UNIVERSITY OF BUEA

P.O. Box 63.

Buea, South West Region

CAMEROON

Tel: (237)33322134/33322690

Fax: (273)33322272

REPUBLIC OF CAMEROON

Peace-Work-Fatherland

FACULTY OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER ENGINEERING

COURSE: INTERNET PROGRAMMING AND MOBILE PROGRAMMING. **CEF 440**

COURSE COORDINATOR: DR VALERY NKEMENI

TASK 4:

System Modeling and Design

GROUP 15 MEMBERS:

1- ELAINE CLINTON NTONGWE-FE22A198
2- ETUNDI ZAMBO JOSIANE-FE22A212
3- FAI NJI JR BAHTINYUY-FE22A215
4- NEBA NELLY ACHA-FE22A255
5- NGWA NATHAN NINJECK-FE22A268

Second semester Task 4 report, (B.ENG) Degree in Computer Engineering.

MAY
2025

Contents

Introduction

The system modelling and design phase plays a crucial role in the successful development of the Attendance Management App. This phase focuses on transforming the functional and non-functional requirements into a structured blueprint that guides implementation. By leveraging system modelling tools and methodologies, we can visualize and analyze the structure, behavior, and interactions within the system to ensure scalability, reliability, and maintainability.

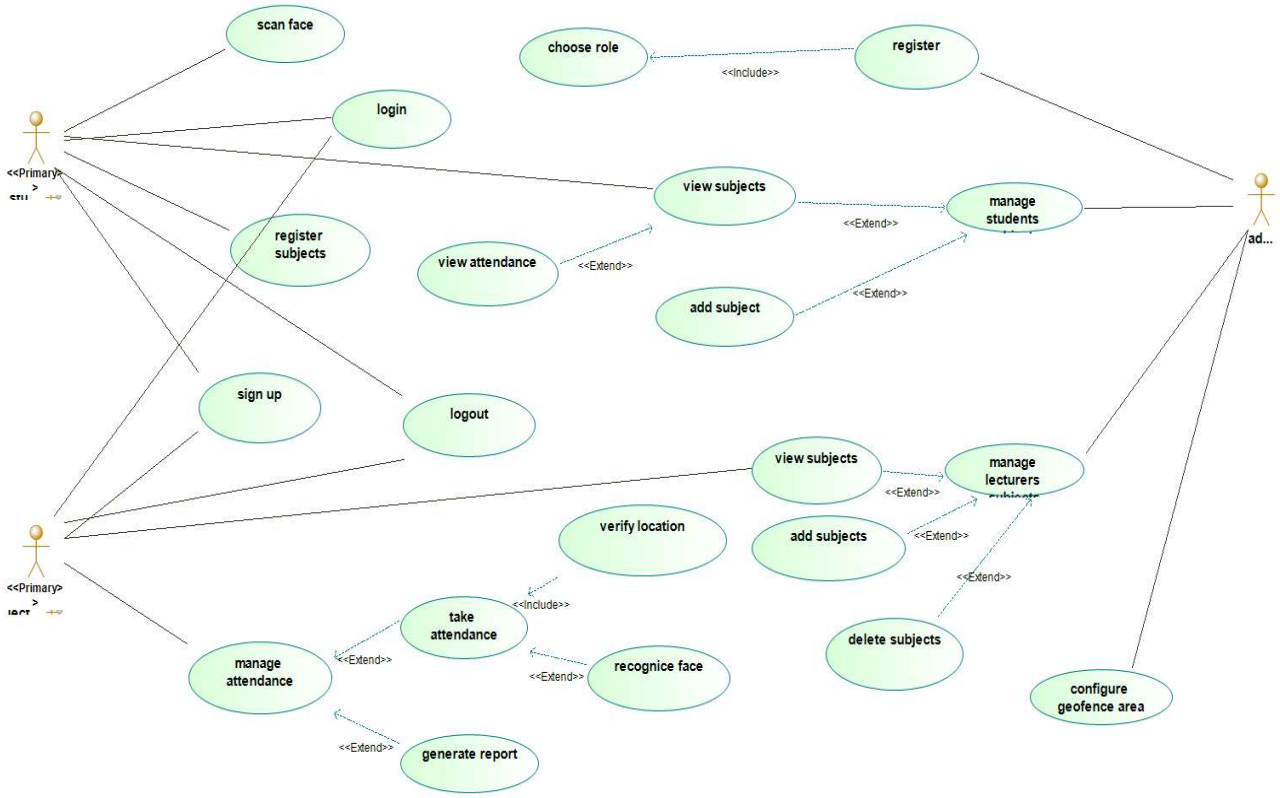
The Attendance Management App aims to automate and secure the process of recording student or staff attendance by integrating modern technologies such as **geofencing** for location validation and **facial recognition** for identity verification. This phase utilizes a variety of UML diagrams—including **context diagrams**, **data flow diagrams**, **use case diagrams**, **sequence diagrams**, **class diagrams**, and **deployment diagrams**—to illustrate the system's components and their interactions comprehensively.

Each model provides a different perspective: the context and DFDs offer a high-level overview, the use case and sequence diagrams capture system behavior and user interaction, the class diagram defines the system's structure, and the deployment diagram outlines its physical architecture. Together, these models form a unified vision for system development.

1. Use Case Diagram

it is a behavioral diagram and is the primary form of system/software requirements for a new software program underdeveloped. Use cases specify the expected behavior (what), and not the exact method of making it happen (how).

- Purpose: Identifies user interactions with the system.
- Use Cases in the Diagram:
- Sign Up: Users register with the system.
- Login: Users authenticate to gain access.
- Mark Attendance: It is taken, once inside the geofence and verified by facial recognition.
- View Attendance: Users can check past attendance logs.
- Admin Control: Admins manage data and generate reports.
- Actors: Student, Lecturers, Admin



2. Class Diagram

It is a structural diagram which shows a set of classes and A set of relationships between classes

Purpose: Describes the system's classes, their attributes, methods, and relationships.

- **Classes (from diagram):**

a. User

- **Attributes**:

userId: Unique identifier for each user.

name: Full name of the user.

email: Contact email.

password: For authentication.

role: Either 'Student' or 'Admin'.

- **Methods**:

login(): Authenticates the user using credentials.

register(): Registers the user into the system.

- **Multiplicity**:

A single User can have **many** attendance records.

Denoted as 1 (User) to 0..* (Attendance).

b. Attendance

- **Attributes** :

attendanceId: Unique record ID.

timestamp: Date and time of marking attendance.

status: Present/Absent.

locationVerified: Boolean for geofencing check.

faceVerified: Boolean for facial recognition check.

- **Methods**:

markAttendance(): Used to log the user's attendance.

- **Multiplicity:** (many Attendance) to 1 (User).

Meaning: Each user can mark multiple attendances, but each attendance belongs to one user.

c. Admin

- **Inheritance**: Inherits from User.
- **Methods**:

viewAllRecords(): Can view all attendance data.

manageUsers(): Can add, update, or delete users.

- **Notes**:

Admin is a specialized type of User, so it inherits all User attributes and methods.

d. LocationService

- **Attributes**:

currentLatitude, currentLongitude

- **Methods**:

isWithinGeofence(): Checks if the user is within the predefined campus boundary.

e. FaceService

- **Attributes**:

capturedImage: The real-time image captured from the device.

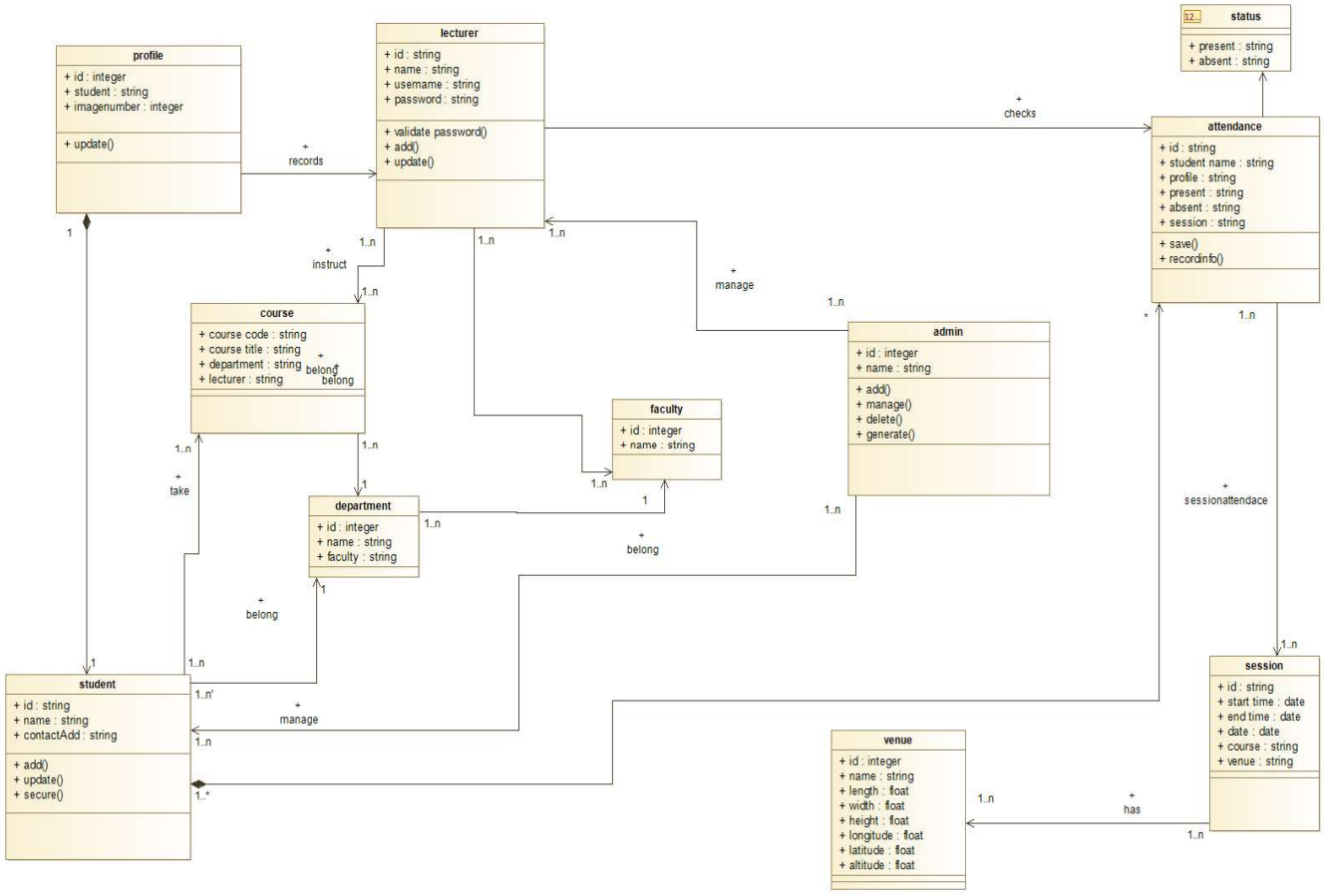
- **Methods**:

verifyFace(): Compares the captured face with the one stored in the system.

- **Relationships**:

- There is an association between the user and attendance, in which a user can mark multiple attendance records. Each attendance is associated with one user.
- There is inheritance(generalization) between the admin and user in which the Admin is a specialized type of User, inheriting all attributes and methods.
- Attendance uses LocationService to verify if the user is within the allowed geofence. And this shows an association
- Attendance uses FaceService to verify the user's identity via facial recognition. Which is also an association

- Admin can manage multiple users (e.g., add, delete, update). This is often implemented via a control or dependency relationship. And this shows an association



3. Sequence Diagrams

It is a diagram that visually represent the interactions between objects or components in a system over time. They focus on the order and timing of messages or events exchanged between different system elements.

- Purpose: Show the flow of messages between system components for specific scenarios.

a. Sign Up Sequence

- Actors: user, App, Database
- Steps:

- **User Enters Registration Details**:

  The user fills out the registration form with required information such as name, email, password, and role.
- **UI Sends Data to System**:

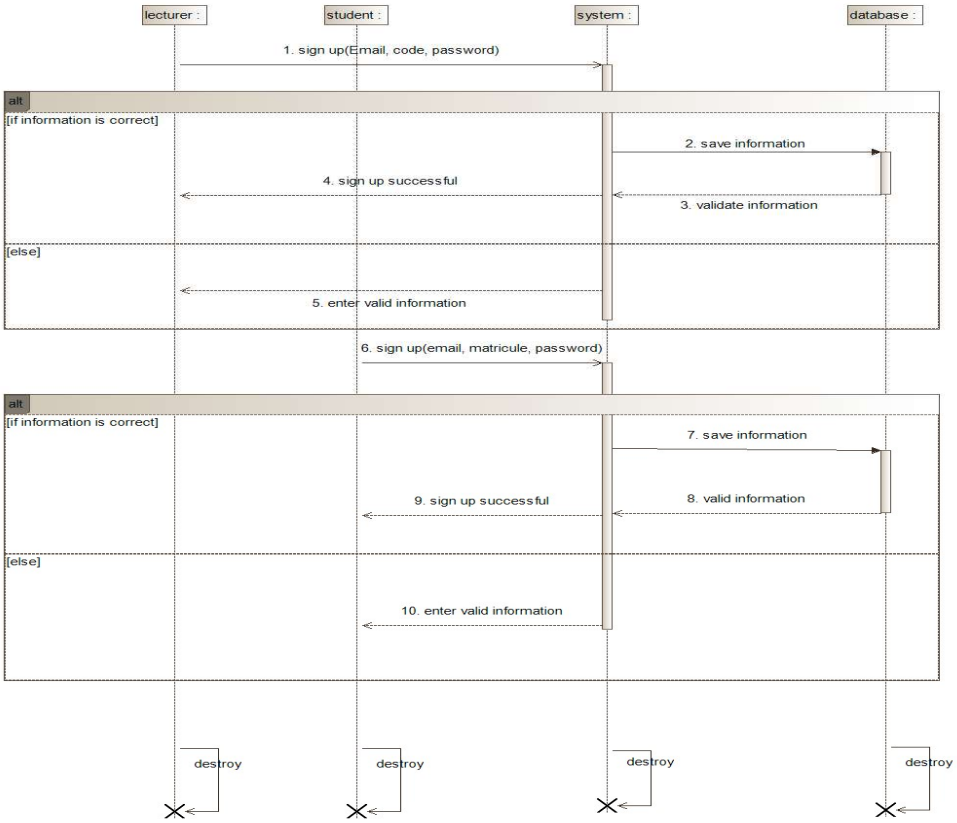
  The UI sends this data to the system controller or backend for processing.
- **System Validates Input**:

  The system checks if the input values are valid (e.g., email format, password strength).
- **System Checks for Existing Account**:

  It queries the database to verify if a user with the same email already exists.
- **Store New User in Database**:

  If no duplicate is found, the system saves the new user data to the database.
- **Return Confirmation to UI**:

  A success response is sent back to the UI, confirming the account creation.



b. Login Sequence

- Actors: User, App, Database
- Steps:

- **User Enters Login Credentials**:

  The user inputs their email and password in the login form.

- **UI Sends Login Request to System**:

  These credentials are passed to the system controller.

- **System Validates Input Format**:

  Checks if the fields are not empty and in valid format.

- **System Queries Database**:

  It checks the database to match the credentials against stored user records.

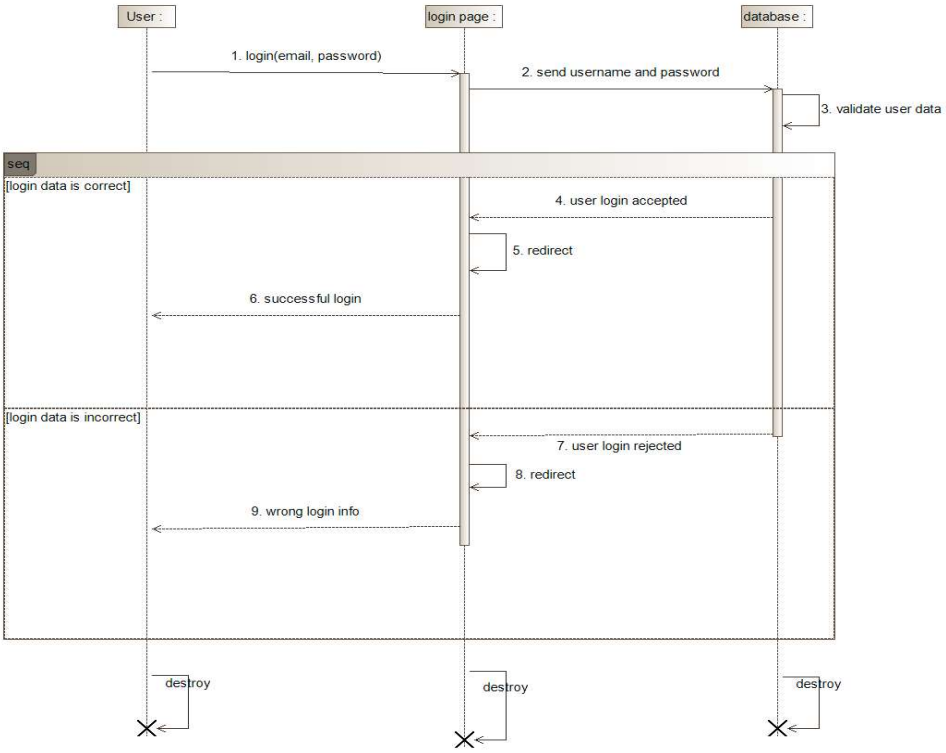
- **Authentication Result**:

  If the credentials match: The system grants access and generates a session/token.

  Otherwise, it sends an authentication failure message.

- **UI Displays Result**:

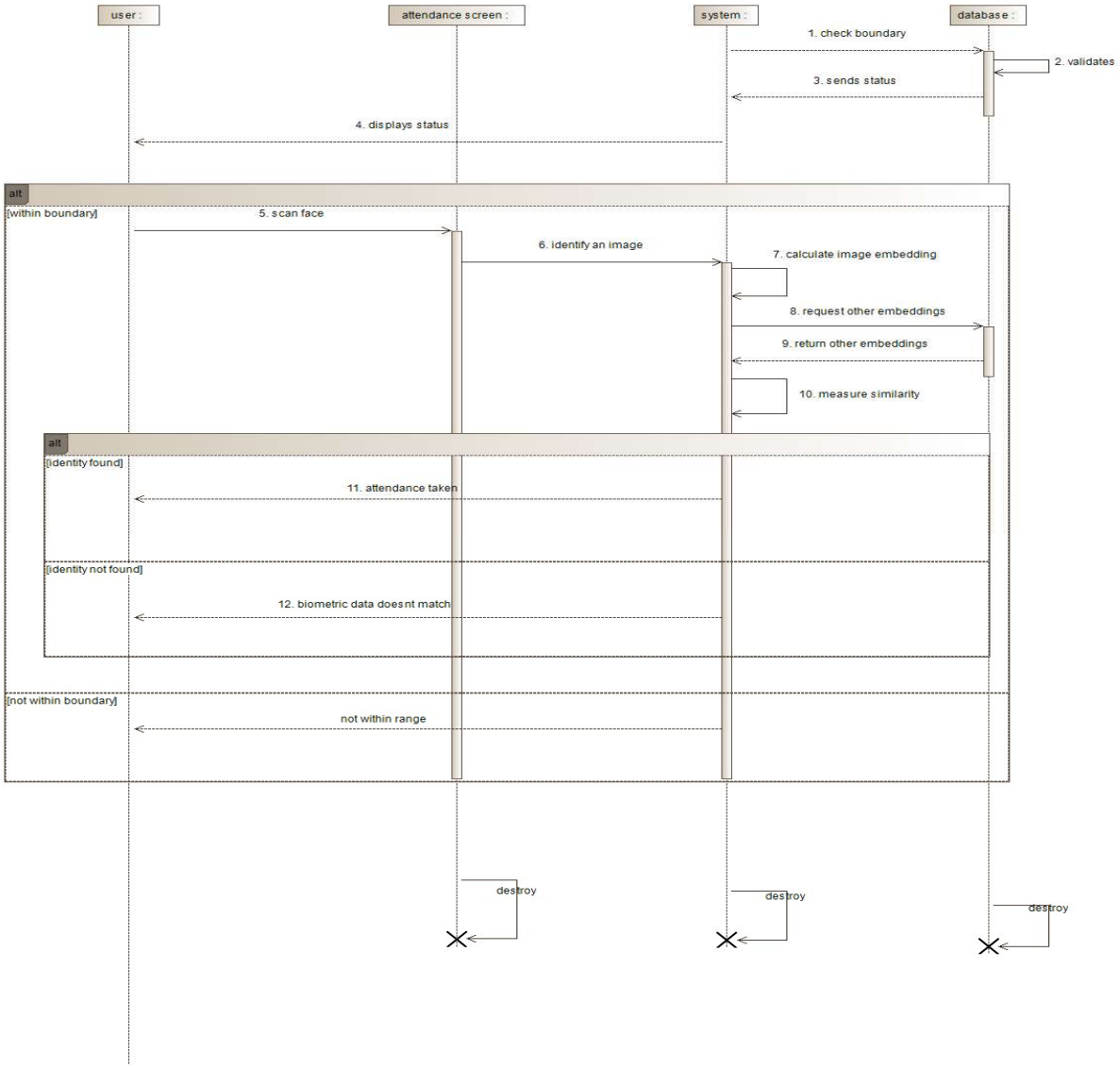
  Either redirects the user to the dashboard or shows an error message.



c. Mark Attendance Sequence

- Actors: Student, App, Geofence API, Face Recognition API, Database
- Steps:

- **User Initiates Attendance Marking**:
  User selects the option to mark attendance in the app.
- **System Requests Current Location**:
  The UI captures the GPS location and sends it to the system.
- **Geolocation Service Validates Location**:
  The system checks whether the user is within the geofenced area (e.g., campus).
- **System Captures Face Image**:
  Using the device camera, a face image is captured and sent to the facial recognition service.
- **Facial Recognition Service Verifies Identity**:
  Compares the captured face with stored facial data.
- **System Combines Results**: If both **location** and **face** are verified successfully, the system proceeds. Otherwise, it notifies the user of the failure.
- **Log Attendance in Database**:
  The system saves the attendance record with timestamp, location status, and face verification result.
- **Notify User**:
  The UI displays success or failure feedback to the user.



4. Context Diagram

It is a high-level visual representation that show the interactions between a system being developed and its external entities, such as users, other systems, or processes.

- Purpose: Illustrates the external entities interacting with the system and the main data flows.
- Key Entities:

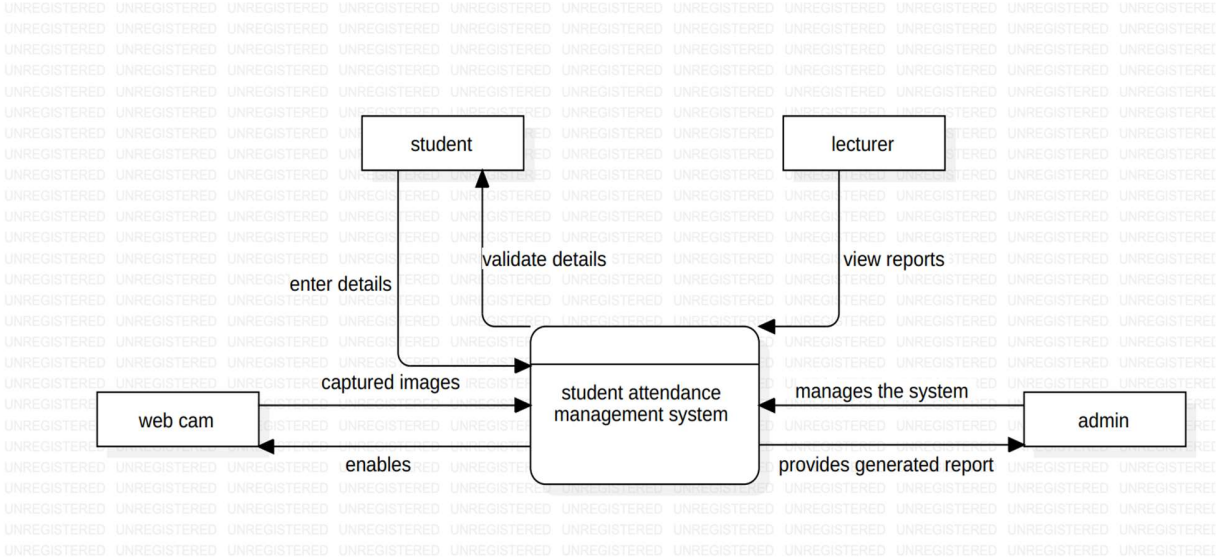
- User (Student/Staff): Interacts with the app to sign up, log in, and mark attendance.

- Admin: Oversees attendance records and manages users.
- Database: Stores user data and attendance logs.
- External APIs: Support for location and facial recognition.

- **Diagram Summary:**

The context diagram from the document shows:

- Users communicating with the system via mobile devices.
- The app accessing geolocation and facial recognition APIs.
- Centralized data storage for attendance and user management.



5. Data Flow Diagram (DFD)

It is a graphical representation of data flow in any system. It is capable of illustrating incoming data flow, outgoing data flow and store data.

- Purpose: Shows how data moves through the system and the processes that handle it.
- **External Entities:**
- **Student**

  Signs up for registration.

  Triggers image acquisition during registration.
- **Admin**

Manages the training set (face images).

Oversees registration, image processing, and reports.

- **Lecturer**

  Views attendance and generated reports.

- **Processes:**

1.0 – Student Registration

A student provides email and other details to register.

Invalid details are flagged and notified.

Valid registration triggers image acquisition.

Student details are saved to the **attendance database**.

2.0 – Image Acquisition

Uses a **webcam** to capture image frames.

Acquired frames are displayed for the user.

Sends frames to the **Face Detection** process.

3.0 – Face Detection

Detects face(s) from the captured frame.

Stores detected faces into a **training set** for recognition.

Sends a test image to the next stage: feature extraction.

4.0 – Feature Extraction

Extracts unique facial features from the test image.

These features are passed to the **Face Recognition** process.

5.0 – Face Recognition

Uses extracted features and training set to recognize the student.

Matches the face and marks attendance in the **attendance database**.

Provides student details for verification.

Passes data to admin for report generation.

6.0 – Admin Activity

Admin can manage the training set and system.

Generates attendance reports using data from the database.

Reports are provided to both **Admin** and **Lecturer**.

- **Data Stores:**

Training Set

Stores the face images captured during registration for use in recognition.

Attendance Database

Stores student registration details and attendance records.

Is accessed by feature extraction, face recognition, admin, and lecturer.

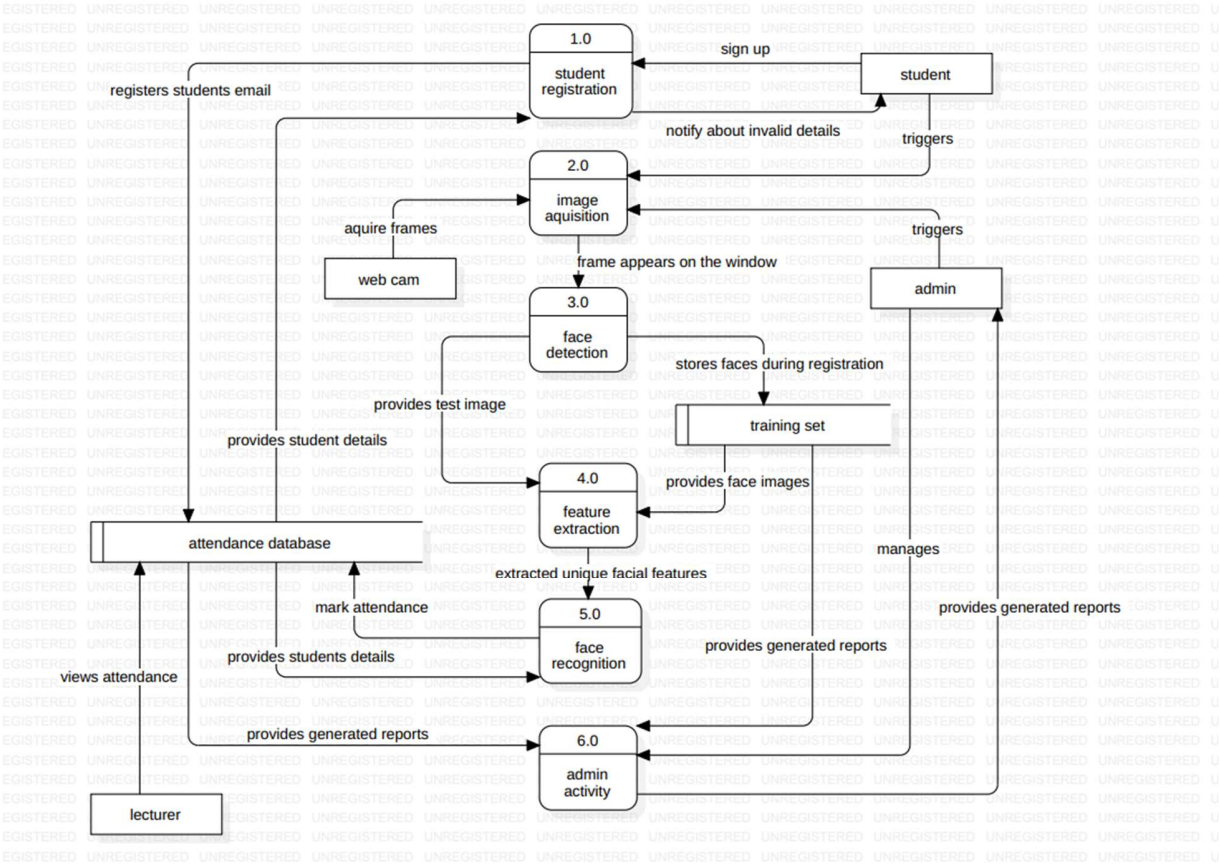
- **Flow Summary:**

Student registers → Face is captured and stored.

System detects and processes face images → Features extracted.

Recognition is performed using training set → Attendance is marked.

Admin and lecturer can access reports and data.



6. Deployment diagram

It is a diagram which shows how the software design turns into the actual physical system where the software will run. They show where software components are placed on hardware devices and shows how they connect with each other.

The deployment diagram below focuses on the core components and their relationships. This includes:

Key Components:

User Layer

Mobile App: For students and faculty to check in/out

Web Dashboard: For administrators and instructors to manage attendance

Server Layer

Load Balancer: Distributes incoming traffic

API Server: Handles all business logic and requests

Auth Service: Manages user authentication and authorization

Face Recognition: Processes facial recognition using ML

Geofencing: Handles location-based attendance validation

Data Layer

Database: Stores user accounts and attendance records

File Storage: Stores face images and generated reports

Redis Cache: Handles session data and quick lookups

External Services

Cloud ML API: Provides facial recognition capabilities

Maps API: Validates GPS locations and geofencing

Notifications: Sends SMS/email alerts and confirmations

This diagram maintains all the essential functionality while being much easier to understand and implement. It addresses the key requirements from our survey:

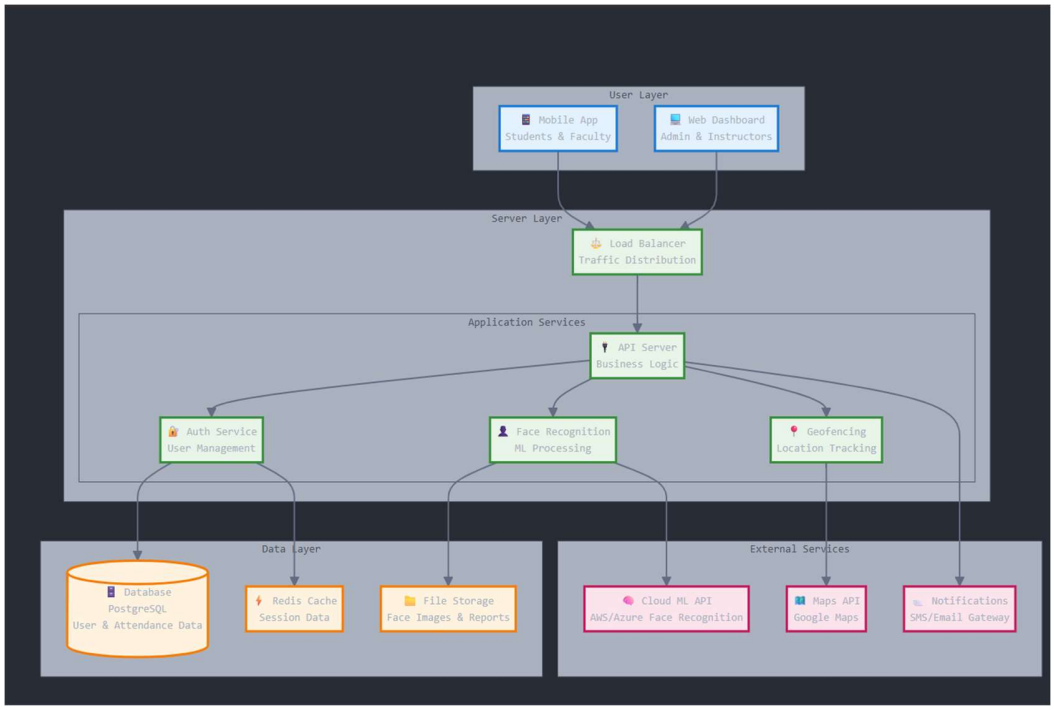
Efficiency: Streamlined architecture for fast response times

Security: Dedicated authentication service

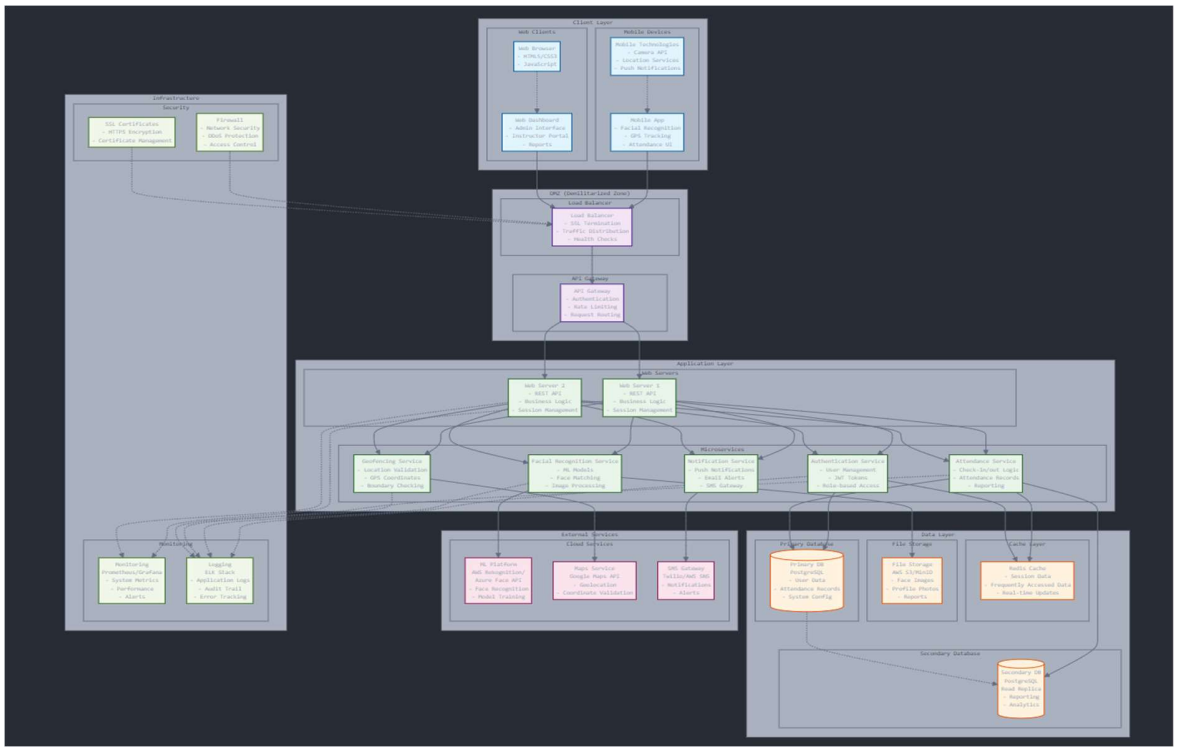
Privacy: Separate storage for sensitive biometric data

Reliability: Load balancing and caching for performance

The diagram is perfect for presentations to stakeholders who need to understand the system without getting overwhelmed by technical details.



The diagram which follows below shows a larger and more clear definition of all the layers in our deployment



Conclusion

The system modelling and design phase has provided a thorough foundation for the development of the Attendance Management App. Through detailed diagrams and analysis, we have captured the system's architecture, user interaction flows, data processes, and technical integrations. The incorporation of **geofencing** and **facial recognition** introduces both convenience and security, ensuring that attendance is accurate, location-verified, and identity-assured.

By clearly defining the system's components and their relationships, this phase reduces ambiguity and enhances communication among stakeholders, developers, and testers. It sets the stage for a smooth transition into the implementation phase, where these models will serve as blueprints for coding and system integration. Ultimately, this design phase ensures that the final application is user-friendly, secure, efficient, and aligned with its intended goals.