Phase 2: College Enrollment System

Group 4: Aakash M, Christian R, Felipe G, Svein Q

Introduction

The Software design document provides a comprehensive design of the College Enrollment System, elaborating on and expanding the specifications laid out in the Software Requirements Specification. It outlines the system architecture, component responsibilities, class designs, Use Case Diagrams, and sequence Diagrams.
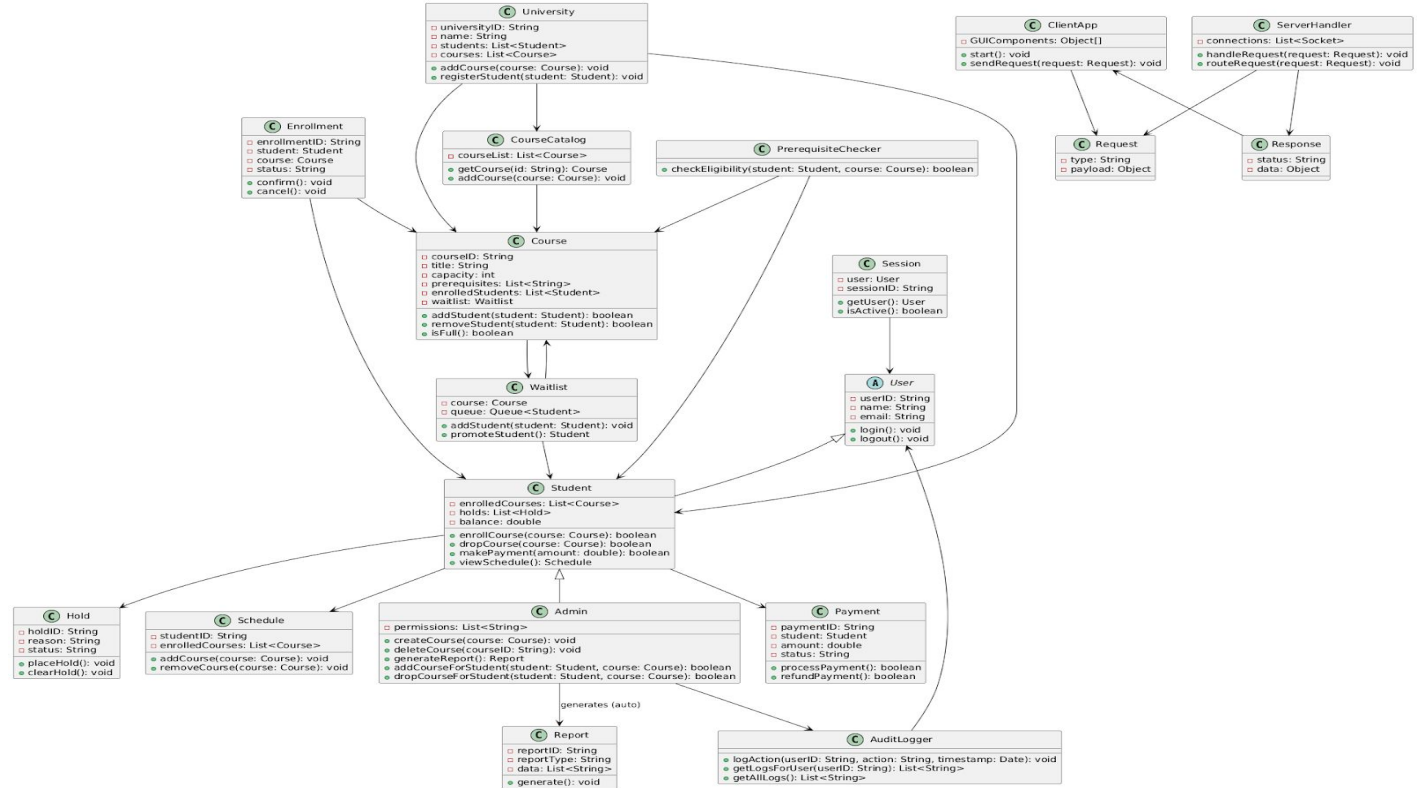
System Architecture

The system follows a client-server architecture. The client application provides the GUI for user interactions, while the server handles business logic and data processing. Communication between the client and server occurs over TCP/IP.
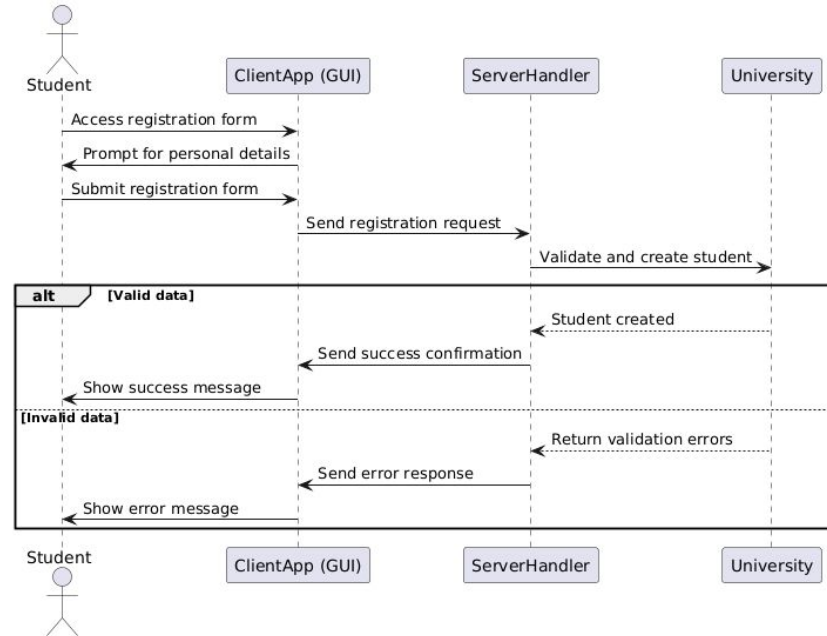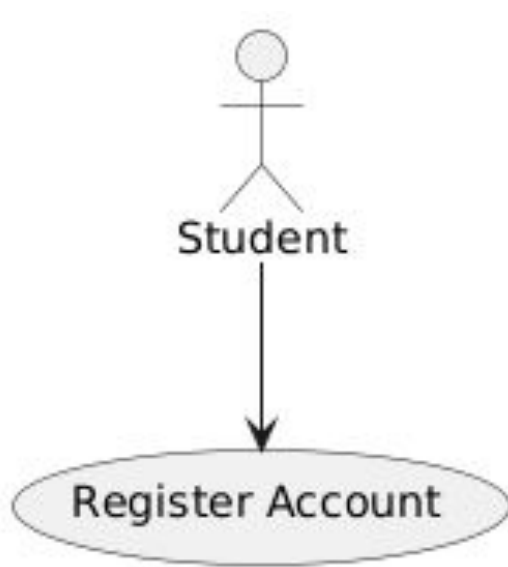
Classes

- User (abstract)
- Student
- Admin
- Course
- University
- Enrollment
- Waitlist
- Payment
- Hold
- Report

- PrerequisiteChecker
- Schedule
- LoginManager
- ServerHandler
- ClientApp
- Request
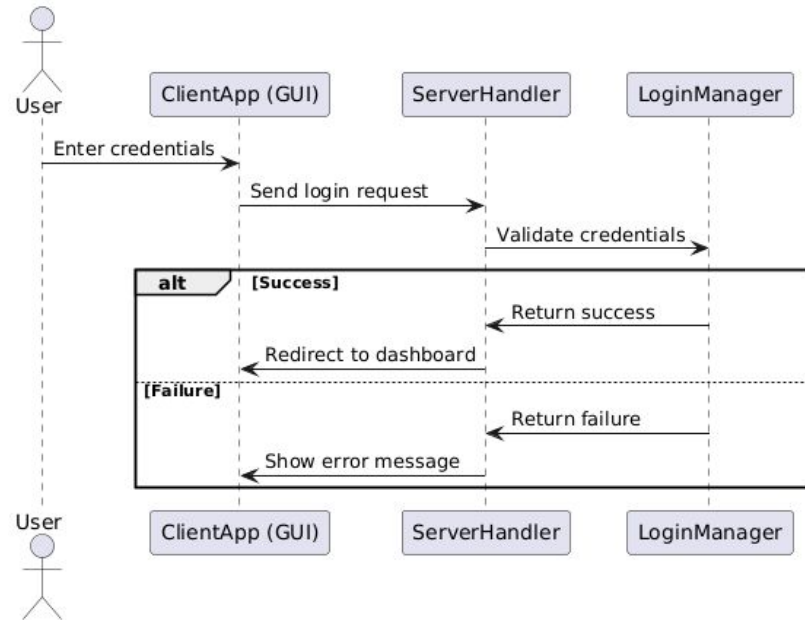- Response
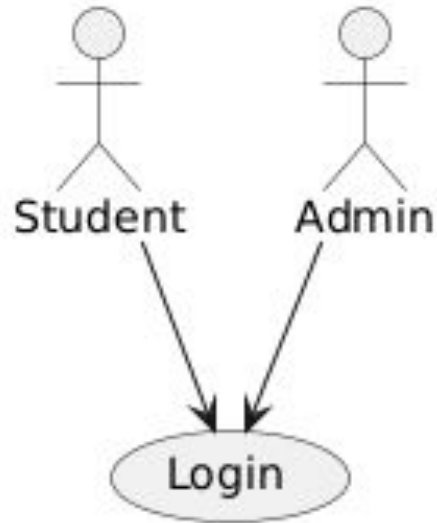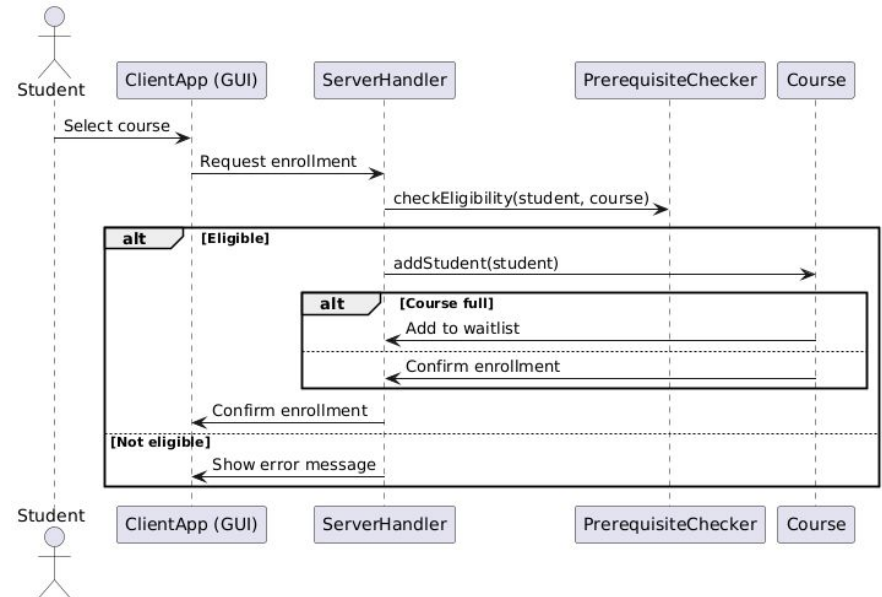- Session
- CourseCatalog
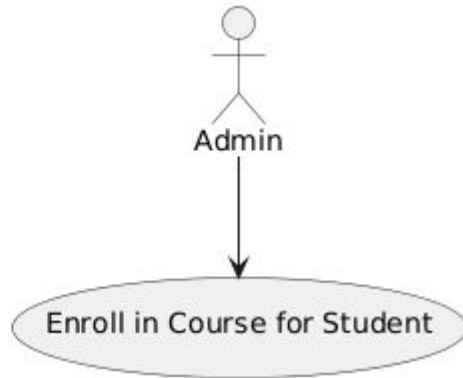- Notification

Class Diagram

University
- □ universityID: String
- □ name: String
- □ students: List<Student>
- □ courses: List<Course>
- ● addCourse(course: Course): void
- ● registerStudent(student: Student): void

ClientApp
- □ GUIComponents: Object[]
- ● start(): void
- ● sendRequest(request: Request): void

ServerHandler
- □ connections: List<Socket>
- ● handleRequest(request: Request): void
- ● routeRequest(request: Request): void

Enrollment
- □ enrollmentID: String
- □ student: Student
- □ course: Course
- □ status: String
- ● confirm(): void
- ● cancel(): void

CourseCatalog
- □ courseList: List<Course>
- ● getCourse(id: String): Course
- ● addCourse(course: Course): void

PrerequisiteChecker
- ● checkEligibility(student: Student, course: Course): boolean

Request
- □ type: String
- □ payload: Object

Response
- □ status: String
- □ data: Object

Course
- □ courseID: String
- □ title: String
- □ capacity: int
- □ prerequisites: List<String>
- □ enrolledStudents: List<Student>
- □ waitlist: Waitlist
- ● addStudent(student: Student): boolean
- ● removeStudent(student: Student): boolean
- ● isFull(): boolean

Session
- □ user: User
- □ sessionID: String
- ● getUser(): User
- ● isActive(): boolean

Waitlist
- □ course: Course
- □ queue: Queue<Student>
- ● addStudent(student: Student): void
- ● promoteStudent(): Student

User
- □ userID: String
- □ name: String
- □ email: String
- ● login(): void
- ● logout(): void

Student
- □ enrolledCourses: List<Course>
- □ holds: List<Hold>
- □ balance: double
- ● enrollCourse(course: Course): boolean
- ● dropCourse(course: Course): boolean
- ● makePayment(amount: double): boolean
- ● viewSchedule(): Schedule

Hold
- □ holdID: String
- □ reason: String
- □ status: String
- ● placeHold(): void
- ● clearHold(): void

Schedule
- □ studentID: String
- □ enrolledCourses: List<Course>
- ● addCourse(course: Course): void
- ● removeCourse(course: Course): void

Admin
- □ permissions: List<String>
- ● createCourse(course: Course): void
- ● deleteCourse(courseID: String): void
- ● generateReport(): Report
- ● addCourseForStudent(student: Student, course: Course): boolean
- ● dropCourseForStudent(student: Student, course: Course): boolean

Payment
- □ paymentID: String
- □ student: Student
- □ amount: double
- □ status: String
- ● processPayment(): boolean
- ● refundPayment(): boolean

generates (auto)

Report
- □ reportID: String
- □ reportType: String
- □ data: List<String>
- ● generate(): void

AuditLogger
- ● logAction(userID: String, action: String, timestamp: Date): void
- ● getLogsForUser(userID: String): List<String>
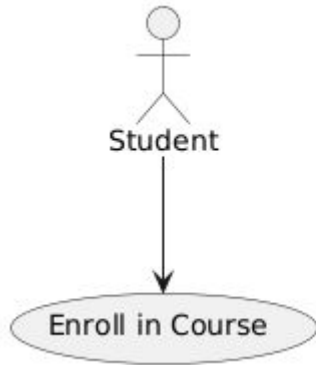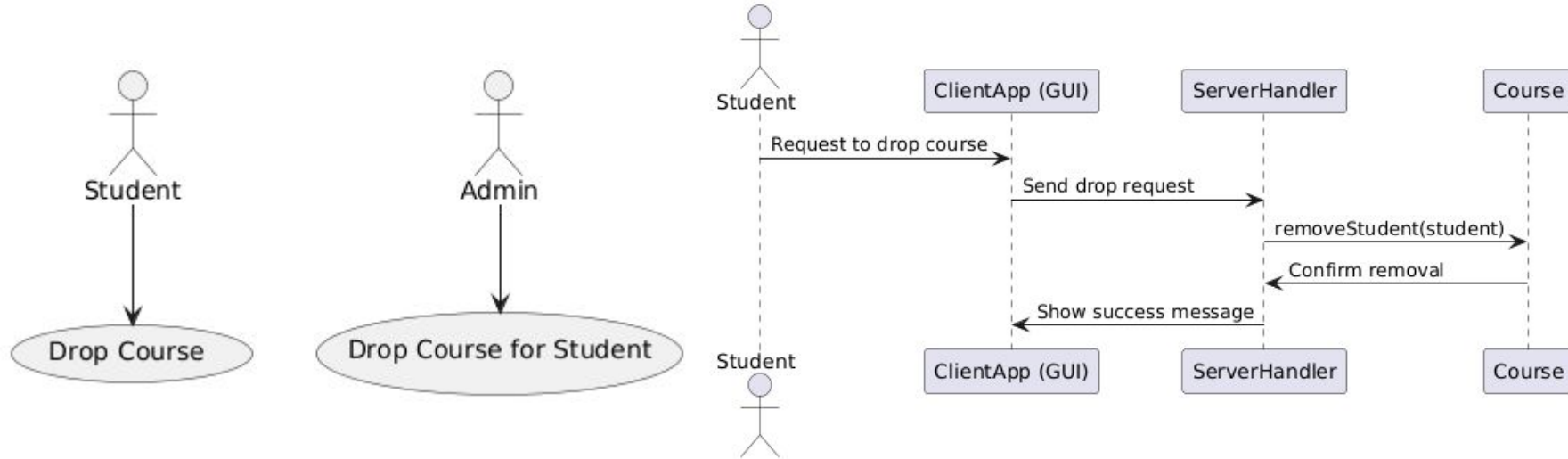- ● getAllLogs(): List<String>
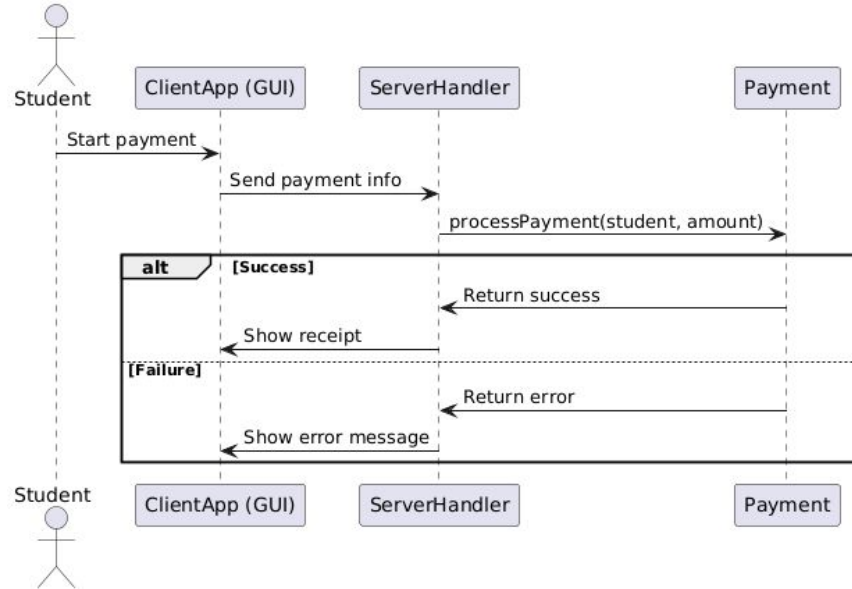
Use Case 01: Student Registration
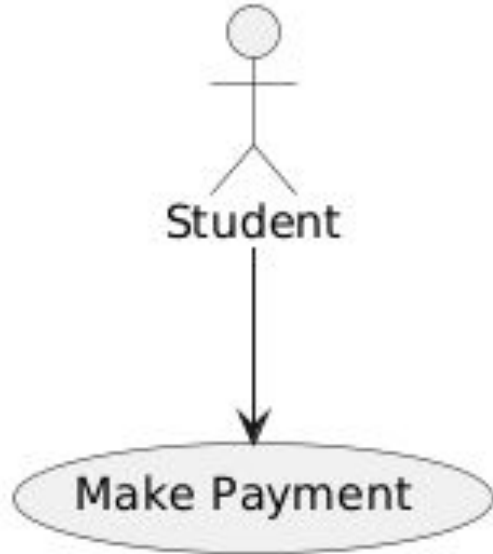
Use Case 02: Login
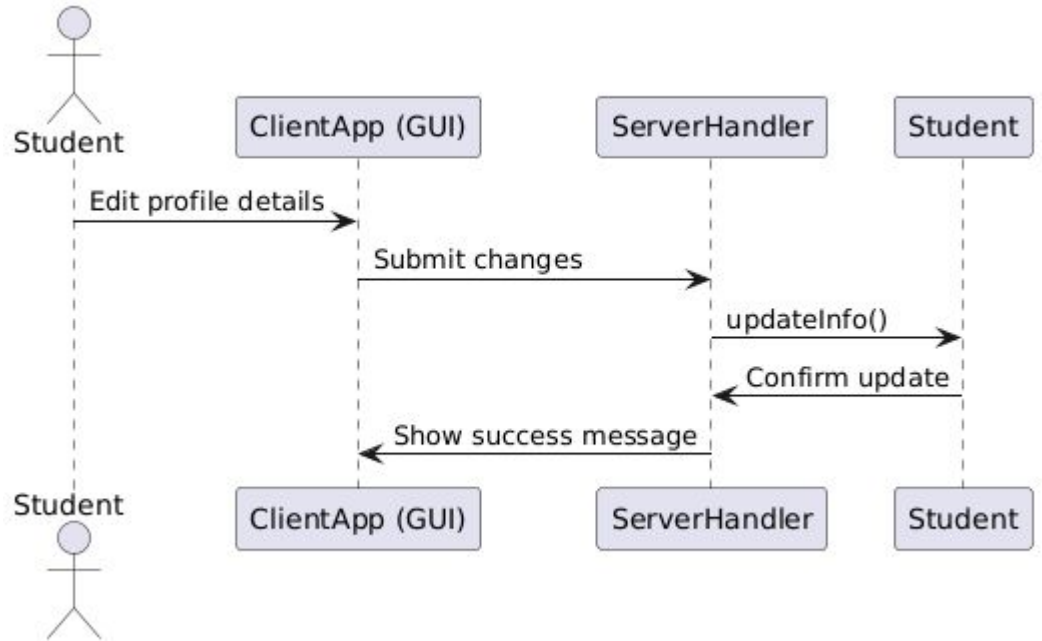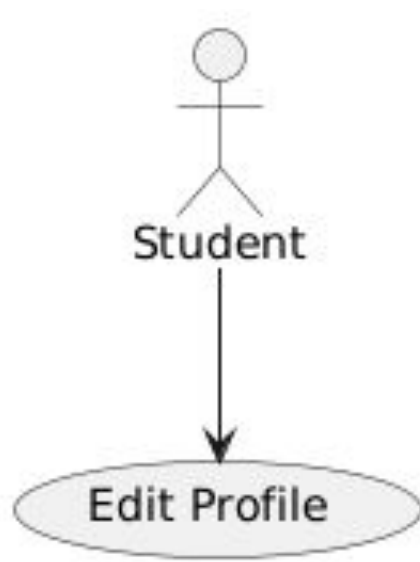
Use Case 03: Course Enrollment
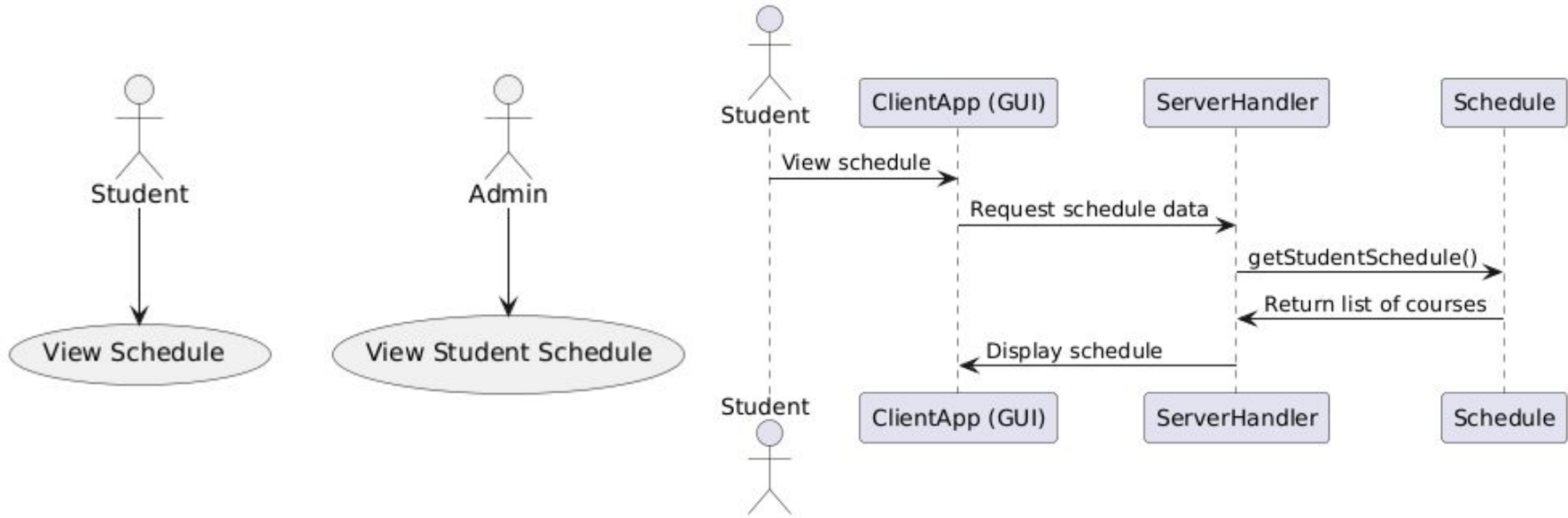
Use Case 04: Drop Course
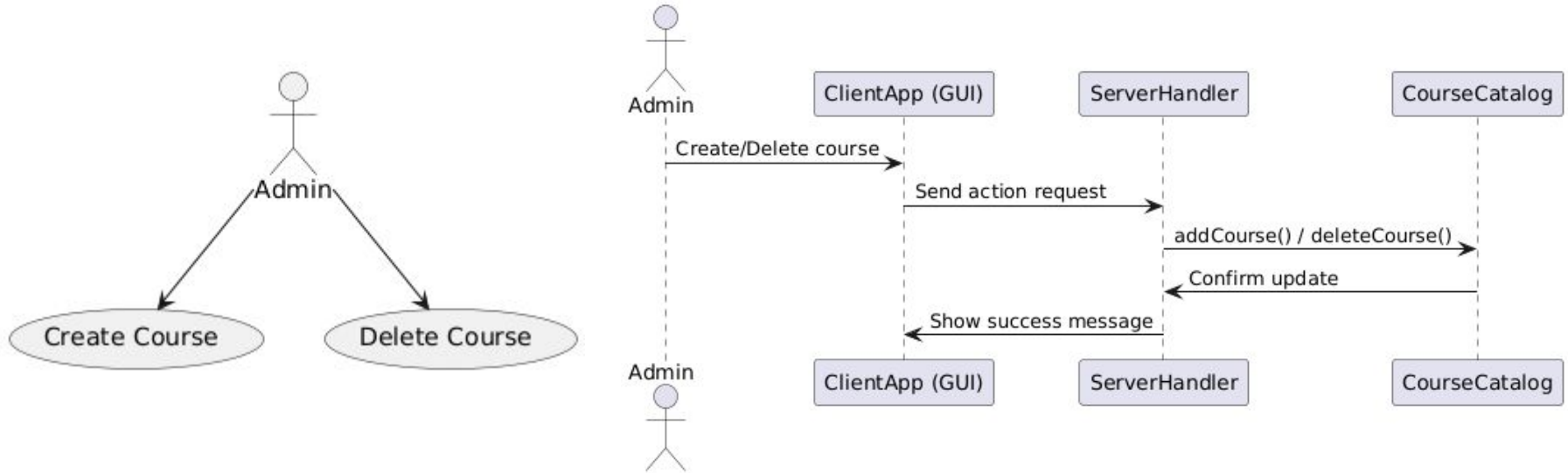
Use Case 05: Payment Processing
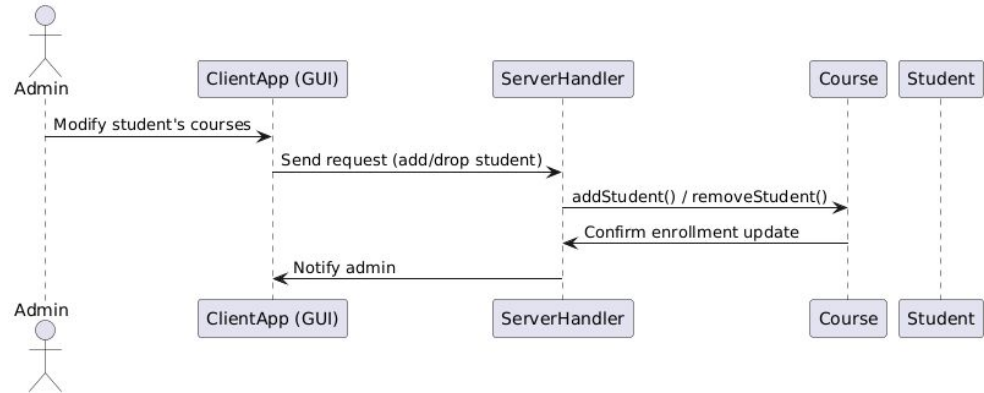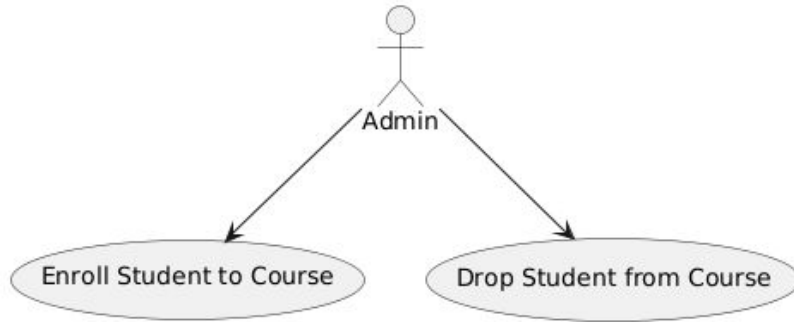
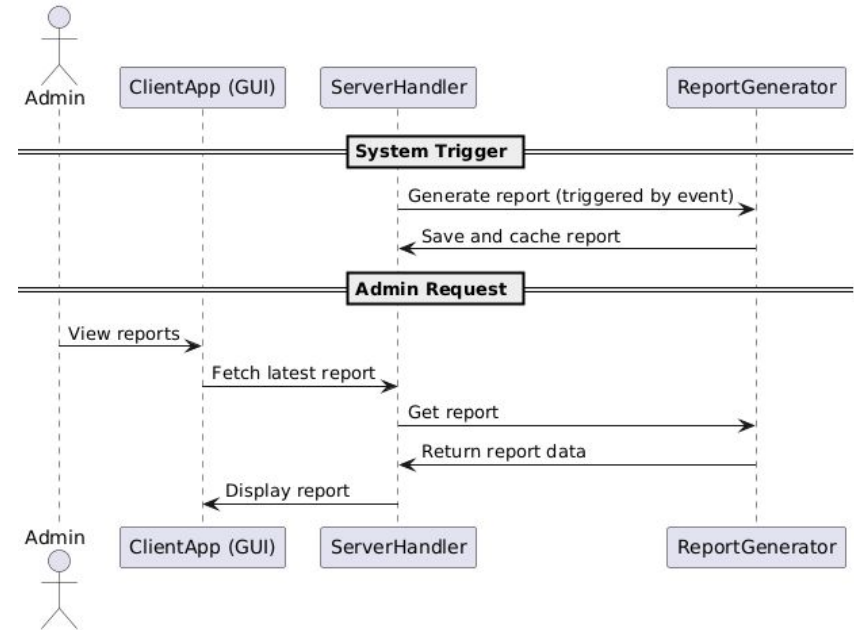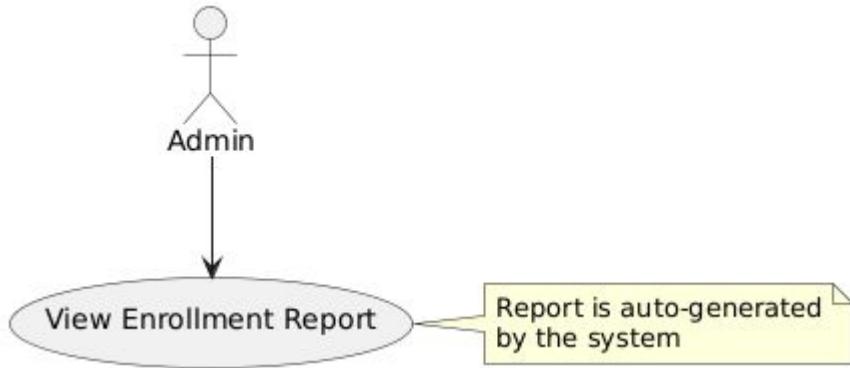# Use Case 06: Manage Student Profile

Use Case 07: View Schedule

Use Case 08: Admin Creates/Deletes Course

Use Case 09: Admin Manages Student Enrollment

Use Case 10: Auto-Generate Reports

Trello Board

Gannt Chart

∨ To-Do

	Task		Assigned	Status ⓘ		Due date ⓘ		Files	Timeline ⓘ	Dependent On ⓘ
☐	Requirements	⊕	👤	Done	⊙	Mar 3			❗ Jan 27 - Ma...	
☐	Design	⊕	👤	Done	✓	Apr 9			✓ Mar 3 - A...	Requirements
☐	Implementation	⊕	👤	Working on it	◯	Apr 28			Apr 9 - 28	Design
☐	Testing	⊕	👤	Not Started	◯	May 3			Apr 28 - May 3	Implementation
☐	Maintenance	⊕	👤	Not Started	◯	May 7			May 3 - 7	Testing

Gantt ▽

🗂 Baseline   📅 Auto Fit   Months ∨   − + ⋯

		Q1 2025			Q2 2025
		January	February	March	April	May
Requirements	Jan 27 - Mar 3					
Design	Mar 3 - Apr 9					
Implementation	Apr 9 - 28					
Testing	Apr 28 - May 3					
Maintenance	May 3 - 7					

THANK YOU