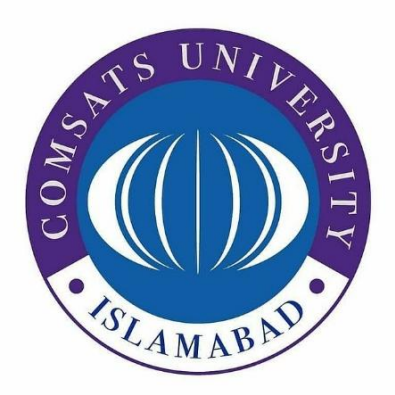


PDF REPORT

MEMBERS:

- Muhammad Masood Hussain (SP23-BSE-119).
- Hammad Shamraiz (Sp23-bse-108).
- Abbas khan (SP23-BSE-115).
- Muhammad Mudasir (Sp23-bse-138).
- Muhammad Hammad Ali Asif (Fa22-bse-037).
- SHOAIB SAKHAWAT (Sp23-bse-135).
- Hamid Anwar (SP23-BSE-124).

Modules:

- Authentication and user Management (Hammad Shamraiz).
- Course And Academic Management (Muhammad Mudasir).
- Mutual Attendence (Muhammad Hammad Ali Asif).
- Auto Attendence(Muhammad Masood Hussain).
- Admin tools and system management (Abbas khan).
- Reports notification and feedback (Hamid Anwar).
- Reminder And Alerts System(SHOAIB SAKHAWAT).

CHAPTER 1

ANALYSIS

Attendance Management System (AMS), showing interactions among three main actors — **Admin**, **Teacher**, and **Student** — and the various functions (use cases) each can perform.

📌 **Actors and Their Roles**

👩‍🏫 **Teacher**

Responsible for managing attendance:

- **Take Manual Attendance**
- **Edit Attendance**
- **View Attendance Records**
- **Send Reminder to Students**
- **View Student List**
- **Download Attendance Report**
- **Send Notifications**
- **Login/Logout/Reset Password/Update Profile**

👩‍🎓 **Student**

Can interact with their own attendance:

- **View Own Attendance**
- **Appeal Attendance**
- **Receive Notifications**
- **Download Attendance History**
- **Give Feedback**
- **Mark Auto Attendance**
- **Login/Logout/Reset Password/Update Profile**

👩‍💼 **Admin**

Has full control and configuration access:

- **Manage Users**
- **Manage Courses**
- **Manage Academic Calendar**
- **Set Attendance Rules**
- **View Attendance Summary**
- **View System Logs**
- **Generate Reports**
- **Backup/Restore Data**
- **Login/Logout/Reset Password/Update Profile**

🔄 Common Use Cases Across Roles

- **Authentication Functions**: Login, Logout, Reset Password, Update Profile – available to all actors.
- **Record Attendance Date and Time**: Shared between multiple attendance-related actions.
- **Select Course for Attendance**: Needed for both manual and auto attendance.
- **Check Proximity for Auto Attendance**: Tied to automatic location-based marking.

🔍 Specialized Functionality

- **Auto Attendance**:
    - Student: Mark Auto Attendance → Check Proximity → Record Attendance Date and Time
- **Manual Attendance**:
    - Teacher: Take Manual Attendance → Record Attendance Date and Time
- **System Maintenance (Admin only)**: Data backup/restore, log viewing, rule setting.

✅ Strengths

- **Comprehensive**: Covers all core attendance-related tasks.
- **Well-separated Roles**: Each actor has specific, relevant permissions.

- **Use of Includes**: Logical connections such as Record Attendance Date and Time reused in both auto/manual marking.
- **Security-Oriented**: Login, password reset, and profile updates for each role.

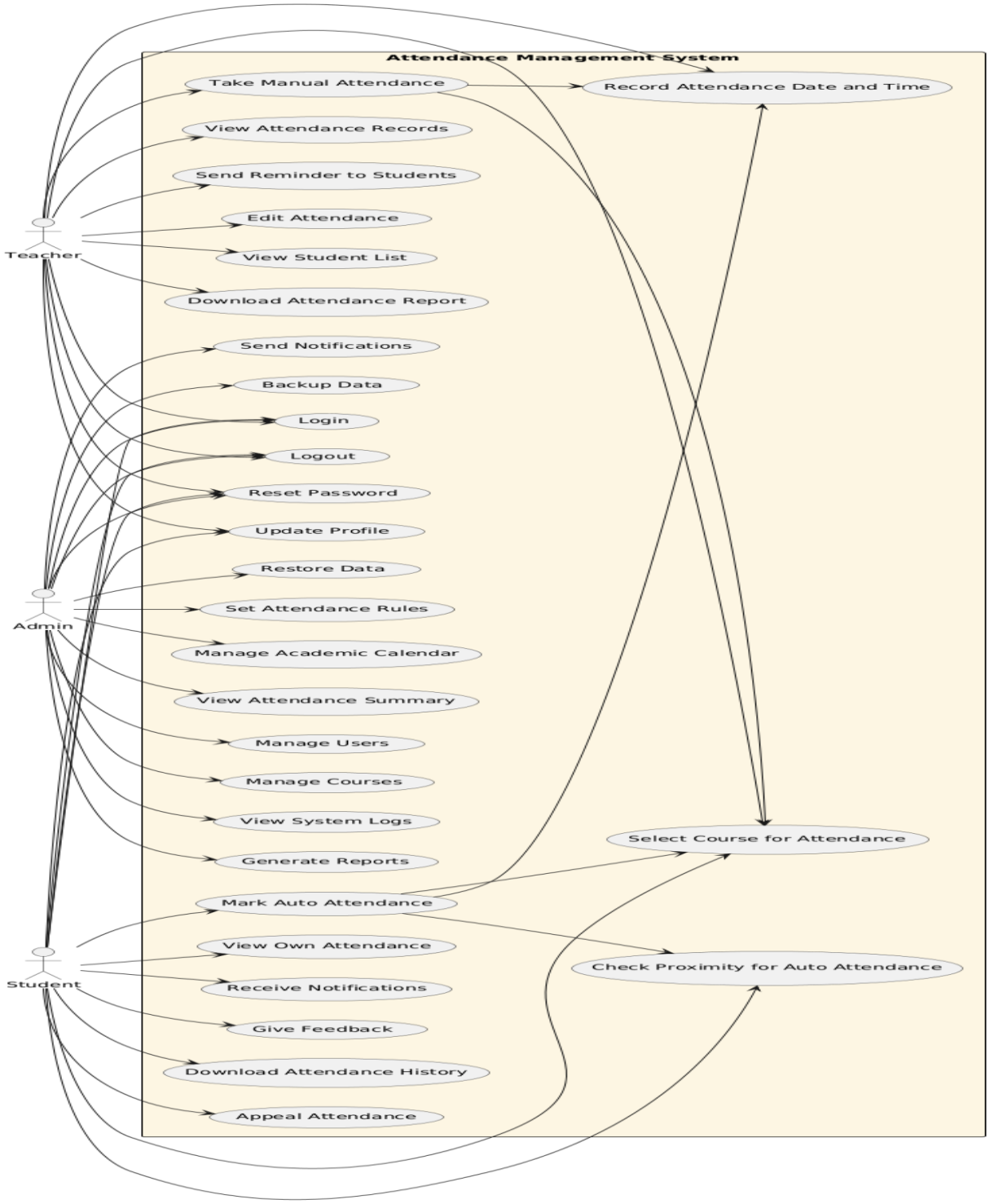
⚠️ Areas for Improvement

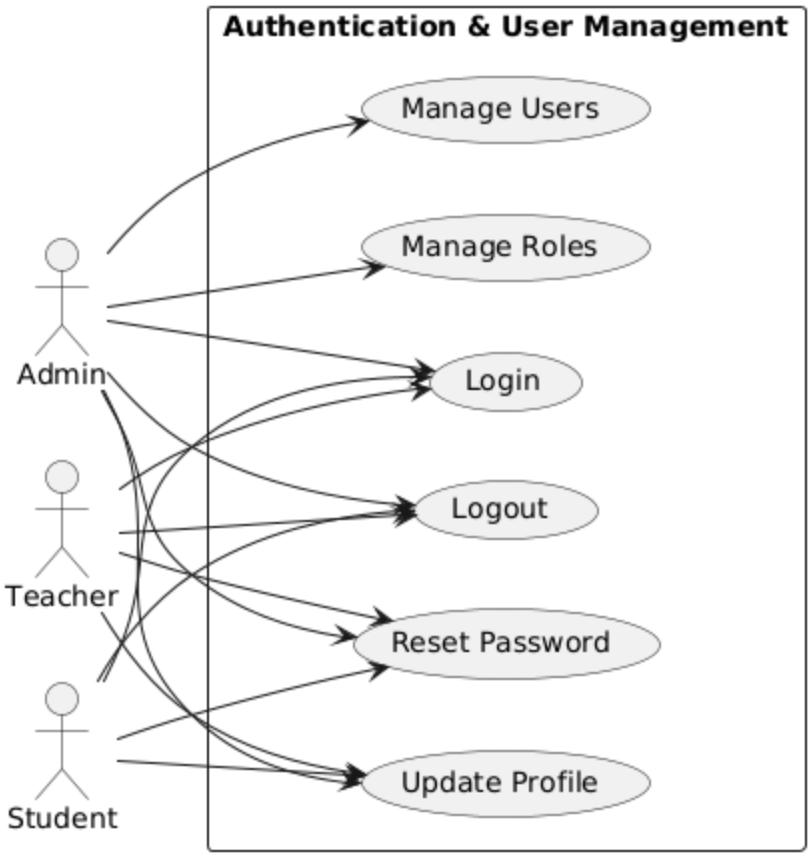
- **Too Many Direct Associations**: Actors are directly connected to many use cases; consider grouping related ones under more generalized actions (like "Manage Attendance").
- **No System Response**: Could improve by adding system reactions or messages (e.g., "Attendance Marked", "Report Generated").
- **Missing Use Case Descriptions**: For detailed system design, textual descriptions (use case specifications) should accompany this diagram.

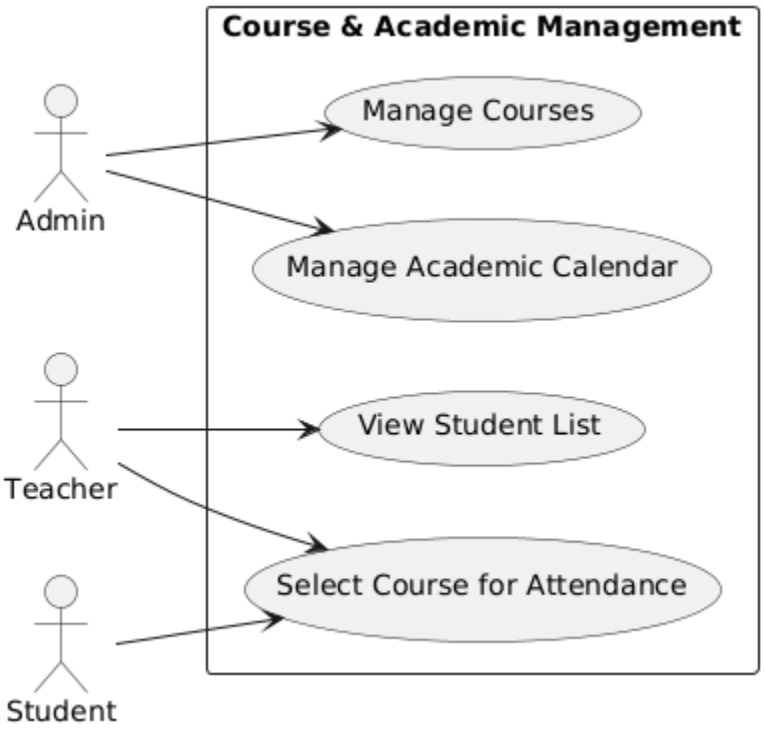
📝 Summary

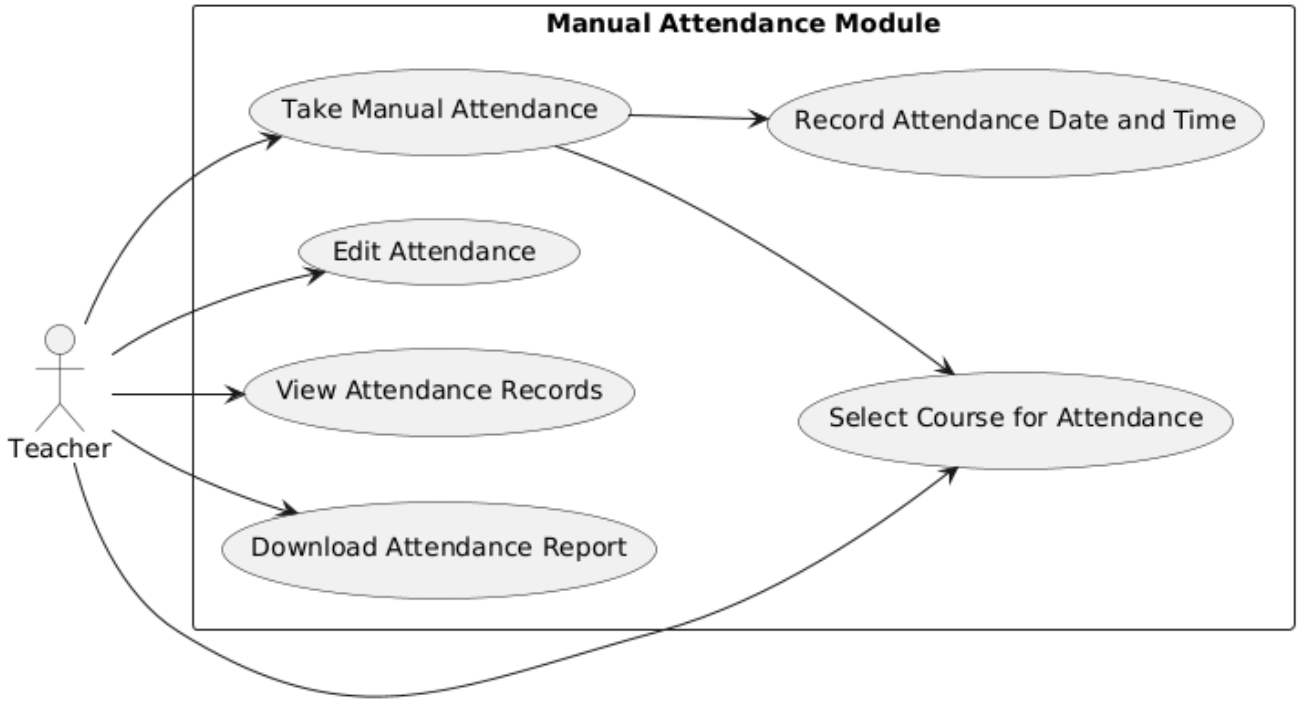
This Use Case Diagram effectively outlines a robust Attendance Management System with functional segregation among Admin, Teacher, and Student roles. It includes both **manual and automated** attendance tracking and administrative features such as **rule-setting and data recovery**. Some refinements in hierarchy or grouping could make the diagram clearer, but functionally, it is solid.

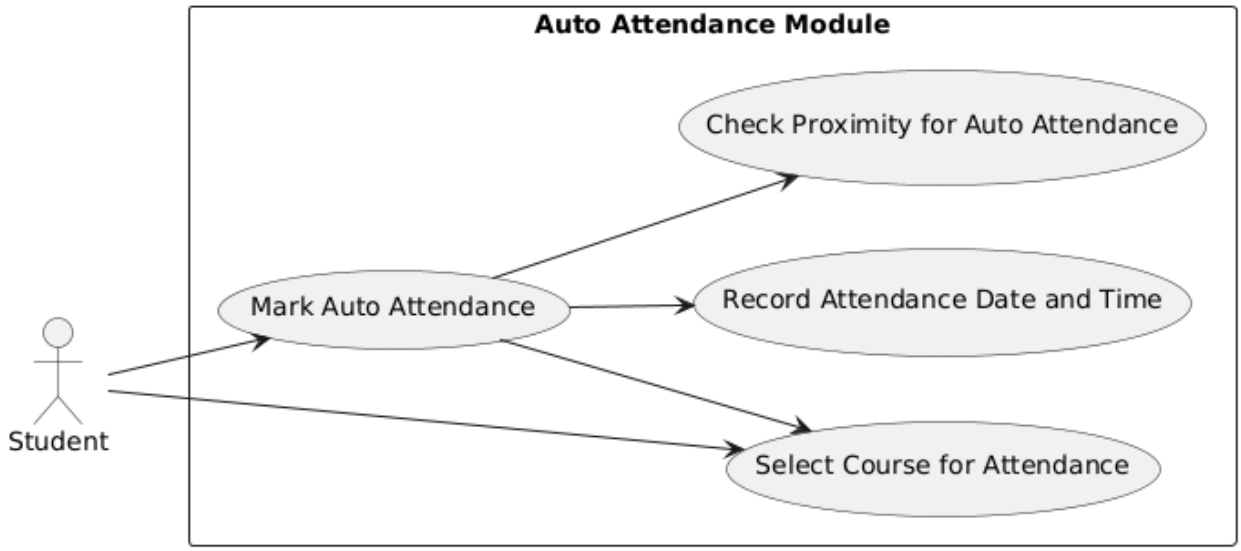
USECASE

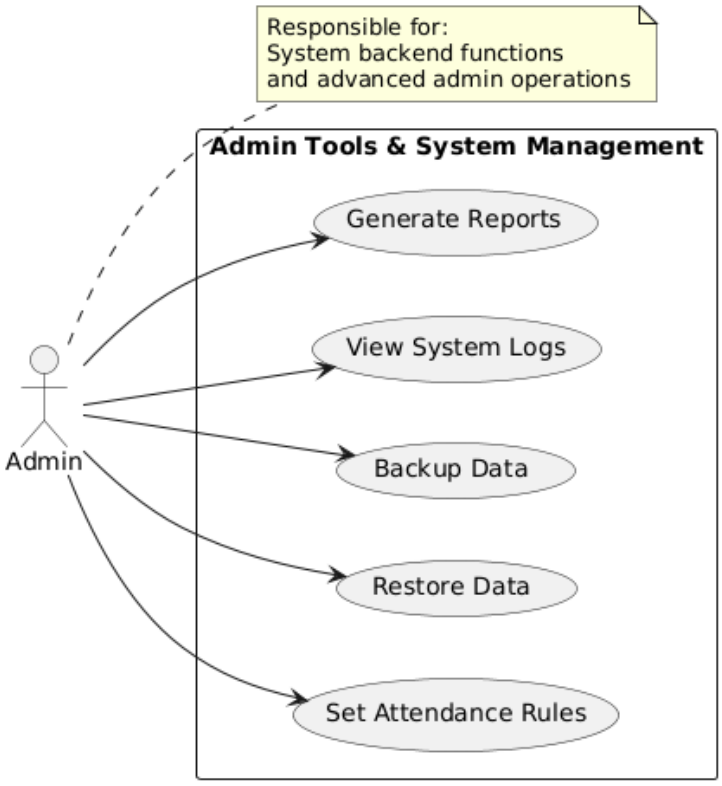

Attendance Management System
Take Manual Attendance
Record Attendance Date and Time
View Attendance Records
Send Reminder to Students
Edit Attendance
View Student List
Download Attendance Report
Send Notifications
Backup Data
Login
Logout
Reset Password
Update Profile
Restore Data
Set Attendance Rules
Manage Academic Calendar
View Attendance Summary
Manage Users
Manage Courses
View System Logs
Generate Reports
Mark Auto Attendance
View Own Attendance
Receive Notifications
Give Feedback
Download Attendance History
Appeal Attendance
Select Course for Attendance
Check Proximity for Auto Attendance
Teacher
Admin
Student

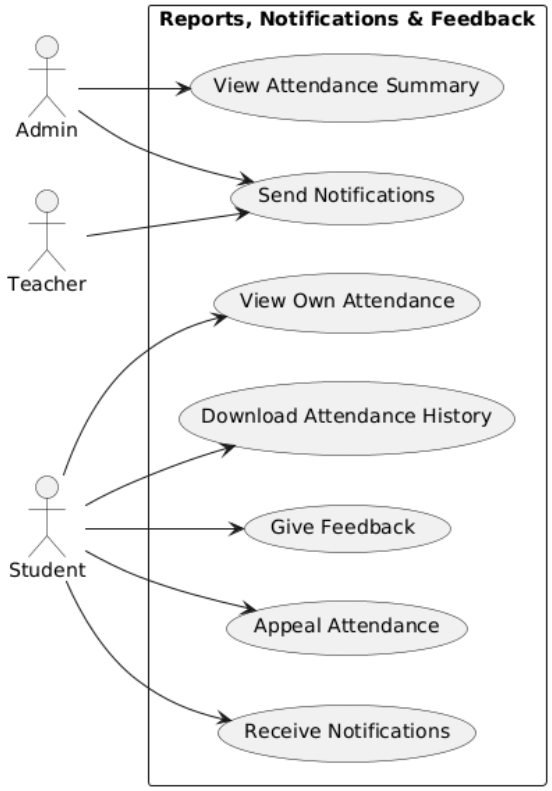

Authentication & User Management
Manage Users
Manage Roles
Login
Logout
Reset Password
Update Profile
Admin
Teacher
Student

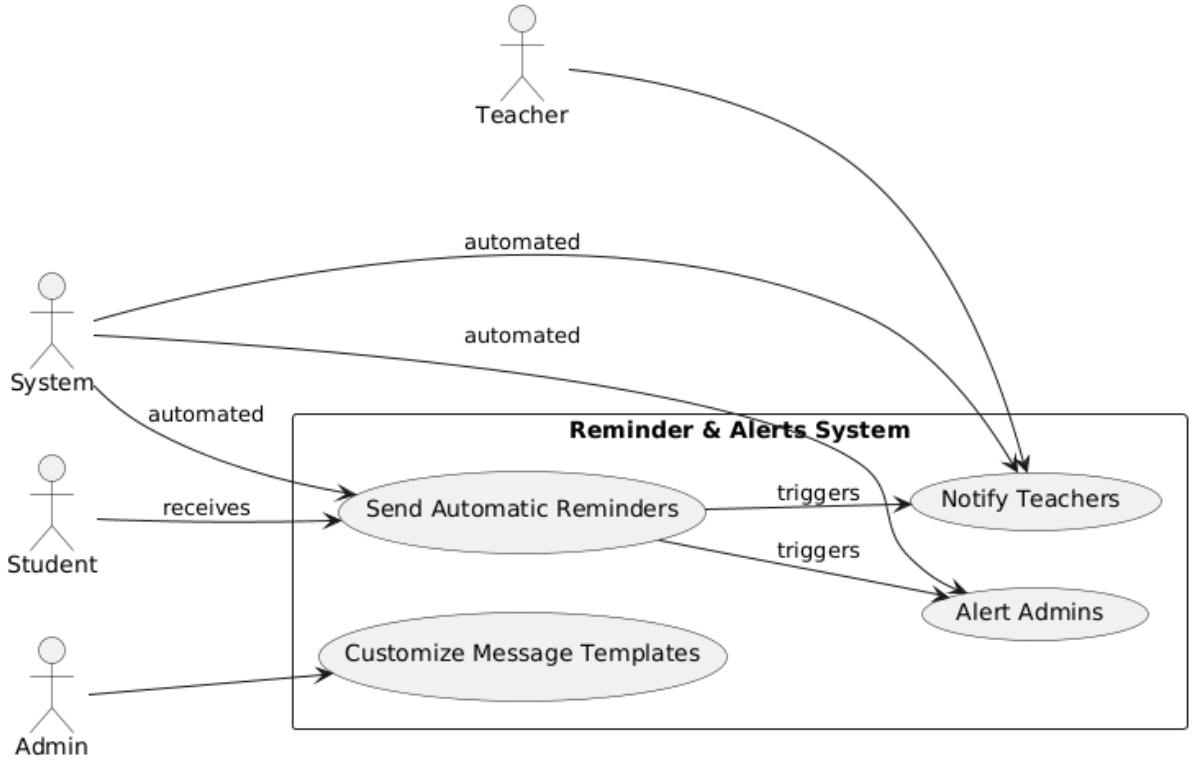

Course & Academic Management
Manage Courses
Manage Academic Calendar
View Student List
Select Course for Attendance
Admin
Teacher
Student


Manual Attendance Module
Take Manual Attendance
Record Attendance Date and Time
Edit Attendance
View Attendance Records
Select Course for Attendance
Download Attendance Report
Teacher


Auto Attendance Module
Check Proximity for Auto Attendance
Mark Auto Attendance
Record Attendance Date and Time
Select Course for Attendance
Student


Responsible for:
System backend functions
and advanced admin operations
Admin Tools & System Management
Generate Reports
View System Logs
Backup Data
Restore Data
Set Attendance Rules
Admin


Reports, Notifications & Feedback
Admin
View Attendance Summary
Send Notifications
Teacher
View Own Attendance
Download Attendance History
Student
Give Feedback
Appeal Attendance
Receive Notifications


Teacher
automated
System
automated
automated
Reminder & Alerts System
Student
receives
Send Automatic Reminders
triggers
Notify Teachers
triggers
Alert Admins
Customize Message Templates
Admin

Chapter 2

Fully Dressed Use Cases

Use Case Name	Login
Scope	Authentication & User Management System
Level	User-goal
Primary Actor	Admin, Teacher, Student
Stakeholders and Interests	- **Users**: Want to securely access the system. - **System Owner**: Wants only authorized users to access the system.
Preconditions	- User must have a valid username and password. - System must be online and accessible.
Success Guarantee	User is authenticated and redirected to their respective dashboard.
Main Success Scenario	1. User navigates to the login page. 2. User enters username and password. 3. System validates credentials. 4. System logs the user in. 5. User is redirected to their dashboard.
Extensions	3a. Invalid credentials: 3a1. System displays error message. 3a2. User retries or resets password. 4a. System unavailable: 4a1. Display maintenance message or error.
Special Requirements	- Must comply with password policy (e.g., complexity). - Must complete within 3 seconds. - Audit login attempts. - Support for CAPTCHA after multiple failed attempts.

Use Case Name:

Course and Academic Management

Scenario:

Admin adds, updates, or removes course records in the academic system.

Triggering Event:

The admin selects the "Manage Courses" option from the system menu**.**

Brief Description:

The Admin manages the list of courses by adding new courses, updating existing ones, or removing outdated courses. Each course includes details such as course name, code, description, and assigned teacher**.**

Primary Actor:

Admin

Supporting Actors:

- Teacher: The course must be assigned to a valid teacher already in the system.
- Student: Students are indirectly affected, as they will enroll in or view the courses added/updated by the admin.

Stakeholders and Interests:

- Admin: Wants to ensure the course catalog is accurate and up-to-date.
- Teachers: Need to be correctly assigned to courses.
- Students: Need access to current and valid course offerings for enrollment and attendance.

Preconditions:

- Admin is logged into the system.
- Course Management module is available and functioning.
- Teacher data must already exist in the system to assign to courses**.**

Postconditions:

- Course data is saved, updated, or removed successfully.
- The changes are reflected in the system immediately.
- Teachers and students see the updated course list where applicable.

Main Flow (Basic Flow):

Step	Actor (Admin)	System Response
1	Selects “Manage Courses” option.	1.1 Displays course management interface.
2	Chooses to add a new course.	2.1 Displays form to enter course details.
3	Enters course name, code, description, and assigns a teacher.	3.1 Validates the input fields.
4	Confirms submission.	4.1 Saves course data to the database. 4.2 Shows success message.
5	(Optional) Selects a course to update or delete.	5.1 Displays selected course information. 5.2 Updates or removes course as requested.

Alternate Flows / Exceptions:

- 3a. If any required field is missing → system shows an error and highlights missing data.
- 3b. If assigned teacher ID does not exist → system rejects input and asks to select a valid teacher.
- 4a. If database error occurs → system shows failure message and logs the error.
- 5a. If the course is currently assigned to students, deletion may be restricted → system warns admin.

Use Case Name: Take Manual Attendance

Scenario:

A teacher manually records student attendance for a selected course.

Triggering Event:

The teacher initiates the attendance process by selecting the "Take Manual Attendance" option in the system.

Brief Description:

This use case allows a teacher to take attendance for a selected course by marking student presence or absence. The attendance date and time are recorded automatically. The teacher can later view, edit, or download attendance records.

Actors:

- **Primary Actor:** Teacher

Related Use Cases:

- **Record Attendance Date and Time** (Included)
- **Select Course for Attendance** (Included)
- **Edit Attendance**
- **View Attendance Records**
- **Download Attendance Report**

Stakeholders and Interests:

- **Teacher**: Wants an easy and reliable way to take and manage attendance.
- **School Administration**: Requires accurate attendance data for monitoring and reporting.
- **Students**: Affected by the accuracy of recorded attendance.

Preconditions:

- The teacher must be logged into the system.
- Courses must already be assigned to the teacher.

Postconditions:

- Attendance for the selected course is recorded with date and time.
- The data is stored in the attendance record database.

Flow of Activities:

Actor Actions (Teacher):

1. Logs into the system.
2. Navigates to the **Manual Attendance Module**.
3. Clicks **Take Manual Attendance**.
4. Selects a course from the list.
5. Marks each student as present or absent.
6. Submits the attendance.

System Responses:

1. Displays the list of assigned courses.
2. Shows the list of students enrolled in the selected course.
3. Records the attendance status along with the current date and time.
4. Confirms successful attendance submission.

Exceptions/Alternative Flows:

- **Course Not Assigned**:
  - System displays an error if no courses are found for the teacher.
- **Partial Submission**:
  - If attendance is submitted without marking all students, the system prompts for confirmation.
- **System Timeout**:
  - If the session times out, the teacher must log in again and restart the process.

Use Case Table: Auto Attendance via Face Recognition

Element	Description
Use Case Name	Auto Attendance (Face Recognition)
Primary Actor	Student
Secondary Actor	System (Face Recognition Module, Camera)
Preconditions	1. Student is enrolled in the course. 2. Student's face is registered in the system. 3. Classroom camera is operational.
Basic Flow	1. **Detect Face**: Camera scans classroom for faces. 2. **Verify Identity**: System matches face with registered student data. 3. **Mark Attendance**: Records "Present" for verified students. 4. **Log Timestamp**: Saves date/time and notifies student (e.g., SMS/app notification).
Alternative Flows	**A1:** Unrecognized face → Flag for manual review. **A2:** Multiple faces detected → Process sequentially. **A3:** Low lighting → Use IR camera or notify staff.
Exceptions	**E1:** Camera failure → Switch to manual attendance. **E2:** Database error → Retry or alert admin. **E3:** Privacy violation → Log incident and anonymize data.
Postconditions	1. Attendance is recorded in the database. 2. Faculty can generate reports.
Related Use Cases	Manual Attendance, Student Registration (Face Enrollment)

Key Features:

- **Accuracy**: Uses deep learning to minimize false positives/negatives.
- **Privacy Compliance**: Blurs faces of non-enrolled individuals.
- **Fallback**: Integrates with manual attendance if tech fails.

Use Case Name	**Generate Reports**
Scope	Admin Tools & System Management
Level	User-goal
Primary Actor	Admin
Stakeholders and Interests	Admin wants to retrieve system reports for auditing, performance monitoring, or record-keeping.
Preconditions	Admin is logged into the system with sufficient privileges.
Success Guarantee	Reports are generated and displayed/downloaded successfully.
Main Success Scenario	1. Admin selects "Generate Reports" option.2. System asks for report type/date range.3. Admin inputs parameters.4. System retrieves data and generates the report.5. Admin views or downloads the report.
Extensions	2a. Admin cancels request → Use case ends.4a. System fails to generate report → Error message shown.
Special Requirements	Reports must be downloadable as PDF. Generation time should be under 5 seconds. Must be accessible only by Admins.

Use Case : Submit Feedback

Use Case Name: Submit Feedback
Actor: Student
Description: Allows a student to submit feedback related to courses, instructors, or the registration process.
Preconditions: Student must be logged into the system.
Postconditions: Feedback is saved in the database and a notification is sent to the admin.

Main Success Scenario:

Student navigates to the feedback form.

Student enters feedback and submits the form.

The system validates the input.

Feedback is stored in the database.

A notification is triggered for the admin.

Alternative Flows:

- Validation fails → error message shown, student corrects input.

Exceptions:

- System/database error while saving → user is shown an error message.

Use Case : Receive Notification

Use Case Name: Receive Notification
Actor: Admin
Description: Admin receives notifications when a new feedback is submitted or when a report is ready.
Preconditions: Admin must be logged in and have notifications enabled.
Postconditions: Admin receives real-time alert in the system dashboard or via email.

Main Success Scenario:

System detects a new event (feedback/report).

Notification Manager generates a message.

Notification is displayed or sent to the admin.

Alternative Flows:

- If admin is offline, notification is stored and displayed on next login.

Exceptions:

- Notification service fails → no message delivered, error logged.

Use Case : View Report

Use Case Name: View Report
Actor: Admin
Description: Allows the admin to view generated reports, including student registrations, feedback statistics, etc.
Preconditions: Report must be generated. Admin must be authenticated.
Postconditions: Report is displayed to the admin.

Main Success Scenario:

Admin navigates to the reports section.

Selects the desired report from the list.

The system fetches and displays the report.

Alternative Flows:

- If no reports exist, a message is shown to the admin.

Exceptions:

- Database error → report fails to load, error message shown.

Use Case 4 Generate Report

Use Case Name: Generate Report
Actor: Admin
Description: Admin can generate various reports such as student registrations, feedback summaries, or usage statistics.
Preconditions: Admin must be authenticated.
Postconditions: A report is generated and saved; notification sent to the admin.

Main Success Scenario:

Admin selects “Generate Report” option.

Admin defines criteria (e.g., date range, course).

System processes the request.

Report is generated and saved.

Notification is sent that the report is ready.

Alternative Flows:

- Invalid criteria → admin asked to correct inputs.

Exceptions:

- System error during report generation → error displayed to admin.

Use Case 5 Send Notification

Use Case Name: Send Notification
Actor: (System use case – triggered by other use cases)
Description: Sends notifications when a new feedback is submitted or a report is generated.
Preconditions: A triggering event (feedback/report) must occur.
Postconditions: Admin receives a system alert.

Main Success Scenario:

System detects an event (new feedback or generated report).

NotificationManager creates a message.

Notification is sent to the admin.

Alternative Flows:

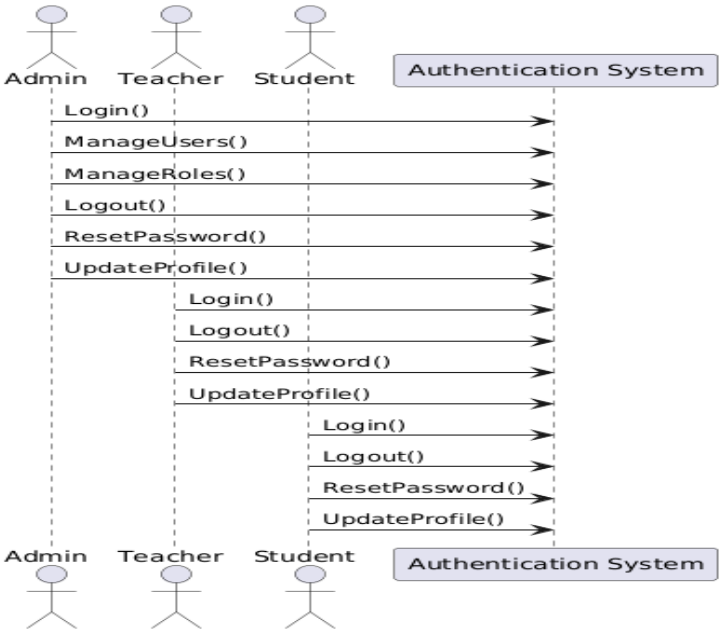
- If admin is unavailable, notification is queued.

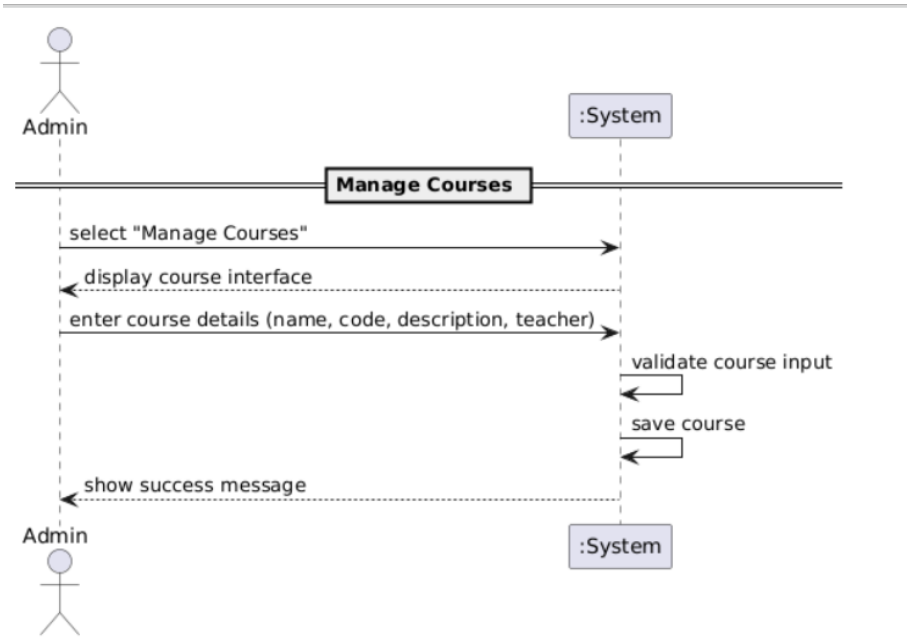
Exceptions:

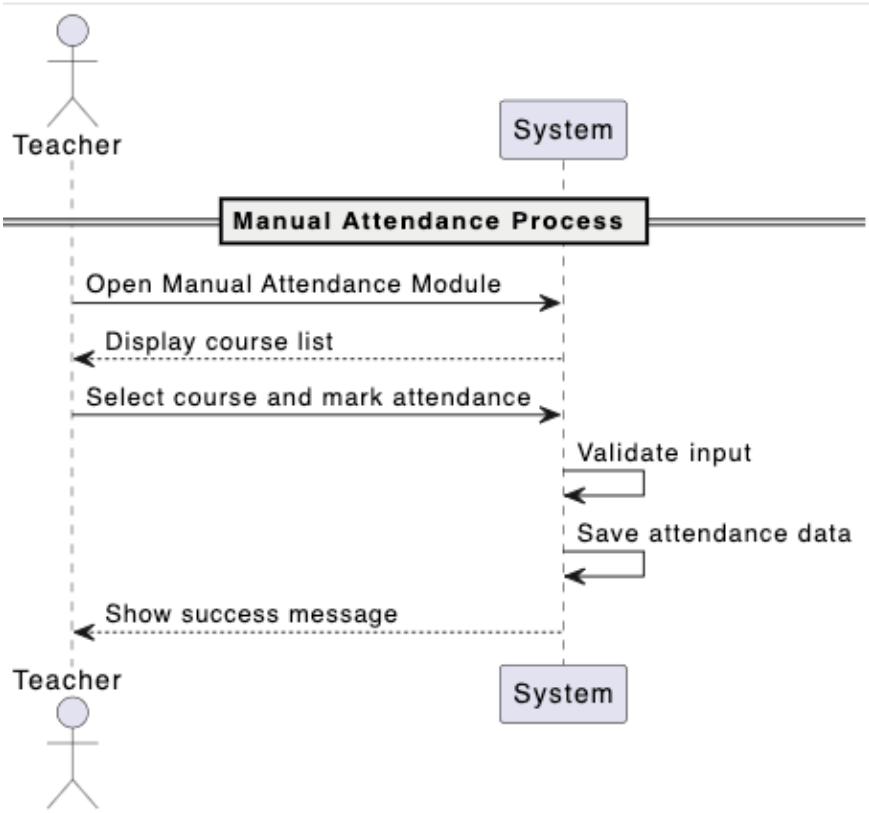
- Notification system failure → log error, retry mechanism initiated.

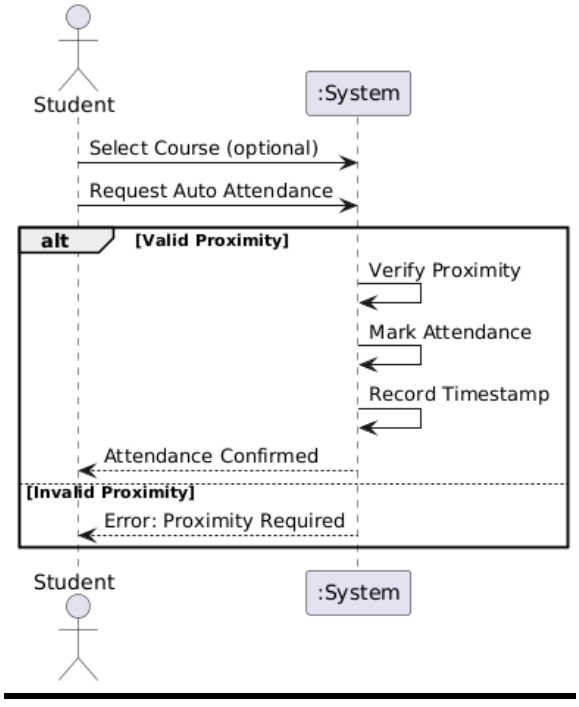
Chapter no 3

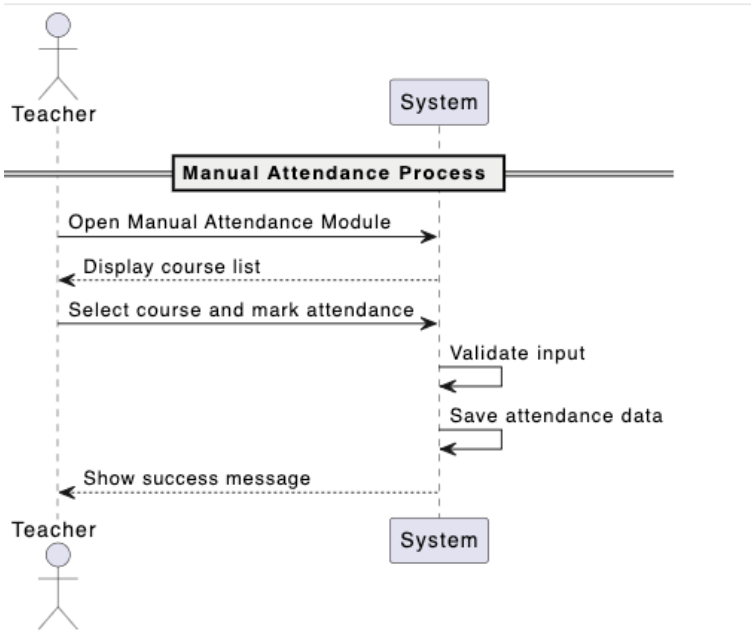
SSDs

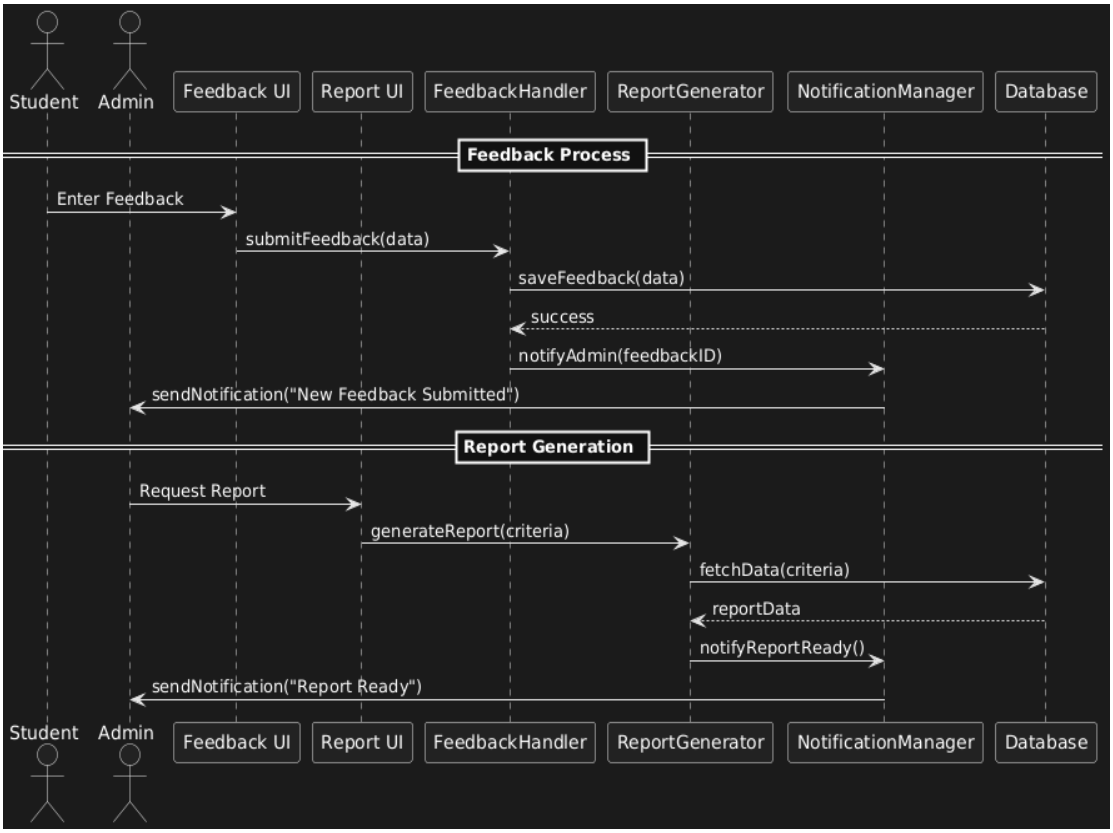


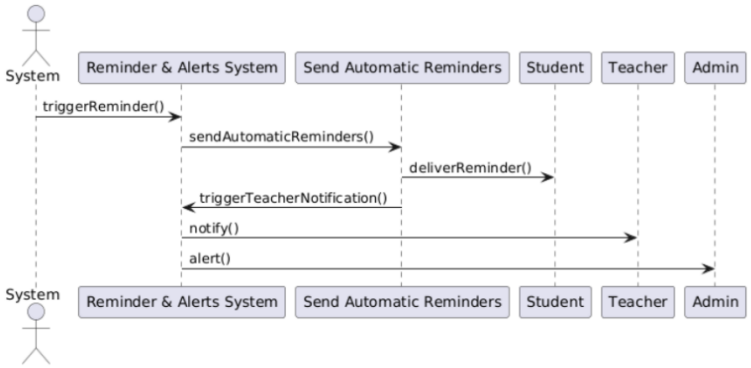



Teacher
System
Manual Attendance Process
Open Manual Attendance Module
Display course list
Select course and mark attendance
Validate input
Save attendance data
Show success message
Teacher
System


Student
:System
Select Course (optional)
Request Auto Attendance
alt
[Valid Proximity]
Verify Proximity
Mark Attendance
Record Timestamp
Attendance Confirmed
[Invalid Proximity]
Error: Proximity Required
Student
:System

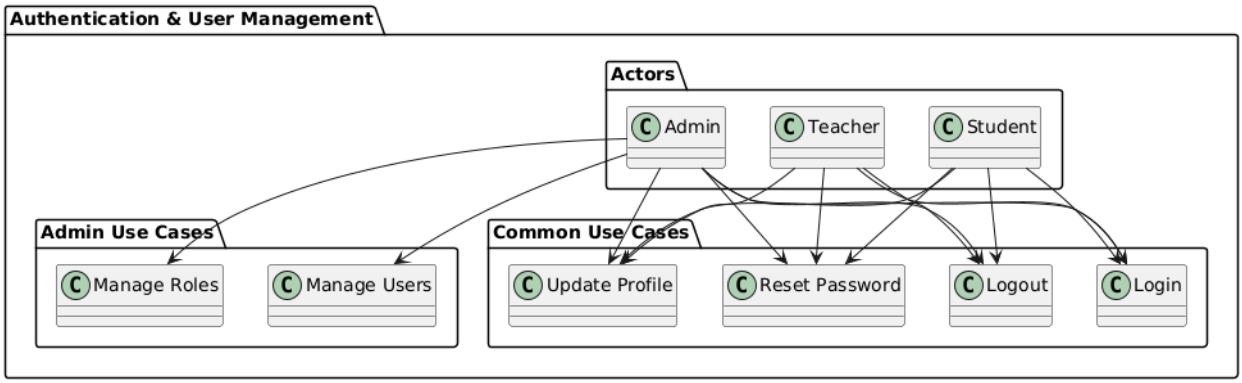

Teacher
System
Manual Attendance Process
Open Manual Attendance Module
Display course list
Select course and mark attendance
Validate input
Save attendance data
Show success message
Teacher
System

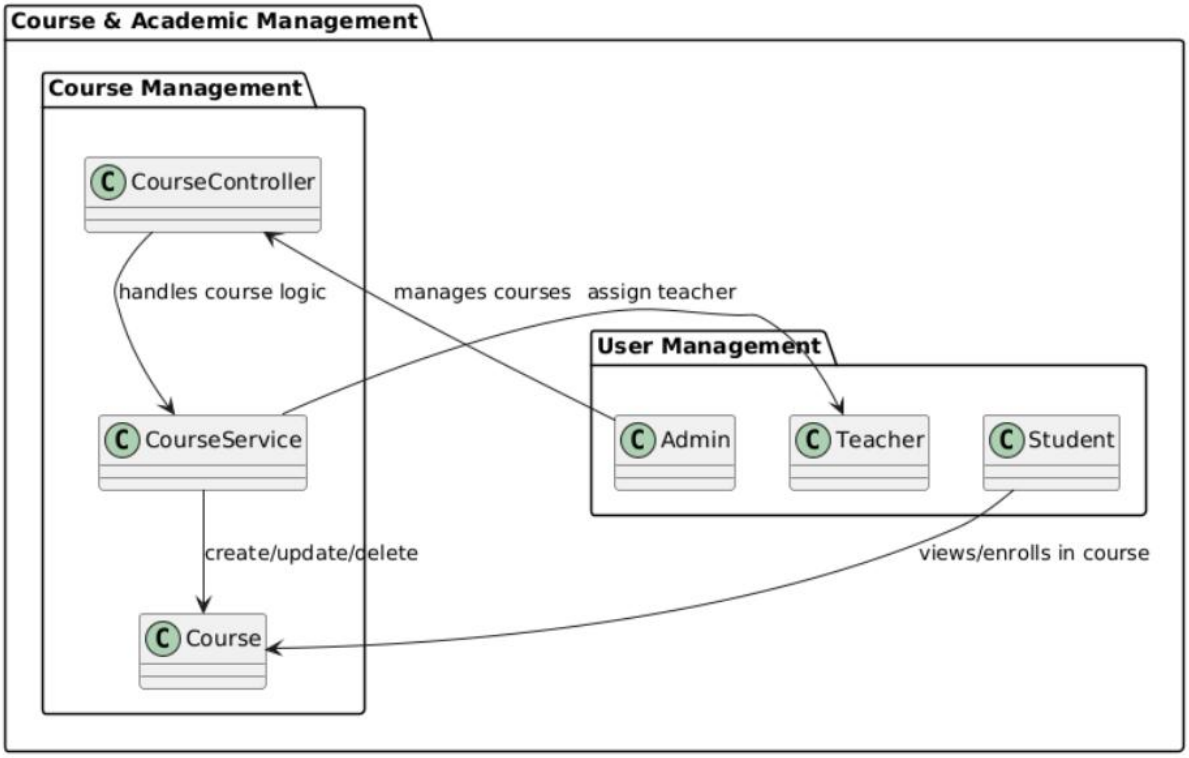

Student
Admin
Feedback UI
Report UI
FeedbackHandler
ReportGenerator
NotificationManager
Database
Feedback Process
Enter Feedback
submitFeedback(data)
saveFeedback(data)
success
notifyAdmin(feedbackID)
sendNotification("New Feedback Submitted")
Report Generation
Request Report
generateReport(criteria)
fetchData(criteria)
reportData
notifyReportReady()
sendNotification("Report Ready")
Student
Admin
Feedback UI
Report UI
FeedbackHandler
ReportGenerator
NotificationManager
Database

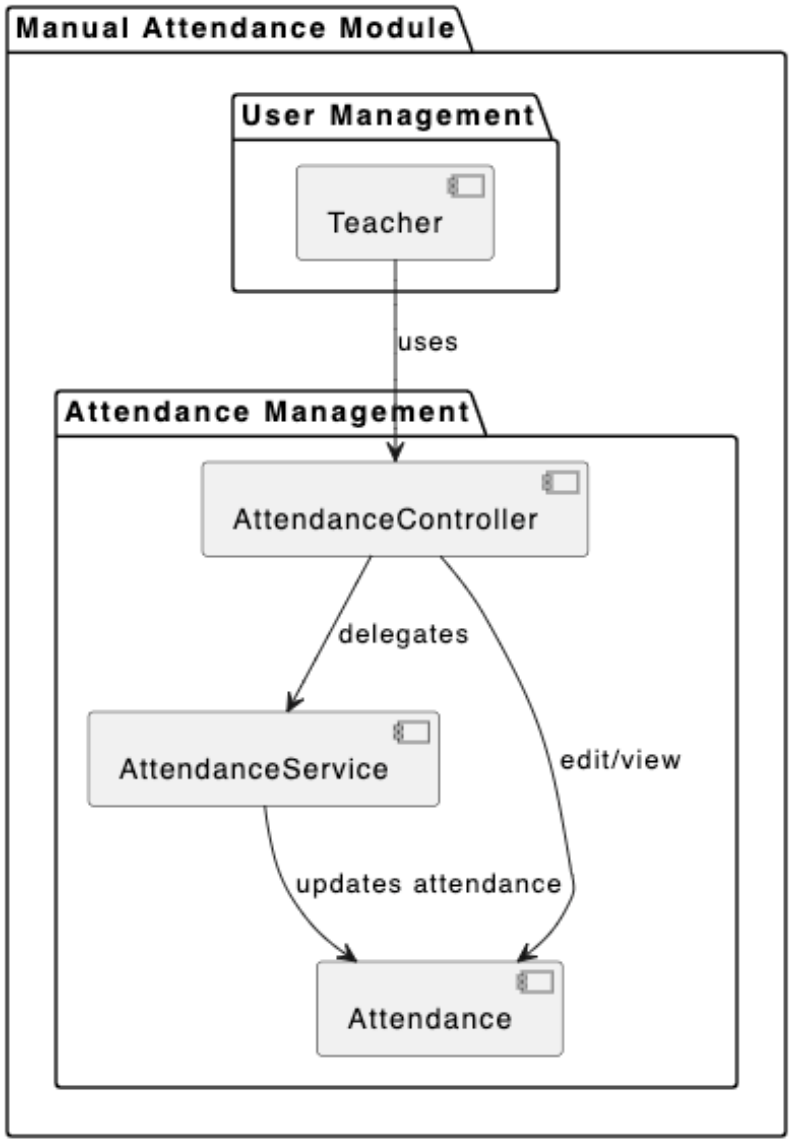

System
Reminder & Alerts System
Send Automatic Reminders
Student
Teacher
Admin
triggerReminder()
sendAutomaticReminders()
deliverReminder()
triggerTeacherNotification()
notify()
alert()
System
Reminder & Alerts System
Send Automatic Reminders
Student
Teacher
Admin

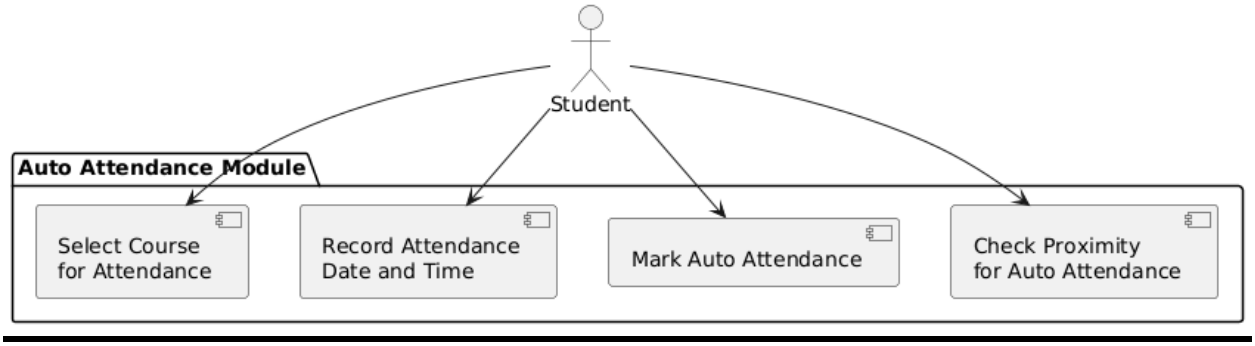
CHAPTER NO 4

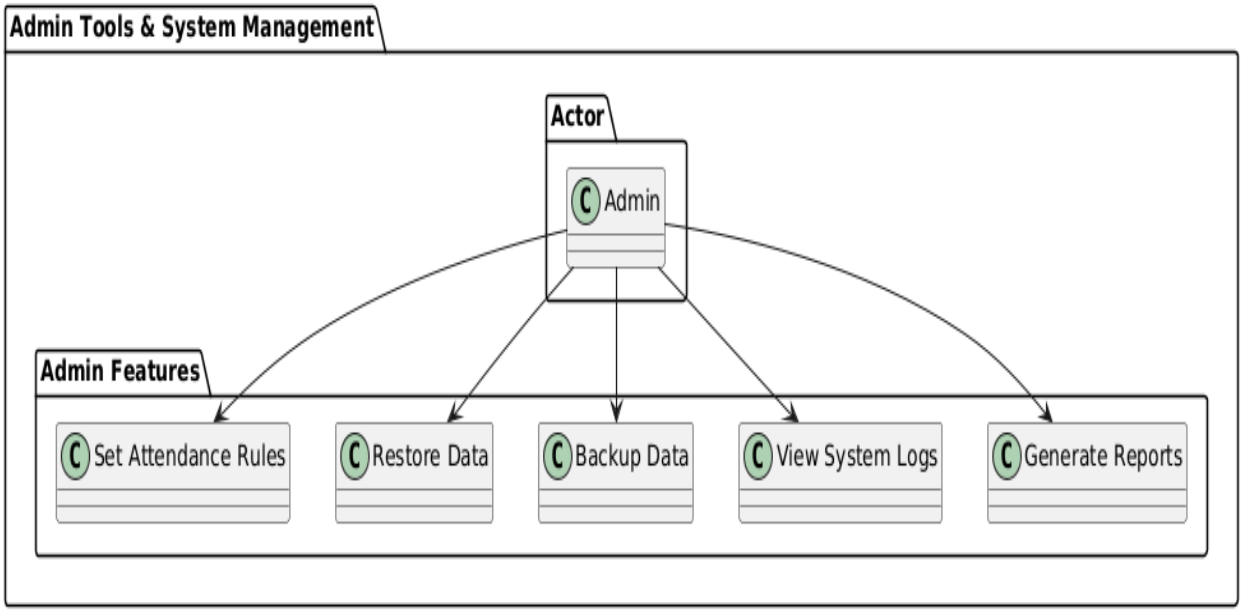
Package diagram

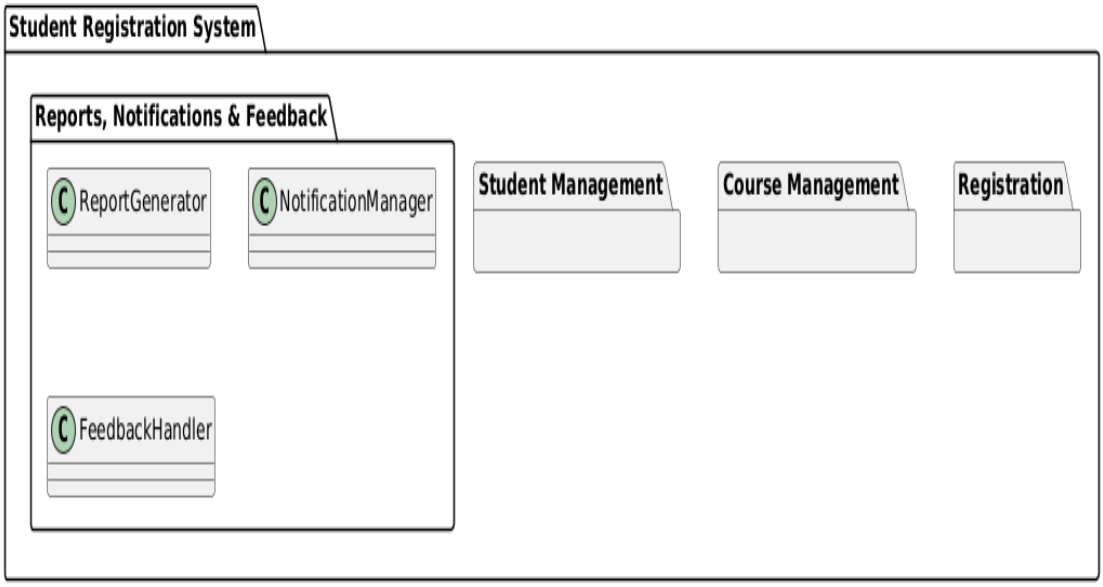


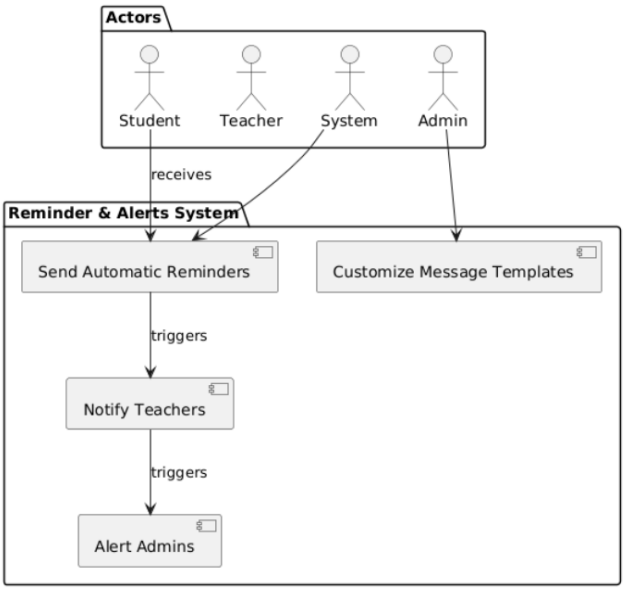



Manual Attendance Module
User Management
Teacher
uses
Attendance Management
AttendanceController
delegates
AttendanceService
edit/view
updates attendance
Attendance


Student
Auto Attendance Module
Select Course for Attendance
Record Attendance Date and Time
Mark Auto Attendance
Check Proximity for Auto Attendance

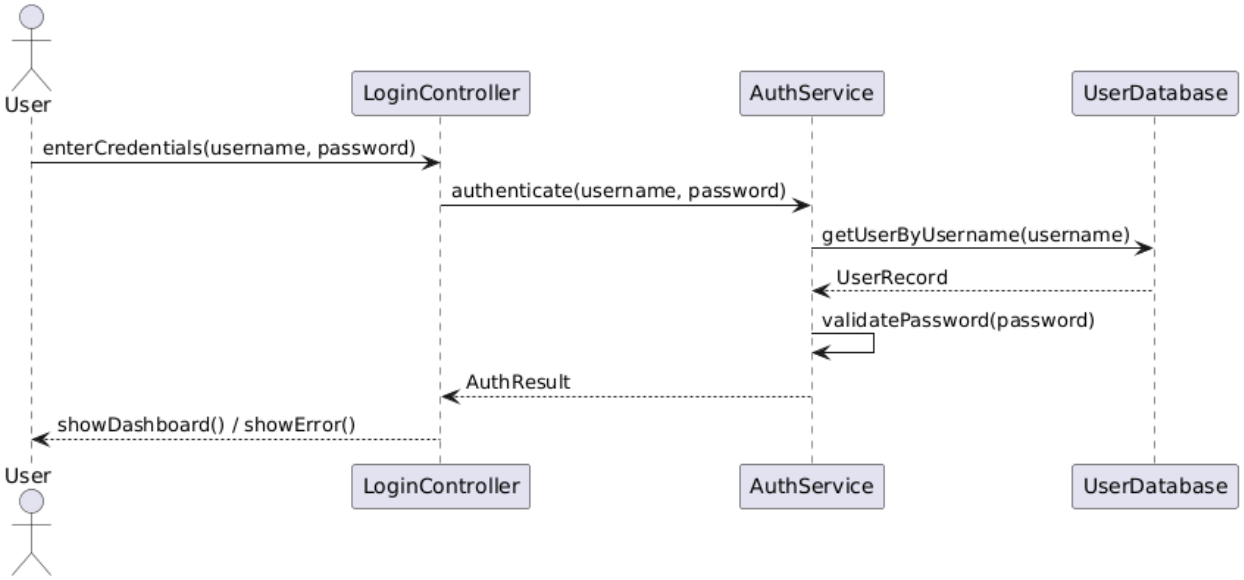

Admin Tools & System Management
Actor
Admin
Admin Features
Set Attendance Rules
Restore Data
Backup Data
View System Logs
Generate Reports

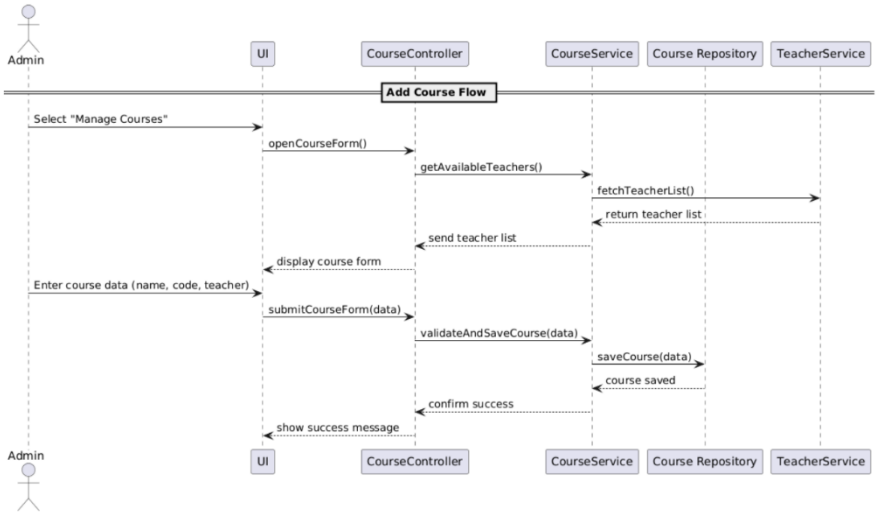

Student Registration System
Reports, Notifications & Feedback
ReportGenerator
NotificationManager
FeedbackHandler
Student Management
Course Management
Registration

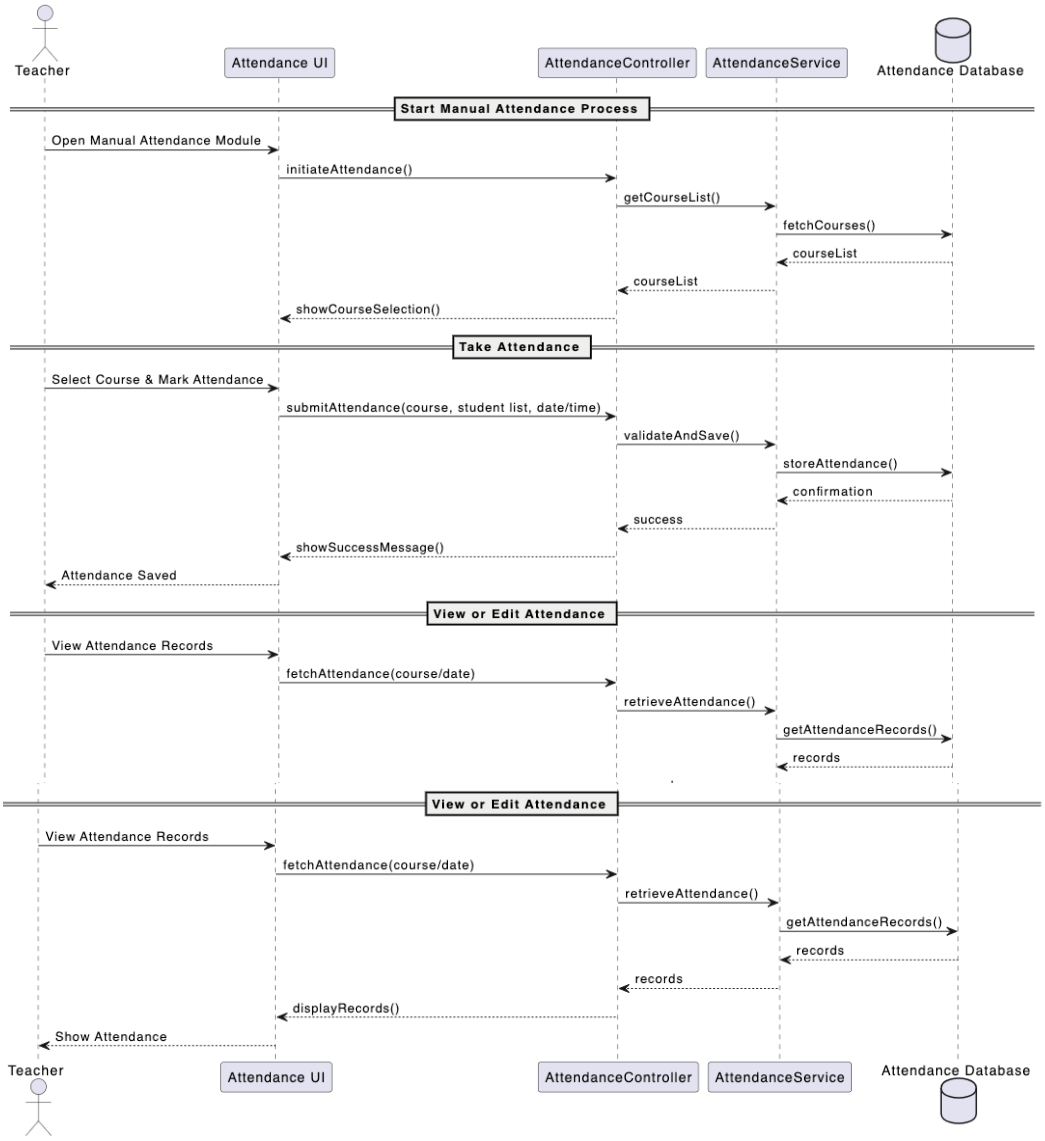

Actors
Student
Teacher
System
Admin
receives
Reminder & Alerts System
Send Automatic Reminders
Customize Message Templates
triggers
Notify Teachers
triggers
Alert Admins

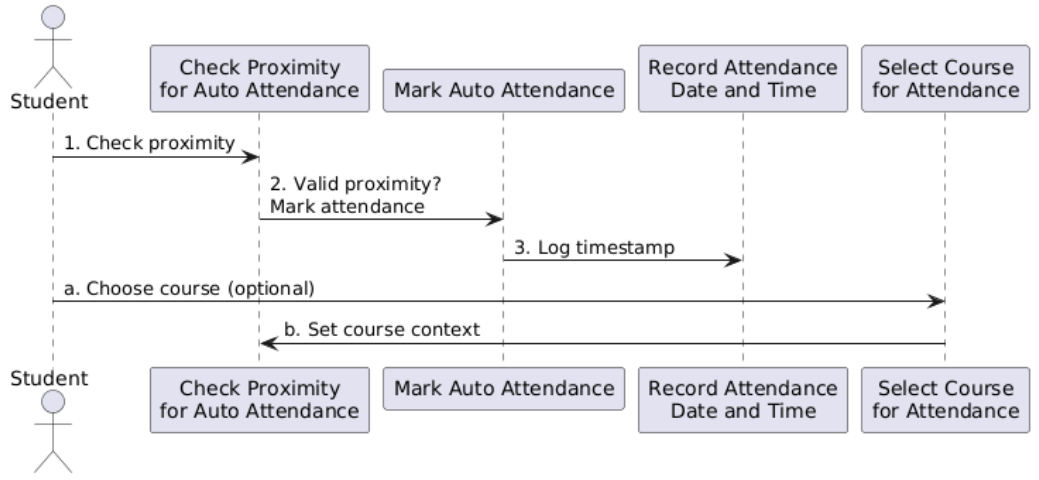
Chapter 5

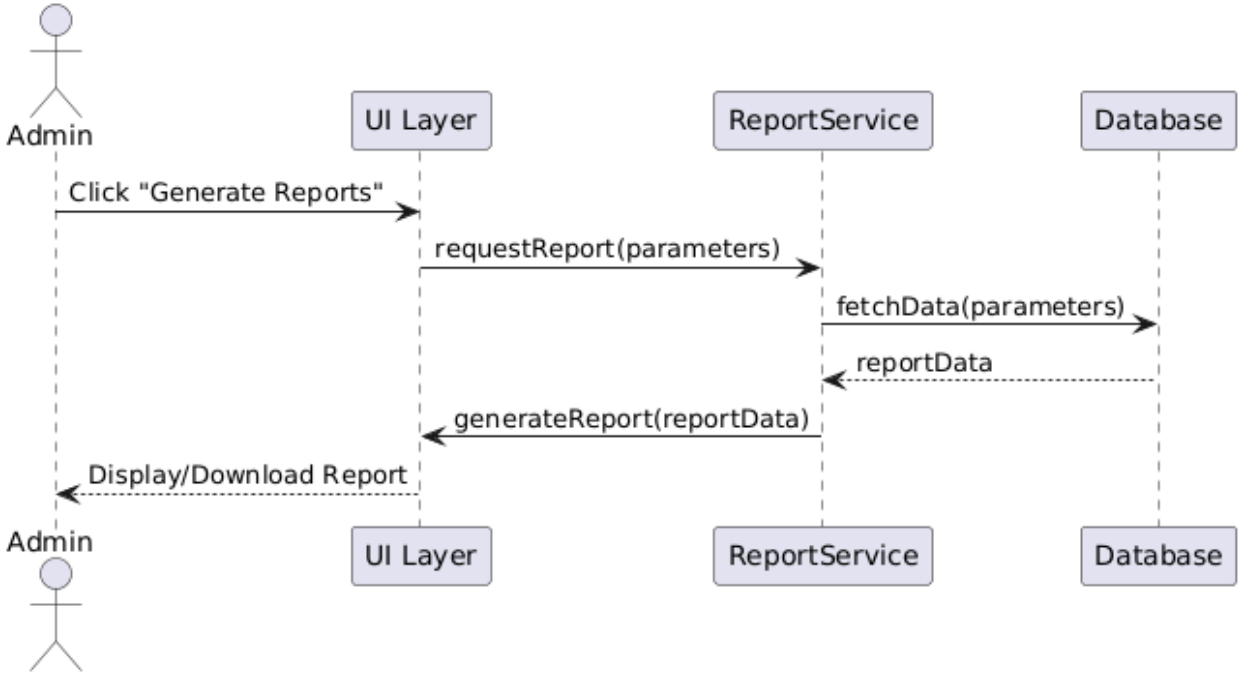
Collaboration Diagrams

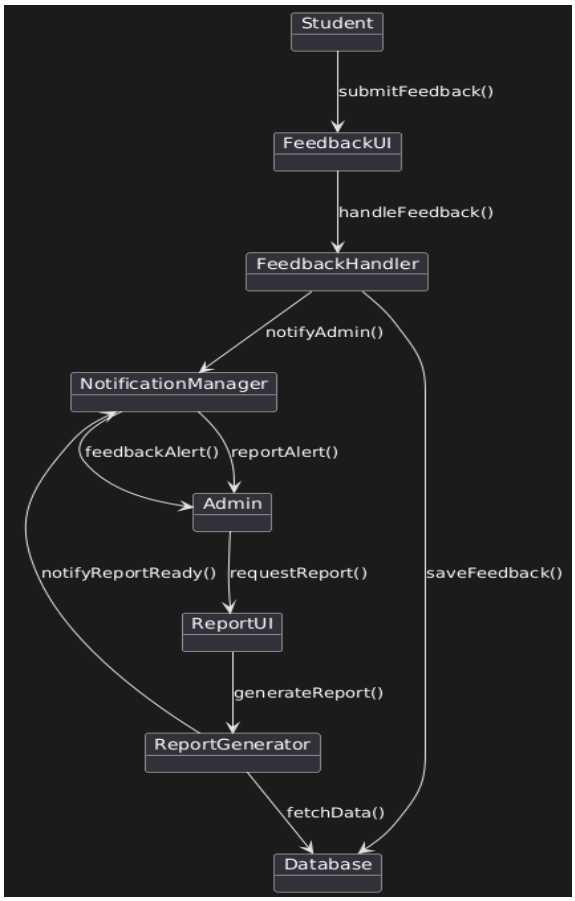


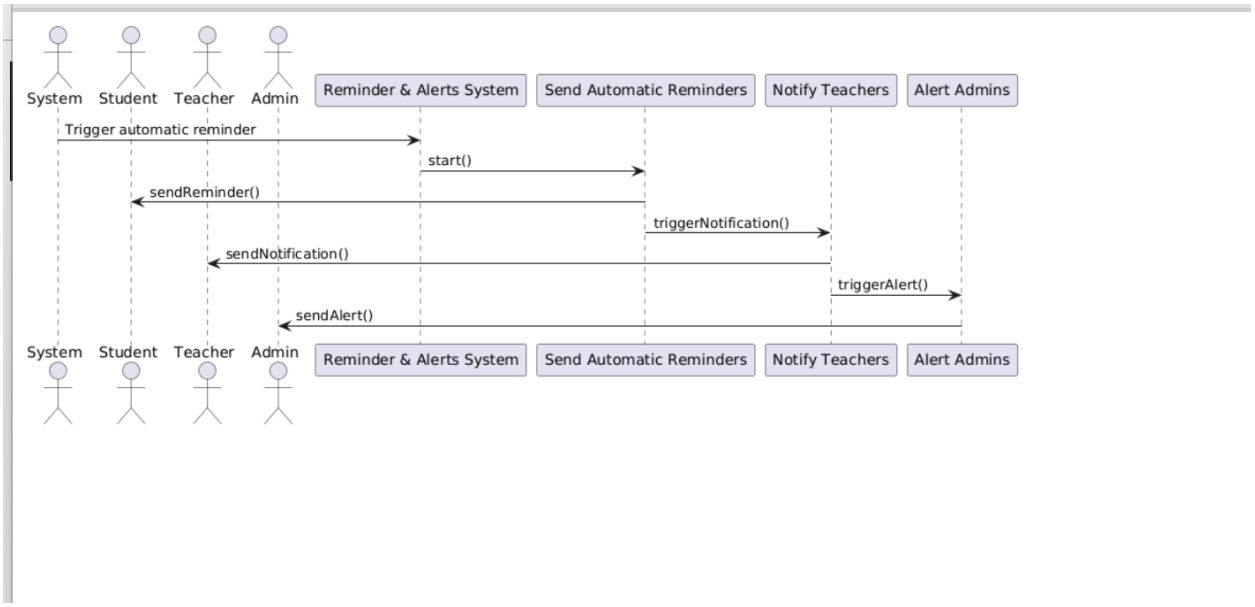



Teacher
Attendance UI
AttendanceController
AttendanceService
Attendance Database
Start Manual Attendance Process
Open Manual Attendance Module
initiateAttendance()
getCourseList()
fetchCourses()
courseList
courseList
showCourseSelection()
Take Attendance
Select Course & Mark Attendance
submitAttendance(course, student list, date/time)
validateAndSave()
storeAttendance()
confirmation
success
showSuccessMessage()
Attendance Saved
View or Edit Attendance
View Attendance Records
fetchAttendance(course/date)
retrieveAttendance()
getAttendanceRecords()
records
View or Edit Attendance
View Attendance Records
fetchAttendance(course/date)
retrieveAttendance()
getAttendanceRecords()
records
records
displayRecords()
Show Attendance
Teacher
Attendance UI
AttendanceController
AttendanceService
Attendance Database


Student
Check Proximity for Auto Attendance
Mark Auto Attendance
Record Attendance Date and Time
Select Course for Attendance
1. Check proximity
2. Valid proximity? Mark attendance
3. Log timestamp
a. Choose course (optional)
b. Set course context
Student
Check Proximity for Auto Attendance
Mark Auto Attendance
Record Attendance Date and Time
Select Course for Attendance

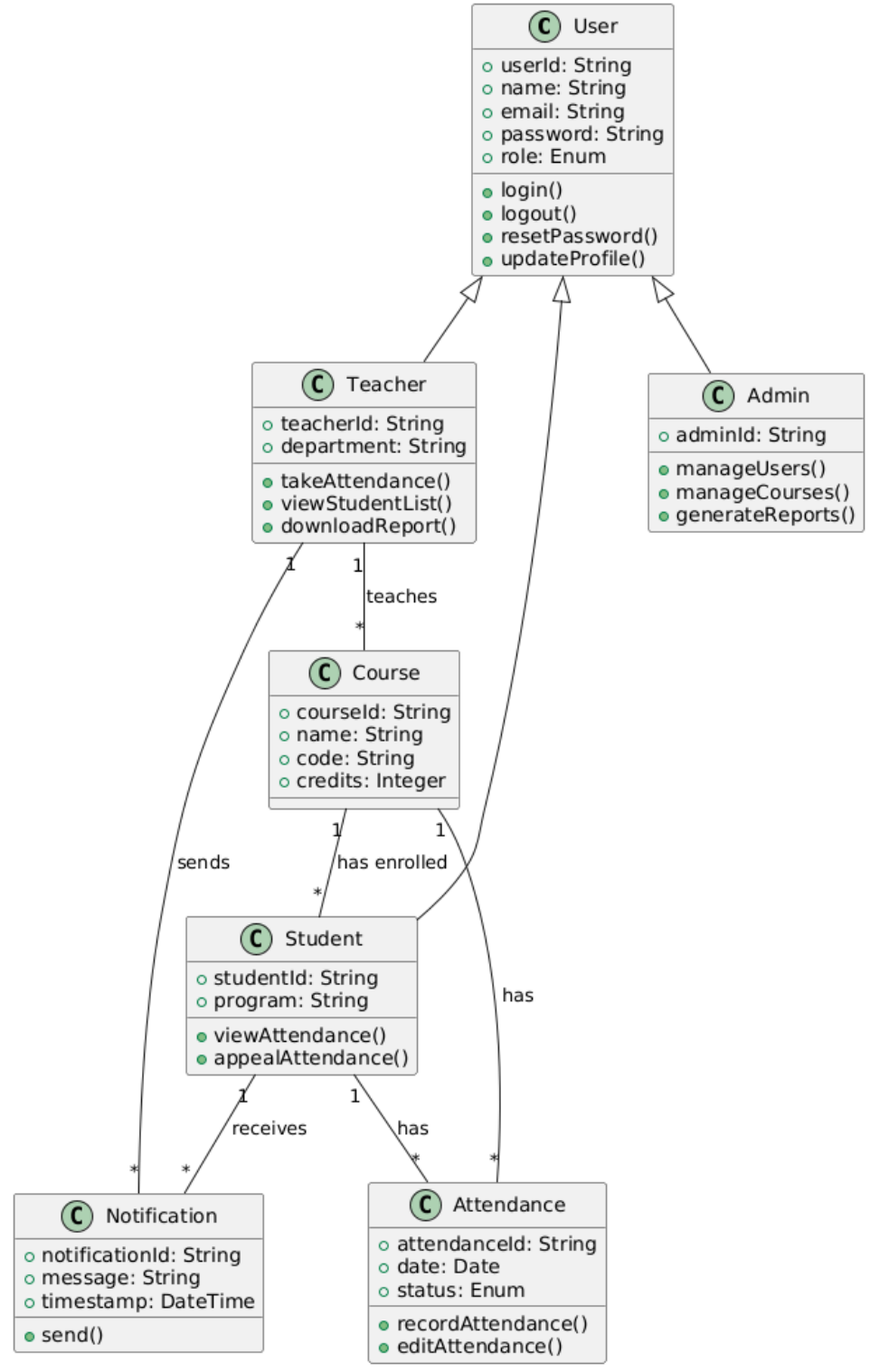

Admin
UI Layer
ReportService
Database
Click "Generate Reports"
requestReport(parameters)
fetchData(parameters)
reportData
generateReport(reportData)
Display/Download Report
Admin
UI Layer
ReportService
Database


Student
submitFeedback()
FeedbackUI
handleFeedback()
FeedbackHandler
notifyAdmin()
NotificationManager
feedbackAlert()
reportAlert()
Admin
notifyReportReady()
requestReport()
saveFeedback()
ReportUI
generateReport()
ReportGenerator
fetchData()
Database


System
Student
Teacher
Admin
Reminder & Alerts System
Send Automatic Reminders
Notify Teachers
Alert Admins
Trigger automatic reminder
start()
sendReminder()
triggerNotification()
sendNotification()
triggerAlert()
sendAlert()
System
Student
Teacher
Admin
Reminder & Alerts System
Send Automatic Reminders
Notify Teachers
Alert Admins

Chapter 6

Class Diagram



Chapter 7

Coding Criteria for Documentation

📁 1. Project Structure

Folder	Description
src/application	Core source code including facial recognition, motion detection, and object tracking modules.
src/application/haar	Pre-trained Haar cascade XML files used for face and body detection (OpenCV).
bin and target	Compiled Java classes and JAR files.
imaes	Contains a readme.md (possibly image assets or markdown info).
src	Image and CSS assets used by the GUI.
Sample.fxml, application.css	JavaFX layout and styling files.
pom.xml	Maven build configuration.
opencv-341.jar	OpenCV Java binding JAR file.

📦 Key Files Overview

File	Purpose
Main.java	Entry point for the application.
FaceDetector.java	Handles face detection using OpenCV and Haar cascades.
FaceRecognizer.java	Implements facial recognition logic.
MotionDetector.java	Detects motion using frame comparison.
OCR.java	Performs optical character recognition.
Database.java	Manages database interactions.
SampleController.java	JavaFX UI logic controller.

🖥️ 2. Language & Libraries

- **Language:** Java (likely Java 8+)
- **Libraries:**
  - **OpenCV** – For face detection, tracking, and recognition.
  - **JavaFX** – GUI development.
  - **Maven** – Dependency and build management.
  -

Installation:

1. Install Java JDK 8+.
2. Set up OpenCV with opencv-341.jar.
3. Build with Maven:
4. mvn clean install

📏 3. Code Standards

- Use descriptive method names: detectFaces(), markAttendance().
- Follow Java naming conventions and structure.
- Add Javadoc comments for all public methods.
- Use consistent indentation and spacing.
- Wrap complex code in comments for clarity.

🧱 4. Modularity

Modules:

- **Image Capture** → FaceDetector.java
- **Training & Recognition** → FaceRecognizer.java
- **Motion Detection** → MotionDetector.java
- **OCR** → OCR.java
- **Database Access** → Database.java
- **UI Logic** → SampleController.java

Each file serves one major purpose—good separation of concerns.

🔧 5. Functions and Classes

Every function should include a **Javadoc docstring**, e.g.:

```
/**
 * Trains the face recognition model using LBPH algorithm.
 * Loads images from the dataset and stores a trained model.
 */
```

public void trainModel() { ... }

⚠️ 6. Error Handling

- Use try-catch for:
  - File access
  - Camera/device issues
  - XML or model loading
- Log errors clearly using System.err.println() or Java Logger.

🧩 7. GUI Standards

- Built with JavaFX FXML layout.
- All inputs (buttons, labels, etc.) are in Sample.fxml.
- GUI assets are grouped in src/.

📄 8. Data Handling

- Uses Haar cascades for detection (src/application/haar/*.xml).
- Face images and possibly logs or output models are handled via file system paths.
- Make all paths platform-independent using Paths.get(...) or File.separator.

🔐 9. Security Considerations

- Avoid hardcoded file paths.
- Sanitize all user inputs before database or file operations.
- Ensure OpenCV loads cascade files safely with try-catch.

📚 10. Documentation Files

- README.md should include:
  - Project description
  - Features (face detection, OCR, motion tracking)
  - Setup instructions (Java, OpenCV)
  - Example output or screenshots

- Add a **Code Documentation Report**:
  - Flowchart or UML diagram of main modules
  - Sample input/output:
    Input: Live webcam feed
    Output: Console log + GUI recognition + DB entry (if implemented)