UNIVERSITY OF BUEA

FACULTY OF ENGINEERING AND TECHNOLOGY

DERPARTMENT OF COMPUTER ENGINEERING

TASK 4: SYSTEM MODELLING AND DESIGN

COURSE CODE: CEF 440

COURSE TITLE: INTERNET PROGRAMMING and MOBILE PROGRAMMING

COURSE INSTUCTOR: DR NKEMENI Valery

BY GROUP 17

DATE: May 2025

SN	NAME	MATRICULE
1	AKENJI FAITH SIRRI	FE22A142
2	DYL PADARAN AMBE MUNJO	FE22A193
3	KONGNYU DESCHANEL	FE22A234
4	NDI BERTRAND	FE22A252
5	NDUKIE EBOKE BLANDINE	FE22A254

Software Design Document (SDD)

Document Information

Field	Value
Document Title	Software Design Document
Version	1.0
Date	May 26, 2025
Project Name	Design and Implementation of a Mobile-Based Attendance Management System Based on Geofencing and Facial Recognition
Prepared By	Group 17

Table of Contents

1. Introduction

1.1 Purpose

This Software Design Document (SDD) provides a comprehensive architectural and design blueprint for the Mobile-Based Attendance Management System that leverages geofencing and facial recognition technologies. The system addresses critical challenges in academic attendance management by replacing traditional manual and RFID-based methods with an automated, secure, and efficient mobile solution.

The document serves as a detailed technical guide for developers, system architects, and stakeholders involved in implementing a mobile application that can efficiently capture student attendance through facial recognition, validate student presence using geolocation within predefined classroom boundaries, and provide real-time attendance monitoring capabilities for both students and instructors.

1.2 Scope

The SDD encompasses the complete design specification for a mobile-based attendance management system that integrates multiple advanced technologies including facial recognition using machine learning, GPS-based geofencing for location validation, secure biometric data storage, and real-time attendance tracking. The system addresses both functional requirements such as automated check-in processes and non-functional requirements including security, performance (5-second check-in limit), and scalability for university-wide deployment.

The scope includes mobile application design for students and faculty, backend system architecture for data processing and storage, integration with existing university systems, and comprehensive security measures for protecting biometric and personal data.

1.3 Intended Audience

This document is primarily intended for:

- Software architects and system designers responsible for high-level design decisions
- Development teams implementing the system components

- Quality assurance engineers developing test strategies
- Project managers overseeing development timelines and resource allocation
- Stakeholders requiring technical understanding of system capabilities
- Operations teams responsible for system deployment and maintenance

2. System Overview

2.1 System Purpose

The Mobile-Based Attendance Management System is designed to revolutionize attendance tracking in higher education institutions by combining facial recognition and geofencing technologies. The system addresses critical limitations of traditional attendance methods including time inefficiency, susceptibility to human error, proxy attendance fraud, and delayed access to attendance records.

Traditional manual sign-in sheets and RFID card systems are prone to manipulation, time-consuming to process, and often result in inaccurate attendance data. The mobile-based solution leverages the ubiquity of smartphones and advanced biometric technologies to create a secure, automated, and real-time attendance management platform that ensures student physical presence while maintaining data integrity and processing efficiency.

2.2 Key Objectives

The primary objectives of the attendance management system include:

Functional Objectives:

- Implement real-time attendance check-in using facial recognition technology with machine learning capabilities
- Validate student physical presence within classroom boundaries using GPS-based geofencing
- Complete the entire check-in process within 5 seconds per student to ensure minimal classroom disruption

- Provide secure storage and management of biometric data with appropriate privacy protections
- Enable real-time attendance monitoring for instructors with filtering capabilities by course, date, or student
- Allow students to view their attendance history and participation status for registered courses
- Integrate face capture and recognition modules using established machine learning libraries

Technical Objectives:

- Develop a robust mobile application architecture capable of handling biometric processing and GPS validation
- Implement secure data transmission and storage mechanisms for sensitive biometric information
- Ensure high accuracy in facial recognition while maintaining processing speed requirements
- Create scalable backend infrastructure supporting university-wide deployment
- Establish reliable geofencing algorithms that accurately define and validate classroom boundaries
- Integrate with existing university information systems for course and enrollment data

2.3 System Boundaries

The Mobile-Based Attendance Management System operates within clearly defined boundaries that establish its scope and interfaces with external entities:

Internal System Scope:

- Mobile application for student attendance check-in with facial recognition capabilities
- Mobile application for instructor attendance monitoring and management
- Backend server infrastructure for data processing, storage, and management
- Facial recognition engine with machine learning capabilities for biometric authentication
- Geofencing service for GPS-based location validation within classroom boundaries

- Database systems for secure storage of biometric data, attendance records, and system configuration
- Real-time notification and reporting services for attendance status updates

External System Interfaces:

- University student information systems for enrollment and course data
- Campus network infrastructure for connectivity and data transmission
- Mobile device GPS services for location tracking and validation
- Mobile device camera systems for facial image capture
- University authentication systems for user access control
- Campus facility management systems for classroom location and boundary data

System Boundaries and Constraints:

- The system operates exclusively within defined classroom boundaries established through GPS coordinates
- Facial recognition functionality requires adequate lighting conditions and clear facial visibility
- System functionality is dependent on mobile device capabilities including camera quality and GPS accuracy
- Network connectivity is required for real-time data synchronization and validation processes

2.4 Success Criteria

Success metrics for the Mobile-Based Attendance Management System implementation include:

Performance Metrics:

- Check-in process completion time not exceeding 5 seconds per student
- Facial recognition accuracy rate of 98% or higher under normal classroom lighting conditions
- GPS location validation accuracy within 95% for defined classroom boundaries
- System availability of 99.5% during class hours and academic periods

- Support for concurrent usage by up to 500 students during peak attendance periods

Functional Metrics:

- Successful elimination of proxy attendance through biometric verification
- Real-time attendance data availability with less than 30-second delay
- Integration with existing university systems achieving 100% data synchronization accuracy
- Student and instructor user adoption rate of 90% within the first semester
- Reduction in attendance processing time by 80% compared to traditional methods

Security and Privacy Metrics:

- Zero incidents of biometric data breaches or unauthorized access
- Compliance with institutional privacy policies and data protection regulations
- Secure encryption of all biometric data in transmission and storage
- Comprehensive audit trails for all system access and data modifications

3. System Architecture

3.1 Architectural Approach

The Mobile-Based Attendance Management System architecture follows a hybrid client-server model optimized for mobile-first deployment with emphasis on biometric processing and real-time location validation. The architectural approach incorporates proven design principles while addressing unique challenges of mobile biometric authentication and geospatial validation.

Architectural Principles:

Mobile-First Design: The system architecture prioritizes mobile device capabilities and constraints, optimizing for battery efficiency, processing power limitations, and network connectivity variations while maintaining high-performance biometric processing.

Security-by-Design: Comprehensive security measures are integrated throughout the architecture with particular emphasis on biometric data protection, secure transmission protocols, and privacy-preserving data storage mechanisms.

Real-Time Processing: The architecture supports real-time facial recognition processing, GPS validation, and attendance data synchronization to meet the 5-second check-in requirement while ensuring data consistency.

Scalable Biometric Processing: The system employs distributed processing patterns that can scale to handle university-wide deployment with thousands of concurrent users while maintaining recognition accuracy and response times.

Modular Service Architecture: Core functionality is decomposed into specialized services including facial recognition, geofencing validation, attendance management, and notification services that can be independently scaled and maintained.

3.2 Quality Attributes

The architectural design addresses critical quality attributes specific to mobile biometric systems:

Performance: The architecture incorporates local biometric processing capabilities, efficient compression algorithms for facial data, and optimized network protocols to ensure sub-5-second response times for attendance check-in operations.

Accuracy: Machine learning model optimization, multiple biometric validation points, and GPS coordinate validation algorithms ensure high accuracy in both facial recognition (98%+) and location validation (95%+) while minimizing false positives and negatives.

Security: Multi-layered security architecture includes biometric template encryption, secure key management, device-level authentication, network encryption, and comprehensive audit logging to protect sensitive student biometric data.

Privacy: Privacy-preserving design principles including biometric template hashing, data minimization, anonymization techniques, and granular consent management ensure compliance with privacy regulations and institutional policies.

Usability: Intuitive mobile interface design, minimal user interaction requirements, clear feedback mechanisms, and accessibility features ensure high user adoption and satisfaction rates among students and faculty.

Reliability: Fault-tolerant design patterns, offline capability for critical functions, data synchronization mechanisms, and comprehensive error handling ensure consistent system operation even under adverse network or device conditions.

4. Design Methodology

4.1 Design Process

The system design follows a structured methodology that ensures comprehensive analysis, systematic design decisions, and thorough validation of design artifacts. The design process incorporates iterative refinement and stakeholder feedback to optimize system characteristics.

Requirements Analysis Phase: Detailed analysis of functional and non-functional requirements establishes the foundation for design decisions. Requirements are analyzed for completeness, consistency, and feasibility.

Architectural Design Phase: High-level architectural decisions are made based on requirement analysis, quality attribute priorities, and constraint considerations. Architectural patterns and technology selections are validated against requirements.

Detailed Design Phase: Component-level design specifications are developed based on architectural guidelines. Interface definitions, data structures, and algorithmic approaches are specified in detail.

Design Validation Phase: Design artifacts are reviewed and validated against requirements through modeling, prototyping, and stakeholder review processes.

4.2 Design Principles

The design methodology adheres to established software engineering principles that promote quality, maintainability, and effectiveness:

Single Responsibility Principle: Each component has a single, well-defined responsibility that promotes cohesion and reduces complexity.

Open-Closed Principle: Components are designed to be open for extension but closed for modification, supporting system evolution without destabilizing existing functionality.

Dependency Inversion Principle: High-level modules are not dependent on low-level modules, promoting flexibility and testability through abstraction.

Interface Segregation Principle: Interfaces are designed to be specific and focused, avoiding unnecessary dependencies and promoting modularity.

4.3 Modeling Techniques

The design process employs various modeling techniques to visualize, analyze, and communicate design concepts:

Structural Modeling: Class diagrams and component diagrams represent static system structure and relationships between system elements.

Behavioral Modeling: Sequence diagrams and activity diagrams illustrate dynamic system behavior and interaction patterns.

Architectural Modeling: Context diagrams and deployment diagrams provide high-level views of system architecture and operational environment.

5. System Context and Environment

5.1 Context Diagram Analysis

The system context diagram illustrates the Mobile-Based Attendance Management System's relationship with external entities in the academic environment. The diagram establishes clear boundaries between the core attendance system and external actors including students, instructors, school administration, and technological infrastructure components.

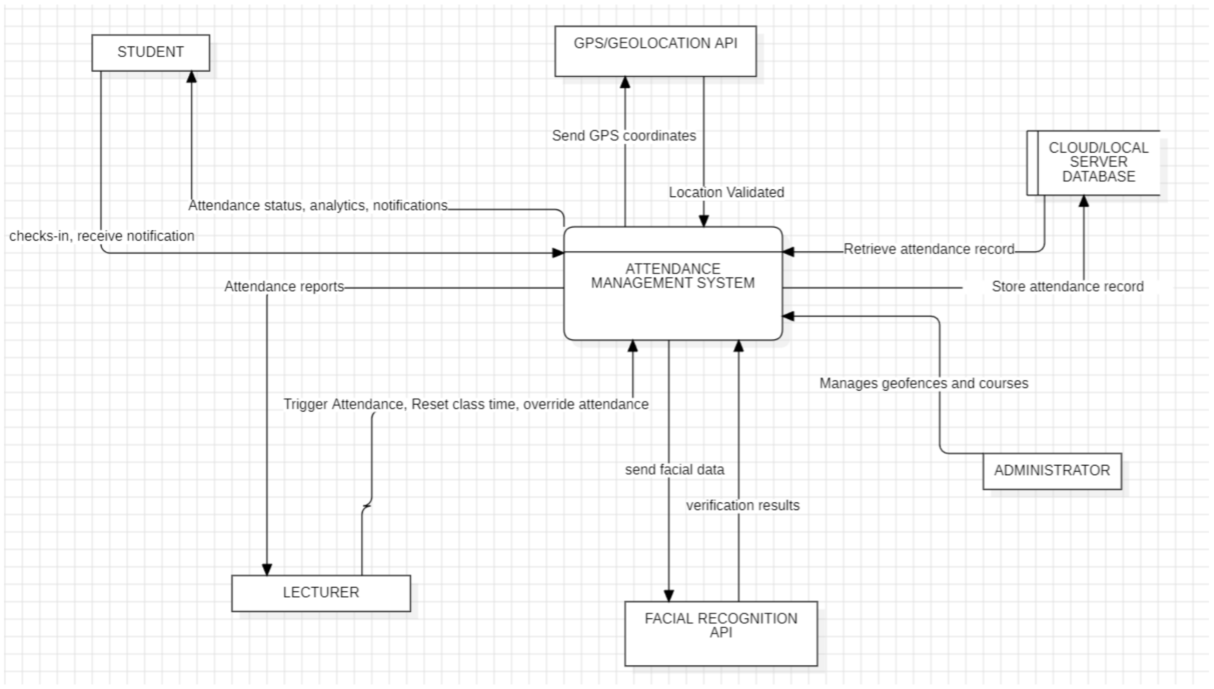

STUDENT
GPS/GEOLOCATION API
Send GPS coordinates
Location Validated
CLOUD/LOCAL SERVER DATABASE
Attendance status, analytics, notifications
checks-in, receive notification
Retrieve attendance record
ATTENDANCE MANAGEMENT SYSTEM
Attendance reports
Store attendance record
Manages geofences and courses
Trigger Attendance, Reset class time, override attendance
send facial data
verification results
ADMINISTRATOR
LECTURER
FACIAL RECOGNITION API

Figure 0-1: Context Diagram (level 0 DFD)

The context diagram demonstrates how the attendance management system serves as the central hub for processing attendance data while interfacing with multiple external entities that provide input data, consume system outputs, or influence system behavior through environmental factors

5.2 Main External Entities

The Mobile-Based Attendance Management System interacts with various external entities within the university ecosystem:

Students: Primary users who utilize mobile devices to check in for class attendance through facial recognition and GPS validation. Students interact with the system through mobile applications that capture biometric data and validate their physical presence within classroom boundaries. Students also access the system to view their attendance history and participation status across registered courses.

Instructors/Faculty: Secondary users who monitor and manage attendance data through dedicated interfaces. Instructors access real-time attendance information, generate attendance reports, and utilize filtering capabilities to analyze attendance patterns by course, date, or individual student. Faculty members may also configure classroom boundaries and attendance policies within their courses.

Geolocation API: External services including GPS location services for geofencing validation, camera systems for facial image capture, and device storage for temporary data caching. These services are essential for core system functionality but operate outside direct system control.

5.3 Environmental Factors

The system operates within a complex academic environment characterized by specific constraints and influences:

Physical Environment: Classroom locations, building layouts, GPS signal availability, and lighting conditions that affect both geofencing accuracy and facial recognition performance. Environmental factors include indoor GPS signal strength, classroom lighting variations, and student positioning relative to mobile devices.

Academic Environment: University policies regarding attendance requirements, privacy regulations, academic schedules, and semester cycles that influence system configuration and operational parameters. The academic calendar affects system usage patterns and peak load requirements.

Technological Environment: Mobile device diversity, operating system variations, network connectivity patterns, and campus IT infrastructure that determine system compatibility and performance characteristics. Device capabilities including camera quality, GPS accuracy, and processing power directly impact system effectiveness.

Regulatory Environment: Privacy laws, educational regulations, biometric data protection requirements, and institutional policies that constrain system design and operational procedures. Compliance requirements influence data handling, storage, and user consent mechanisms.

5.4 Interface Requirements

Each external interface requires specific protocols, data formats, and interaction patterns to ensure successful integration and operation. Interface requirements address both functional capabilities and quality attributes such as performance, security, and reliability.

6. Data Flow Analysis

6.1 Data Flow Diagram Overview

The data flow diagram for the Mobile-Based Attendance Management System illustrates how data is processed by a system in terms of inputs and outputs. Its focus is on the flow of information, where data comes from, where it goes, and where it gets stored, the movement of biometric data, location information, and attendance records through various processing stages. The diagram reveals the complex data transformations required to convert raw facial images and GPS coordinates into verified attendance records while maintaining security and privacy requirements.

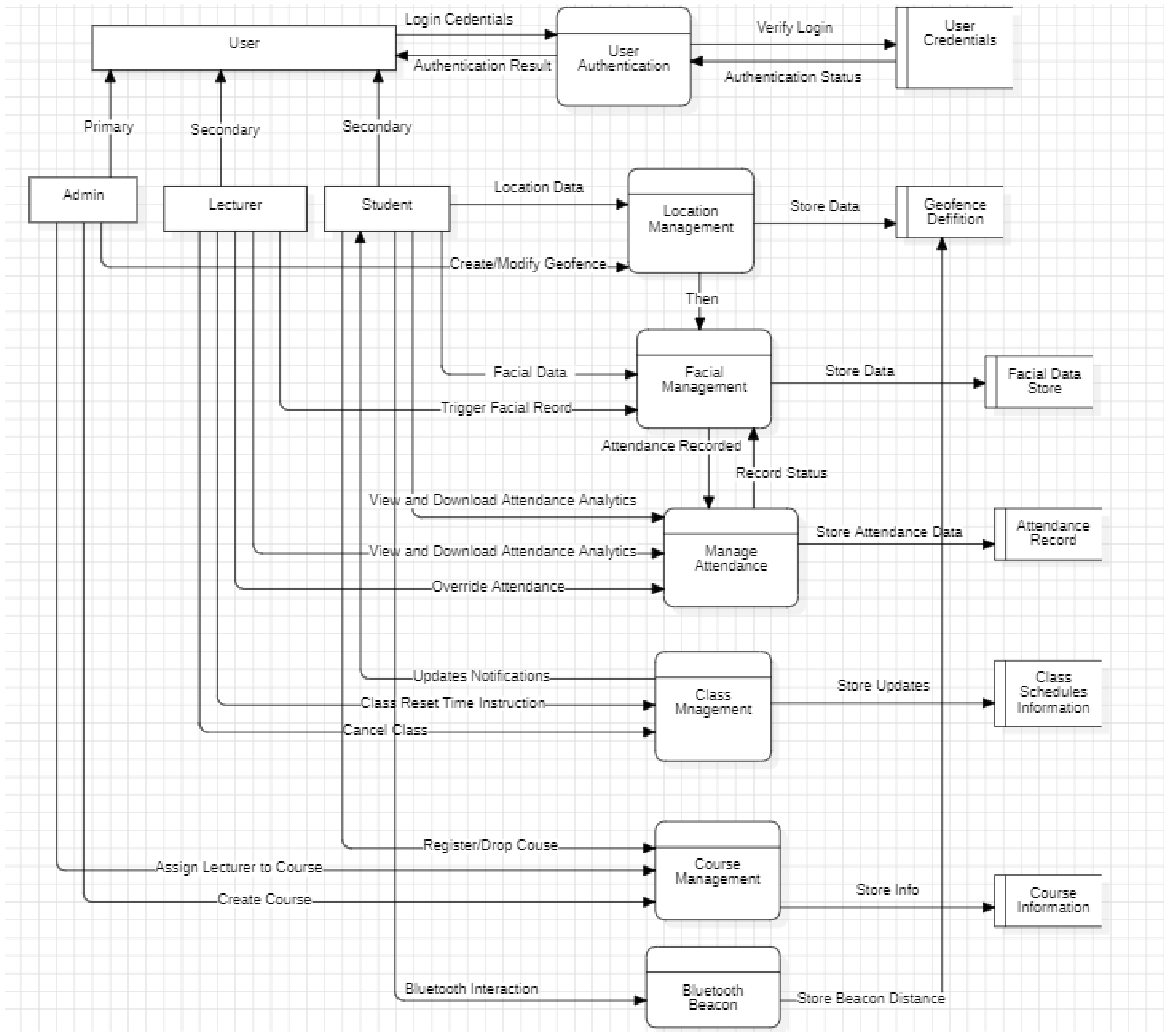


Figure 0-2: Dataflow Diagram

The data flow analysis demonstrates how the system processes multiple data streams simultaneously, including real-time facial recognition processing, GPS coordinate validation, and attendance record generation, while maintaining data integrity.

This diagram breaks down the "Attendance Management System" into its main processes.

6.2 External Entities

- Student
- Educator
- Administrator

6.3 Processes

- **P1: Manage User Authentication** (Handles FR1)
- **P2: Manage Facial Data** (Handles FR2, FR3, FR10)
- **P3: Manage Location Data** (Handles FR4, FR12)
- **P4: Record Attendance** (Handles FR5)
- **P5: Manage Attendance Records** (Handles FR6, FR11, FR14)
- **P6: Manage Class Schedule & Notifications** (Handles FR7, FR8)
- **P7: Provide Attendance Analytics** (Handles FR9)
- **P8: Manage Courses** (Handles FR13)
- **P9: Facilitate Bluetooth Interaction** (Handles FR15)

6.4 Data Stores

- **DS1: User Credentials** (Stores usernames, hashed passwords)
- **DS2: Facial Data Store** (Stores encrypted facial templates)
- **DS3: Geofence Definitions** (Stores coordinates and details of geofenced areas)
- **DS4: Attendance Records** (Stores student ID, course ID, timestamp, location, facial verification status)
- **DS5: Class Schedules** (Stores course, start time, end time, lecturer)
- **DS6: Course Information** (Stores course details, registered students, assigned lecturers)

6.5 Data Flow (One Selected Examples):

- **Student -> P1:** Login Credentials
- **P1 -> DS1:** Validate Credentials
- **DS1 -> P1:** Authentication Status

7. Functional Requirements and Use Cases

7.1 Use Case Diagram Analysis

The use case diagram for the Mobile-Based Attendance Management System illustrates the main functionalities of the system and how different types of users interact with it and provides a comprehensive view of system functionality from multiple user perspectives, illustrating the interactions between students, instructors, and system administrators with core attendance management features. The diagram emphasizes the biometric authentication and location validation capabilities that distinguish this system from traditional attendance methods.

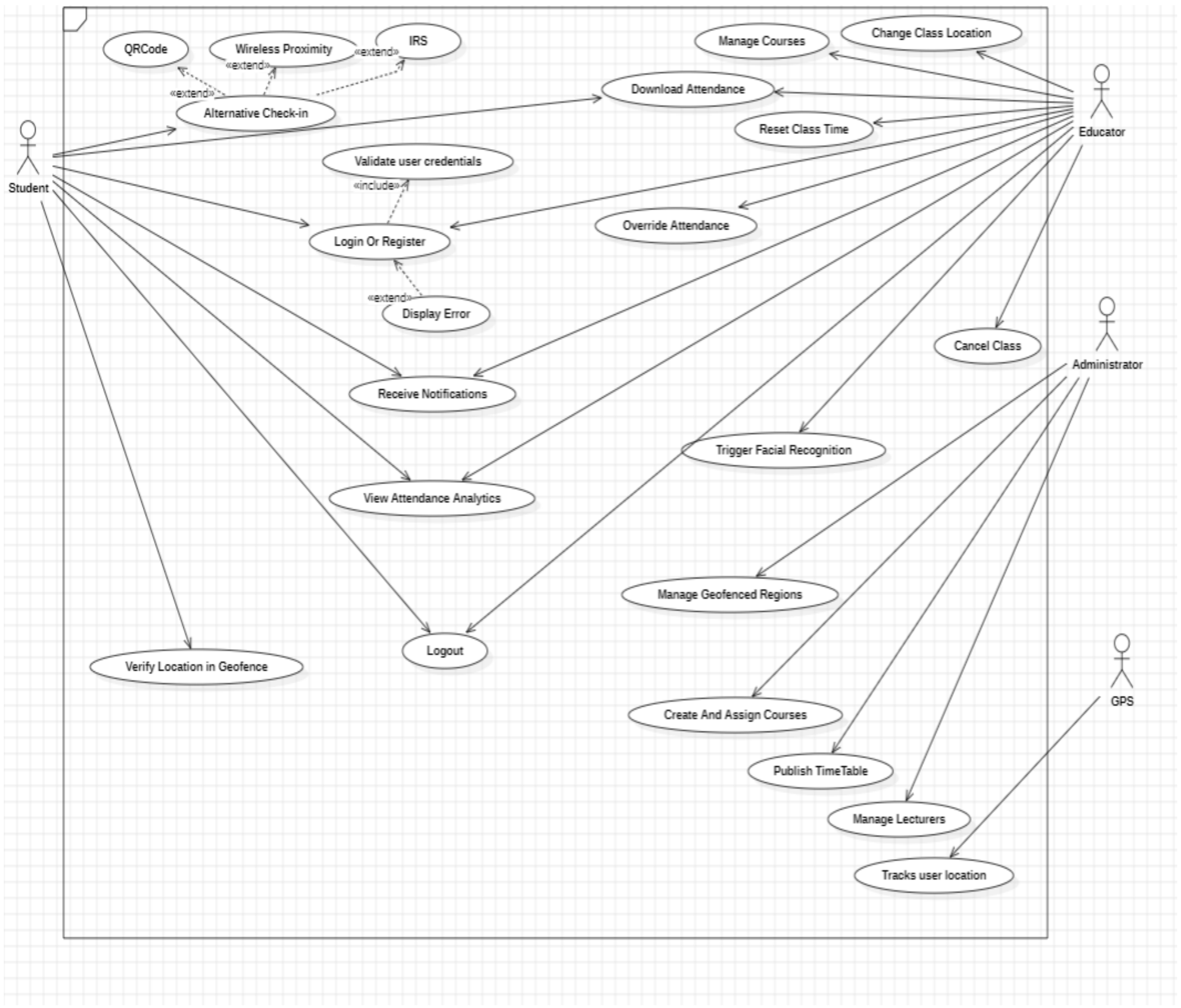


Figure 0-3: Use Case Diagram

7.2 Actor Identification

Primary Actor: Student (Left-Aligned)

The **Student** is the central user of the system who utilize the mobile application to register their attendance through biometric authentication and location validation. Students represent the primary system users who drive the core attendance check-in functionality and benefit from real-time access to their attendance history and participation status across registered courses. From the diagram, they have access to a wide range of functionalities.

Secondary Actors (Right-Aligned)

Educator (Lecturer)

Educators interact with various parts of the system primarily to **manage sessions and attendance**. They monitor and manage student attendance data through dedicated instructor interfaces. Instructors access real-time attendance information, generate attendance reports, analyze participation patterns, and utilize filtering capabilities to view attendance data by course, date, or individual student.

Administrator

The administrator has broader configuration and management powers as seen on the diagram.

GPS

Tracks User Location – Ensures students are within valid class areas, enabling the verify location.

7.2 Use Case Specifications

Each use case represents specific functional requirements optimized for the attendance management domain:

Student Attendance Check-In Use Cases:

Biometric Attendance Registration: Students initiate attendance check-in by capturing facial images through the mobile application. The system performs real-time facial recognition against enrolled templates while simultaneously validating student location within classroom boundaries. Success criteria include sub-5-second processing time, 98% recognition accuracy, and secure biometric data handling.

Location Validation: The system automatically captures and validates student GPS coordinates against predefined classroom geofences. Students must be physically present within specified boundaries before attendance registration is permitted. The system provides immediate feedback regarding location validation status and guides students to correct positioning if necessary.

Attendance History Access: Students can view comprehensive attendance records including participation status, attendance percentages, and historical data for all registered courses. The system provides filtering and search capabilities while maintaining privacy controls and data access restrictions.

Instructor Attendance Management Use Cases:

Real-Time Attendance Monitoring: Instructors access live attendance data during class sessions, observing student check-in status and participation levels in real-time. The system provides visual indicators, attendance statistics, and immediate notifications of attendance changes.

Attendance Report Generation: Instructors generate comprehensive attendance reports with filtering capabilities by course, date range, or individual student. Reports include participation statistics, trend analysis, and export capabilities for integration with grading systems.

Classroom Boundary Configuration: Instructors define and modify geofencing boundaries for their classrooms, adjusting GPS coordinate parameters and validation tolerances based on physical classroom characteristics and attendance policy requirements.

7.4 Use Case Relationships

Relationships between use cases provide insights into system complexity and reusability opportunities:

Include Relationships: Common functionality that is shared across multiple use cases, promoting reusability and consistency in system behavior.

Extend Relationships: Optional functionality that enhances basic use cases under specific conditions, providing flexibility and customization capabilities.

Generalization Relationships: Abstract use cases that capture common patterns and are specialized by concrete use cases, promoting design consistency and maintainability.

7.5 Traceability to Requirements

Each use case is traced to specific functional requirements, ensuring complete coverage and providing a basis for validation and testing activities.

8. System Structure and Components

8.1 Class Diagram Analysis

The class diagram represents the static structure of the system, showing classes, attributes, methods, and relationships. This diagram provides the foundation for object-oriented implementation and establishes the structural organization of system components.

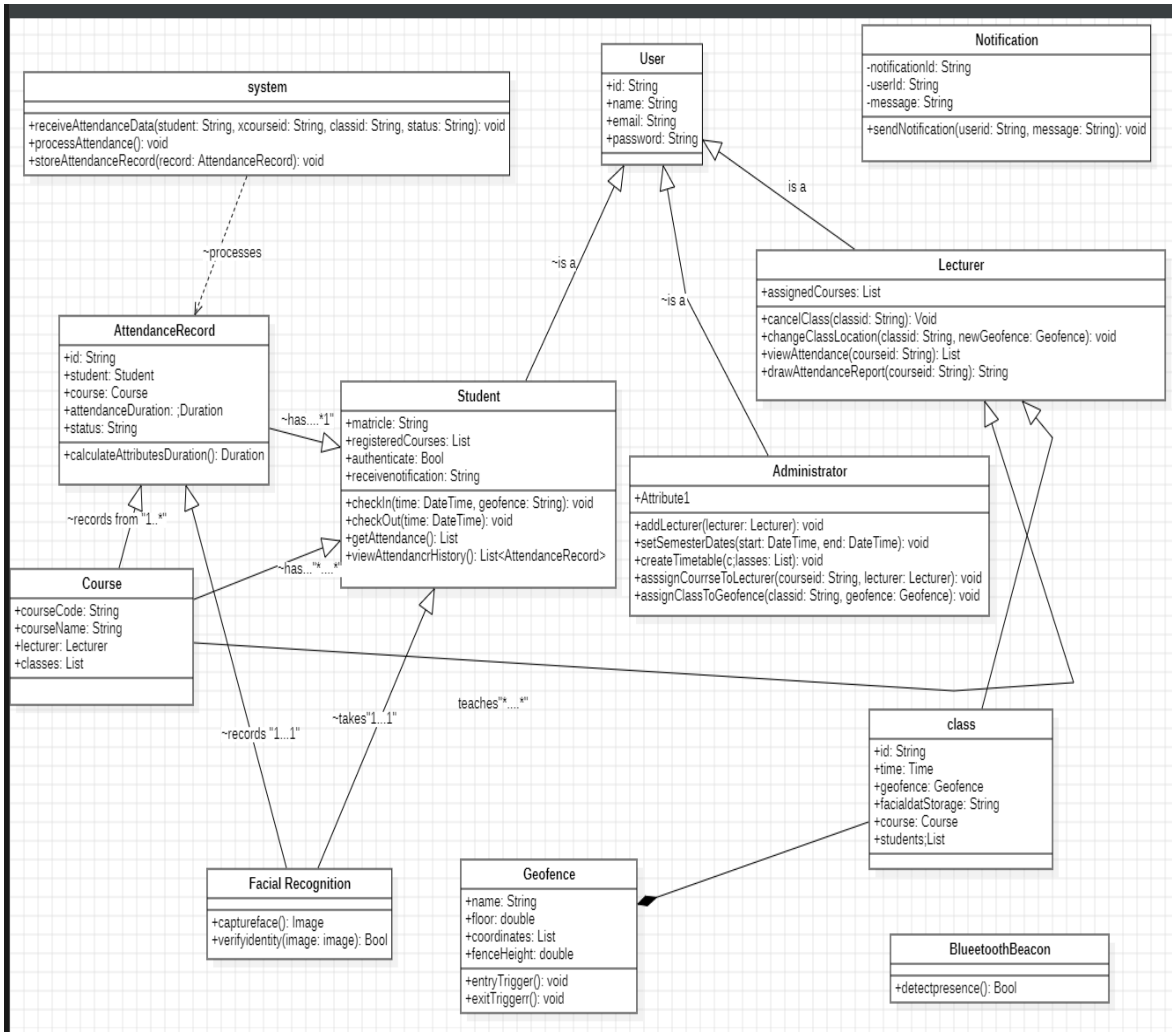

system
+receiveAttendanceData(student: String, xcourseid: String, classid: String, status: String): void
+processAttendance(): void
+storeAttendanceRecord(record: AttendanceRecord): void
User
+id: String
+name: String
+email: String
+password: String
Notification
-notificationId: String
-userId: String
-message: String
+sendNotification(userid: String, message: String): void
is a
~processes
~is a
~is a
Lecturer
+assignedCourses: List
+cancelClass(classid: String): Void
+changeClassLocation(classid: String, newGeofence: Geofence): void
+viewAttendance(courseid: String): List
+drawAttendanceReport(courseid: String): String
AttendanceRecord
+id: String
+student: Student
+course: Course
+attendanceDuration: ;Duration
+status: String
+calculateAttributesDuration(): Duration
~has....*1"
Student
+matricle: String
+registeredCourses: List
+authenticate: Bool
+receivenotification: String
+checkIn(time: DateTime, geofence: String): void
+checkOut(time: DateTime): void
+getAttendance(): List
+viewAttendancrHistory(): List<AttendanceRecord>
Administrator
+Attribute1
+addLecturer(lecturer: Lecturer): void
+setSemesterDates(start: DateTime, end: DateTime): void
+createTimetable(c;lasses: List): void
+asssignCourrseToLecturer(courseid: String, lecturer: Lecturer): void
+assignClassToGeofence(classid: String, geofence: Geofence): void
~records from "1..*"
~has..."*...*"
Course
+courseCode: String
+courseName: String
+lecturer: Lecturer
+classes: List
teaches"*....*"
~takes"1...1"
~records "1...1"
class
+id: String
+time: Time
+geofence: Geofence
+facialdatStorage: String
+course: Course
+students;List
Facial Recognition
+captureface(): Image
+verifyidentity(image: image): Bool
Geofence
+name: String
+floor: double
+coordinates: List
+fenceHeight: double
+entryTrigger(): void
+exitTriggerr(): void
BlueetoothBeacon
+detectpresence(): Bool

Figure 0-4: Class Diagram

8.2 Component Architecture

The system is organized into logical components that encapsulate related functionality and provide well-defined interfaces:

Presentation Layer Components: User interface components responsible for presenting information to users and capturing user input. These components handle formatting, validation, and user interaction management.

Data Access Components: Components responsible for managing data persistence, retrieval, and manipulation. These components abstract database operations and provide consistent data access interfaces.

Integration Components: Components that handle communication with external systems, including adapters, connectors, and protocol handlers.

Utility Components: Shared components that provide common functionality such as logging, security, configuration management, and error handling.

8.3 Class Relationships

The class structure employs various relationship types that define how components interact and depend on each other:

Inheritance Relationships: Hierarchical relationships that promote code reuse and polymorphic behavior through specialization of general classes.

Composition Relationships: Strong ownership relationships where contained objects have no independent existence, ensuring encapsulation and lifecycle management.

Aggregation Relationships: Weak association relationships where objects can exist independently while participating in larger structures.

Association Relationships: General relationships between classes that represent business or logical connections without ownership implications.

Dependency Relationships: Usage relationships where one class depends on another for specific functionality, influencing design decisions about coupling and modularity.

8.4 Implementation: Description of Classes

Our system consists of 12 classes detailed explained below;

1. **System**

The System class represents the core functionality of the smart attendance management system. It has the following methods:

- receiveAttendanceData(student: String, xcourseid: String, classid: String, status: String): Receives attendance data for a student.
- processAttendance(): Processes the attendance data.
- storeAttendanceRecord(record: AttendanceRecord): Stores the attendance record.

2. **User**

The User class represents a user of the system, with the following attributes:

- id: String
- name: String
- email: String
- password: String

The User class has a method sendNotification(userid: String, message: String) to send notifications to users.

3. **Lecturer**

The Lecturer class is a specialized type of User and has the following methods:

- assignedCourses: List: Holds the list of courses assigned to the lecturer.
- cancelClass(classid: String): Cancels a class.
- changeClassLocation(classid: String, newGeofence: Geofence): Changes the location of a class.
- viewAttendance(courseid: String): Views the attendance for a course.

- drawAttendanceReport(courseid: String): Generates an attendance report for a course.

4. **Student**

The Student class is a specialized type of User and has the following attributes:

- matricle: String: The student's matriculation number.
- registeredCourses: List: The list of courses the student is registered for.
- authenticate: Bool: A flag indicating whether the student is authenticated.
- receivenotification: String: The notification received by the student.

The Student class has a method calculateAttributesDuration() to calculate the duration of the student's attendance.

5. **AttendanceRecord**

The AttendanceRecord class represents a record of a student's attendance, with the following attributes:

- id: String: The unique identifier of the attendance record.
- student: Student: The student associated with the attendance record.
- course: Course: The course associated with the attendance record.
- attendanceDuration: Duration: The duration of the student's attendance.
- status: String: The status of the attendance record.

6. **Course**

The Course class represents a course in the system, with the following attributes:

- courseCode: String: The code of the course.
- courseName: String: The name of the course.
- lecturer: Lecturer: The lecturer assigned to the course.
- classes: List: The list of classes associated with the course.

7. **Class**

The Class class represents a specific class within a course, with the following attributes:

- id: String: The unique identifier of the class.
- time: Time: The time of the class.
- geofence: Geofence: The geofence associated with the class.
- facialdatStorage: String: The storage location for facial data.
- course: Course: The course associated with the class.
- students: List: The list of students enrolled in the class.

8. **Geofence**

The Geofence class represents a geographical area, with the following attributes:

- name: String: The name of the geofence.
- floor: double: The floor level of the geofence.
- coordinates: List: The list of coordinates defining the geofence.
- fenceHeight: double: The height of the geofence.

The Geofence class has two methods: entryTrigger() and exitTrigger(), which are triggered when a user enters or exits the geofence, respectively.

9. **Facial Recognition**

The FacialRecognition class is responsible for facial recognition, with the following methods:

- captureface(): Captures the face of a user.
- verifyidentity(image: image): Verifies the identity of a user based on the provided image.

10. **BluetoothBeacon**

The BluetoothBeacon class is responsible for detecting the presence of users, with the following method:

- detectpresence(): Detects the presence of a user.

11. Notification

The Notification class represents a notification sent to a user, with the following attributes:

- notificationId: String: The unique identifier of the notification.
- userId: String: The ID of the user receiving the notification.
- message: String: The message content of the notification.

The Notification class has a method sendNotification(userid: String, message: String) to send notifications to users.

Explaining class relationships and multiplicities

1. **User and Lecturer/Student Relationship**:
    - The User class is the parent class for both Lecturer and Student classes.
    - The relationship between User and Lecturer/Student is an "is-a" relationship, meaning Lecturer and Student are specialized types of User.
    - This is represented by the inheritance arrow from Lecturer and Student to User.
2. **Student and AttendanceRecord Relationship**:
    - The Student class has a one-to-many relationship with the AttendanceRecord class.
    - This is represented by the "~has..." association between Student and AttendanceRecord, with the multiplicity "... *" on the AttendanceRecord side, indicating that a student can have multiple attendance records.
3. **AttendanceRecord and Course Relationship**:
    - The AttendanceRecord class has a one-to-one relationship with the Course class.
    - This is represented by the "~records from 1 ... 1" association between AttendanceRecord and Course, indicating that each attendance record is associated with a single course.

4. **Course and Lecturer Relationship**:
   - The Course class has a one-to-one relationship with the Lecturer class.
   - This is represented by the "lecturer: Lecturer" attribute in the Course class, indicating that each course has a single lecturer assigned to it.
5. **Course and Class Relationship**:
   - The Course class has a one-to-many relationship with the Class class.
   - This is represented by the "classes: List" attribute in the Course class, indicating that a course can have multiple classes associated with it.
6. **Class and Student Relationship**:
   - The Class class has a many-to-many relationship with the Student class.
   - This is represented by the "students: List" attribute in the Class class, indicating that a class can have multiple students enrolled in it.
7. **Geofence and Class Relationship**:
   - The Class class has a one-to-one relationship with the Geofence class.
   - This is represented by the "geofence: Geofence" attribute in the Class class, indicating that each class is associated with a single geofence.
8. **Facial Recognition and User Relationship**:
   - The FacialRecognition class has a one-to-one relationship with the User class.
   - This is represented by the "verifyidentity(image: image)" method in the FacialRecognition class, which takes an image as input to verify the identity of a user.
9. **Bluetooth Beacon and User Relationship**:
   - The BluetoothBeacon class has a one-to-many relationship with the User class.
   - This is represented by the "detectpresence()" method in the BluetoothBeacon class, which can detect the presence of multiple users.

10. **Notification and User Relationship**:

    - The Notification class has a one-to-one relationship with the User class.
    - This is represented by the "sendNotification(userid: String, message: String)" method in the Notification class, which sends a notification to a specific user.

The multiplicities in the class diagram indicate the cardinality of the relationships between the classes. For example, the "~has..." association between Student and AttendanceRecord has a multiplicity of "... *", meaning that a student can have multiple attendance records.

9. System Behavior and Interactions

9.1 Sequence Diagram Analysis

Sequence diagrams illustrate the dynamic behavior of the system by showing how objects interact over time to accomplish specific scenarios. These diagrams provide detailed insights into message flows, timing constraints, and component responsibilities.

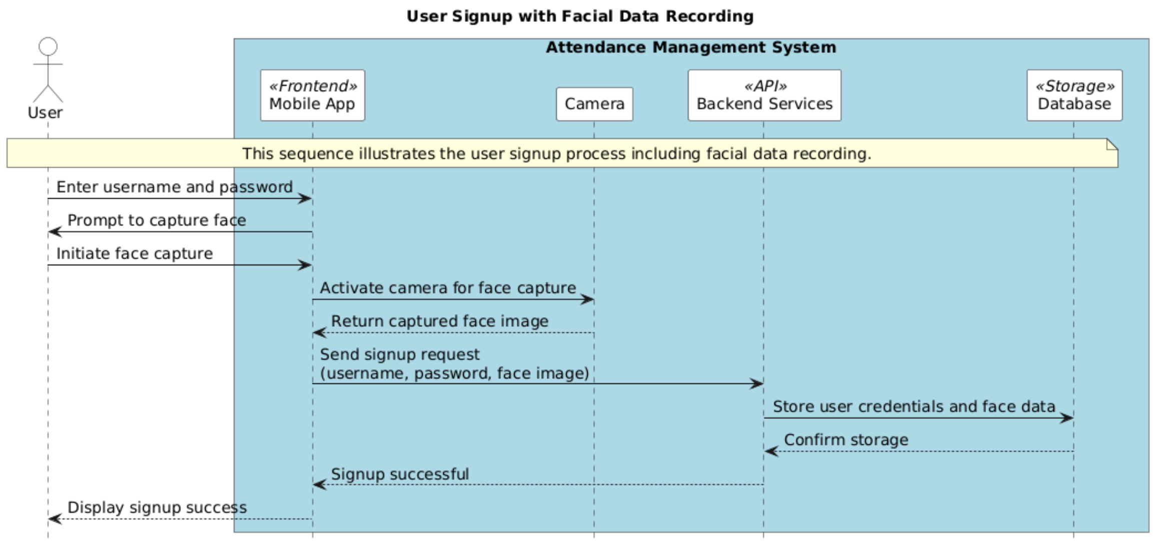
The following six sequence diagrams are included, covering the core functionalities of the system:

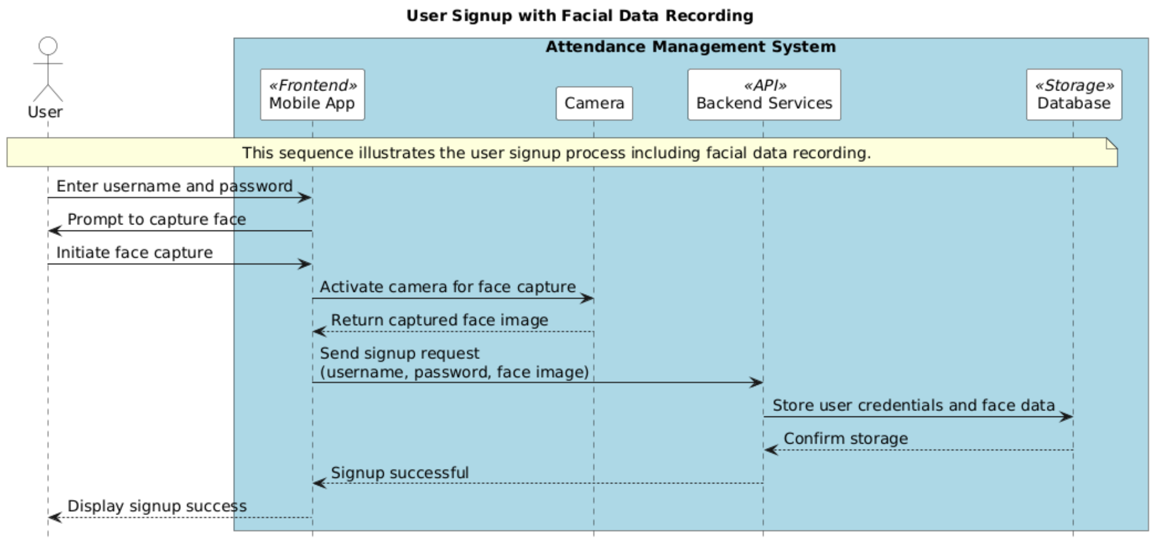
1. User Authentication (FR1)
2. Student Attendance Recording (FR4, FR5, FR2, FR3)
3. Lecturer Triggers Facial Recognition (FR10, FR4)
4. Lecturer Manages Attendance (FR6, FR11, FR14)
5. Course Management (FR13)

Each diagram is described in detail, with placeholders for their visual representations. These diagrams were created using PlantUML, ensuring clarity and professional presentation.

9.2 User Authentication

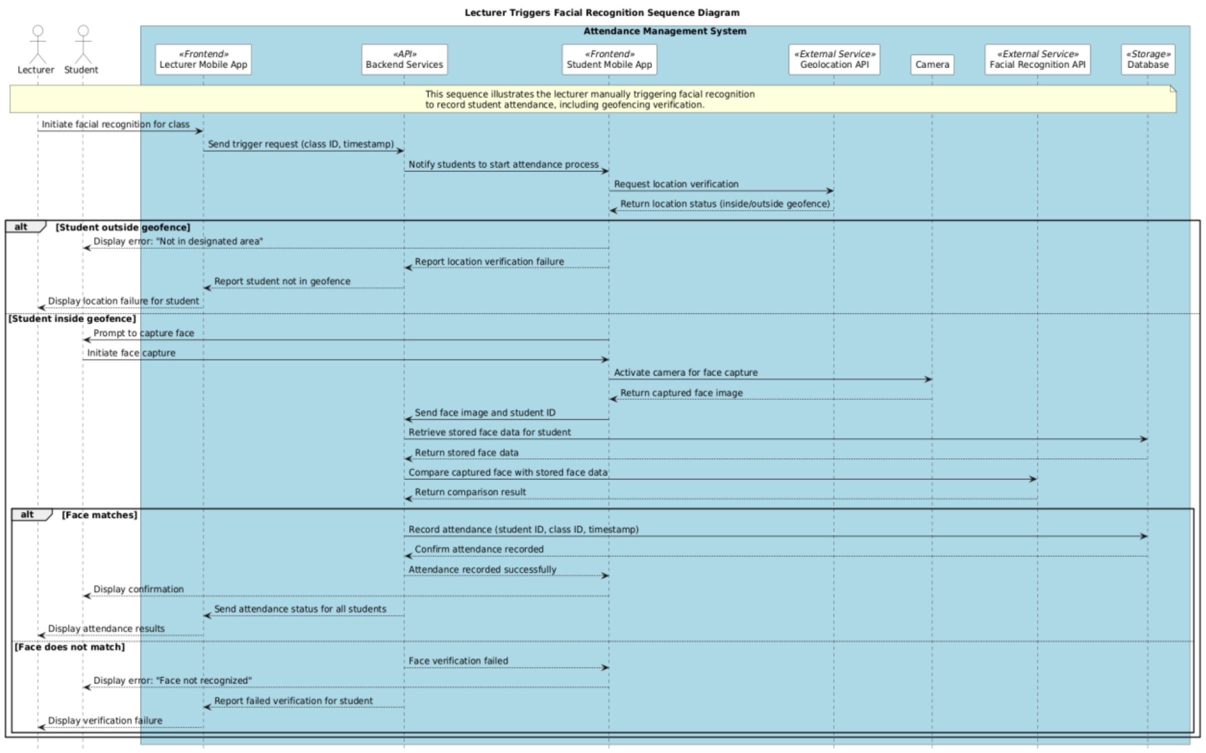
The User Authentication sequence diagram illustrates the login process for students, lecturers, and administrators, incorporating both credential verification and facial recognition, as per an enhanced requirement. The user enters their credentials, followed by a facial capture, which is verified against stored data.


User Signup with Facial Data Recording
Attendance Management System
User
«Frontend» Mobile App
Camera
«API» Backend Services
«Storage» Database
This sequence illustrates the user signup process including facial data recording.
Enter username and password
Prompt to capture face
Initiate face capture
Activate camera for face capture
Return captured face image
Send signup request (username, password, face image)
Store user credentials and face data
Confirm storage
Signup successful
Display signup success


User Signup with Facial Data Recording
Attendance Management System
User
«Frontend» Mobile App
Camera
«API» Backend Services
«Storage» Database
This sequence illustrates the user signup process including facial data recording.
Enter username and password
Prompt to capture face
Initiate face capture
Activate camera for face capture
Return captured face image
Send signup request (username, password, face image)
Store user credentials and face data
Confirm storage
Signup successful
Display signup success

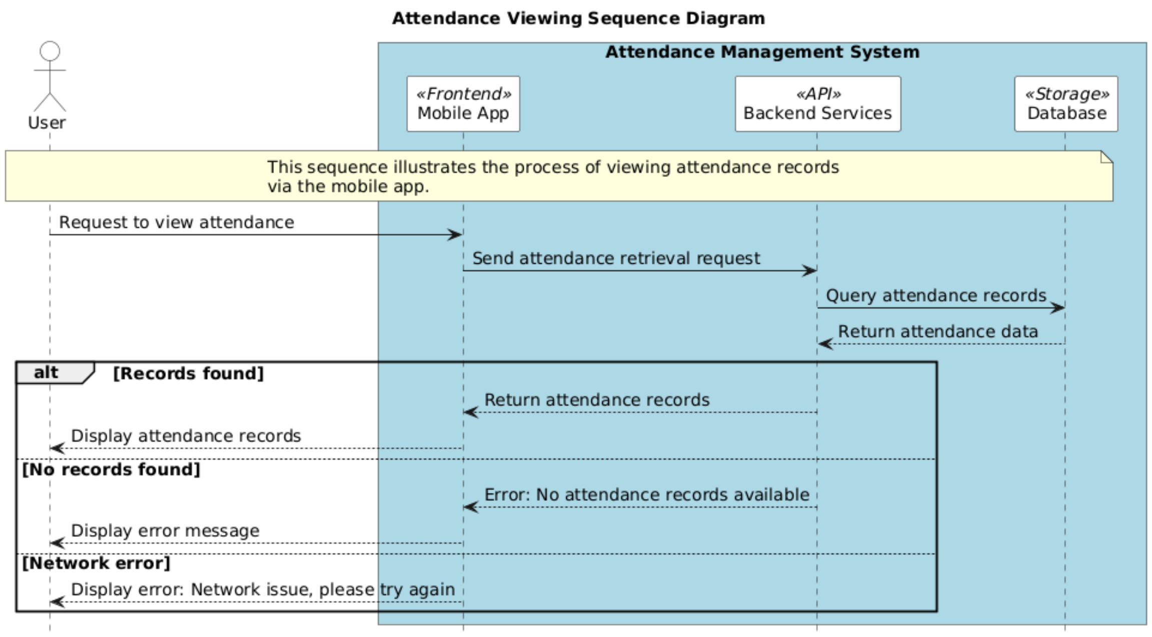
9.3 Student Attendance Recording

This diagram depicts the process of a student recording their attendance, involving geofencing (FR4) and facial recognition (FR5). The student's location is verified within a geofenced area before their face is captured and verified to record attendance.


Lecturer Triggers Facial Recognition Sequence Diagram
Attendance Management System
Lecturer
Student
«Frontend» Lecturer Mobile App
«API» Backend Services
«Frontend» Student Mobile App
«External Service» Geolocation API
Camera
«External Service» Facial Recognition API
«Storage» Database
This sequence illustrates the lecturer manually triggering facial recognition to record student attendance, including geofencing verification.
Initiate facial recognition for class
Send trigger request (class ID, timestamp)
Notify students to start attendance process
Request location verification
Return location status (inside/outside geofence)
alt [Student outside geofence]
Display error: "Not in designated area"
Report location verification failure
Report student not in geofence
Display location failure for student
[Student inside geofence]
Prompt to capture face
Initiate face capture
Activate camera for face capture
Return captured face image
Send face image and student ID
Retrieve stored face data for student
Return stored face data
Compare captured face with stored face data
Return comparison result
alt [Face matches]
Record attendance (student ID, class ID, timestamp)
Confirm attendance recorded
Attendance recorded successfully
Display confirmation
Send attendance status for all students
Display attendance results
[Face does not match]
Face verification failed
Display error: "Face not recognized"
Report failed verification for student
Display verification failure

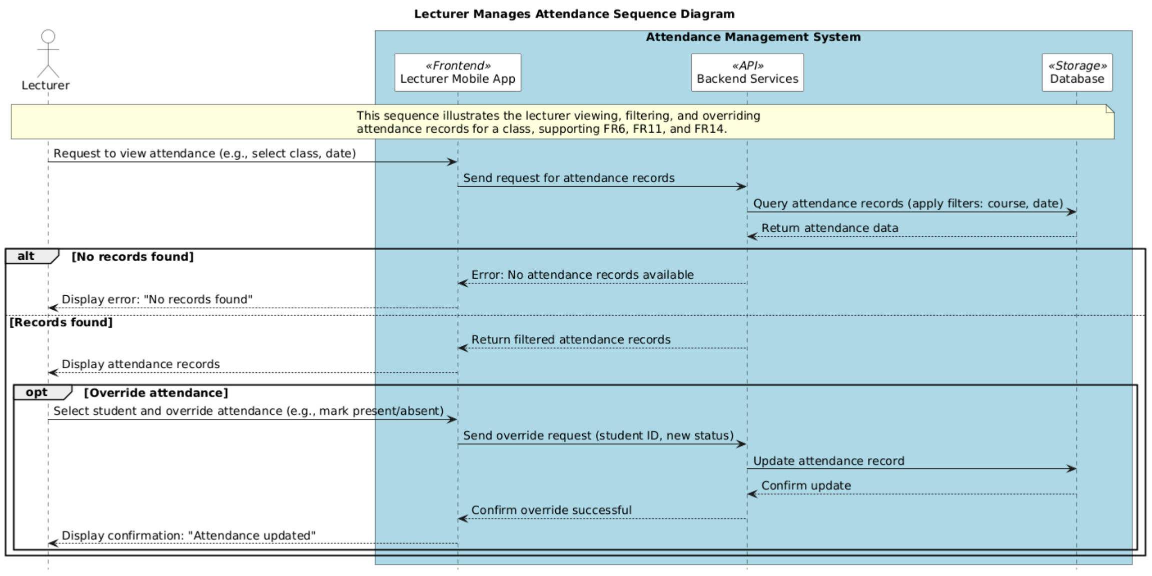
9.4 Lecturer Triggers Facial Recognition

The Lecturer Triggers Facial Recognition diagram shows how a lecturer initiates the facial recognition process for a class (FR10), with geofencing verification (FR4) to ensure students are in the designated area. This addresses timing conflicts, such as late lecture starts.


Attendance Viewing Sequence Diagram
Attendance Management System
User
«Frontend» Mobile App
«API» Backend Services
«Storage» Database
This sequence illustrates the process of viewing attendance records via the mobile app.
Request to view attendance
Send attendance retrieval request
Query attendance records
Return attendance data
alt [Records found]
Return attendance records
Display attendance records
[No records found]
Error: No attendance records available
Display error message
[Network error]
Display error: Network issue, please try again

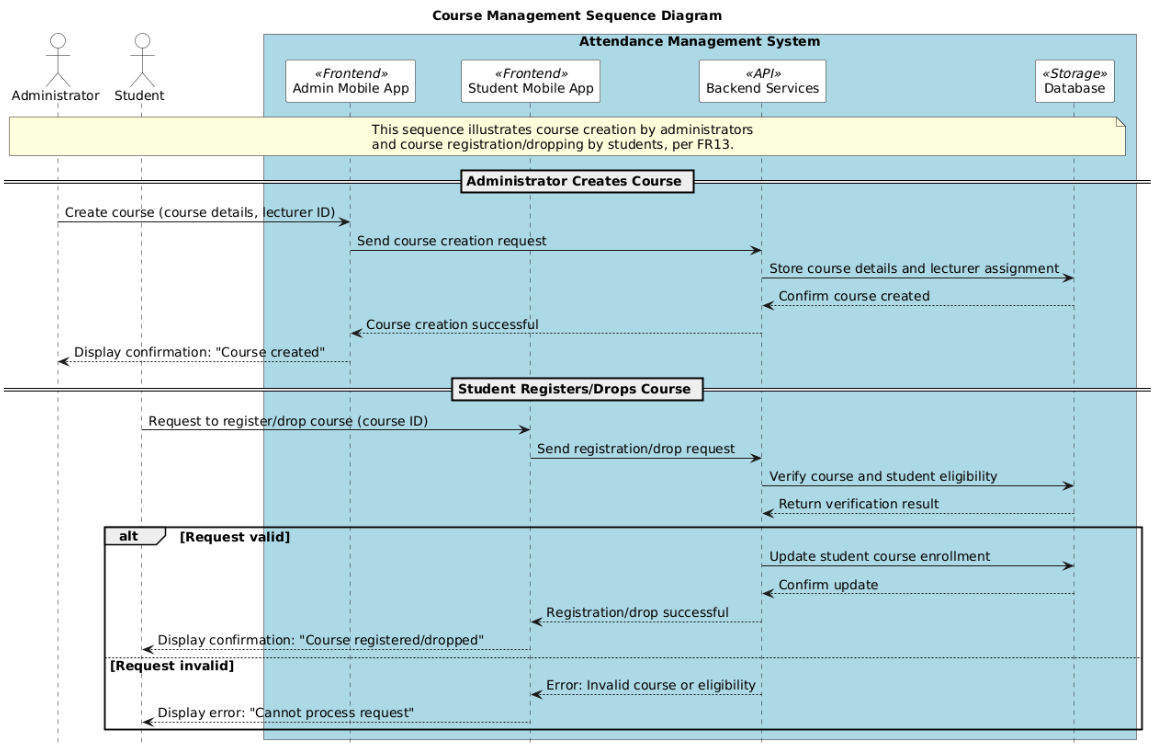
9.5 Lecturer Manages Attendance

This diagram illustrates how a lecturer views, filters (FR11), and overrides (FR14) attendance records (FR6). It supports manual adjustments for cases like student dismissals or valid absences, ensuring flexibility in attendance management.


Lecturer Manages Attendance Sequence Diagram
Attendance Management System
Lecturer
«Frontend» Lecturer Mobile App
«API» Backend Services
«Storage» Database
This sequence illustrates the lecturer viewing, filtering, and overriding attendance records for a class, supporting FR6, FR11, and FR14.
Request to view attendance (e.g., select class, date)
Send request for attendance records
Query attendance records (apply filters: course, date)
Return attendance data
alt [No records found]
Error: No attendance records available
Display error: "No records found"
[Records found]
Return filtered attendance records
Display attendance records
opt [Override attendance]
Select student and override attendance (e.g., mark present/absent)
Send override request (student ID, new status)
Update attendance record
Confirm update
Confirm override successful
Display confirmation: "Attendance updated"

9.6 Course Management

The Course Management diagram details how administrators create courses and assign lecturers, and how students register or drop courses (FR13). It includes validation to ensure eligibility and proper course data.


Course Management Sequence Diagram
Attendance Management System
Administrator
Student
«Frontend» Admin Mobile App
«Frontend» Student Mobile App
«API» Backend Services
«Storage» Database
This sequence illustrates course creation by administrators and course registration/dropping by students, per FR13.
Administrator Creates Course
Create course (course details, lecturer ID)
Send course creation request
Store course details and lecturer assignment
Confirm course created
Course creation successful
Display confirmation: "Course created"
Student Registers/Drops Course
Request to register/drop course (course ID)
Send registration/drop request
Verify course and student eligibility
Return verification result
alt [Request valid]
Update student course enrollment
Confirm update
Registration/drop successful
Display confirmation: "Course registered/dropped"
[Request invalid]
Error: Invalid course or eligibility
Display error: "Cannot process request"

The five sequence diagrams presented in this report comprehensively cover the core functionalities of the mobile-based attendance management system, as specified in the SRS. Each diagram illustrates the interactions between actors and system components, ensuring alignment with functional requirements (FR1, FR2, FR3, FR4, FR5, FR6, FR10, FR11, FR12, FR13, FR14). The diagrams incorporate error handling and address challenges like timing conflicts, network dependencies, and invalid data, ensuring a robust design.

These diagrams serve as a critical tool for developers, testers, and stakeholders to understand the system's workflow, facilitating effective implementation and validation. Future steps include refining these diagrams based on stakeholder feedback and integrating them into the development process.

10. Deployment Architecture

10.1 Deployment Diagram Analysis

The deployment diagram show the physical deployment of software artifacts on hardware nodes.

It illustrates the physical deployment of system components across hardware and software infrastructure. This diagram provides insights into system topology, resource requirements, and operational characteristics.

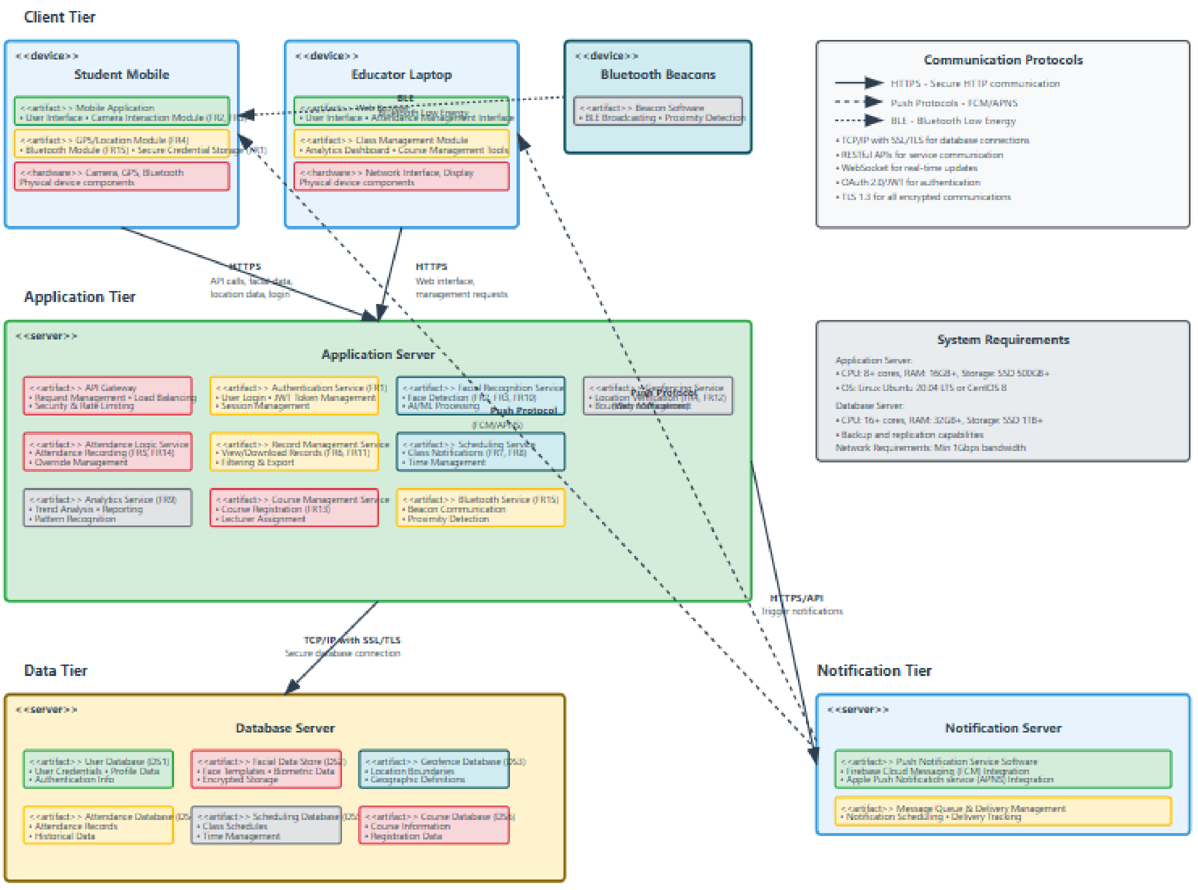

Client Tier
<<device>> Student Mobile
<<artifact>> Mobile Application • User Interface • Camera Interaction Module (FR2, FR3)
<<artifact>> GPS/Location Module (FR4) • Bluetooth Module (FR15) • Secure Credential Storage (NR1)
<<hardware>> Camera, GPS, Bluetooth Physical device components
<<device>> Educator Laptop
<<artifact>> Web Browser • User Interface • Attendance Management Interface
BLE Bluetooth Low Energy
<<artifact>> Class Management Module • Analytics Dashboard • Course Management Tools
<<hardware>> Network Interface, Display Physical device components
<<device>> Bluetooth Beacons
<<artifact>> Beacon Software • BLE Broadcasting • Proximity Detection
Communication Protocols
HTTPS - Secure HTTP communication
Push Protocols - FCM/APNS
BLE - Bluetooth Low Energy
• TCP/IP with SSL/TLS for database connections
• RESTful APIs for service communication
• WebSocket for real-time updates
• OAuth 2.0/JWT for authentication
• TLS 1.3 for all encrypted communications
HTTPS API calls, facial data, location data, login
HTTPS Web interface, management requests
Application Tier
<<server>> Application Server
<<artifact>> API Gateway • Request Management • Load Balancing • Security & Rate Limiting
<<artifact>> Authentication Service (FR1) • User Login • JWT Token Management • Session Management
<<artifact>> Facial Recognition Service • Face Detection (FR2, FR3, FR10) • AI/ML Processing
<<artifact>> Geofencing Service • Location Verification (FR4, FR12) • Boundary Management
Push Protocol (FCM/APNS)
Push Protocol (Web Notifications)
<<artifact>> Attendance Logic Service • Attendance Recording (FR5, FR14) • Override Management
<<artifact>> Record Management Service • View/Download Records (FR6, FR11) • Filtering & Export
<<artifact>> Scheduling Service • Class Notifications (FR7, FR8) • Time Management
<<artifact>> Analytics Service (FR9) • Trend Analysis • Reporting • Pattern Recognition
<<artifact>> Course Management Service • Course Registration (FR13) • Lecturer Assignment
<<artifact>> Bluetooth Service (FR15) • Beacon Communication • Proximity Detection
System Requirements
Application Server:
• CPU: 8+ cores, RAM: 16GB+, Storage: SSD 500GB+
• OS: Linux Ubuntu 20.04 LTS or CentOS 8
Database Server:
• CPU: 16+ cores, RAM: 32GB+, Storage: SSD 1TB+
• Backup and replication capabilities
Network Requirements: Min 1Gbps bandwidth
HTTPS/API Trigger notifications
TCP/IP with SSL/TLS Secure database connection
Data Tier
<<server>> Database Server
<<artifact>> User Database (DS1) • User Credentials • Profile Data • Authentication Info
<<artifact>> Facial Data Store (DS2) • Face Templates • Biometric Data • Encrypted Storage
<<artifact>> Geofence Database (DS3) • Location Boundaries • Geographic Definitions
<<artifact>> Attendance Database (DS4) • Attendance Records • Historical Data
<<artifact>> Scheduling Database (DS5) • Class Schedules • Time Management
<<artifact>> Course Database (DS6) • Course Information • Registration Data
Notification Tier
<<server>> Notification Server
<<artifact>> Push Notification Service Software • Firebase Cloud Messaging (FCM) Integration • Apple Push Notification service (APNS) Integration
<<artifact>> Message Queue & Delivery Management • Notification Scheduling • Delivery Tracking

Figure 0-5: Deployment Diagram

10.2 Infrastructure Components

The deployment architecture consists of various infrastructure components that provide the runtime environment:

Deployment Diagram Components:

Nodes (Hardware):

- **User Device (Mobile/Laptop):** Represents student's and educator's smartphones, tablets, or laptops.
- **Application Server:** Hosts the core application logic and APIs.
- **Database Server:** Hosts the databases storing system data.
- **Notification Server:** Manages and sends push notifications.
- **Bluetooth Beacons (Optional Infrastructure):** Physical Bluetooth devices if used for proximity.

Artifacts (Software Components deployed on Nodes):

On User Device:

Mobile Application:

- User Interface
- Camera Interaction Module (FR2, FR3)
- GPS/Location Module (FR4)
- Bluetooth Module (FR15)
- Secure Credential Storage (local aspect of FR1)

On Application Server:

- **Authentication Service:** (Implements FR1)
- **Facial Recognition Service:** (Implements FR2, FR3, FR10) - May involve specialized libraries or AI models.
- **Geofencing Service:** (Implements FR4, FR12)
- **Attendance Logic Service:** (Implements FR5, FR14)
- **Record Management Service:** (Implements FR6, FR11)
- **Scheduling & Notification Service:** (Connects to Notification Server, implements FR7, FR8)

- **Analytics Service:** (Implements FR9)
- **Course Management Service:** (Implements FR13)
- **Bluetooth Interaction Service:** (Implements FR15)
- **API Gateway:** Manages requests to backend services.

On Database Server:

- **User Database Artifact:** (Corresponds to DS1: User Credentials)
- **Facial Data Store Artifact:** (Corresponds to DS2: Facial Data Store)
- **Geofence Database Artifact:** (Corresponds to DS3: Geofence Definitions)
- **Attendance Database Artifact:** (Corresponds to DS4: Attendance Records)
- **Scheduling Database Artifact:** (Corresponds to DS5: Class Schedules)
- **Course Database Artifact:** (Corresponds to DS6: Course Information)

On Notification Server:

- **Push Notification Service Software:** (e.g., Firebase Cloud Messaging (FCM), Apple Push Notification service (APNS) integration)

Communication Paths (Connections between Nodes):

- **User Device to Application Server:** HTTPS (for API calls, data submission like facial data, location, login).
- **Application Server to Database Server:** Secure database connection protocol (e.g., TCP/IP with SSL/TLS).
- **Application Server to Notification Server:** Secure protocol for triggering notifications (e.g., HTTPS API call).
- **Notification Server to User Device:** Platform-specific push notification protocols (FCM/APNS).
- **User Device to Bluetooth Beacons (if applicable):** Bluetooth Low Energy (BLE).

Diagram Structure Description:

1. **Client Tier:**
   - Multiple <<device>> User Device nodes (e.g., "Student Mobile," "Educator Laptop").
   - Inside each User Device node, deploy <<artifact>> Mobile Application or <<artifact>> Web Browser.
   - Show dependencies from the Mobile/Web App to local hardware like Camera, GPS, and Bluetooth module.
2. **Application Tier:**
   - A <<server>> Application Server node.
   - Deploy the various <<artifact>> Service components (Authentication Service, Facial Recognition Service, etc.) and the <<artifact>> API Gateway onto this server.
3. **Data Tier:**
   - A <<server>> Database Server node.
   - Deploy the <<artifact>> Database Artifacts (User DB, Facial Data DB, etc.) onto this server.
4. **Notification Tier (Optional, could be part of App Server or a separate service):**
   - A <<server>> Notification Server node.
   - Deploy <<artifact>> Push Notification Service Software onto it.
5. **Bluetooth Infrastructure (If applicable):**
   - Multiple <<device>> Bluetooth Beacon nodes.
6. **Connections:**
   - Draw communication paths (lines) between these nodes.
   - Label the paths with the communication protocol (e.g., HTTPS, BLE, TCP/IP).
   - For example, a line from User Device to Application Server labeled "HTTPS".
   - A line from Application Server to Database Server labeled "Secure DB Protocol".
   - A line from Application Server to Notification Server labeled "HTTPS/API".
   - A line from Notification Server to User Device labeled "Push Protocol (FCM/APNS)".
   - If beacons are used, a line from Bluetooth Beacon to User Device labeled "BLE".

10.3 Deployment Strategies

The system supports various deployment strategies that address different operational requirements:

Single-Server Deployment: Simplified deployment for development, testing, or small-scale production environments where all components run on a single server.

Multi-Tier Deployment: Distributed deployment across multiple specialized servers that provides better performance, scalability, and fault tolerance.

Cloud Deployment: Deployment to cloud platforms that provides elastic scaling, managed services, and geographic distribution capabilities.

Hybrid Deployment: Combination of on-premises and cloud resources that balances control, security, and scalability requirements.

10.4 Scalability Considerations

The deployment architecture incorporates scalability mechanisms that allow the system to handle varying loads:

Horizontal Scaling: Addition of multiple server instances to distribute load and increase capacity without modifying individual components.

Vertical Scaling: Enhancement of individual server capabilities through increased CPU, memory, or storage resources.

Load Balancing: Distribution of incoming requests across multiple server instances to optimize resource utilization and response times.

Caching Strategies: Implementation of caching at various levels to reduce database load and improve response times.

10.5 High Availability and Disaster Recovery

The deployment design includes provisions for maintaining system availability and recovering from failures:

Redundancy: Multiple instances of critical components to eliminate single points of failure and ensure continued operation during component failures.

Failover Mechanisms: Automatic detection of component failures and redirection of traffic to healthy instances with minimal service disruption.

Backup and Recovery: Regular backup procedures and tested recovery processes that ensure data protection and rapid restoration capabilities.

Monitoring and Alerting: Comprehensive monitoring of system health, performance metrics, and automated alerting for proactive issue resolution.

11. Design Assumptions

The design is based on various assumptions that will be validated during implementation:

User Behavior Assumptions: Expected patterns of user interaction, usage volumes, and functional requirements that influence capacity planning and interface design.

Technology Assumptions: Availability and stability of selected technologies, third-party services, and integration platforms.

Environmental Assumptions: Characteristics of the deployment environment, network infrastructure, and operational procedures.

Business Process Assumptions: Stability of business requirements, organizational processes, and stakeholder priorities.

15. Conclusion

15.1 Design Summary

This Software Design Document provides a comprehensive architectural and design specification for the software system. The design addresses functional requirements through a well-structured architecture that incorporates proven patterns and practices. The modular design approach promotes maintainability, scalability, and flexibility while ensuring that quality attributes are addressed throughout the system.

The design methodology employed systematic analysis and modeling techniques to ensure comprehensive coverage of requirements and thorough validation of design decisions. The resulting architecture provides a solid foundation for implementation while accommodating future enhancement and evolution requirements.

15.2 Implementation Readiness

The design documentation provides sufficient detail to guide implementation activities while maintaining flexibility for detailed design decisions during development. The comprehensive analysis of requirements, constraints, and quality attributes ensures that implementation teams have clear guidance for making consistent decisions.

The modular architecture and well-defined interfaces enable parallel development activities while maintaining system integration integrity. The detailed specifications provide clear contracts between components and establish validation criteria for testing activities.

15.4 Future Considerations

The design incorporates extensibility mechanisms that support future enhancements and evolution. The modular architecture enables incremental improvements and feature additions without destabilizing existing functionality. The comprehensive documentation and standardized approaches facilitate maintenance and knowledge transfer activities.

The scalability characteristics built into the design provide growth capacity for increased user loads and expanded functionality. The deployment architecture supports various operational scenarios and can adapt to changing infrastructure requirements.

The design provides a solid foundation for delivering a high-quality software system that meets business requirements, technical objectives, and user expectations while providing a platform for future growth and enhancement.