**PROJECT X: REQUIREMENTS**

**Lingo, Ivan Kyle L.**

**Ventura, Ella Mae B.**

**Matinagnos, Nikki C.**

**High-Level Design (HLD) Document Template**

**1. Project Overview**

* **Project Name: PROJECT X: Automated Attendance System**
* **Prepared By: Team Introverts**
* **Date:**
* **Version:**
* **Purpose:**  
  Brief description of the system and its goals.

**2. Scope**

* Features covered in this design
* Out of scope items (if any)

**3. Architecture Diagram**

* Visual diagram showing major system components and how they interact.
* Can include front-end, back-end, APIs, databases, external services, etc.

*(Use tools like Lucidchart, Draw.io, or Figma for diagrams.)*

**4. Technology Stack**

| **Layer** | **Technology Used** |
| --- | --- |
| Front-End | (e.g., React, Angular) |
| Back-End | (e.g., Node.js, Django) |
| Database | (e.g., PostgreSQL, MongoDB) |
| API Protocol | (e.g., REST, GraphQL) |
| Authentication | (e.g., OAuth2, JWT) |
| Hosting/Infra | (e.g., AWS, Firebase) |

**5. Module Descriptions**

Describe major components/modules of the system:

**5.1 Front-End**

* Features
* User roles and views

**5.2 Back-End**

* Services
* Business logic
* Security and validation

**5.3 Database**

* Schema overview (ER diagram or table list)
* Key relationships

**5.4 APIs**

* REST endpoints or GraphQL queries/mutations
* Example requests/responses

**6. Data Flow Diagram (DFD)**

* Describe how data flows between components.
* Can include Level 0 (context), Level 1 diagrams.

**7. Integration Points**

* Third-party services (e.g., payment gateway, analytics)
* Internal/external APIs
* Webhooks or data pipelines

**8. Security Considerations**

* Authentication & Authorization
* Encryption (data at rest and in transit)
* Secure coding practices

**9. Scalability and Performance**

* Load expectations
* Caching
* Asynchronous processing (queues, background jobs)

**10. Error Handling and Logging**

* Error tracking strategy
* Tools (e.g., Sentry, Logstash)
* Alerting and monitoring

**11. Deployment Strategy**

* Environments (dev, staging, prod)
* CI/CD pipeline
* Rollback strategy

**12. Appendices**

* Glossary
* References
* Links to documents or code repositories

**PROJECT OVERVIEW**

The Automated Attendance System is designed to automate the process of student attendance tracking, reporting, and verification. It ensures secure access to the system and validates user authenticity.

**SYSTEM REQUIREMENTS**

**R1. User Roles and Access**

R1.1. The system shall support three main roles: Lecturer, Student, and Administrator.

R1.2. Only registered lecturers shall record attendance.

R1.3. Attendance shall only be recorded using authorized devices.

R1.4. Administrators shall manage lecturers, students, courses, and attendance records

**R2. Attendance Recording & Data Storage**

R2.1. Lecturers shall record attendance via registered devices (mobile, tablet, or computer).

R2.2. Attendance records are stored in a cloud-based database.

R2.3. Each attendance entry is linked to a course and includes details such as student, lecturer, date, and time.

R2.4. Attendance records shall be accessed in real-time.

**R3. Device Registration & Tracking**

R3.1. Lecturers shall register multiple devices for attendance tracking.

R3.2. The system shall track device locations for assistance.

R3.3. Only registered devices shall interact with the attendance system.

**R4. Student Enrollment & Management**

R4.1. Students shall check their attendance status.

R4.2. The system stores student details, including name, university ID, and profile picture (captured via the system).

R4.3. Administrators can add, update, and delete student records.

**R5. Reporting & Data Access**

R5.1. Available reports:

R5.1.1. Attendance reports by student, course, or date range.

R5.1.2. Student enrollment per course.

R5.1.3. Lecturer and registered device lists.

R5.2. Reports are accessible to lecturers and administrators.

**R6. Photo Capture & Storage**

R6.1. Lecturers shall capture student photos for attendance verification.

R6.2. Photos are stored securely, with IDs and names recorded in the database.

**R7. System Architecture & API Access**

R7.1. The system operates via a REST API for database interactions.

R7.2. All actions, including attendance, enrollment, and reporting, are managed through the API.

R7.3. API requests are authenticated and authorized.

**R8. Testing Criteria**

R8.1. Unit Testing: Test each function (e.g., attendance logging, student enrollment).

R8.2. System Testing: Perform end-to-end testing to ensure proper integration.

R8.3. User Acceptance Testing (UAT): Test system compliance with user requirements.

**SYSTEM COMPONENTS**

**Frontend (Client-Side)**

A web and mobile app (built using React or Flutter) provides interfaces for lecturers, students, and administrators.

A camera module captures student photos.

**Backend (Server-Side)**

REST API (Node.js/Express): Handles requests from the frontend.

Authentication Module: Manages user logins.

Attendance Processing Module: Records attendance.

Report Generation Module: Creates attendance reports.

Device Tracking Module: Tracks registered devices.

**Database (Cloud-Based)**

User & Role Management: Stores student, lecturer, and admin details.

Course & Enrollment Data: Manages student-course enrollments.

Attendance Records: Maintains attendance logs.

Photo Storage: Saves student images for verification.

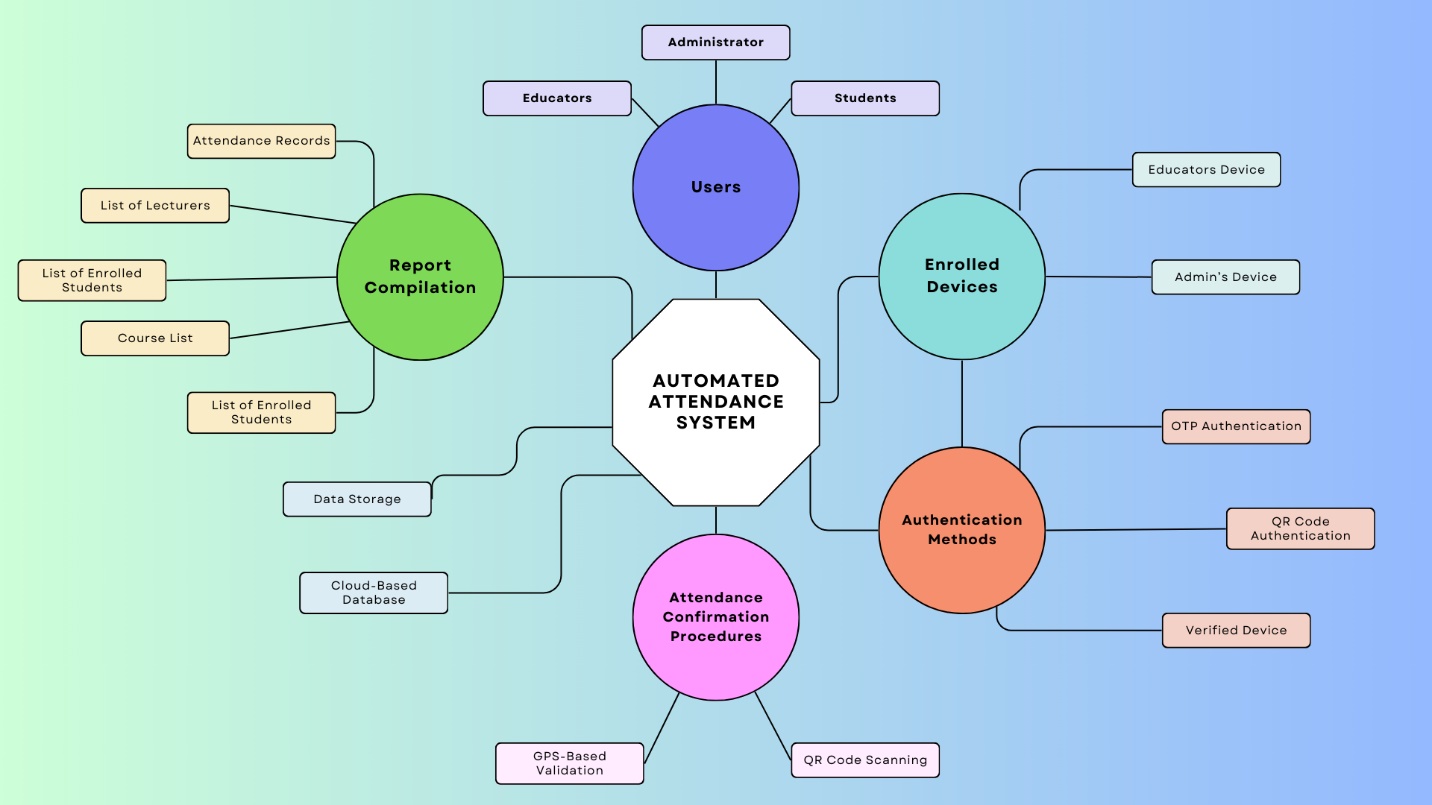
**External Services (Optional Enhancements)**

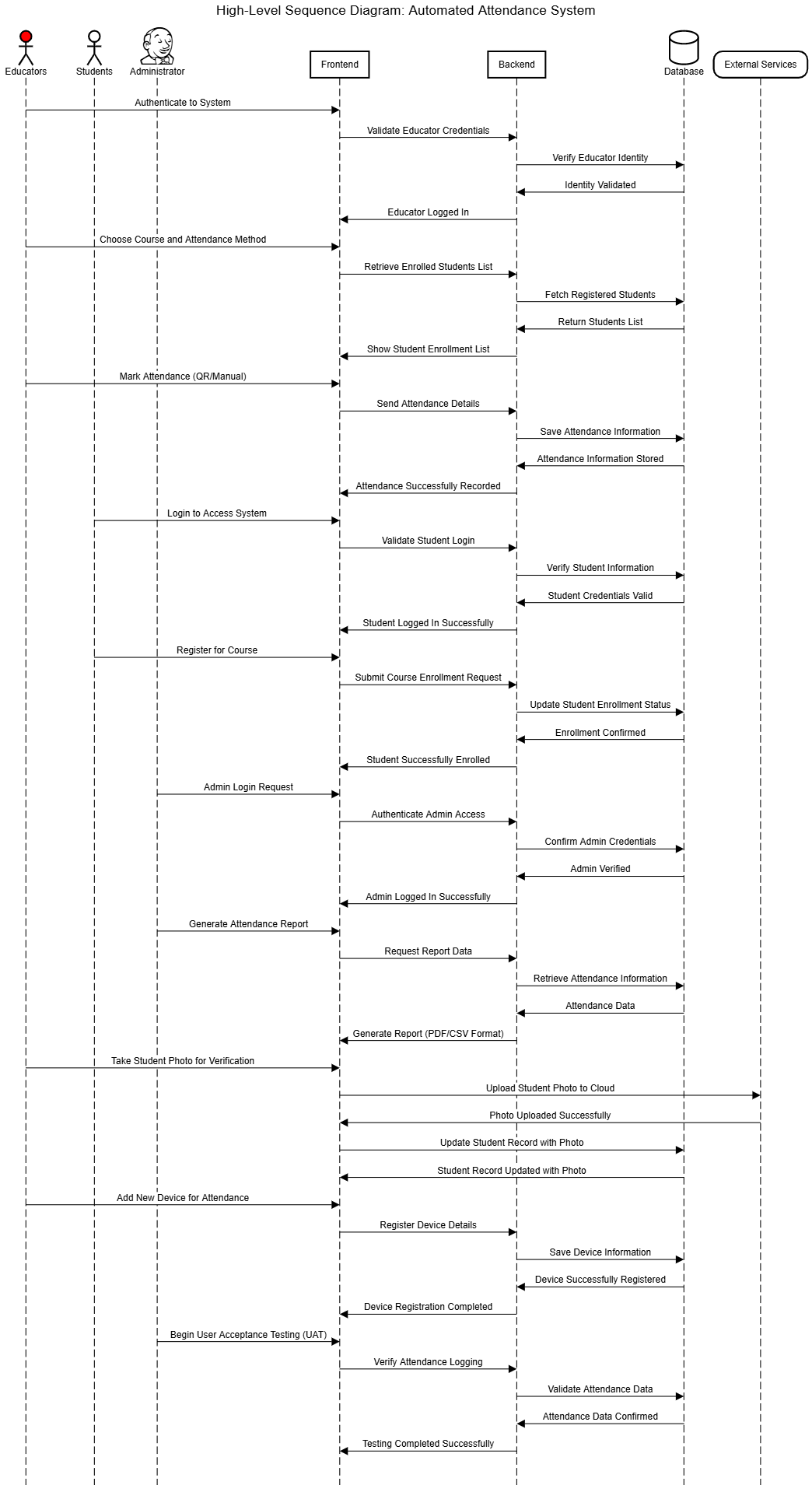
Cloud Storage (AWS S3, Firebase Storage): Stores student images.

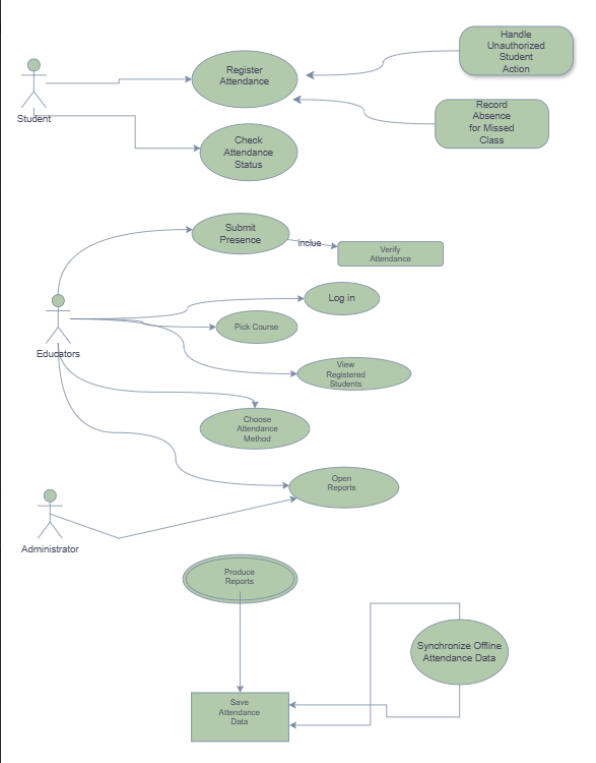
GPS Tracking API (Google Maps API): Locates lecturer devices.

Notification Services (Twilio, Firebase Cloud Messaging): Sends attendance reminders.

**CONCEPTUAL MODEL**



**HIGH-LEVEL SEQUENCE DIAGRAM**

**USE CASE DIAGRAM  
**

**1. Mark Student Attendance**

**Actors:**

Primary: Lecturer

Secondary: Student, Administrator

**Preconditions:**

Lecturer is logged in.

A class session is scheduled**.**

Students are registered.

**Main Flow:**

1. Lecturer logs in and selects a course.
2. The system displays enrolled students.
3. Lecturer selects an attendance method (QR code, biometric, manual).
4. Students mark their attendance.
5. The system validates student presence and updates records.
6. Lecturer submits attendance.
7. Data is stored in the database, and reports are generated.

**Alternate Flows:**

1. **Invalid Student Attempt:** Unauthorized students cannot mark attendance.
2. **Missed Attendance:** Students who do not mark attendance within the timeframe are marked absent.
3. **Offline Mode:** Attendance data is stored locally and synced later.

**Postconditions:**

1. Attendance is recorded in the database.
2. Lecturers and administrators can access reports.
3. Students can view their attendance status.

**2. Generate Attendance Report**

**Actors:**

1. **Primary:** Administrator
2. **Secondary:** Lecturer

**Preconditions:**

1. The user must be logged in.
2. Attendance records must exist.

**Main Flow:**

1. The user logs in and navigates to the Reports section.
2. Filters are applied (by course, date range, student, or lecturer).
3. The report is generated in table, PDF, or CSV format.
4. The user can download or print the report.

**Alternate Flows:**

1. **Invalid Date Range**: If no records exist for the selected range, the user is prompted to adjust filters.
2. **Export Options:** Reports can be exported in multiple formats.
3. **Access Restriction:** Lecturers can only generate reports for their assigned classes, while administrators have full access**.**

**Postconditions:**

1. Reports are generated and ready for download.
2. Users can use reports for record-keeping or analysis.