**PROJECT X - AUTOMATED ATTENDANCE APP**

**Use Case Diagrams**

Group Name:

March 14, 2025

**TRIPLE THREAT (+1)**

Submitted by:

**John Kenneth Gade**

**Cyril Lloyd Balanay**

**Ivan Vasay**

**Keynt Harly Adol**

Submitted to:

**Terry Watts**

**USE CASE DIAGRAMS**

1. **Registration and Login**

A diagram of a user authentication system

AI-generated content may be incorrect.

* 1. **Actors:** Lecturer, Student
  2. **Preconditions:** User is not logged in.
  3. **Scenario:** 
     1. User opens the app and selects "Register" or "Login."
     2. If registering, the user inputs details (name, email, password, and role).
     3. The system validates the information and creates an account.
     4. If logging in, the user enters credentials.
     5. The system verifies the credentials and grants access.
  4. **Postconditions**: User gains access to the system.

1. **Class Management**

A diagram of a class management system

AI-generated content may be incorrect.

* 1. **Actors:** Lecturer
  2. **Preconditions:** Lecturer is logged in.
  3. **Scenario:** 
     1. Lecturer navigates to the "Manage Classes" section.
     2. Lecturer creates a new class, assigns subjects, and adds students manually or via a class code.
     3. Lecturer edits or deletes classes as needed.
  4. **Postconditions:** Classes are created, updated, or removed.

1. **Student Joins a Class**

A diagram of a class program

AI-generated content may be incorrect.

* 1. **Actors**: Student
  2. **Preconditions:** Student is logged in.
  3. **Scenario:** 
     1. Student enters a class code provided by the lecturer.
     2. The system validates the code.
     3. The student is enrolled in the class.
  4. **Postconditions**: Student is successfully added to the class.

1. **QR Code Attendance Scanning**

A diagram of a qr code attendance system

AI-generated content may be incorrect.

* 1. **Actors:** Lecturer, Student
  2. **Preconditions:** Lecturer is logged in, students are enrolled.
  3. **Scenario:** 
     1. Lecturer generates a QR code for the session.
     2. Students scan the QR code.
     3. The system verifies the scan with time and GPS location.
     4. Attendance is recorded.
  4. **Postconditions:** Attendance is updated in the system.

1. **Attendance Marking via Geofencing**

A diagram of a geofencing system

AI-generated content may be incorrect.

* 1. **Actors:** Student, System
  2. **Preconditions:** Student is within the predefined geofenced area.
  3. **Scenario:** 
     1. Student enters the classroom.
     2. The system detects their location.
     3. The system automatically marks attendance.
     4. Student receives a confirmation notification.
  4. **Postconditions:** Attendance is recorded without manual interaction.

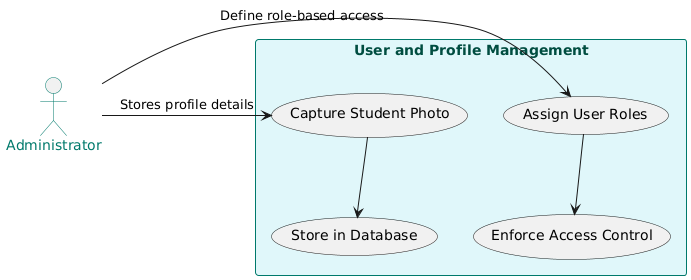
1. **Generate Reports**

A diagram of attendance reporting system

AI-generated content may be incorrect.

* 1. **Actors:** Lecturer
  2. **Preconditions:** Attendance records exist.
  3. **Scenario:** 
     1. Lecturer navigates to the attendance reports section.
     2. Lecturer selects a class and date range.
     3. The system generates the report.
     4. Lecturer exports the report in PDF or Excel format.
  4. **Postconditions:** Lecturer obtains attendance records.

1. **User and Profile Management**

****

* 1. **Actors:** Administrator
  2. **Preconditions:**
     1. The Administrator is logged into the system.
     2. The system database is active and accessible.
     3. The device used to capture student photos is registered and authorized.
     4. Users (students or lecturers) must have valid IDs and enrollment details for profile creation.
  3. **Scenario:** 
     1. Admin captures student photos and stores profile details.
     2. Admin assigns roles (Administrator, Instructor, Student) with specific access permissions.
  4. **Postconditions:** 
     1. Student profiles with photos are stored.
     2. Role-based access control is applied.

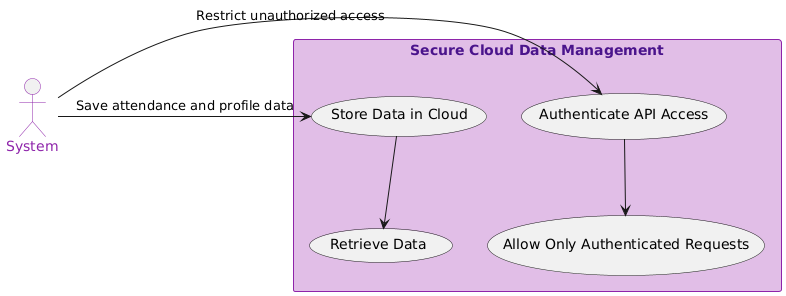
1. **Device Management**

**A diagram of a device management

AI-generated content may be incorrect.**

* 1. **Actors: Administrator, Instructor**
  2. **Preconditions:**
     1. The **Administrator or Instructor** is logged in.
     2. The device attempting to access the system must be **registered** and authorized.
     3. The **GPS feature** must be enabled on the device for location tracking.
     4. The system must have access to **location services**.
     5. Device tracking requires **permissions** to access location history..
  3. **Scenario:** 
     1. System links devices to user accounts during registration.
     2. Only authorized devices can access attendance functions.
     3. In case of a lost device, the system tracks and displays the last known GPS location.
  4. **Postconditions:** 
     1. Only authorized devices can access the system.
     2. Lost devices can be located.

1. **Cloud Data Management**

****

* 1. **Actors:** System
  2. **Preconditions:**
     1. The cloud server must be active and accessible.
     2. The system’s API authentication service must be operational.
     3. Only authenticated users can initiate cloud data operations.
     4. Internet connectivity is required for data retrieval and storage.
  3. **Scenario:** 
     1. The system stores all data in the cloud.
     2. Only authenticated API requests can access or modify the data.
  4. **Postcondition:** Data is securely stored and accessible only to authenticated users.

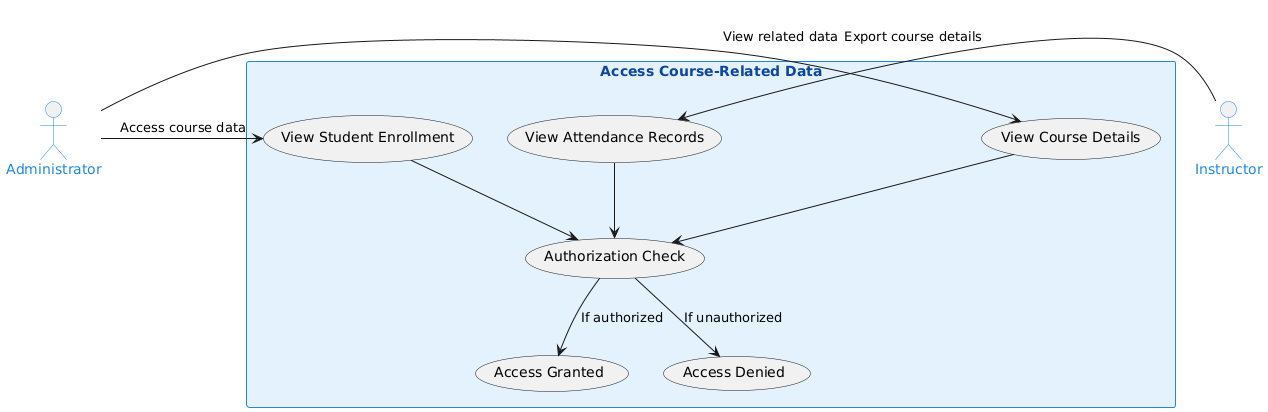
1. **Attendance View-Only Access**

**A diagram of a student

AI-generated content may be incorrect.**

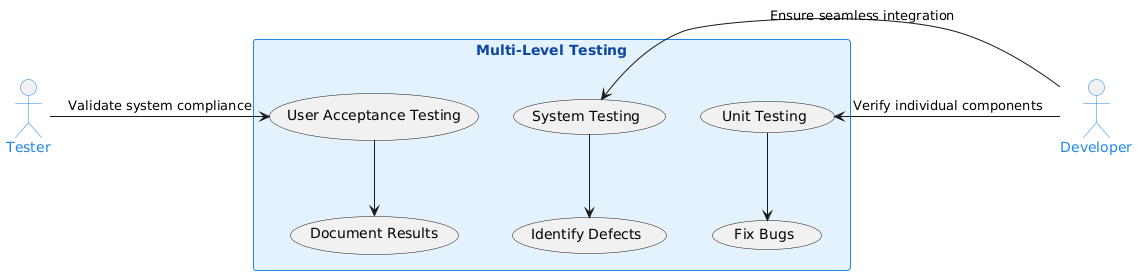
* 1. **Actors:** Student
  2. **Preconditions:**
     1. The Student must be logged into the system.
     2. The student must be enrolled in at least one class.
     3. The attendance records must already exist in the database.
     4. The view-only restriction policy is applied, preventing data modification.
  3. **Scenario:** 
     1. Student logs in and views attendance records.
     2. System restricts modification access.
  4. **Postcondition:** Students can only view their attendance​.

1. **Access Course-Related Data**

****

* 1. **Actors:** Administrator, Instructor
  2. **Preconditions:** 
     1. The user is logged in with appropriate permissions.
     2. The authorization system is active and enforces access control.
     3. Course-related data (student enrollment, attendance records, course details) exists in the database.
  3. **Scenario:** 
     1. The user selects the "Course Data" section.
     2. The system verifies the user’s authorization level.
     3. If authorized, the system grants access to course-related data:
     4. Student enrollment information.
     5. Attendance records.
     6. Course details.
     7. The user can view or export the data (if permitted).
  4. **Postconditions:** 
     1. Authorized users can access and review course-related data.
     2. Unauthorized users are denied access.

1. **Multi-Level Testing**

****

* 1. **Description:** This use case covers User Acceptance Testing (UAT), System Testing, and Unit Testing.
  2. **Actors:** Tester, Developer
  3. **Preconditions:** 
     1. The system is ready for testing.
     2. Test plans and test cases are prepared.
  4. **Scenario:** 
     1. The tester initiates User Acceptance Testing (UAT) to validate the system against client expectations.
     2. The developer performs System Testing to verify integration of components.
     3. Unit Testing is conducted on individual components.
  5. **Postconditions:** 
     1. Test results are documented.
     2. Defects are identified and reported.

1. **Documentation and Version Control**

**A diagram of a system

AI-generated content may be incorrect.**

* 1. **Description:** This use case covers the formal documentation and version management process using Git.
  2. **Actors:** Developer, Administrator
  3. **Preconditions:** 
     1. The system code and documentation are ready for versioning.
     2. Git repositories are accessible.
  4. **Scenario:** 
     1. Developer creates formal documentation, including:
        1. System Requirements
        2. System Design
        3. User Acceptance Test (UAT)
     2. The documentation and code are committed to Git for versioning and safekeeping.
     3. Git repositories are shared with the Administrator.
  5. **Postconditions:** 
     1. The system documentation is versioned and accessible.
     2. Git repository access is granted.

1. **High-Level and Detailed Design Documentation**

**A diagram of a design documentation

AI-generated content may be incorrect.**

* 1. **Description:** This use case covers the creation of high-level diagrams (Use Case, UML, and Conceptual) and detailed design documentation.
  2. **Actors:** Developer, Designer
  3. **Preconditions:** 
     1. The system architecture and components are defined.
  4. **Scenario:** 
     1. Developer creates High-level Diagrams:
        1. Use Case Diagram.
        2. UML Diagram.
        3. Conceptual Diagram.
     2. Designer creates Detailed Design Documentation:
        1. Component Diagram.
        2. Class Diagram.
        3. Activity and Sequence Diagrams.
     3. The documentation is reviewed and approved.
  5. **Postconditions:** 
     1. The design documentation is stored in the system repository.
     2. Review feedback is applied.

1. **Implementation and Code Reviews**

**A diagram of a product

AI-generated content may be incorrect.**

* 1. **Description:** This use case covers the code implementation, code reviews, and team standups.
  2. **Actors:** Developer, Reviewer, Team
  3. **Preconditions:** 
     1. The development phase is ongoing.
     2. Code is ready for review.
  4. **Scenario:** 
     1. The developer writes well-structured, documented code.
     2. The team conducts daily standups to track progress:
        1. What was done yesterday.
        2. What is being done today.
        3. Any impediments.
     3. The code undergoes review:
        1. Developer presents the code.
        2. Reviewers provide feedback.
        3. The final working code is demonstrated.
  5. **Postconditions:** 
     1. Code is reviewed, refined, and stored in the repository.
     2. Team progress is tracked.