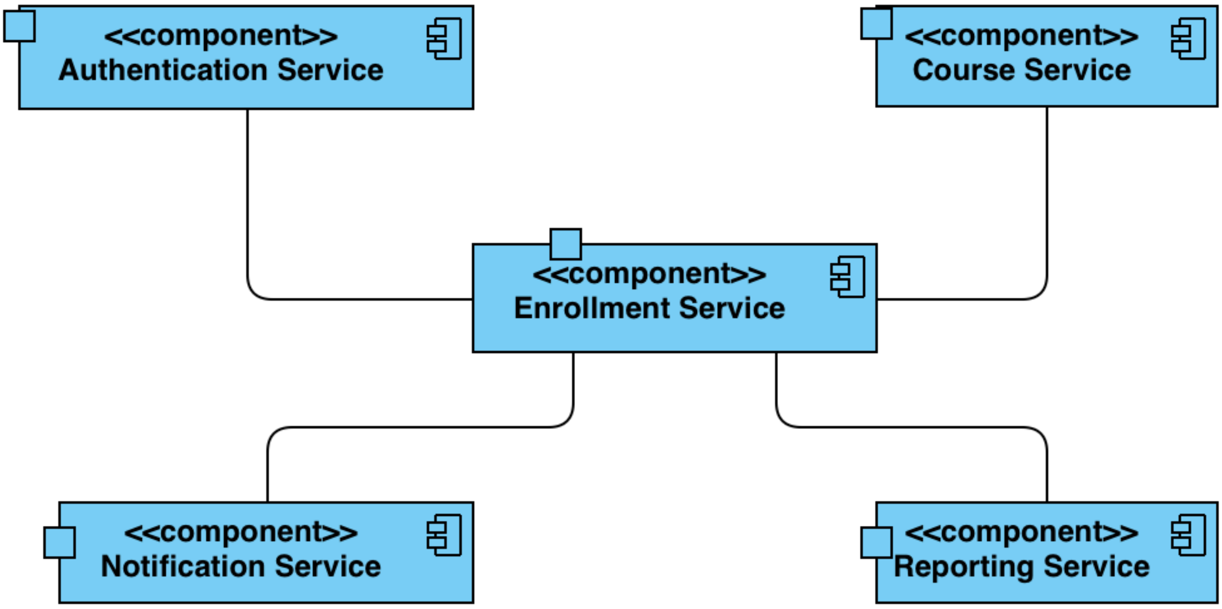
Brief Description of Layered Architecture:

- **Layer 1:** User Interface (Login screen, dashboard)
- **Layer 2:** Business Logic Layer (Handles operations like registration, enrollment)
- **Layer 3:** Database Layer (Stores users, courses, enrollments)
- **Communication:** Via API calls (Client to Server)

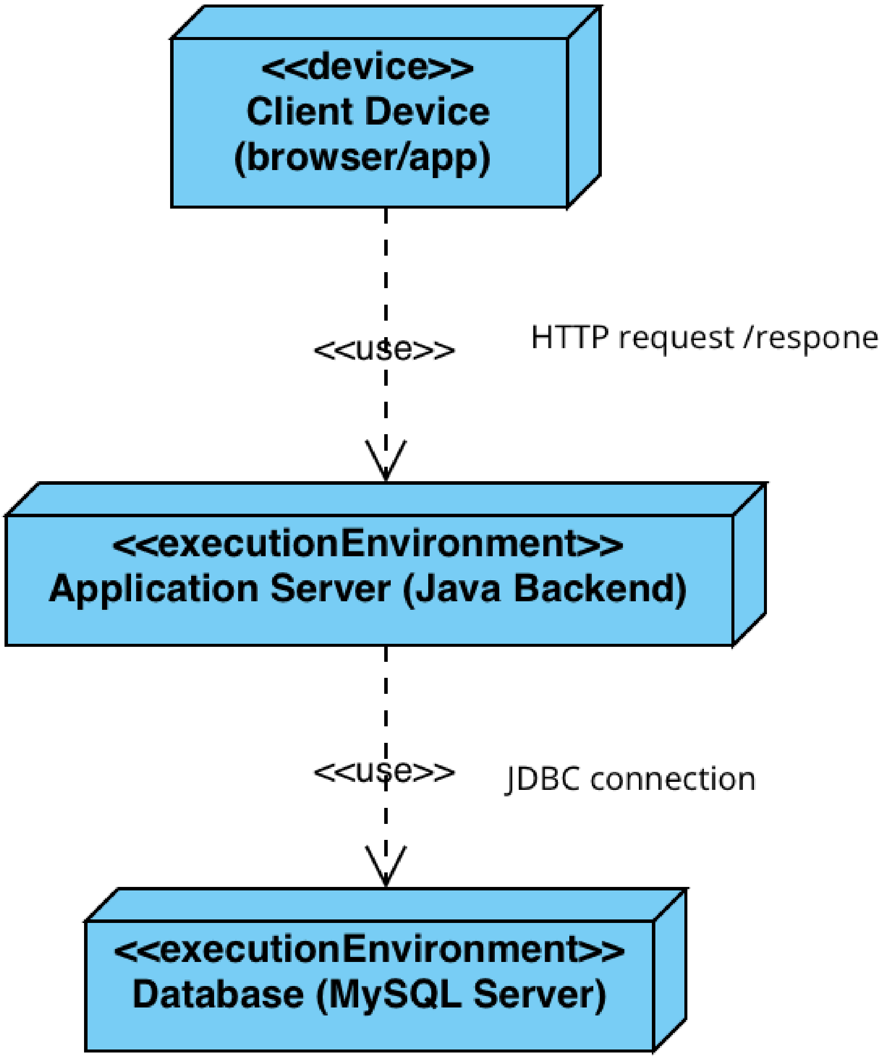
UML PACKAGE DIAGRAM



COMPONENT DIAGRAM



DEPLOYMENT DIAGRAM

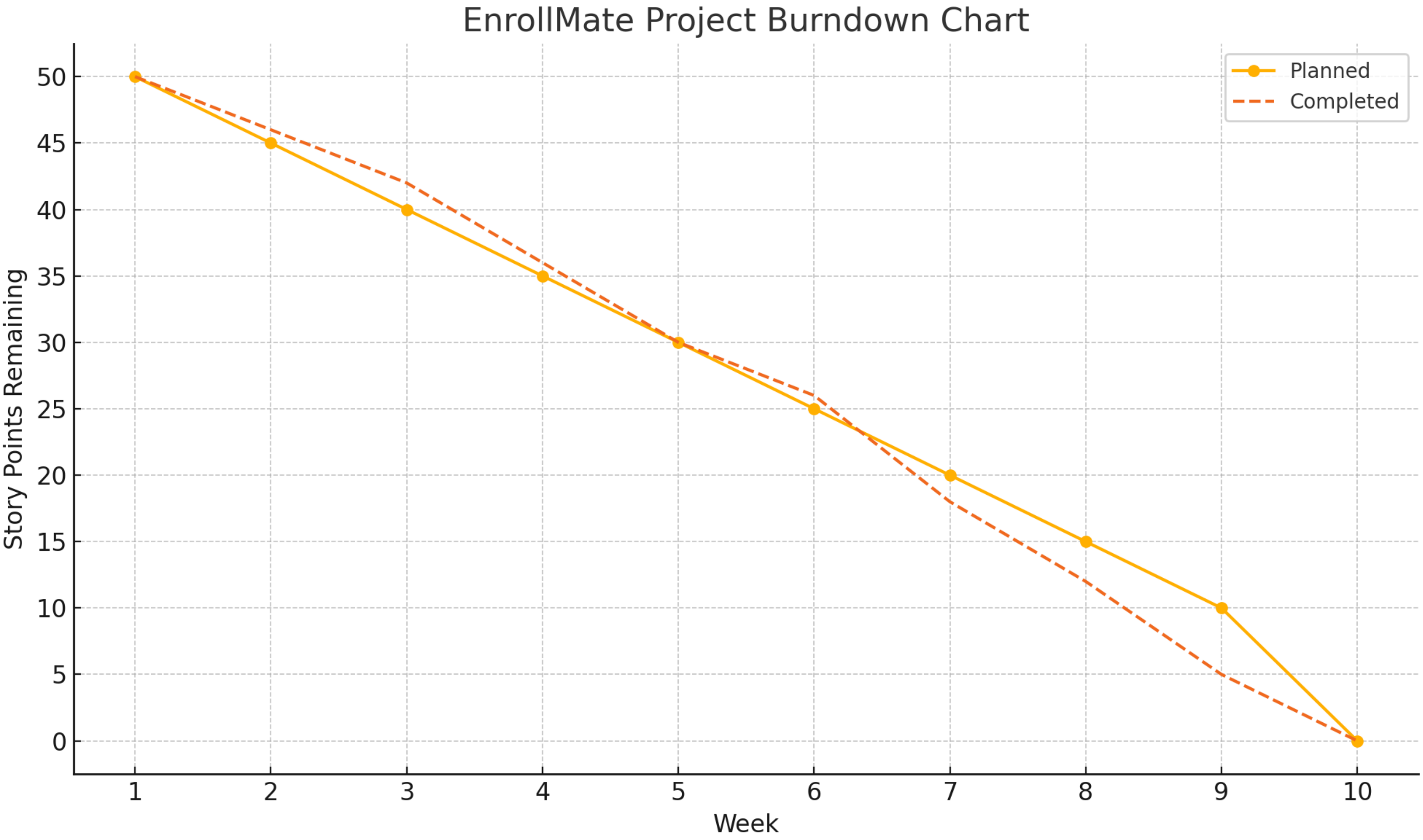


23

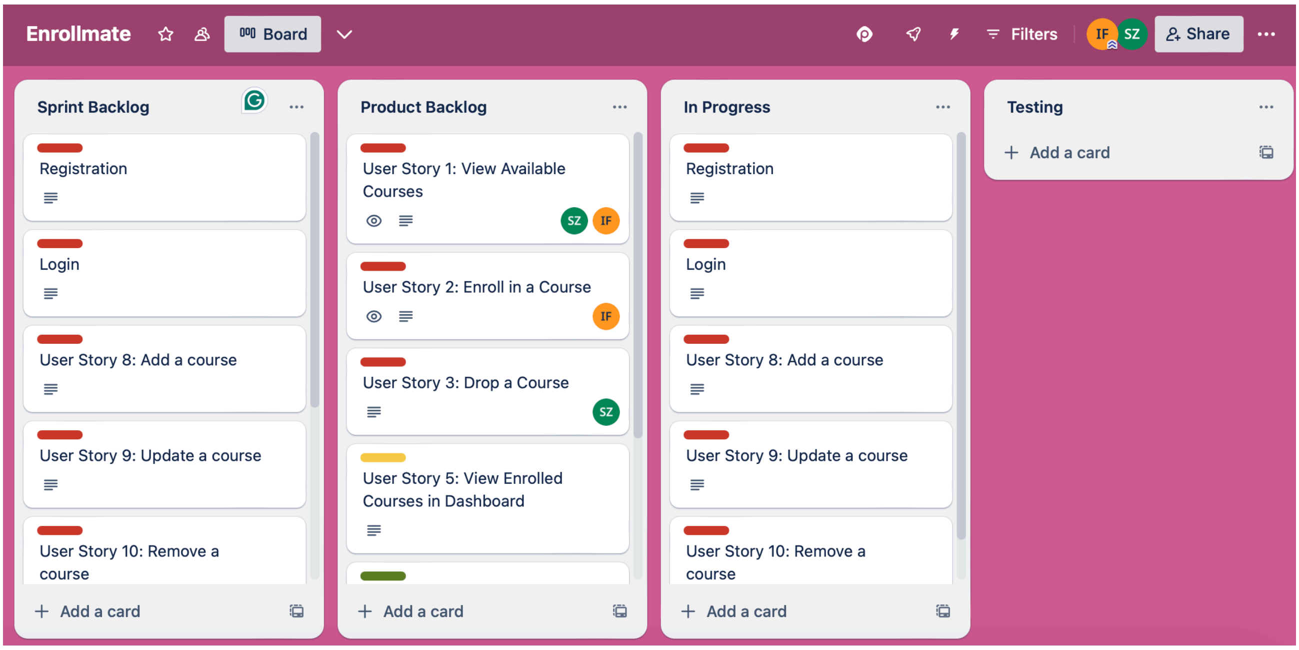
IMPLEMENTATION

24

PRODUCT BURNDOWN CHART



TRELLO BOARD



Enrollmate

Sprint Backlog

- Registration
- Login
- User Story 8: Add a course
- User Story 9: Update a course
- User Story 10: Remove a course

+ Add a card

Product Backlog

- User Story 1: View Available Courses (SZ, IF)
- User Story 2: Enroll in a Course (IF)
- User Story 3: Drop a Course (SZ)
- User Story 5: View Enrolled Courses in Dashboard

+ Add a card

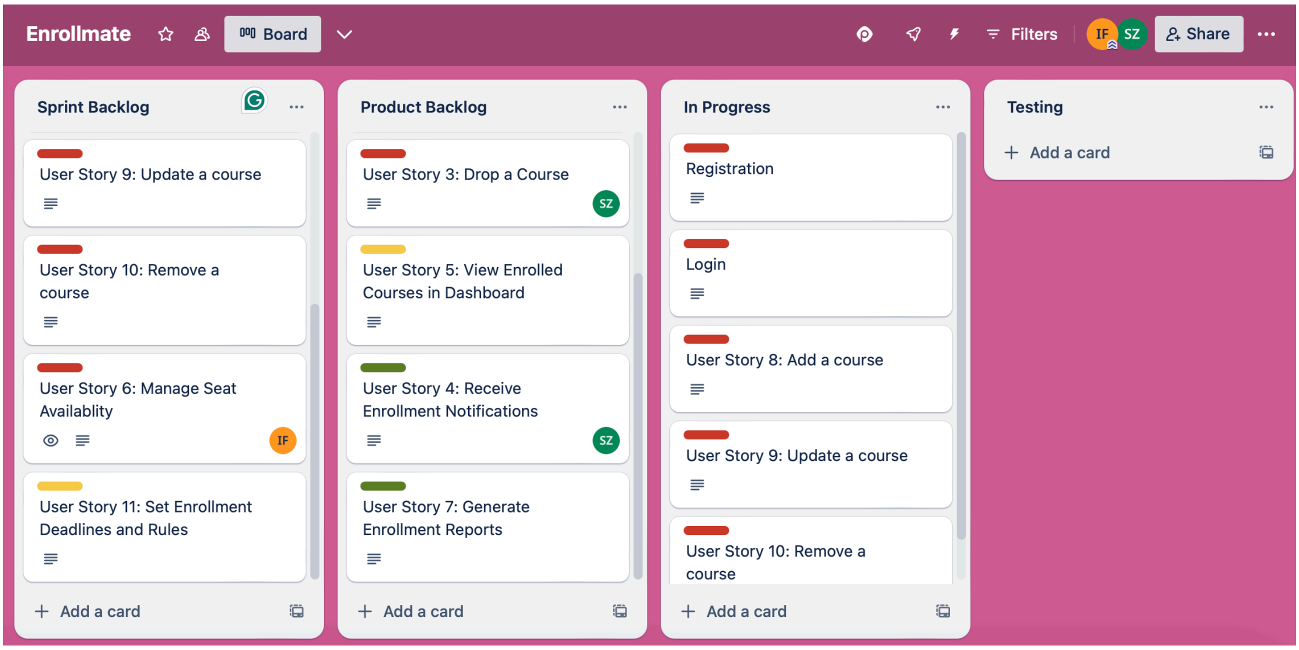
In Progress

- Registration
- Login
- User Story 8: Add a course
- User Story 9: Update a course
- User Story 10: Remove a course

+ Add a card

Testing

+ Add a card



Enrollmate

Sprint Backlog

- User Story 9: Update a course
- User Story 10: Remove a course
- User Story 6: Manage Seat Availablity (IF)
- User Story 11: Set Enrollment Deadlines and Rules

+ Add a card

Product Backlog

- User Story 3: Drop a Course (SZ)
- User Story 5: View Enrolled Courses in Dashboard
- User Story 4: Receive Enrollment Notifications (SZ)
- User Story 7: Generate Enrollment Reports

+ Add a card

In Progress

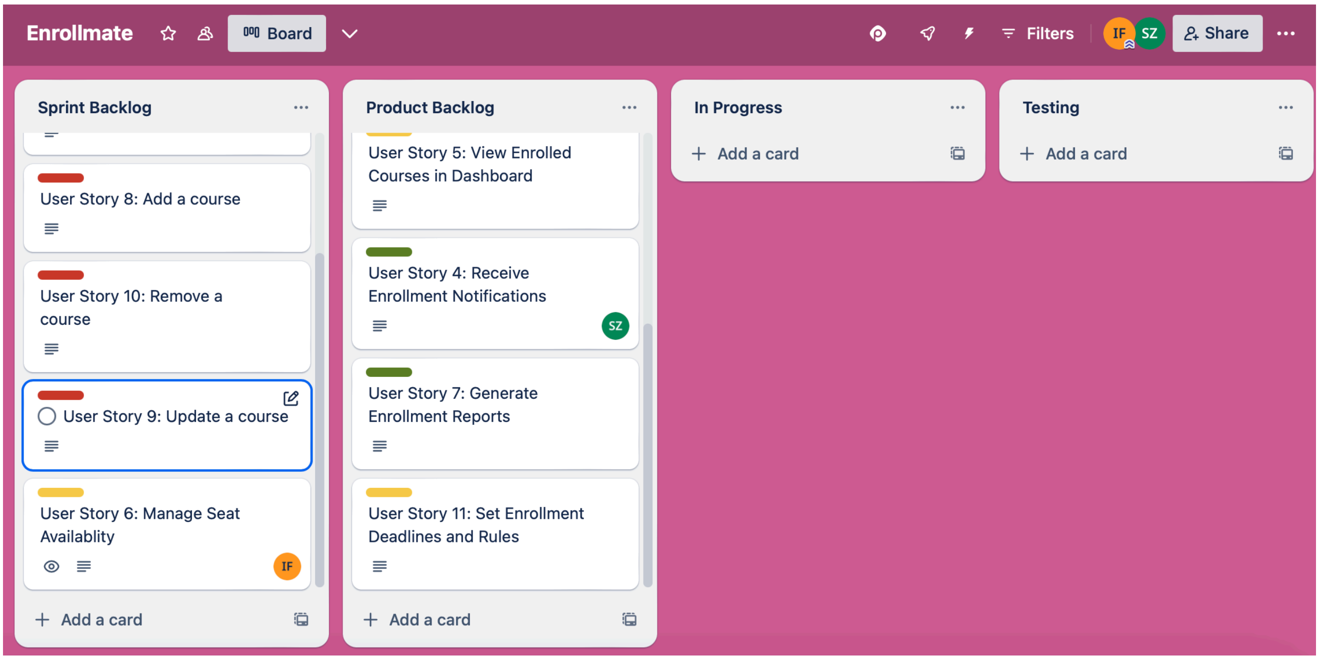
- Registration
- Login
- User Story 8: Add a course
- User Story 9: Update a course
- User Story 10: Remove a course

+ Add a card

Testing

+ Add a card

TRELLO BOARD



Sprint Backlog

- User Story 8: Add a course
- User Story 10: Remove a course
- User Story 9: Update a course
- User Story 6: Manage Seat Availablity — IF

+ Add a card

Product Backlog

- User Story 5: View Enrolled Courses in Dashboard
- User Story 4: Receive Enrollment Notifications — SZ
- User Story 7: Generate Enrollment Reports
- User Story 11: Set Enrollment Deadlines and Rules

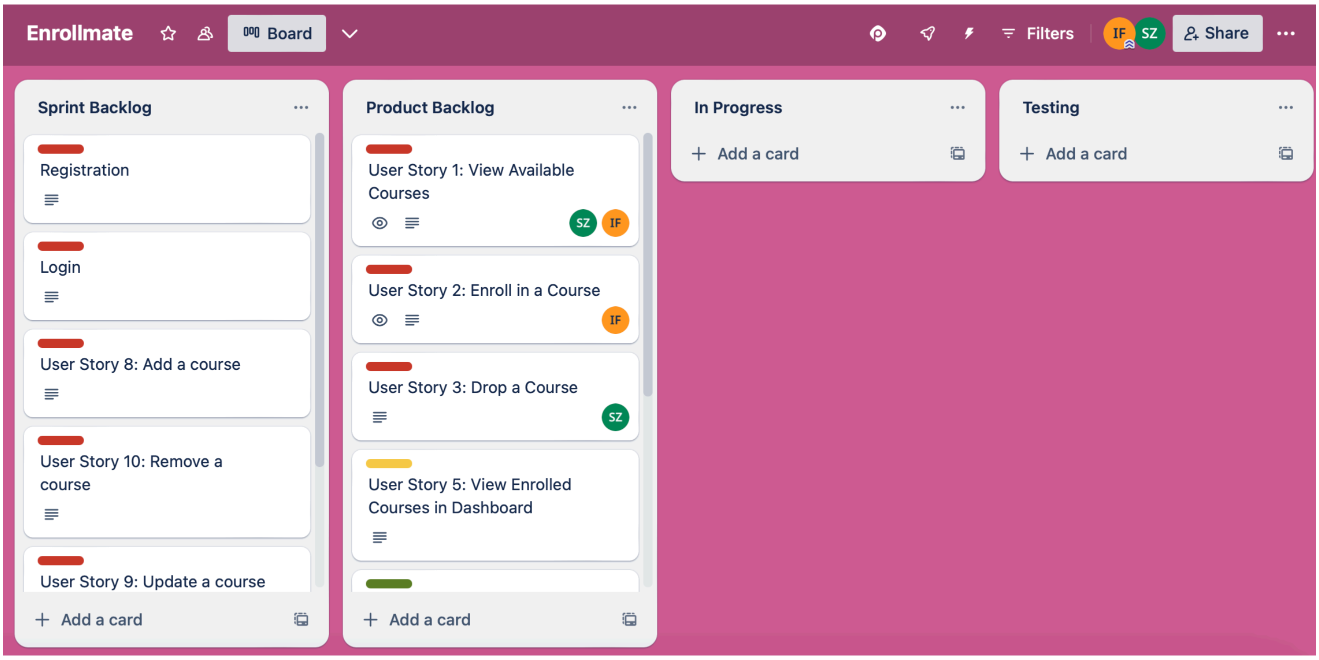
+ Add a card

In Progress

+ Add a card

Testing

+ Add a card



Sprint Backlog

- Registration
- Login
- User Story 8: Add a course
- User Story 10: Remove a course
- User Story 9: Update a course

+ Add a card

Product Backlog

- User Story 1: View Available Courses — SZ IF
- User Story 2: Enroll in a Course — IF
- User Story 3: Drop a Course — SZ
- User Story 5: View Enrolled Courses in Dashboard

+ Add a card

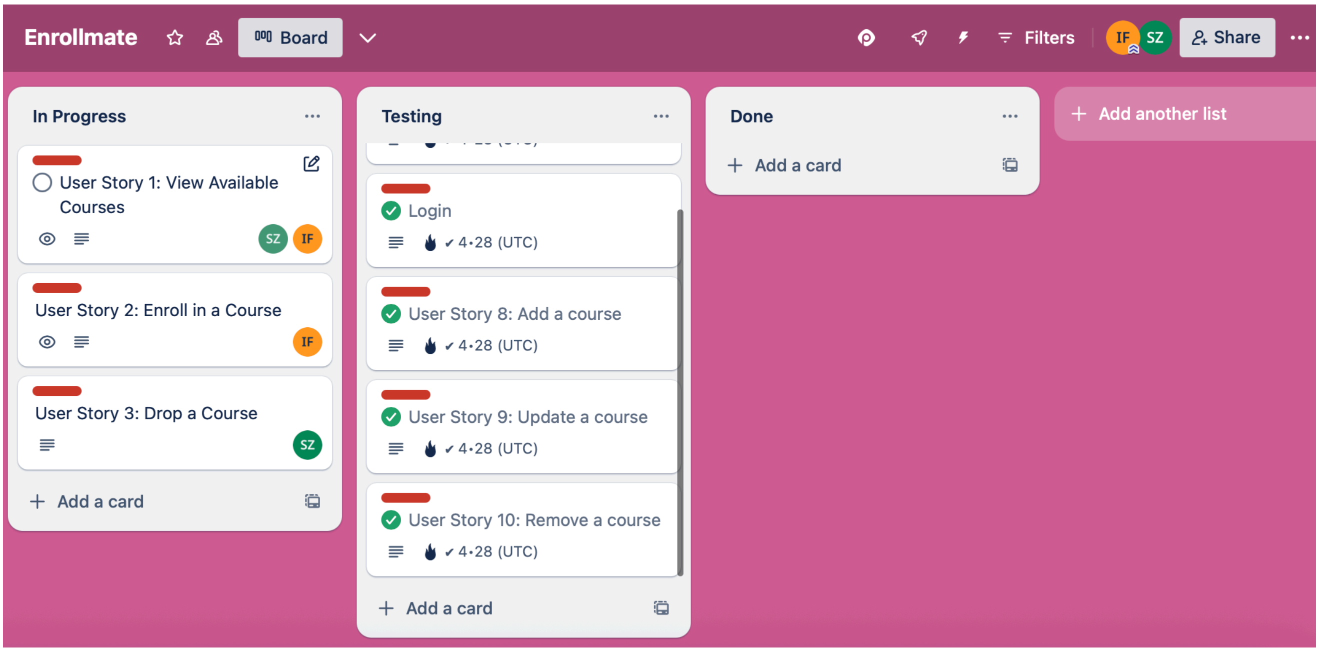
In Progress

+ Add a card

Testing

+ Add a card

TRELLO BOARD



In Progress

- User Story 1: View Available Courses — SZ IF
- User Story 2: Enroll in a Course — IF
- User Story 3: Drop a Course — SZ

+ Add a card

Testing

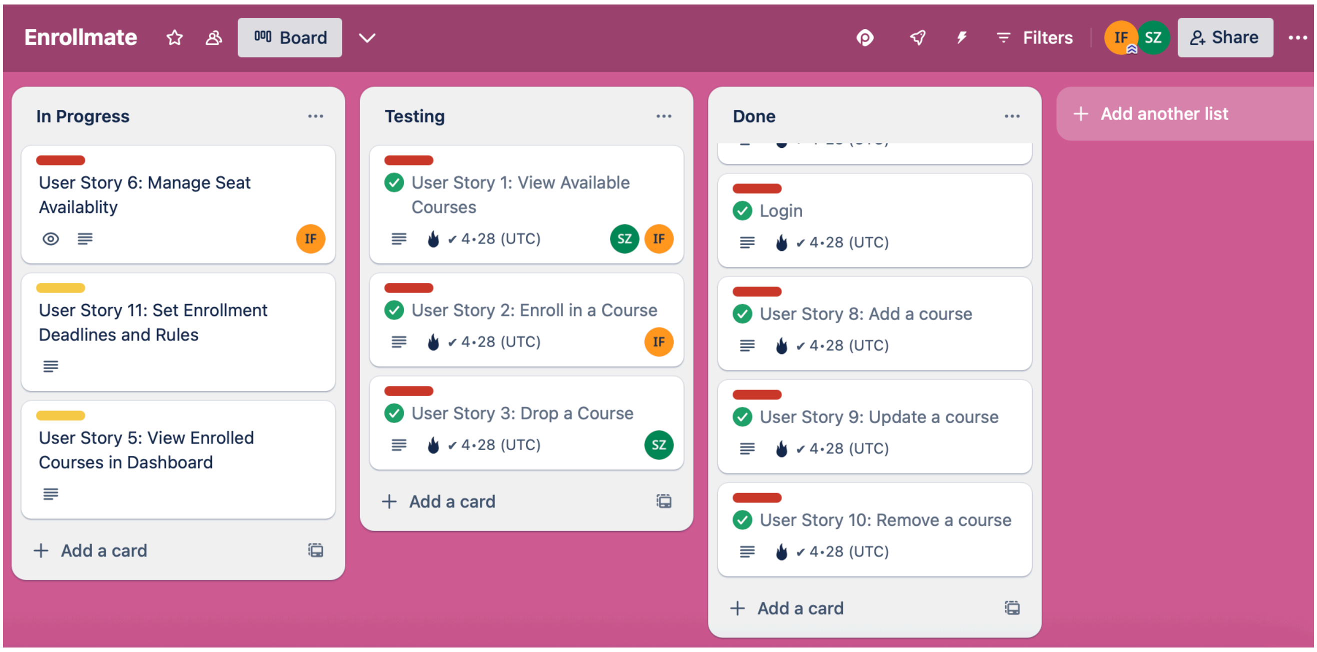
- ✓ Login — ✓ 4-28 (UTC)
- ✓ User Story 8: Add a course — ✓ 4-28 (UTC)
- ✓ User Story 9: Update a course — ✓ 4-28 (UTC)
- ✓ User Story 10: Remove a course — ✓ 4-28 (UTC)

+ Add a card

Done

+ Add a card

+ Add another list



In Progress

- User Story 6: Manage Seat Availablity — IF
- User Story 11: Set Enrollment Deadlines and Rules
- User Story 5: View Enrolled Courses in Dashboard

+ Add a card

Testing

- ✓ User Story 1: View Available Courses — ✓ 4-28 (UTC) — SZ IF
- ✓ User Story 2: Enroll in a Course — ✓ 4-28 (UTC) — IF
- ✓ User Story 3: Drop a Course — ✓ 4-28 (UTC) — SZ

+ Add a card

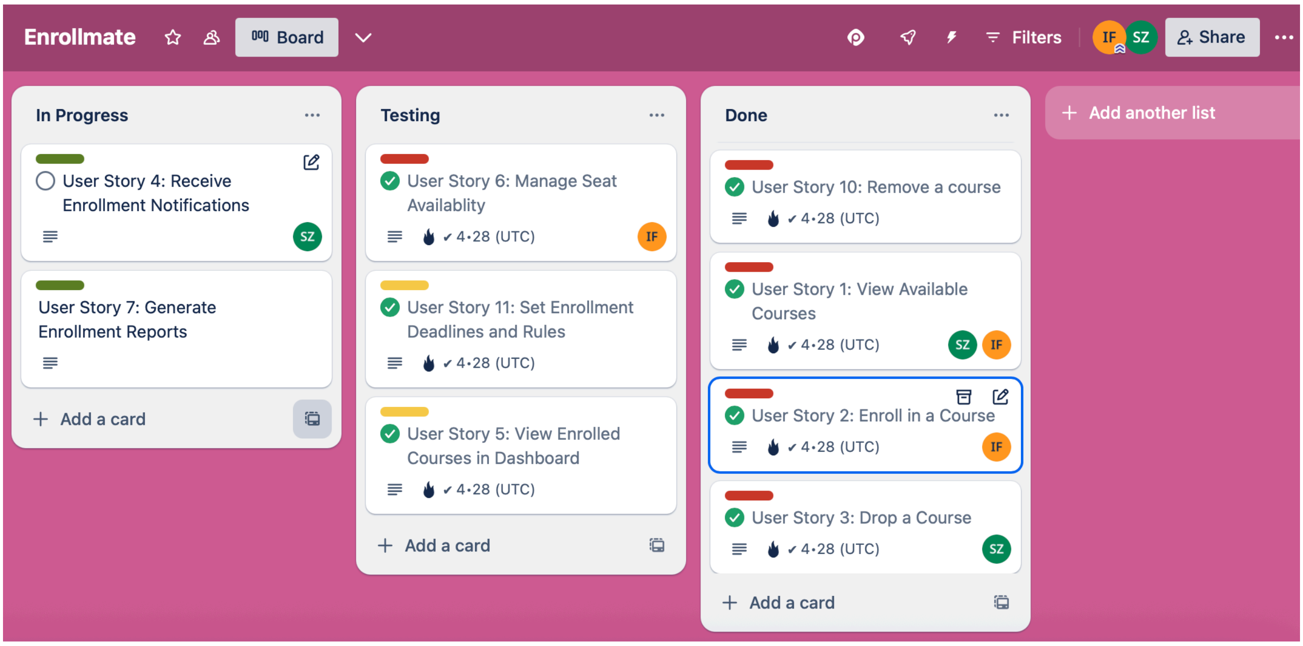
Done

- ✓ Login — ✓ 4-28 (UTC)
- ✓ User Story 8: Add a course — ✓ 4-28 (UTC)
- ✓ User Story 9: Update a course — ✓ 4-28 (UTC)
- ✓ User Story 10: Remove a course — ✓ 4-28 (UTC)

+ Add a card

+ Add another list

TRELLO BOARD



In Progress

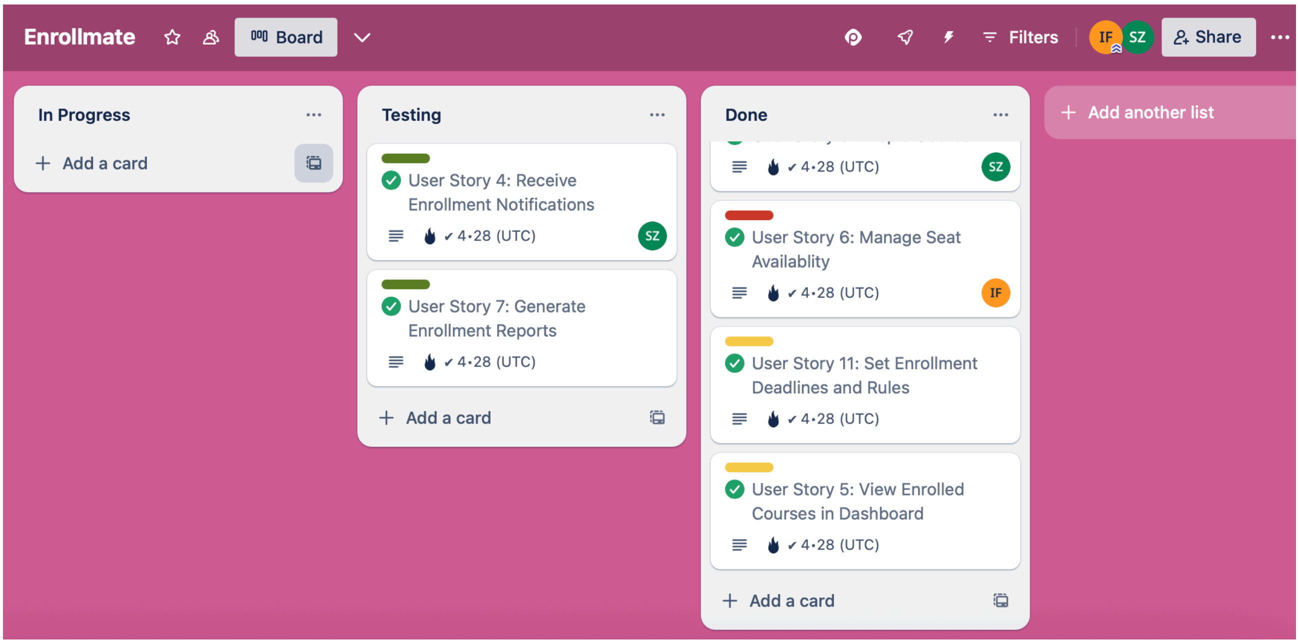
- ○ User Story 4: Receive Enrollment Notifications — SZ
- User Story 7: Generate Enrollment Reports

Testing

- ✓ User Story 6: Manage Seat Availablity — ✓ 4·28 (UTC) — IF
- ✓ User Story 11: Set Enrollment Deadlines and Rules — ✓ 4·28 (UTC)
- ✓ User Story 5: View Enrolled Courses in Dashboard — ✓ 4·28 (UTC)

Done

- ✓ User Story 10: Remove a course — ✓ 4·28 (UTC)
- ✓ User Story 1: View Available Courses — ✓ 4·28 (UTC) — SZ IF
- ✓ User Story 2: Enroll in a Course — ✓ 4·28 (UTC) — IF
- ✓ User Story 3: Drop a Course — ✓ 4·28 (UTC) — SZ



In Progress

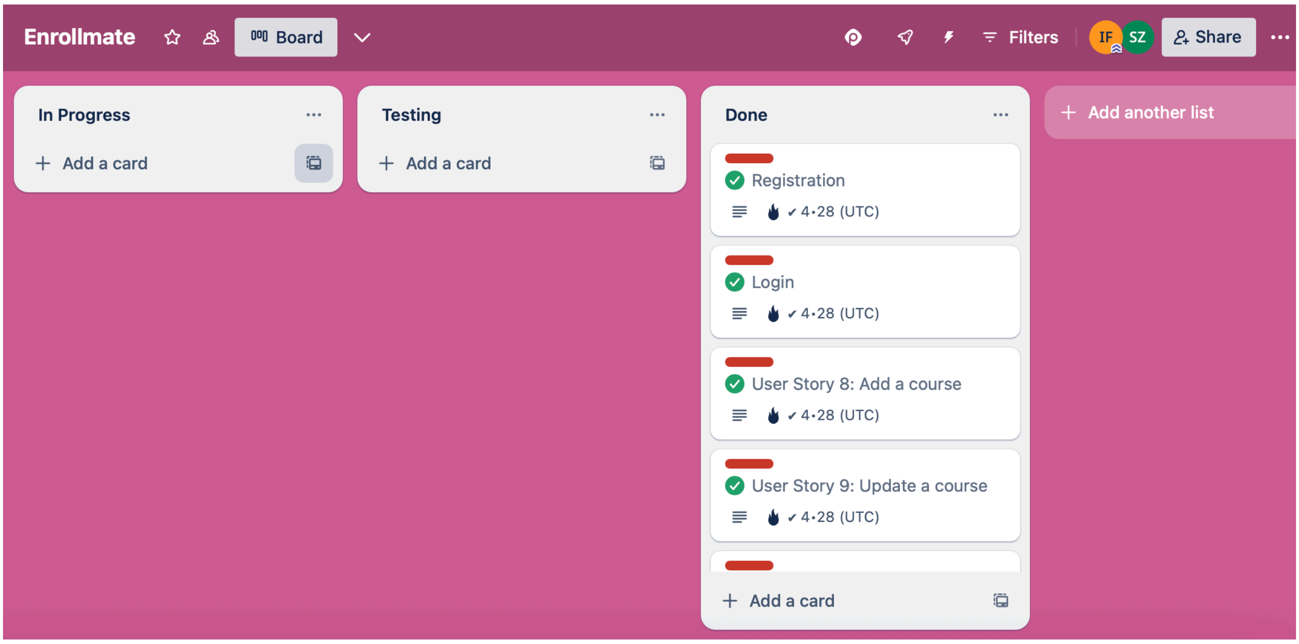
Testing

- ✓ User Story 4: Receive Enrollment Notifications — ✓ 4·28 (UTC) — SZ
- ✓ User Story 7: Generate Enrollment Reports — ✓ 4·28 (UTC)

Done

- ✓ 4·28 (UTC) — SZ
- ✓ User Story 6: Manage Seat Availablity — ✓ 4·28 (UTC) — IF
- ✓ User Story 11: Set Enrollment Deadlines and Rules — ✓ 4·28 (UTC)
- ✓ User Story 5: View Enrolled Courses in Dashboard — ✓ 4·28 (UTC)

TRELLO BOARD

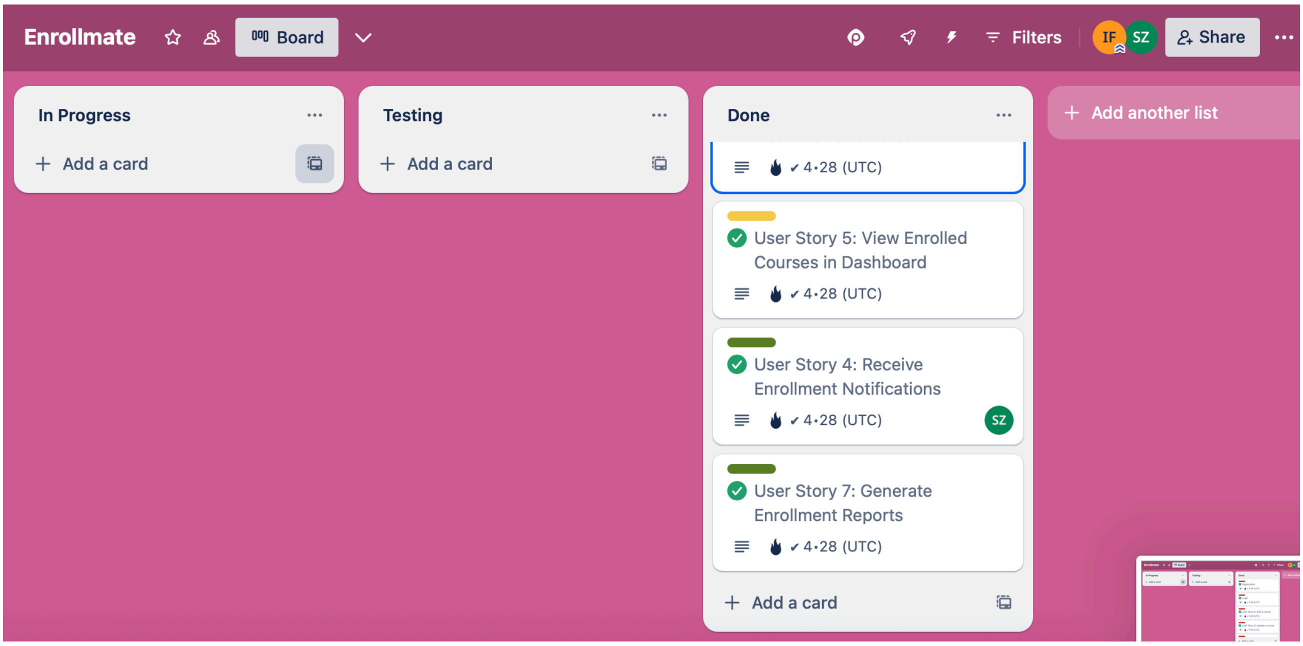


In Progress

Testing

Done

- Registration — ✓ 4-28 (UTC)
- Login — ✓ 4-28 (UTC)
- User Story 8: Add a course — ✓ 4-28 (UTC)
- User Story 9: Update a course — ✓ 4-28 (UTC)



In Progress

Testing

Done

- ✓ 4-28 (UTC)
- User Story 5: View Enrolled Courses in Dashboard — ✓ 4-28 (UTC)
- User Story 4: Receive Enrollment Notifications — ✓ 4-28 (UTC) — SZ
- User Story 7: Generate Enrollment Reports — ✓ 4-28 (UTC)

BLACK BOX TESTING

31

WHITE BOX TESTING

WORK DIVISION

- **Zainab Fatima:** Registration, Login, Database Connectivity, Black Box Testing.
- **Shayane Zainab:** Enrollment Management, Admin Panel (Add/Remove Courses), White Box Testing.
- **Both:** Documentation, Testing, Final Report Preparation.

LESSON LEARNT

- Improved understanding of Agile project management.
- Effective use of Trello for sprint tracking.
- Practical application of black box and white box testing.
- Managing deadlines through Gantt charts.
- Better collaboration and communication skills.
- Handling database integration and deployment on local servers.
- Understanding of architectural designs like layered and modular architecture.

GITHUB REPOSITORY

The following is the GITHUB link to our project:
https://github.com/ZainabFatima10/Enrollmate