REPUBLIC OF CAMEROON
****
Peace-Work-Fatherland
****
MINISTER OF HIGHER EDUCATION
****
UNIVERSITY OF BUEA

REPUBLIQUE DU CAMEROON
****
PAIX-Travail-Patrie
****
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR
****
UNIVERSITE DE BUEA

FACULTY OF ENGINEERING AND TECHNOLOGY

COURSE TITLE:
INTERNET PROGRAMMING (J2EE) AND MOBILE PROGRAMMING
COURSE CODE:
CEF440

TASK 3 REPORT

System Modelling and Design

Course Intructor: Dr. Nkemeni Valery

GROUP IV

S/N	Names	Matricules
1	ARREY ABUNAW REGINA EBAI	FE22A142
2	AWA ERIC ANGELO JUNIOR	FE22A162
3	FAVOUR OZIOMA	FE22A217
4	OBI OBI ETCHU JUNIOR	FE22A291
5	VERBURINYUY JERVIS NYAH	FE22A324

Table of Content

Table of Content.....2
INTRODUCTION.....3
CONTEXT DIAGRAM.....4
USE CASE DIAGRAM.....7
CLASS DIAGRAM.....10
SEQUENCE DIAGRAM.....13
DATA FLOW DIAGRAM.....16
DEPLOYMENT DIAGRAM.....20
CONCLUSION.....24
List of figures.....26

INTRODUCTION

In today's digital-first academic environment, traditional methods of managing attendance are no longer efficient, secure, or scalable. Manual sign-ins and paper registers are prone to errors, proxy attendance, and administrative overhead. To address these challenges, we have designed and modeled a Mobile-Based Attendance Management System that leverages facial recognition and geofencing to deliver a secure, real-time, and user-friendly solution.

This report documents the system modeling phase of our mobile application project — a critical step that translates abstract ideas into structured, visual representations. By applying industry-standard modeling techniques and UML diagrams, we explored the system's architecture, behavior, data flow, and deployment strategy in a clear, comprehensive manner.

Our objective is to create a system that not only automates attendance tracking but also ensures accuracy, accountability, and transparency for students, instructors, and administrators alike. The modeling artifacts presented in this report form the blueprint for a robust application that is ready to scale, adapt, and meet real-world needs in educational institutions.

Through this report, we aim to demonstrate how thoughtful system modeling fosters clarity, reduces development risks, and lays a strong foundation for building intelligent mobile solutions.

I - CONTEXT DIAGRAM

Definition

A Context Diagram (also known as a Level 0 Data Flow Diagram) is a high-level representation of the entire system as a single process, showing its interaction with external entities. It helps stakeholders visualize system boundaries and understand how data flows in and out of the system without revealing internal processes or data stores.

Purpose

- To provide a simple overview of how the Mobile-Based Attendance Management System interacts with the external environment.
- To help stakeholders understand system boundaries and external dependencies at a glance.
- To serve as a foundation for further system design and requirement analysis.

Key Elements

Elements	Description
System	Represented as a single process named "Attendance System"
External entities	People or systems that interact with the Attendance System.
Dataflows	Arrows indicating the data exchanged between the system and external entities.
No data stores	Context diagrams do not show internal data stores or sub-processes.

Description of External Entities and Interactions

❖ Students

Send face images and GPS location via the mobile app.

Receive confirmation of attendance status (marked or rejected).

❖ GPS Service

Receives location data from the system.

Returns geofence verification (in-bound or out-of-bound).

❖ Facial Recognition API

Receives face images from the system.

Returns identity verification (match or no match).

❖ Lecturers

Receive attendance reports and class schedules.

May review and manually override attendance records.

❖ Admins

Configure settings, manage users, upload student face data.

Define geofence parameters and review verification logs.

❖ Database

Stores and retrieves attendance records, student data, and schedules.

Supports report generation and system operations.

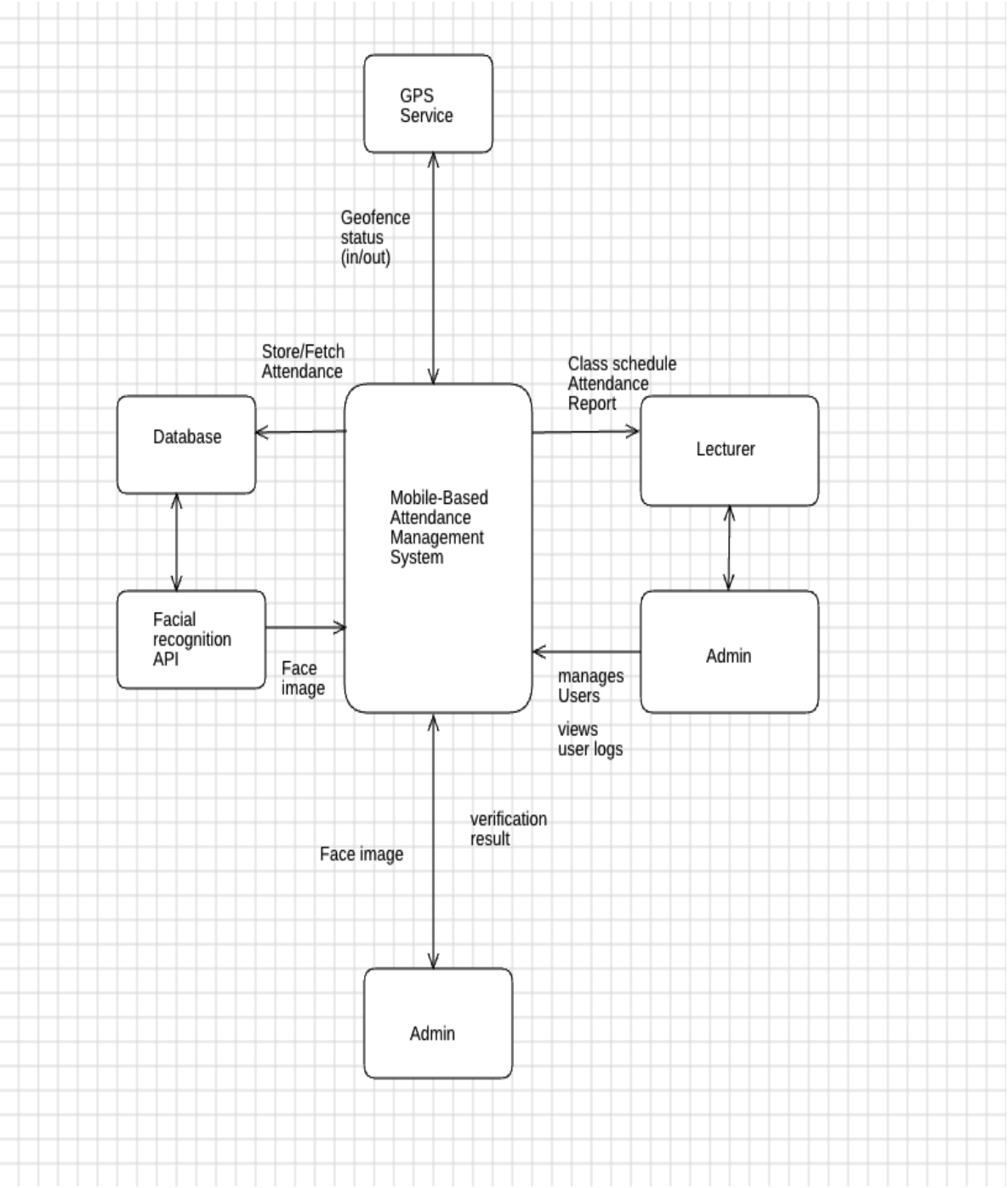

fig 1: Context diagram

The context diagram captures the essential interactions between the **Attendance System** and external entities. It illustrates how face images, GPS locations, verification statuses, and reports flow between users, APIs, and system components. This serves as a crucial blueprint for understanding the system's boundaries and external dependencies.

II - USE CASE DIAGRAM

Definition

A Use Case Diagram is a type of UML (Unified Modeling Language) diagram that visually represents:

- What the system does – its functionalities.
- Who interacts with the system – the actors (users or external systems).
- How they interact – through specific actions (use cases).

Purpose

- To capture the functional requirements of the system at a high level.
- To illustrate user interactions and system behavior.
- To help stakeholders understand user roles and responsibilities within the system.

System Overview

System Name: Attendance Management System

Primary Actors:

- Students
- Instructors
- Administrators

Auto-specific Use cases

❖ **Student Use cases**

Use case	Description
Sign up	Student registers by submitting personal info and facial biometrics. System saves the profile in the database.

Check-in	Student logs in (via face recognition or password). The system verifies their GPS location and face match, then records attendance with timestamp.
View Attendance	Student accesses their attendance history, which can be filtered by date or course.
Submit Permission request	Student sends a justification (e.g., medical note) for absence to the instructor for review.

❖ **Instructor Use cases**

Use case	Description
View Student Attendance	Instructor reviews real-time attendance data, including visual indicators (e.g., red for absences).
Manual Override	Instructor manually updates attendance records if face recognition or geofence validation fails.
Filter/Search	Instructor filters attendance data by student name, course, or date.

❖ **Administrator Use cases**

Use case	Description
User Management	Admin creates or removes user accounts and assigns roles (student/instructor).
Set Geofence Coordinates	Admin defines GPS boundaries for classrooms to validate check-ins.
Data Management	Admin manages data integrity, database backups, and compliance with privacy/security policies.

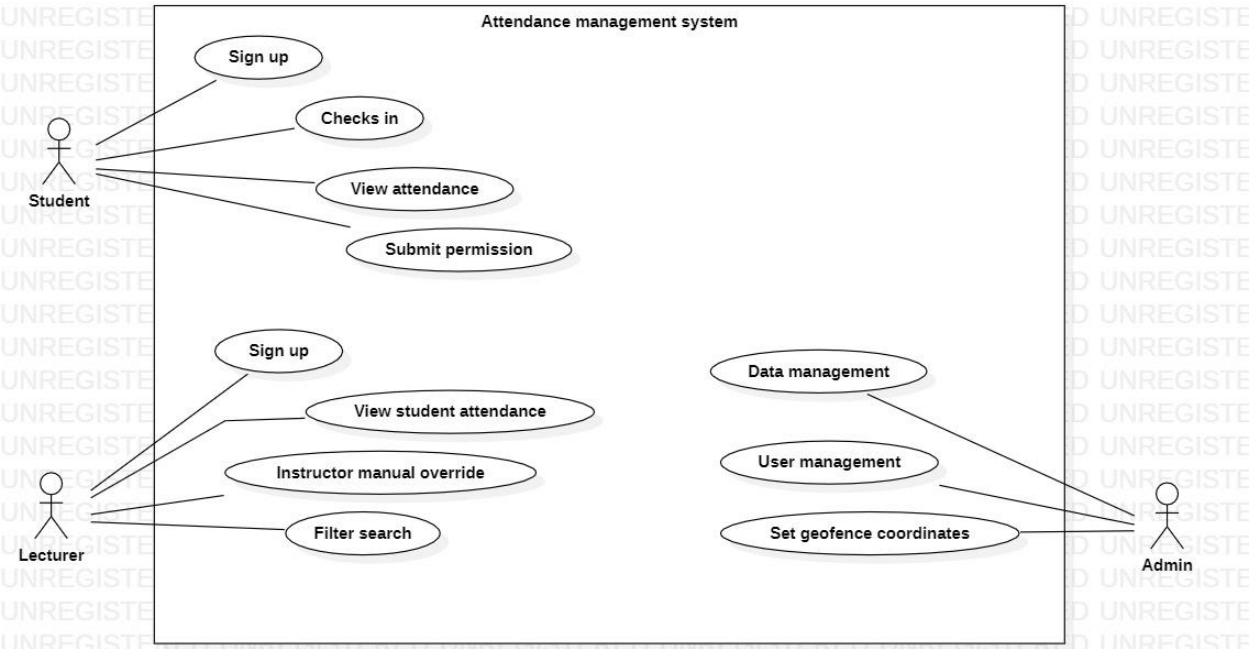

fig 2: Use case diagram

The Use Case Diagram provides a clear overview of the system's functional scope and the roles of each user type. By mapping out system capabilities and their corresponding users, this diagram serves as a foundation for detailed system design and testing.

III - CLASS DIAGRAM

Definition

A Class Diagram is a UML (Unified Modeling Language) diagram used to represent the static structure of a system. It shows the system's classes, their attributes, methods, and relationships, providing a blueprint of how the system is organized and how its components interact.

Key Elements

Element	Description
Classes	Core building blocks representing real-world entities or concepts in the system.
Attributes	Properties or variables associated with a class.
Methods	Actions or operations that can be performed by the class.
Relationships	Connections between classes: inheritance, association, or composition.

Class diagram Structure

❖ Core classes and their roles

✧ User (Abstract Base Class)

- Attributes: *userID (Unique identifier), name, email, image*
- Methods: *login(), logout()*
- Inheritance: *Parent to: Student, Instructor, Admin*

✧ Student

- Attributes: *studentID, courseList[] (Courses enrolled)*

- Methods: *checkIn() – Uses facial recognition and geofence validation, viewHistory() – Views attendance history.*
- Relationships: *Associated with Course (Enrollment)*

✧ Instructor

- Attributes: *instructorID, assignedCourses[]*
- Methods: *overrideAttendance() - Manually edit attendance, viewReports - Generate attendance analytics*
- Relationships: *Associated with Course (Teaches)*

✧ Admin

- Attributes: *adminID*
- Methods: *manageUsers() - Add/remove users, defineGeofence() - Set geofence parameters, configureSystem() - Adjust system-wide settings*

✧ Course

- Attributes: *courseID, courseName, schedule, instructorID*
- Methods: *getAttendance() - Retrieve attendance data*
- Relationships: *Composes a Geofence (Each course defines a unique location boundary)*

✧ Geofence

- Attributes: *geofenceID, latitude, longitude, radius*
- Methods: *boundary() - Validates location check-in*
- Relationships: *Composed by Course*

✧ Notification

- Attributes: *notificationID, recipientID, message, timestamp*
- Methods: *send() - Sends alerts to users*
- Relationships: *Associated with User (as recipient)*

❖ Key Relationships

Type	Description
Inheritance	*Student*, *Instructor* and *Admin* inherit from *User*
Association	*Student* ↔ *Course*, *Instructor* ↔ *Course*, *Notification* → *User*
Composition	*Course* → *Geofence* (Geofence cannot exist without it's course)

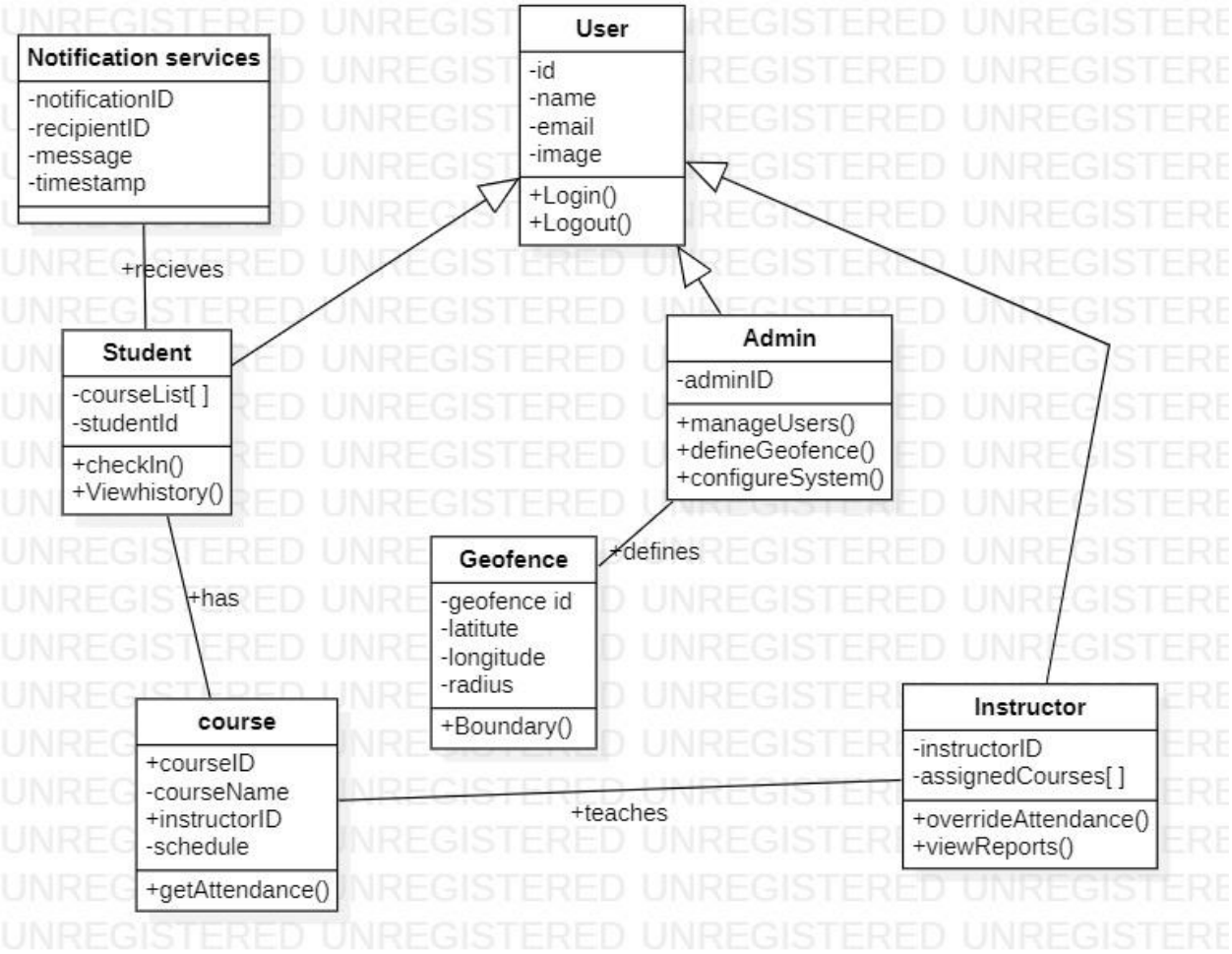

fig 3: Class diagram

IV - SEQUENCE DIAGRAM

Definition

A **Sequence Diagram** is a UML behavioral diagram that models the **interaction between objects** in a **time-sequenced order**. It illustrates how system components and users communicate to achieve a specific use case.

Purpose

- To model message flows between objects.
- To show the chronological order of operations for key functionalities.
- To visualize internal workings of a feature (especially useful for developers and designers).
- To analyze potential bottlenecks or failure points in interactions.

Use Case Modeled: Student Attendace Check-in

❖ Actors and Components

Entity	Role
Student (Actor)	Initiates login and check-in process.
Mobile App	Interface for user interaction and communication with backend.
Server	Processes requests, verifies identity, and records attendance.
GPS Service	Validates geolocation against geofence.
Facial recognition API	Matches submitted image with stored biometric data.
Database	Stores attendance records and user data.

Instructor dashboard	Reflects real-time attendance updates.

❖ Interaction Flow

1. **Login Sequence**
   - ✓Student opens the Mobile App.
   - ✓App prompts for face scan or password.
   - ✓App sends credentials to the Server.
   - ✓Server verifies and responds with login success/failure.
2. **Class Selection & Geofence Validation**
   - ✓Student selects a class from the app.
   - ✓App retrieves GPS location from the device.
   - ✓App sends location data to the Server.
   - ✓Server calls GPS Service to validate location.
   - ✓GPS Service returns location status ("In class" or "Out of bounds").
   - ✓Server responds to App with location confirmation.
3. **Facial Recognition**
   - ✓App captures student's face image.
   - ✓Sends image to the Server.
   - ✓Server forwards image to Facial Recognition API.
   - ✓API returns match result (success/failure).
   - ✓If successful, server proceeds to record attendance.
   - ✓If failed, allow 3 retries, then alert instructor.
4. **Attendance Recording**
   - ✓Server creates timestamped attendance entry.
   - ✓Saves it in the Database.

✓Updates Instructor Dashboard in real-time.

✓Sends confirmation to student: "Attendance marked!"

❖ Error handling Scenarios

- **Face mismatch**: 3 retry attempts → notify instructor.
- **Out-of-geofence**: Attendance blocked → instructor alert.
- **Network issues**: Cache data locally → sync when online.

❖ Security highlights

- **TLS 1.3** encryption for all communication.
- **Mathematical representations** used for biometric data (not raw images).
- Fully **GDPR compliant** system for student data privacy.

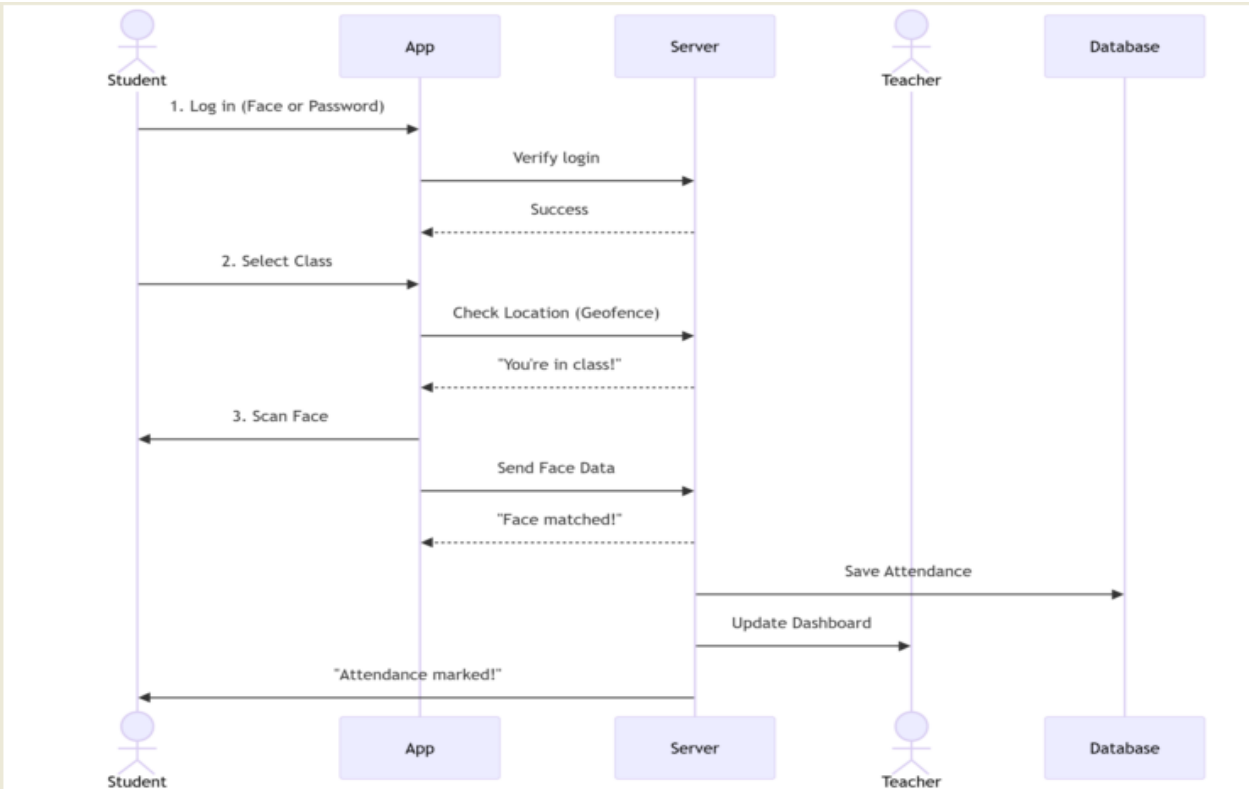

fig 4: Sequence diagram

V - DATA FLOW DIAGRAM (DFD)

Definition

A **Data Flow Diagram (DFD)** is a graphical tool that shows how **data flows through a system**, how it is **processed**, and how it **interacts** with external entities. It is widely used in the **requirement analysis** and **system design** stages of software development.

Purpose

- To visualize how data is collected, processed, and stored in the system.
- To separate process logic from the physical implementation.
- To improve understanding of the information flow and dependencies.
- To assist with system redesign, optimization, or documentation.

Key Elements

Element	Description
External entities	People or systems that send/receive data to/from the system (e.g., Student, Instructor).
Processes	Actions or operations that transform input data into output data (e.g., Verify Location).
Data stores	Storage areas where data is held for retrieval (e.g., Attendance DB).
Data flows	Arrows representing movement of data between elements.

DFD Level 1: Attendance Check-in Process

1. Student Login

Process: Authenticate Student

Input: Login credentials (username, password or biometric)

Output: Authentication response (success/fail)

Entities Involved: Student, Authentication Server

2. Geofence Verification

Process: Verify Location

Input: GPS coordinates from the student's device

Output: Geofence validation result (Inside / Outside)

Entities Involved: Student, GPS Service

3. Facial Recognition

Process: Validate Face

Input: Face image from the mobile app

Output: Match result (Match / No Match)

Entities Involved: Student, Facial Recognition API

4. Attendance Recording

Process: Record Attendance

Input: Verified student ID, timestamp

Output: Entry saved to database, dashboard updated

Data Stores: Attendance Database

Entities Involved: Instructor (via real-time dashboard)

5. Error Handling

Facial Recognition Fail: Trigger retry mechanism or notify instructor.

Outside Geofence: Block check-in and notify dashboard for manual override.

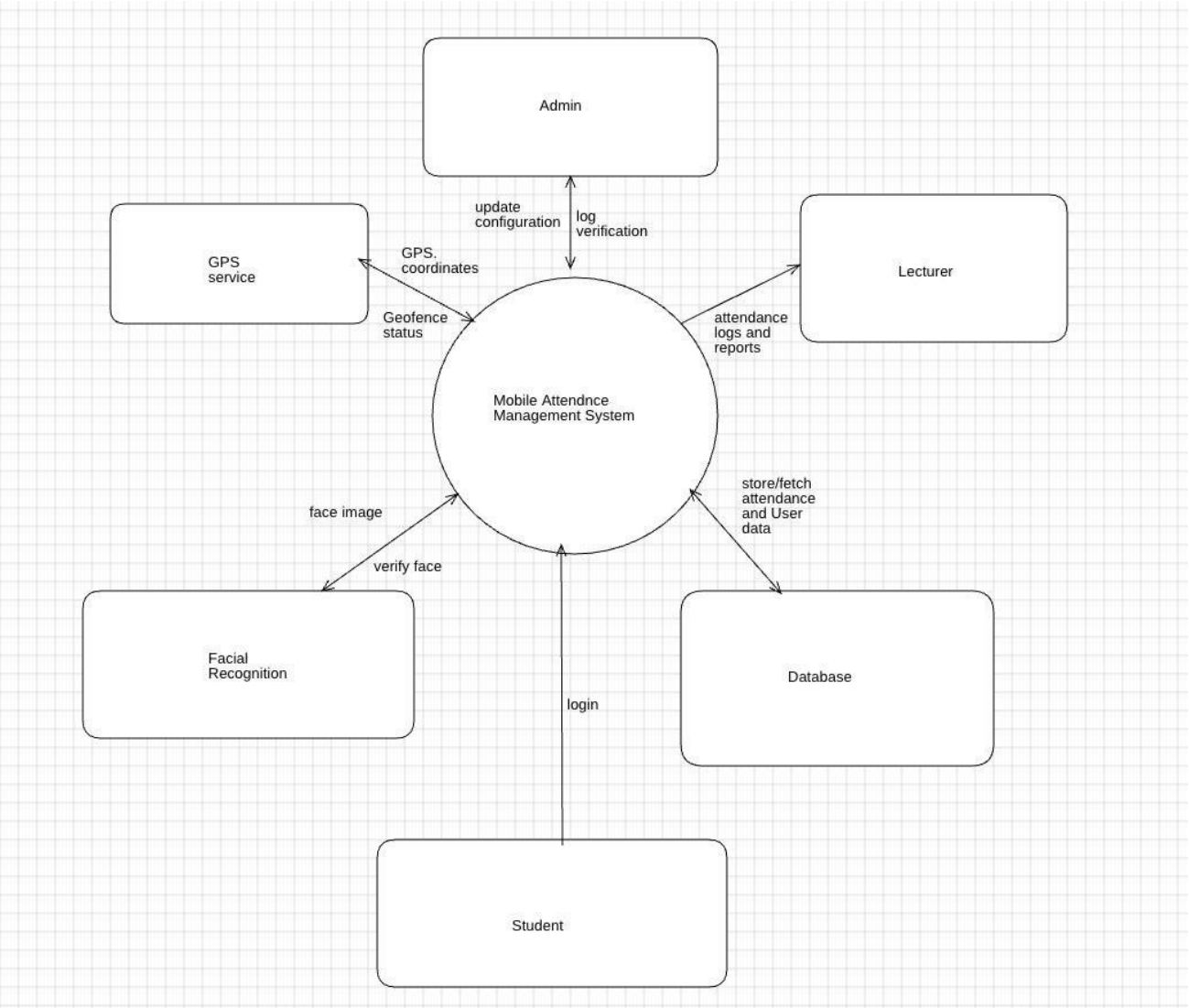

fig 5: Level 0 Data flow diagram

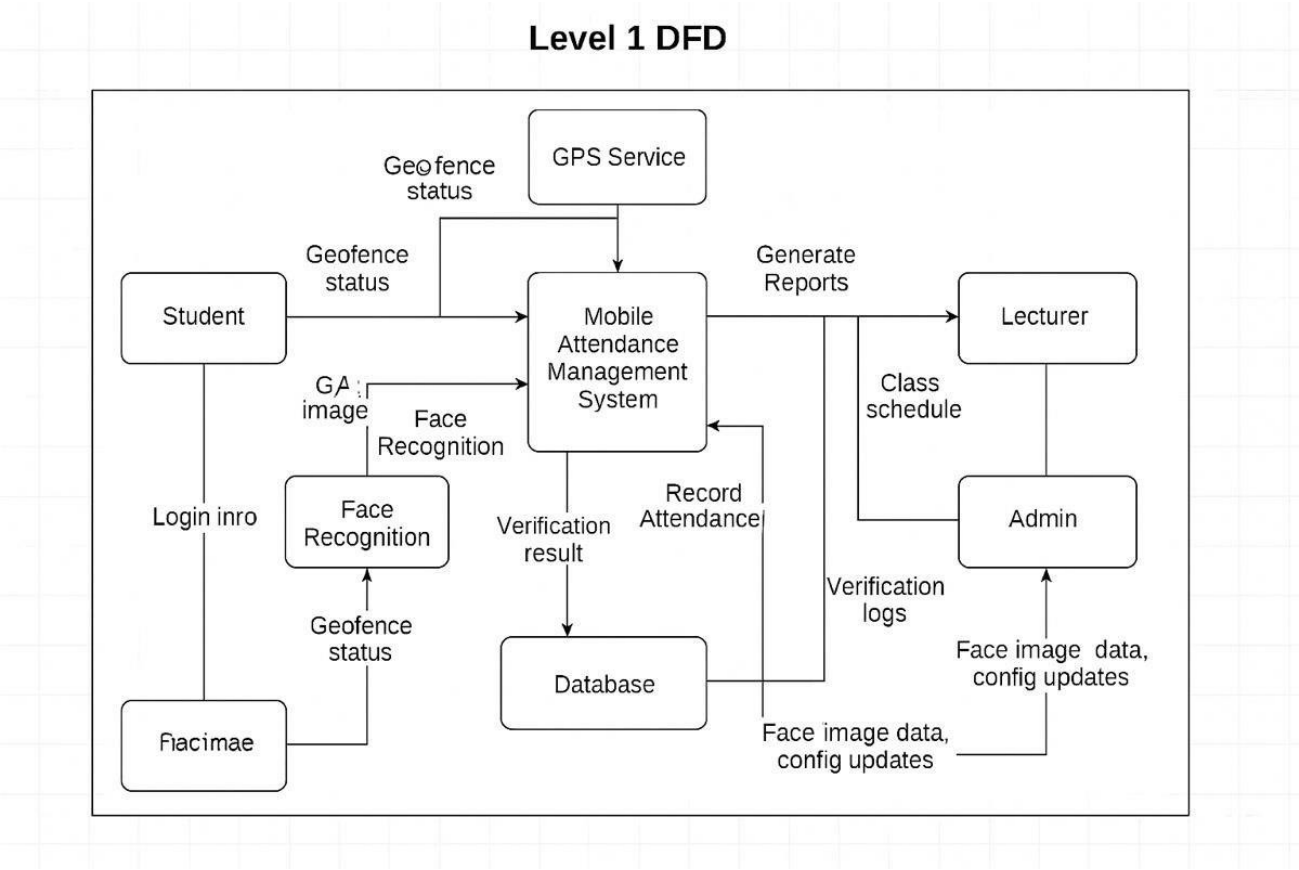

fig 6: Level 1 Data flow diagram

This DFD outlines the **logical flow of data** during a student check-in. It emphasizes the **sequential and conditional steps** involving location, biometric validation, and real-time updates, ensuring **accuracy**, **security**, and **transparency** in attendance management.

VI - DEPLOYMENT DIAGRAM

Overview

The deployment diagram illustrates the physical architecture of the Attendance Management System, showing how software components are deployed across hardware nodes. This includes the mobile client, cloud services, and external APIs such as GPS and facial recognition. It helps stakeholders understand how the system operates in a live environment and ensures scalability, reliability, and security.

Nodes and Components

- **Mobile App (Client Node):**
  - Platform: Android/iOS
  - Responsibilities:
    - ✧ User interface for students and instructors.
    - ✧ Captures user input (login credentials, face image).
    - ✧ Sends GPS coordinates.
    - ✧ Displays status messages (e.g., “Attendance Marked”).
    - ✧ Communicates securely with the backend using HTTPS.
- **Cloud Environment (Server Node):**
  - Hosted On: Cloud-based infrastructure (e.g., AWS, Azure, GCP).
  - Components Deployed:
    a. API Server:
       1. Acts as the core logic layer.
       2. Handles routing, request processing, and validation.

3. Interfaces with the database and services.

b. Authentication Service:
    1. Verifies credentials or facial biometric match.
    2. Manages session tokens and encryption protocols.
    3. Ensures secure and authenticated access.

c. Face Recognition Service:
    1. Receives and processes facial images.
    2. Compares against stored biometric templates.
    3. Returns a match/no match result.

d. GPS Geolocation Module:
    1. Verifies whether the device is within the allowed geofence.
    2. Cross-references student coordinates with pre-set class boundaries.

e. Database:
    1. Stores user data, attendance logs, geofence coordinates, and notifications.
    2. Supports queries for reporting, analytics, and dashboard display.
    3. Backed up regularly to ensure integrity and compliance.

Communication and Protocols

- All client-server communications use HTTPS for encrypted data exchange.
- Biometric data is sent as mathematical vectors — no raw images are stored.
- Location data is validated in real-time via geolocation services.

- System adheres to GDPR and local data privacy regulations.

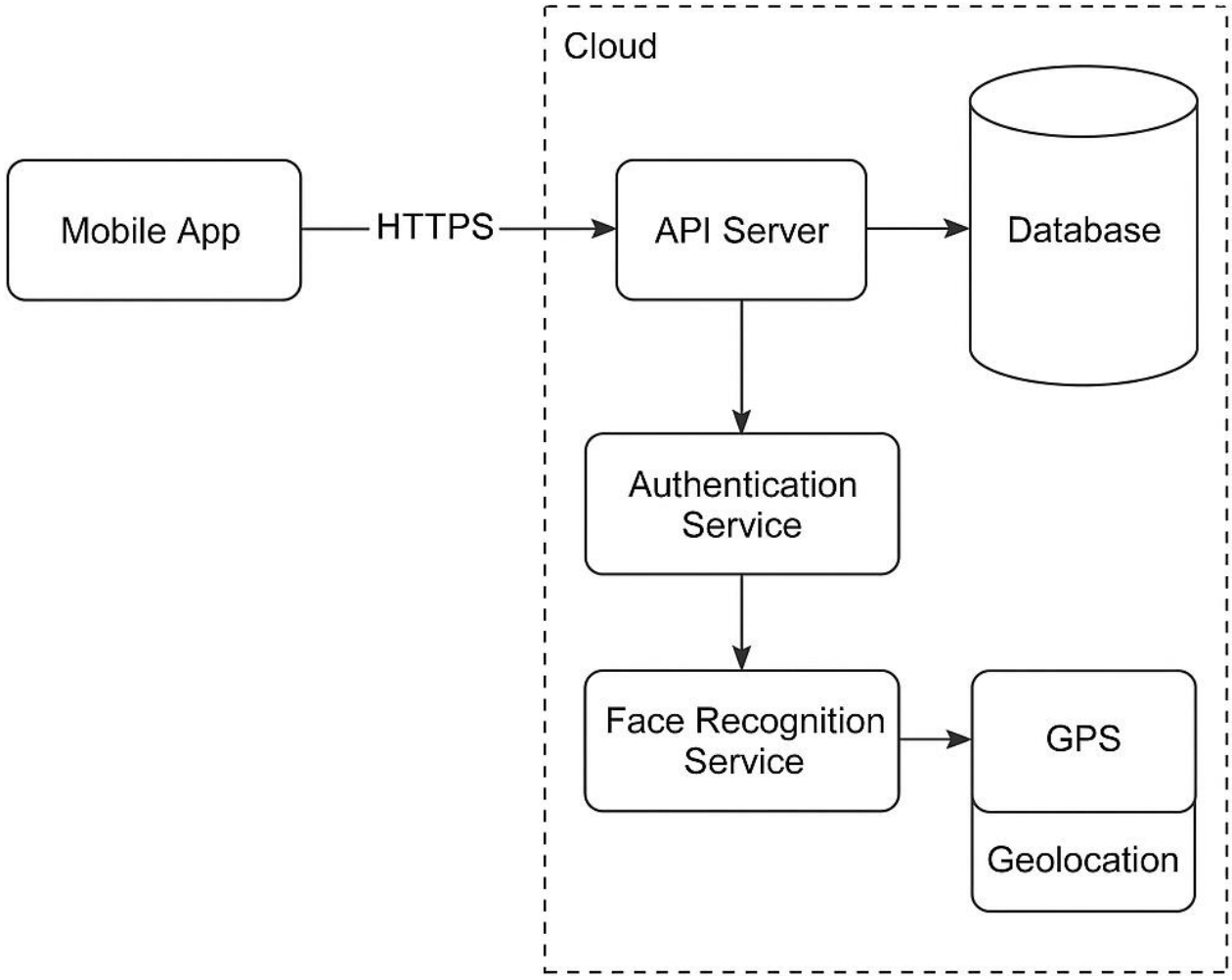

fig 7: Deployment diagram

Deployment Highlights

Scalability: Components can be independently scaled based on load (e.g., face recognition service can scale during peak hours).

Modularity: Services are decoupled, supporting microservices or containerized deployment (e.g., using Docker or Kubernetes).

Security: End-to-end encryption, minimal data exposure, secure API gateways.

Resilience: Mobile app can cache offline attempts and sync when connectivity resumes.

CONCLUSION

System modeling played a pivotal role in shaping the development of our Mobile-Based Attendance Management System. By using structured diagrams such as the context diagram, use case diagram, class diagram, sequence diagram, data flow diagram, and deployment diagram, we were able to visualize and understand the full scope of our application before implementation.

Each diagram contributed uniquely:

- The Context Diagram clarified the boundaries of our system and its interaction with external entities.
- The Use Case Diagram helped identify core functionalities and user roles, guiding feature prioritization.
- The Class Diagram revealed the underlying structure and relationships between data models, enabling a clean and maintainable backend design.
- The Sequence Diagram exposed the internal workflow and real-time interactions critical for check-in processes, including face recognition and geofencing.
- The Data Flow Diagram illuminated how data traverses the system, ensuring that transformations and validations were clearly mapped.
- Finally, the Deployment Diagram solidified our understanding of how the application would operate in the real world — across devices, cloud infrastructure, and third-party services — with an emphasis on security, scalability, and reliability.

Through this modeling process, we uncovered potential challenges early, refined our architecture, and aligned the entire team with a shared technical vision. These diagrams didn’t just document our system — they shaped it.

List of figures

fig 1: Context diagram.....6

fig 2: Use case diagram.....9

fig 3: Class diagram.....12

fig 4: Sequence diagram.....15

fig 5: Level 0 Data flow diagram.....18

fig 6: Level 1 Data flow diagram.....19

fig 7: Deployment diagram.....22