REPUBLIQUE DU CAMEROUN

PAIX – TRAVAIL – PATRIE

MINISTERE DE L'ENSEIGNEMENT SUPERIEURE

REPUBLIC OF CAMEROON

PEACE – WORK – FATHERLAND

MINISTRY OF HIGHER EDUCATION

FACULTY OF ENGINEERING AND TECHNOLOGY

PO BOX 63

BUEA, SOUTH WEST REGION

DEPARTMENT OF COMPUTER ENGINEERING

COURSE CODE: CEF440

COURSE TITLE: INTERNET PROGRAMMING AND MOBILE DEVELOPMENT

COURSE INSTRUCTOR: Dr. NKEMENI VALERY

TASK 3 REPORT: Software Requirements and Specifications (SRS) Document for the Mobile-Based Attendance Management System Based on Geofencing and Facial Recognition

Presented by:

NAMES	MATRICLE
1. MEKOLLE ASHLEY ARREYMANYOR	**FE22A247**
2. AKO RUTH ACHERE	**FE22A144**
3. NGUIMENANG ZEUFACK KEREINE	**FE22A266**
4. TUMUTOH LYDIE-KASEY BIHNUI	**FE22A321**
5. TIWA DELAN NDIKA	**FE22A319**

MAY 2025

Software Requirements and Specification (SRS) Document

Project Name: MOBILE-BASED ATTENDANCE MANAGEMENT SYSTEM BASED ON GEOFENCING AND FACIAL RECOGNITION
Date: 03-05-2025

Version	1.0	**Status**	~~Validated~~

Prepared By	Group 12	**Role**	In the Task3 distribution document

Version History:

Version	**Primary Authors**	**Description of Version**	**Completion Date**

Review History and Approval:

Approving Party	**Version Approved**	**Signature**	**Updated Date**

Requirements Document Approval History

Reviewer	Version Reviewed	Signature	Updated Date

1. INTRODUCTION 5
  1.1. Purpose 5
  1.2. Project Overview 5
  1.3. Document Conventions 6
  1.4. Intended Audience and Reading Suggestions 7
  1.5. Product Scope 9
2. Overall Description 10
  2.1. Background 10
  2.2. Problem Statement 10
  2.3. Project Objectives 10
  2.4. System Interfaces 12
  2.5. Subsystem Interconnection 12
  2.6. Product Features 13
  2.7. Operating Environment 15
3. Design and Implementation Constraints 17
  3.1. Design Constraints 17
  3.2. Implementation 18
4. Project strategy 21
  4.1. Project Work Strategy 21
  4.2. Project Milestone 21
5. Technology Task To use 24
  5.1. Mobile Application Development 24
  5.2. Backend Development 24
  5.3. Database Management 25
6. System Requirements 25
  6.1. User Registration and Authentication 25
  6.2. Facial Recognition for Attendance 26
  6.3. Geofencing Location Verification 27
  6.4. Attendance Recording 28

6.5. Instructor Attendance Monitoring 29
6.6. Admin Management Panel 29
6.7. Notification System 29
6.8. Real-Time Attendance Monitoring 30
6.9. Error Dispute Mechanism 30
6.10. Role-Based Access Control (RBAC) 31
6.11. Students-Assisted Attendance Logging by Lecturers 32
7. Non-Functional Requirements 33
7.1. Performance Requirements 33
7.2. Scalability Requirements 34
7.3. Reliability Requirements 34
7.4. Usability Requirements 34
7.5. Security Requirements 34
7.6. Maintainability Requirements 35
7.7. Portability Requirements 35
7.8. Cost Efficiency Requirements 35
7.9. Recovery Requirements 35
7.10. Data Retention Requirements 36
7.11. Auditability Requirements 36
7.12. Fault Tolerance Requirements 36
8. Technical Constraints and Feasibility 37
9. Future Work 39
10. Conclusion 39

1. INTRODUCTION

1.1. Purpose

This Software Requirements Specification (SRS) document provides a detailed description of the functional and non-functional requirements for the Mobile-Based Attendance Management System based on Geofencing and Facial Recognition technologies.

The purpose of this document is to outline what the system is intended to achieve, how it should behave, and the expectations from both a user and system perspective. It will serve as a foundation for the design, development, and testing of the application, ensuring that all stakeholders have a common and clear understanding of the product's capabilities and constraints.

It will also help:

- Developers to understand what they need to build
- Designers to know how the user interface should behave
- Testers to create tests that ensure all functionalities work correctly and efficiently
- Project manager (Group Leader) to track progress and scope

The final validated version of this SRS will be referred to throughout the project lifecycle to make sure that the final product meets the intended requirements.

Product Identification

- **Product Name**: Mobile-Based Attendance Management System Based on Geofencing and Facial Recognition
- **Version**: 1.0
- **Date**: 03 May 2025
- **Prepared By**: Group 12
- **Approved By**: Dr. NKEMENI VALERY (CEF440 Course Instructor)
- **Document Status**: ~~Validated~~

1.2. Project Overview

The Mobile-Based Attendance Management System (MBAMS) aims to automate the student attendance process in educational institutions by utilizing facial recognition for identity verification and geofencing to confirm a student's physical presence within the classroom boundary. The application will be deployed on Android mobile devices initially.

The goal is to minimize errors, prevent proxy attendance, reduce processing time, and provide real-time access to attendance data records.

1.3. Document Conventions

This document adheres to specific standards and typographical conventions to ensure consistency, readability, and clarity throughout the Software Requirements Specification (SRS).

i. **Formatting and Style:**

- **Font Family**:
  - The primary font used in this document is **Times New Roman** for all standard text.
- **Font Sizes**:
  - **Document Title**: 20pt, Bold, Center-aligned.
  - **Heading 1 (H1)** (Major Sections): 20pt, Bold.
  - **Heading 2 (H2)** (Subsections): 16pt, Bold.
  - **Heading 3 (H3)** (Sub-subsections): 14pt, Bold.
  - **Heading 4 (H4)** (Minor headings): 12pt, Bold.
  - **Body Text**: 12pt, Regular.
- **Emphasis**:
  - Key terms, definitions, and important points are presented in **bold**.
  - Italics may be used sparingly to emphasize specific notes or warnings.
- **Spacing**:
  - All body text and headings are formatted with **1.5 line spacing**.
  - Paragraphs are separated by a consistent space to ensure a clean and easy-to-read layout.
- **Alignment**:
  - Paragraphs are **justified**, ensuring neat left and right edges for improved readability and a professional appearance.
- **Margins**:
  - Standard margins of **1 inch (2.54 cm)** on all sides are maintained throughout the document.

ii. **Requirement Priority Notation**

The following notation is used to prioritize the importance of requirements:

- **High**: Essential for system functionality; must be implemented in the initial release.
- **Medium** Important but not critical; can be implemented in subsequent updates if time allows
- **Low**: Nice-to-have features that enhance user experience, performance and usability but are not necessary for core operations.

1.4. Intended Audience and Reading Suggestions

1.4.1. Intended Audience

This Software Requirements Specification (SRS) document is created for a variety of individuals and groups who are involved in or impacted by the development, deployment, and maintenance of the Mobile-Based Attendance Management System based on Geofencing and Facial Recognition.

Each group of stakeholders plays a unique and important role, and specific parts of this document will be most relevant to them. Below is a detailed explanation:

1. Project Team Members (Developers, Designers, and Engineers)

- **Who they are**: They are technical professionals responsible for designing, coding, testing, and deploying the application.
- **Why they need this document**: They will use this SRS to understand exactly what the system should do (functional requirements), how it should behave (non-functional requirements), and any constraints they must work within (e.g., device compatibility, performance expectations).

2. University Administrators

- **Who they are**: Administrative staffs and lecturers who will use the application to manage and monitor student attendance.
- **Why they need this document**: They need to verify that the system's functionalities meet their operational needs (such as real-time attendance tracking, filtering, and viewing history). They also ensure the system supports the academic policies regarding attendance.

3. End Users (Students and Lecturers)

- **Who they are**: The individuals (students attending classes and lecturers managing courses and student's attendance) who interact directly with the mobile application daily.
- **Why they need this document**: To understand the capabilities of the system from a user experience perspective, what actions they can perform, how attendance is captured, and how their privacy and data are handled.

4. Course Instructor

- **Who he is**: A Professional who oversees the system's analysis, ensure the requirements are correctly understood, and validate the alignment of project outcomes with initial goals.
- **Why he needs this document**: He is responsible for reviewing the document for completeness, consistency, and technical feasibility and ensuring it matches the project goals.

1.4.2. Reading Suggestion

To help different readers effectively navigate and understand this Software Requirements Specification document, the following reading guidelines are recommended:

1. **For Developers and Technical Team Members:**

- Focus first on the Functional Requirements and System Features sections to understand what exactly needs to be implemented.
- Carefully review the Non-Functional Requirements such as system performance, security, maintainability, and usability expectations.
- Pay attention to Technical Constraints and Assumptions which may impact development choices, such as device compatibility (Android) and third-party services (GPS, Face Recognition APIs).
- The System Overview should be reviewed early to grasp the big picture of the solution before diving into technical details.

2. For Project Managers, and Course Instructor:

- Start with the Introduction and Purpose sections to understand the goals of the system.
- Review the Requirement Prioritization to understand which features are critical, and the Validation Criteria to assess how requirement fulfillment will be evaluated.
- Focus also on the Dependency Relationships and Risk Assessment areas to identify possible project risks and required risk mitigation strategies.

3. For University Administrators and Stakeholders (Decision Makers):

- Begin by reading the Overall Description and Scope sections to understand what the system will offer and its operational boundaries.
- Look into the User Interface and Usability Requirements to ensure that the system aligns with the institution's policies and is easy for staff and students to use.

4. For End Users (Students and Lecturers):

- Focus primarily on the User Requirements and User Interface Design sections to get a clear picture of how they will interact with the system.
- Review the Attendance Check-In Procedures, Security and Privacy Requirements, and Attendance History Access.

1.5. Product Scope

This project aims to develop a mobile application for academic institutions to manage student attendance through biometric facial recognition and geofencing. The system will provide role-based access to students, lecturers, and administrators and include real-time data processing, robust notification systems, and analytics for engagement monitoring. Below is a scope of what the system must do and behave like.

Functional Scope

- The system must ensure secure login and registration using credentials and facial recognition (for students and Lecturers).
- The system must perform real-time attendance check-in using facial recognition and geofencing to verify identity and presence in class respectively.
- The system must ensure that Lecturers can change session status (such as ongoing, postponed, and completed), triggering attendance windows.
- The system must provide tailored interfaces for students, lecturers, and admins showing attendance data and tools, ensuring efficient User Experience.
- The system must send notifications for reminders, confirmations, and absence alerts of attendance and courses to keep users informed and engaged.
- The system must ensure that Students can report attendance errors to lecturers for a specific course such lecturers or admins would review and resolve ensuring direct access to lecturers and admins.
- The system must ensure that access to different features is tailored based on user's roles and privileges.

Non-Functional Scope

- The system must ensure 99% uptime during academic hours.
- The system must ensure Fast facial recognition and accurate geolocation, and support a high check-in volume of at least ten (10) attendance check-in per second.
- The system must ensure End-to-end encryption of user data and ensure legal compliance for biometric data.
- It would be built on Android devices first before deployment and would be later built too scale iOS devices.
- The system must have a modular design for easy updates and system growth.
- The system should Logs all key actions for traceability.
- The system must have an Intuitive UI and would later support for users with disabilities such as mutes, deafs, visually impaired, and others.
- The system should be built with open-source tools to reduce expenses.

2. Overall Description

2.1. Background

Attendance management is a fundamental administrative activity in educational institutions. Traditionally, attendance is recorded manually using paper sheets or semi-digital methods like RFID cards or Excel spreadsheets. These approaches are prone to human error, manipulation (proxy attendance), time wastage, and difficulties in real-time tracking.
As mobile technology becomes more powerful and accessible, it offers an opportunity to automate attendance processes. Technologies like **geofencing** (location-based verification) and **facial recognition** (biometric identification) can drastically improve accuracy, reliability, and efficiency. By leveraging mobile devices, GPS, cameras, and cloud backend systems, this system intends to modernize and secure the attendance recording process, thus promoting transparency and reducing administrative workload.

2.2. Problem Statement

Traditional attendance methods in universities and schools face multiple challenges such as:

- **Manual Errors:** Mistakes often occur when recording or transferring data.
- **Proxy Attendance:** Students can sign in on behalf of their peers.
- **Time-Consuming:** Calling roll or handling sign-in sheets wastes valuable lecture time.
- **Delayed Reports:** Processing attendance data and generating reports is slow.
- **Lack of Real-Time Verification:** Administrators cannot verify whether a student was truly present at the time of class.

There is a pressing need for a secure, efficient, real-time attendance management solution that:

- Verifies both physical location and identity.
- Minimizes manual intervention.
- Enables quick reporting and tracking.

2.3. Project Objectives

2.3.1. General Objective

The aim of this project is to ensure that the Group 12 students taking the course CEF440 to whom this project |***Design and Implementation of a Mobile-based Attendance Management system based on Geofencing and Facial Recognition***| is assigned to should learn to design and develop a secure, efficient, and user-friendly **mobile** attendance management system that automates and validates student attendance using geofencing and facial recognition technologies, thereby eliminating manual errors, reducing impersonation, and ensuring accountability and transparency in academic attendance monitoring.

2.3.2. Specific Objectives

- **To eliminate impersonation and proxy attendance through facial recognition technology.**
  - The system will use live face scanning with AI-based verification to confirm student identity in real time.
- **To ensure students are physically present in class using geofencing.**
  - The system will validate student check-ins based on their proximity to the correct geofence venue generated by the Administrator and saved to the Database.
- **To automate attendance tracking and real-time reporting.**
  - Attendance data will be collected and processed, verified, and logged automatically, reducing errors and administrative overhead.
- **To implement a role-based user interface for students, lecturers, and administrators.**
  - Each user type will have tailored access and functionality to streamline attendance workflows and minimize complexity.
- **To provide real-time notifications and reminders to enhance participation.**
  - Students will receive alerts for upcoming classes, check-in confirmations, and missed sessions to improve class punctuality, while Lecturers and school Admins would receive alerts of student's complaints and system technical errors recorded.
- **To incorporate a transparent dispute resolution process for attendance records.**
  - Students will be able to submit formal disputes with supporting evidence for absences or incorrect logs, and lecturers/admins can review and respond accordingly.
- **To generate insightful analytics and reports for monitoring attendance trends.**
  - Dashboards will visualize attendance statistics using charts, icons, and color codes for easy interpretation by all stakeholders.
- **To ensure system security, data privacy, and compliance with legal standards.**
  - All user data, including biometrics, will be encrypted and handled in line with data protection laws, ensuring user consent and confidentiality.
- **To design the system for scalability and extensibility.**
  - The architecture will support high concurrency, modular updates, and future integration with academic systems.

- **To offer accessibility features for inclusive use by all students.**
    - The system will include visual cues, simple interfaces, and consideration for users with disabilities in future iterations.
- **Leadership, communication and Technical Understanding**
    - Members of Group 12 will learn how to build mobile applications and they would go through a series of class presentations building their skills in effective communication.

2.4. System Interfaces

The Mobile-Based Attendance System will interact with several key components:

- **Mobile App Interface**: Students and teachers use a mobile app to check-in and monitor attendance.
- **GPS Interface**: The app uses device location to confirm students are within the allowed school/class area.
- **Camera Interface**: Students take a live selfie to verify their identity using facial recognition.
- **Authentication Interface**: Users log in securely to the app through a basic authentication system.
- **Database Interface**: The app connects to a cloud database to store attendance records, user information, and face data.

2.5. Subsystem Interconnection

Subsystem	Responsibilities	Interconnection
Student Mobile App	▪ Capture GPS location. ▪ Perform face scan. ▪ Submit attendance request.	▪ Connects to GPS API for location validation. ▪ Connects to Camera API for face recognition. ▪ Sends verified check-in request to Backend Server.
Instructor Mobile App Module	▪ Monitor live attendance. ▪ Approve/decline attendance if needed. ▪ Generate daily reports.	▪ Communicates with Backend Server to fetch real-time attendance data. ▪ Displays student check-in information.
Geofencing Module	▪ Create virtual boundaries around classrooms. ▪ Monitor entry/exit within geofenced area.	▪ Accesses GPS coordinates from the mobile app. ▪ Sends "allowed" or "denied" check-in signal

		based on location validation.
Facial Recognition Engine	▪ Capture and compare face biometric data. ▪ Authenticate student identity	▪ Embedded inside the mobile app. ▪ Captures live face and matches against stored templates fetched from Database.
Backend Server (APIs)	▪ Validate check-in data. ▪ Manage business logic. ▪ Process authentication and verification requests.	▪ Receives and processes requests from Mobile Apps. ▪ Communicates with Database for storage and retrieval.
Cloud Database	▪ Store all data: user profiles, face biometrics, attendance records, course data.	▪ Centralized repository. ▪ Provides data to both Mobile Apps and Admin Dashboard
Admin Dashboard	▪ Manage system users. ▪ View, filter, and generate attendance reports. ▪ Monitor system logs.	▪ Fetches data through APIs exposed by Backend Server. ▪ Provides a graphical UI for easier administration.

2.6. Product Features

1. Real-time Attendance Check-in

- ➢ Students can check-in using the app immediately after entering the classroom or any time within the course hours.
- ➢ Timestamp is recorded automatically with the check-in event.
- ➢ Allows instructors to monitor attendance live without waiting for end-of-class reports.

2. Facial Recognition Verification

- ➢ Uses a built-in live detection system to prevent use of photos/screenshots for face verification (anti-spoofing).
- ➢ Face templates are securely processed and encrypted before storage.
- ➢ Repeated failed face recognition attempts alert the system admin for review.
- ➢ Students may update face data securely if major changes happen (e.g., injury, glasses).

3. Geofencing-Based Location Validation

- Virtual boundaries (geofences) are created using latitude, longitude, altitude, and radius.
- Students can only check-in if they are physically within the boundary and the altitude.
- Accuracy threshold can be adjusted (e.g., 10 meters, 20 meters) depending on classroom size.
- Prevents remote check-ins from homes or outside the campus.

4. Fast and Efficient Process

- Modular and optimized code ensures quick loading of the facial recognition model and GPS reading.
- Minimal steps required: open app → check-in → complete.
- Average check-in time goal is **less than 5 seconds** under good network conditions.
- Reduces crowding or delays at the beginning of class.

5. Secure Authentication

- Students and instructors must first register with verified email or student ID.
- Passwords are stored using strong hashing techniques (e.g., bcrypt).
- May optionally implement Two-Factor Authentication (2FA) for higher security.
- Sessions automatically expire after a period of inactivity to prevent unauthorized access.

6. Attendance History for Students

- Students can access a clear calendar view or list view of attendance.
- Shows details like check-in time, geolocation verification status, and face match status.
- Alerts students if they are nearing attendance shortfall (e.g., less than 25% required attendance).
- Encourages students to be more responsible for their attendance.

7. Instructor Dashboard

- Instructors can monitor attendance in real-time as students check-in.
- They can generate attendance summaries by course, session, week, month or end of year semester.
- Color-coded charts or graphs show attendance trends over time (GREEN for presence and RED for absence).
- Can communicate to System Administrator to manually override student's attendance for exceptional cases (e.g., medical leave, family emergency).

8. Face Data and Attendance Data Storage

- All sensitive data (faces, attendance, credentials) is encrypted both during transfer (SSL/TLS) and at rest (in the database).
- Compliance with data privacy regulations (e.g., GDPR if applicable).

- Regular backups to prevent data loss.
- Database optimized for quick read/write access even under high loads.

9. Admin Management Panel

- Admins can add or remove users, update course schedules, adjust classroom geofence zones.
- System settings (e.g., face match sensitivity, location accuracy) can be modified.
- Logs of all administrative actions are recorded for auditing and transparency.
- Role-based access control ensures that only admins have critical privileges.

10. Push Notifications (Optional / Bonus Feature)

- Students receive reminders about upcoming classes and attendance deadlines.
- Instructors can send quick notices about class changes (e.g., venue change, cancelations etc).
- Notifications can be customized based on student preferences (email, in-app, SMS).

2.7. Operating Environment

2.7.1. System Platforms

1. **Hardware Requirements**
   - **Mobile Devices**:
     - Android smartphones (minimum Android 10.0 and above).
     - iOS devices (minimum iOS 8 and above) for future development.
     - Front-facing camera (minimum 5MP recommended for better facial recognition).
     - GPS functionality enabled for geofencing accuracy.
   - **Server Hardware**:
     - Minimum 4 vCPUs, 8 GB RAM (for a medium user base: ~500+ concurrent users).
     - SSD storage (minimum 100GB for fast read/write operations).
     - Reliable internet connection with high uptime (e.g., 99.9% availability).
     - Backup power supply (UPS or Generator) to maintain server uptime.
2. **Software Requirements**
   - **Mobile Application**:
     - Developed using **React Native** (for Android and iOS cross-platform compatibility).
     - Facial recognition library

  - Google Play Services (for location-based services on Android).

- **Backend Server**:
  - Operating System: Ubuntu Server 22.04 LTS or higher (recommended for security and stability).
  - Web Server: Nginx or Apache.
  - Backend Framework: Node.js (Express)
  - Database: PostgreSQL (recommended for scalability) or MySQL.
  - Cloud Services (optional): AWS EC2, Google Cloud, or Render for hosting backend and database.

3. **Server Requirements**
   - SSL/TLS Certificate for secure communication (HTTPS).
   - Firewall configuration to restrict unauthorized access.
   - Regular automatic backups scheduled (daily incremental and weekly full backups).
   - Scalability support if user base grows (load balancers, auto-scaling groups).

2.7.2. Limitations

- **Hardware Dependency**: Poor camera quality or old GPS hardware can affect accuracy of facial recognition and location verification.
- **Internet Connectivity**: Real-time attendance submission requires active internet connection (WiFi or mobile data). Offline mode may not be supported initially.
- **Mobile OS Permissions**: If users deny access to camera or location services, check-in will fail.
- **Environmental Conditions**: Poor lighting can interfere with face detection.
- **Device Compatibility**: The app may not perform optimally on very old mobile devices (low RAM, outdated OS versions).

2.7.3. Constraints

- **Security Compliance**: Biometric and location data must be encrypted and stored securely, following applicable data protection laws (e.g., local regulations).
- **Authentication Requirement**: Only registered and verified users (students and staff) will be able to access the application.
- **Scalability Constraint**: The initial deployment will target small to medium amount of users (500-1000 users). Further optimization will be needed for university-wide deployment.
- **Maintenance and Updates**: Regular updates will be required to maintain facial recognition accuracy and adapt to new mobile OS versions.

3. Design and Implementation Constraints

3.1. Design Constraints

The UI of the Mobile-Based Attendance Management System will be designed with a focus on simplicity, user-friendliness, and accessibility for students, instructors, and administrators. The UI will follow mobile-first design principles to ensure seamless usage across Android and iOS devices.

3.1.1. Design Principles

- **Simplicity**: Minimalist layout with only essential elements displayed per screen.
- **Consistency**: Consistent color scheme, button styles, fonts, and navigation flow.
- **Accessibility**: Text readability, large buttons, and intuitive icons for easier interaction.
- **Responsiveness**: Adaptable UI for various device screen sizes.

3.1.2. Primary UI Components

Screen	Description
Login/Registration Screen	Allows users to sign up (with face registration) and login securely.
Home Dashboard	Displays quick options based on user role (Attendance Check-in, View Records, notification, profile/settings etc.).
Attendance Check-in	Displays GPS location status and triggers facial recognition before marking attendance.
Attendance Records	▪ Students can view their own attendance ▪ Instructors can view attendance of the students taking their courses.
Admin Panel	Admins can add students/instructors, configure geofencing parameters, monitor system activity.
Settings/Profile	Users can update limited profile settings (e.g., profile picture, password).
Notifications	System alerts about successful check-ins, missed attendances, or system updates.

3.1.3. Style Guide

- **Color Scheme**:
  - Primary Color: BLUE which signifies Trust and clarity for an educational background.
  - Secondary Color: WHITE for Backgrounds.
  - Success Color: GREEN for Successful check-ins and alerts.
  - Warning Color: RED for Error messages, failed check-ins.
- **Typography**:
  - Font Family:
  - Font Sizes:
    - Titles: 20pt
    - Headings: 14–18pt
    - Body Text: 12–14pt
- **Icons and Buttons**:
  - Clear and recognizable icons (e.g., check-in, map pin for location).
  - Primary action buttons are filled, secondary actions are outlined.

3.1.4. Wireframes

Wireframes will be created using tools like **Figma** to visualize:

- Screen layouts
- Navigation flow
- Key interaction points (camera activation for facial capture, GPS map display for geofencing)

3.2. Implementation

3.2.1. Technologies and Tools

1. **Mobile Development**
   - **Framework**: React Native (Version 0.73 or higher)
     - Cross-platform development (Android and iOS).
     - Reusable UI components, faster updates using Hot Reload.

- Access to device features (camera, GPS) via React Native libraries (e.g., react-native-camera, react-native-geolocation-service).

2. **Backend Development**
   - **Runtime Environment**: Node.js (Latest LTS Version)
     - Fast, scalable backend services.
     - Efficient handling of asynchronous operations (important for real-time attendance logging).
   - **Framework**: Express.js (Version 4.18+)
     - Lightweight, minimalistic server-side framework.
     - Simplifies API development for mobile communication.
3. **Database**
   - **Database Management System**: PostgreSQL (Version 15+)
     - Relational database, ideal for structured attendance records.
     - Support for geospatial data types (PostGIS) — helpful for location-based queries (geofencing validation).
   - **Database Tools**:
     - pgAdmin 4 (for database management).
     - Prisma ORM for easier integration with Node.js.
4. **Version Control**
   - **Tool**: Git
   - **Repository Hosting**: GitHub
     - Branching strategy will be used (e.g., "main" for production and for development, and names for each group member's branch).
     - Code reviews and pull requests required before merging.

3.2.2. Development Methodologies and Standards

1. **Agile Methodology**:
   - Development will be iterative and incremental.
   - Weekly sprints and stand-up meetings will be held to track progress.
   - Continuous requirement gathering/feedback from stakeholders.
2. **Coding Standards**:
   - JavaScript/TypeScript style guides (Airbnb style guide).
   - Proper documentation for APIs (Swagger).

- Unit tests (e.g., Jest for Node.js and React Native).
- Mobile UI components will adhere to platform-specific design guidelines (Material Design for Android, Human Interface Guidelines for iOS).

3. **Security Standards**:
   - JWT (JSON Web Tokens) for secure user authentication.
   - Passwords and sensitive data will be hashed using bcrypt.
   - Secure APIs with HTTPS, CORS policies, and input validation.

3.2.3. User Roles and Permission

3.2.4. Standard Data Exchange Formats

1. **Data Formats**:
   - **JSON** (JavaScript Object Notation) will be used for all data exchanges between client and server (APIs).
   - **Multipart/Form-Data**: For file uploads (like capturing facial images during registration).

2. **API Communication**:
   - RESTful API standards (HTTP methods: GET, POST, PUT, DELETE).
   - Proper status codes and error messages (e.g., 200 OK, 400 Bad Request, 401 Unauthorized).

3. **Data Encryption**:
   - Sensitive information (e.g., biometric data) will be encrypted in transit (HTTPS) and at rest (database encryption techniques).

4. Project strategy

The **Project Strategy** defines the overall approach for planning, developing, managing, and delivering the Mobile-Based Attendance Management System. It ensures that the project is executed in a structured, efficient, and predictable way, minimizing risks and delays.

4.1. Project Work Strategy

The project will follow an **Agile Development Methodology**, particularly the **Scrum framework**. This is chosen because:

- ➢ Requirements might evolve based on stakeholder feedback.
- ➢ It allows for incremental delivery of functional components.
- ➢ It promotes continuous testing, user feedback, and improvements at every stage.

Each sprint will last **1 weeks**, and at the end of each sprint, a working component or feature will be demonstrated.
The project phases include:

1. **Requirement Analysis**: Understanding user needs and technical feasibility.
2. **System Design**: Designing system architecture, database schemas, UI mockups.
3. **Development**: Coding frontend, backend, integrating machine learning models, GPS/geofencing.
4. **Testing**: Conducting unit tests, integration tests, performance tests.
5. **Deployment**: Launching the system on the cloud/mobile platforms.
6. **Maintenance**: Fixing bugs and updating the system based on user feedback.

4.2. Project Milestone

Milestone	**Tasks Description**	**Expected Completion Time**
1. Mobile App Development Process Research	▪ Types of Mobile applications ▪ Mobile apps programming languages ▪ Mobile app development frameworks ▪ Mobile app architectures and design patterns ▪ Requirement engineering process ▪ Mobile app development cost estimation	One Week
2. Project Initialization	▪ Form the project team (developers, designers, testers). ▪ Assign roles (Project Manager, Lead Developer, Lead Designer, Team Members).	One week

	▪ Set up communication tools (WhatsApp group, GitHub). ▪ Define high-level goals, deliverables, and initial timelines.	
3. Requirements Gathering & Analysis	▪ Stakeholder identification ▪ Requirement gathering techniques (surveys, interviews, brainstorming, reverse engineering, etc) ▪ Data gathering ▪ Data cleaning ▪ User reluctance assessment	
4. Requirement Analysis	▪ Review and analyze the requirements gathered (Completeness, clarity, technical feasibility, dependency relationships) ▪ Identify inconsistencies, ambiguities, and missing information ▪ Prioritize requirements based on importance and feasibility ▪ Classify requirements (functional and non-functional) ▪ Develop the software requirement specification (SRS) ▪ Validate Requirements with Stakeholders	
5. System Modelling and Design	▪ Context Diagram ▪ Dataflow Diagrams ▪ Use Case Diagram ▪ Sequence Diagrams ▪ Activity Diagram ▪ Class Diagrams ▪ Deployment Diagram	
6. UI Design	▪ App Identity ▪ Visual design ▪ High Fidelity Prototype.	
7. Environment Setup for Frontend and Implementation	▪ Set up development environment for React Native (VS Code, Android studio). ▪ Build student login and registration pages. ▪ Implement dashboard for different pages. ▪ Create check-in button linked to face recognition and GPS. ▪ Build admin interfaces for instructors to monitor students.	
8. Database design and implementation	▪ Data Elements ▪ Conceptual Design ▪ ER Diagram ▪ Database implementation	

9. Environment Setup for Backend	▪ Set up GitHub repositories for version control. ▪ Install backend frameworks and libraries (Express.js, Sequelize ORM). ▪ Connecting database to backend ▪ Establish continuous integration pipeline (GitHub Actions).	
10. Backend Development (Node.js API)	▪ Build RESTful APIs for user authentication (login, signup, logout etc). ▪ Create APIs for attendance records (mark attendance, retrieve attendance list). ▪ Integrate geolocation service (GPS coordinate validation). ▪ Secure APIs with JWT (JSON Web Tokens) authentication and session configuration. ▪ Build Admin endpoints (manage all student attendance and Instructors Activities). ▪ Integrating APIs with Frontend	
11. Facial Recognition Integration	▪ Train a simple machine learning face model or use internal APIs (e.g., React Native facial recognition, OpenCV with face recognition). ▪ Integrate camera functionality into the app. ▪ Verify face matching before allowing attendance submission. ▪ Optimize model to work within 5 seconds (as per system requirement).	
12. Geofencing Module Integration	▪ Set up classroom geofence coordinates. ▪ Integrate geolocation services in the mobile app. ▪ Validate student location before allowing check-in. ▪ Alert users if outside geofence boundaries. ▪ Combine GPS and Face Recognition for final attendance confirmation.	
13. Testing Phase	▪ Perform Unit Testing (test individual modules such as login, GPS location fetch, face scan). ▪ Perform Integration Testing (test how backend and frontend work together). ▪ Perform System Testing (end-to-end testing). ▪ Perform Security Testing (ensure only authorized users can access APIs).	
14. User Acceptance Testing (UAT)	▪ Select real users (students and instructors) to test the system. ▪ Collect feedback on user-friendliness, speed, and accuracy. ▪ Document reported bugs or issues.	

15. Documentation and Final Report Submission	▪ Prepare a **User Manual** (how to use the app for students and instructors). ▪ Write **Technical Documentation** (API documentation, architecture diagrams, database structure). ▪ Submit project reports.	

5. Technology Task To use

This project will utilize a modern and scalable technology stack to ensure high performance, security, and ease of maintenance for the Mobile-Based Attendance Management System based on Geofencing and Facial Recognition. Below are the selected technologies, tools, databases, frameworks, and methodologies to be used throughout the development lifecycle.

5.1. Mobile Application Development

5.1.1. Framework: React Native

- React Native enables the development of a single mobile application for both Android and iOS platforms, reducing development time and maintenance costs.
- It offers native performance and integrates well with device hardware features such as GPS for geofencing and Camera APIs for facial recognition.

5.1.2. Supporting Libraries

- **Facial Recognition**:
    - ***face-api.js*** for on-device facial detection and recognition, providing a lightweight solution.
    - Alternatively, we can Train a simple machine learning face model using python OpenCV.
- **Geofencing and Location Services**:
    - ***react-native-background-geolocation*** and ***react-native-maps*** will be used for tracking and managing user locations with high precision.

5.2. Backend Development

5.2.1. Framework: Node.js with Express.js

- Node.js offers an asynchronous, event-driven architecture ideal for building scalable and real-time APIs.
- Express.js provides a simple yet powerful framework for defining server routes, handling authentication, and managing database communication.

5.2.2. Authentication

- ***JWT (JSON Web Tokens)*** will be used for secure user authentication, ensuring that sessions are stateless and scalable.

5.3. Database Management

5.3.1. Database: PostgreSQL

- A highly reliable, secure, and feature-rich relational database will be used to manage all system data including users, attendance logs, facial templates, and geolocation records.
- The ***PostGIS*** extension will be enabled to perform advanced spatial queries needed for geofencing operations.

5.4. Version Control

- Tool: Git with remote repositories on GitHub
    - All source code, documentation, and configuration files will be version-controlled to ensure proper collaboration, code review, and rollback capabilities.

6. System Requirements

6.1. User Registration and Authentication

Description: Before accessing the system, users (students, instructors, admins) must create an account and log in. The system ensures that only authorized individuals can use attendance services.

Rank: High

Explanation: Authentication is foundation for any system. Student sign-up and login with biometric verification ensure secure and accurate identity confirmation, which is critical for attendance tracking. Without this, the system cannot function reliably.

6.1.1. User Actions and System Responses:

- **Action 1:** User opens the app and clicks "Sign Up."
- **System Response:** The app displays a registration form requesting email, password, full name, and user role (student or instructor).

  - **Action 2:** User fills the registration form and submits containing details which are different for Students, Lecturers and Administrators.
  - **System Response:**
    - The system validates email format, password strength, and ensures the email is not already registered.
    - If validation passes, user data is stored in the database securely.
    - System sends a confirmation notification.
- **Action 3:** User returns and clicks "Login."
- **System Response:**
  - Login form is displayed asking for email and password.
  - On submission, system checks credentials.
  - If correct, JWT token is generated and user is authenticated.
  - User is redirected to their respective dashboard (Student or Instructor dashboard).
- **Action 4:** Forgot Password.
- **System Response:**
  - User inputs email address.
  - System sends a password reset link to registered email.

6.1.2. Functional Requirements

- **FR1**: The system shall allow users (students, lecturers, admins) to register securely via email or institution credentials.
- **FR2**: The system shall implement authentication using encrypted password storage.
- **FR3**: The system shall allow users to log in with registered credentials.
- **FR4**: The system shall validate user input fields during registration and login.
- **FR5**: The system shall allow password reset using email verification.

6.2. Facial Recognition for Attendance

Rank: High

Explanation: This is the core feature for verifying student attendance. To prevent fraudulent attendance, each student must verify their identity using facial recognition. This ensures that only the actual enrolled student can mark themselves present, making it indispensable for the system's primary purpose.

6.2.1. User Actions and System Responses:

- **Action 1:** Student opens "Attendance" and clicks "Scan Face."
- **System Response:**
  - App triggers the device's front camera.
  - System asks the user to move their face within the frame verifying physical human.

- **Action 2:** User aligns their face and clicks "Capture."
- **System Response:**
  - Captured facial image is temporarily stored.
  - System uses facial recognition algorithm to process the facial template and compares the current facial template with the one on the Database saved during signup
  - If a match is found, system proceeds to next step (Geofencing check).
  - If match fails, system denies attendance and shows “Face Verification Failed” message.
- **Action 3:** If successful, System displays message: “Face verified successfully! Proceeding to GPS verification.”

6.2.2. Functional Requirements

- **FR1**: The system shall require users to scan their faces for identity verification during attendance check-in.
- **FR2**: The system shall match the captured facial data against stored facial profiles in the database.
- **FR3**: The system shall allow attendance marking only upon successful facial verification.
- **FR4**: The system shall notify users if facial verification fails and provide a retry option.
- **FR5**: The system shall store facial recognition logs for verification audits.

6.3. Geofencing Location Verification

Rank: High

Explanation: Geofencing is a core requirement for the system, as it ensures that students can only mark attendance when they are physically present in the authorized class location. Without geofencing, the system cannot reliably prevent fraudulent check-ins from remote locations, undermining the integrity of attendance tracking. While implementation may require additional research (e.g., defining accurate boundaries, handling dynamic location changes), it is a must-have feature for the system to function as intended.

6.3.1. User Actions and System Responses:

- **Action 1:** After face verification, user clicks "Verify GPS."
- **System Response:**
  - App requests permission to access GPS location always while using the app.
  - System captures device latitude and longitude in real-time.
- **Action 2:** User stays within the designated classroom zone.
- **System Response:**
  - If within allowed radius (e.g., 50 meters of class location), and within the altitude of the class (depending on the configurations of the venue on the database), system allows attendance submission.

    - Message displayed: “Location Verified. You can submit your attendance.”
- **Action 3:** User tries outside classroom zone.
- **System Response:**
    - System detects user is outside the geofenced area.
    - Message displayed: “You are not in the attendance zone. Cannot mark attendance.”

6.3.2. Functional Requirements

- **FR1**: The system shall capture the user's location when attempting to check-in.
- **FR2**: The system shall compare the user's current location against the predefined geofence for the classroom.
- **FR3**: The system shall allow attendance submission only if the user is within the approved geofence radius.
- **FR4**: The system shall deny attendance submission attempts made outside the geofence and notify the user.
- **FR5**: The system shall log geolocation metadata for every attendance event

6.4. Attendance Recording

Rank: High

Explanation: Immediate recording and validation of attendance, along with notifications, enhance user trust and engagement. This is critical for a seamless experience.

6.4.1. User Actions and System Responses:

- **Action 1:** User clicks "Submit Attendance."
- **System Response:**
    - System re-verifies session validity (face and location results).
    - If valid, attendance is recorded with:
        - User ID
        - Date and Time
        - Geolocation (lat/long)
        - Status (Present)
    - Confirmation message displayed: “Attendance recorded successfully!”
- **Action 2:** User tries to submit again.
- **System Response:**
    - System checks duplicate submission prevention.
    - Message displayed: “Attendance already submitted for today.”

6.5. Instructor Attendance Monitoring

Rank: High

Explanation: Instructors can see which students have attended each session in real time. They can track, filter, and export attendance records for reporting.

6.5.1. User Actions and System Responses:

- **Action 1:** Instructor logs into their dashboard.
- **System Response:**
  - System loads instructor's course list.
  - Instructor selects a course.
- **Action 2:** Instructor views today's attendance.
- **System Response:**
  - System provides a table showing students' names, ID numbers, and attendance status which will be noted in different colours (GREEN for present and RED for absent)
  - Instructor can filter by date or student name.
- **Action 3:** Instructor clicks "Export Report."
- **System Response:**
  - System generates attendance report in PDF or Excel.
  - File is downloaded to instructor's device.

6.6. Admin Management Panel

Rank: High

Explanation: Admins control system data by managing users, courses, settings, and monitoring overall system health.

6.7. Notification System

Rank: Medium

Explanation: The app communicates important events and errors to users through real-time notifications. Notifications improve user engagement and remind students of classes, but the system can function minimally without them initially. They are useful but not critical.

6.7.1. Functional Requirements

- **FR1**: The system shall send attendance check-in reminders to students before class sessions.
- **FR2**: The system shall send confirmation notifications after a successful check-in.

- **FR3**: The system shall send administrative alerts for incomplete or irregular attendance.
- **FR4**: The system shall allow users to customize notification preferences (email, app notification, SMS).
- **FR5**: The system shall log and track all notification deliveries for auditing.

6.8. Real-Time Attendance Monitoring

Rank: Medium

Explanation: The system provides real-time updates to instructors and admins, allowing them to see attendance activity as it happens; who has attended, who hasn’t, and who is currently checking in.

6.8.1. Functional Requirements

- **FR1**: The system shall allow authorized users (Students/admin/lecturers) to view live attendance data.
- **FR2**: The system shall dynamically update attendance records without manual refresh.
- **FR3**: The system shall provide filters to view attendance by course, class, and date.
- **FR4**: The system shall timestamp every attendance record submitted.
- **FR5**: The system shall restrict monitoring access based on user roles.
- **FR6**: The system shall auto-refresh monitoring data at regular intervals (e.g., 1 minute).
- **FR7**: The system shall trigger alerts for abnormal patterns (e.g., bulk simultaneous check-ins).

6.9. Error Dispute Mechanism

Rank: Medium

Explanation: Important for addressing errors, but it can be implemented after the core attendance features are stable. It adds value but isn’t required for initial functionality.

6.9.1. User Actions and System Responses:

- **Action 1:** Student clicks on "Report an Issue" and choose a course
- **System Response:**
  - System displays a dispute form asking for:
    - Student Details
    - Reason for Dispute
    - Image Attachment (selfie or screenshot)
- **Action 2:** Student fills and submits the dispute form.
- **System Response:**
  - System saves Dispute ID to the database under "Pending Review" status.
  - System sends the dispute to the email of the instructor/Lecturer.
  - Notification sent to instructor/admin for review.

- **Action 3:** Admin opens "Disputes."
- **System Response:**
  - System shows list of pending disputes.
  - Admin/Instructor can Approve or Reject with comments.
- **Action 4:** Admin update the dispute status and changes the Attendance

6.9.2. Functional Requirements

- **FR1**: The system shall allow students/lecturers to raise a dispute against recorded attendance or take leaves.
- **FR2**: The system shall provide a form for users to select the course and describe the dispute reason.
- **FR3**: The system shall send all dispute as an email to the Lecturer or Administrative the dispute is addressed to and a dispute ID is saved in a database for approval.
- **FR4**: The system shall allow administrators to view, approve, or reject dispute requests.
- **FR5**: The system shall update the attendance record if a dispute is approved.
- **FR6**: The system shall notify users about the status of their dispute (approved/rejected).
- **FR7**: The system shall maintain a dispute log for auditing purposes.

6.10. Role-Based Access Control (RBAC)

Rank: High

Explanation: Ensures security and proper segregation of duties between students, lecturers, and admins. This is critical for system integrity and must be implemented early.

6.10.1. User Actions and System Responses:

- **Action 1:** User logs into the system.
- **System Response:**
  - System identifies user’s role from JWT token (student, instructor, admin).
  - Based on the role, system dynamically loads only the permitted pages:
    - Student: Attendance submission page, history.
    - Instructor: Attendance monitoring, reporting tools.
    - Admin: Full management dashboard.

6.10.2. Functional Requirements

- **FR1**: The system shall define and enforce multiple user roles (Admin, Lecturer, Student).
- **FR2**: The system shall allow only authorized roles to access specific modules (e.g., monitoring, disputes, check-in).
- **FR3**: The system shall restrict sensitive actions (e.g., modifying attendance records) to Admins and Lecturers only.
- **FR4**: The system shall ensure that students can only view their own attendance data.
- **FR5**: The system shall audit and log all actions taken by different roles for accountability.

6.11. Students-Assisted Attendance Logging by Lecturers

Rank: High

Explanation:

This feature allows lecturers to assist with attendance marking by using facial recognition to detect multiple students at once during a class session. Lecturers can access a dedicated "Class Attendance Capture" page on the mobile app, where the system uses the device camera to scan and identify students who are physically present. Once recognized, the system automatically logs their attendance under the appropriate course and session without requiring individual student actions. This will help students without internet connection to Log in their Attendance

6.11.1. User Actions and System Responses:

User Action	**System Response**
Lecturer logs into the mobile app using credentials.	System authenticates credentials and grants access to the Lecturer Dashboard.
Lecturer navigates to "Class Attendance Capture" page.	System loads the facial recognition module and activates the device camera.
Lecturer starts a scanning session.	System scans the environment and detects multiple faces simultaneously.
System matches detected faces against the database of enrolled students.	For each match, system verifies identity and displays a list of recognized students on-screen.
Lecturer confirms the attendance list.	System records attendance for all confirmed students, linked to the specific course, date, and session.
Lecturer ends the session.	System saves all attendance logs to the database and returns lecturer to the Dashboard.

6.11.2. Functional Requirements:

- **FR1**: The system shall provide a secure login system for lecturers.
- **FR2**: The system shall allow lecturers to navigate to a page designed for mass facial recognition attendance.
- **FR3**: The system shall activate the device's camera when the lecturer starts an attendance session.
- **FR4**: The system shall detect and identify multiple faces within a camera frame.

- **FR5**: The system shall match detected faces with registered students for the specific course.
- **FR6**: The system shall display the names and IDs of successfully recognized students for lecturer confirmation.
- **FR7**: The system shall allow lecturers manually select the student ID to be checked in
- **FR8**: The system shall record attendance entries into the database after lecturer confirmation.
- **FR9**: The system shall associate each attendance log with the correct course code, class session, and timestamp.
- **FR10**: The system shall notify the lecturer if any detected student is not enrolled in the course.
- **FR11**: The system shall ensure data privacy by encrypting facial data during processing.

7. Non-Functional Requirements

Non-functional requirements describe how the system must behave. They define system attributes such as performance, reliability, usability, security etc.

7.1. Performance Requirements

Rank: High

Explanation: Slow or inaccurate performance would frustrate users and undermine trust in the system.

- The system must allow a student to complete facial recognition attendance within **5 seconds**.
- Geolocation verification must process within **3 seconds** after location services are turned on.
- The mobile app must handle up to **500 concurrent users** without system slowdown.
- The system must refresh real-time attendance dashboards every **30 seconds** during live monitoring.
- The database queries (e.g., fetching a class attendance list) must return results within **2 seconds** for up to 1,000 records.

7.2. Scalability Requirements

Rank: High

Explanation: The system must handle peak loads, especially during class start times.

- The system architecture should be modular to allow easy expansion to more users (e.g., from one school to many schools).
- The database and backend servers must support horizontal scaling (adding more servers easily).

7.3. Reliability Requirements

Rank: High

Explanation: The system must be reliable during key hours to ensure attendance tracking isn't disrupted.

- System uptime must be **at least 99.9%** per day.
- System should have automatic backup mechanisms daily to prevent data loss.

7.4. Usability Requirements

Rank: High

Explanation: A poorly designed interface would discourage adoption and usage.

- The mobile app must have a **simple and intuitive UI** accessible by users with minimal technical skills.
- Use large buttons, clear labels, and error-free navigation paths for better student experience.
- Onboarding process (first time use) should guide users through setting permissions (camera, GPS) effectively within **3 minutes**.

7.5. Security Requirements

Rank: High

Explanation: Protecting sensitive data is non-negotiable for compliance and user trust.

- Facial recognition data must be **encrypted** both at rest and in transit.
- JWT Authentication tokens must expire after **2 hours** and require refresh.
- Role-Based Access Control (RBAC) must strictly prevent unauthorized access to admin or instructor pages.
- Detect and block GPS spoofing attempts during geolocation verification.

7.6. Maintainability Requirements

Rank: High

Explanation: Ensures the system can be updated and fixed efficiently over time.

- Code should be well-commented and follow consistent style guidelines (e.g., Airbnb for Node.js, Prettier for JavaScript/React Native).
- A detailed developer's guide must be provided for future team members.
- All modules must be loosely coupled to allow independent updating (microservices).

7.7. Portability Requirements

Rank: Medium

Explanation: Starting with Android covers most users, with iOS support added later.

- The mobile application must be compatible with:
  - **Android 10.0 and above**
- Application updates would be made to be easily downloadable via Google Play Store and Apple App Store.

7.8. Cost Efficiency Requirements

Rank: High

Explanation: Ensures the project stays within budget while maintaining performance.

- Using **open-source technologies** (e.g., React Native, Node.js, PostgreSQL).
- Deploying on **cost-effective cloud solutions** (e.g., AWS Free Tier etc).
- Minimizing the need for expensive hardware — standard smartphones and mid-level servers are sufficient.

7.9. Recovery Requirements

Rank: Medium

Explanation: Important for data integrity but can be refined post-launch.

- In case of system crash or unexpected shutdown:
  - Recovery must be automatic within **5 minutes** without manual intervention.
  - Critical data like attendance records and user information must be restored from the last successful backup.
- The system must perform **incremental backups** every 12 hours and **full backups** once daily.

7.10. Data Retention Requirements

Rank: Medium

Explanation: Requires research but is important for compliance and record-keeping.

- Attendance records must be stored securely for a minimum of **5 years** before eligible for archival or deletion.
- Historical data must be accessible for report generation, audits, and compliance for at least **5 academic years**.
- Data deletion must be carried out only after approval and following a proper data disposal policy to prevent unauthorized access.

7.11. Auditability Requirements

Rank: High

Explanation: Essential for transparency, dispute resolution, and security audits.

- The system must maintain comprehensive **audit logs** recording:
  - All attendance submissions (with timestamp, device ID, GPS coordinates).
  - User login and logout activities.
  - Administrative actions (e.g., manual attendance adjustments, account modifications).
- Audit logs must be **immutable** (cannot be altered or deleted by users).
- Audit logs must be retained for at least **3 years** for legal and compliance purposes.
- Only authorized roles (e.g., Admin, Compliance Officer) should have access to view audit logs.

7.12. Fault Tolerance Requirements

Rank: Low

Explanation: Important for reliability but can be improved iteratively after launch.

- The system should continue operating correctly even if:
  - One server node fails (use of **load balancers** and **auto-healing instances**).
  - A database replica becomes unavailable (read/write split architecture for PostgreSQL).
- Mobile app must cache attendance locally when offline, and **retry** synchronization without losing data once connection is restored.
- GPS or Camera failures should trigger **user notifications** with fallback options or error reporting to admin.
- Graceful degradation must occur — if facial recognition service is temporarily unavailable, users must be allowed to retry within **5 minutes**.

8. Technical Constraints and Feasibility

Below is a categorized list of technical constraints along with an assessment of their feasibility:

1.Hardware & Device Requirements:

Constraint	Feasibility	Notes
Camera Access (for facial recognition during sign-up/login/attendance)	Feasible	Most modern smartphones have cameras, and permission requests are standard in mobile apps.
GPS/Location Services (for geofencing attendance validation)	Feasible	GPS is widely available, but accuracy (~1m) may require additional validation (e.g., Wi-Fi/BLE beacons).
Biometric Storage & Processing (secure facial template storage & real-time matching)	Partially Feasible	Requires robust encryption (e.g., AES-256) and optimized ML models (e.g., TensorFlow Lite) to meet the 5-second latency target.

2. Performance & Scalability:

Constraint	Feasibility	Notes
Facial Recognition ≤5 sec/scan	Feasible with Optimization	Requires lightweight ML models (e.g., FaceNet) and edge processing to reduce server load.
Geofencing Accuracy (1m)	Challenging	Standard GPS accuracy is ~5m; may need hybrid solutions (GPS + Wi-Fi/BLE beacons).
10 Check-ins/Second (Scalability)	Feasible	Achievable with load-balanced cloud infrastructure (e.g., AWS/Azure) and efficient DB indexing
99% Uptime (6 AM–7 PM)	Feasible	Requires redundant servers, failover mechanisms, and monitoring (e.g., Kubernetes clusters).

3. Security & Compliance:

Constraint	Feasibility	Notes
Biometric Data Encryption (in transit/at rest)	Feasible	Standard practice using HTTPS, TLS, and AES-256 encryption.
Explicit User Consent for Biometrics	Feasible	GDPR/regional compliance can be implemented via opt-in prompts.
Role-Based Access Control (RBAC)	Feasible	Straightforward with JWT tokens or OAuth 2.0.

4. Software & Integration:

Constraint	Feasibility	Notes
Cross-Platform (Android to iOS)	Feasible	React Native supports both, but may require platform-specific tweaks.
Real-Time Notifications (Push/Email)	Feasible	Firebase Cloud Messaging (FCM) or AWS SNS can handle this.
Dynamic Geofencing (Pre-Approved Venues Only)	Feasible	Geofence APIs (Google Maps/Mapbox) allow boundary updates via admin inputs.

5. Usability & Accessibility:

Constraint	Feasibility	Notes
Colorblind-Friendly UI (Icons/Patterns)	Feasible	WCAG-compliant design tools (e.g., Figma plugins) simplify implementation.
Offline Support	Explicitly Excluded	Not required per the document (real-time only).

6. Cost & Maintenance:

Constraint	Feasibility	Notes
Open-Source Stack (React Native, Node.js, PostgreSQL)	Feasible	Reduces licensing costs but may require expertise for optimization.
Modular Codebase	Feasible	Requires disciplined architecture (e.g., microservices).

Key Challenges & Risks:

1. Geofencing Accuracy: Achieving 1m precision may need supplemental technologies (e.g., Bluetooth beacons), increasing complexity.

2. Biometric Processing Latency: Edge computing (on-device face matching) may be needed to meet the 5-second target.

3. iOS Compatibility: Delayed iOS support could exclude some users initially.

Most constraints are feasible with existing technologies, but geofencing accuracy and biometric latency require careful optimization. The system's reliance on real-time connectivity (no offline mode) and open-source tools keeps costs manageable. Legal compliance (biometric data) and accessibility testing are critical next steps.

9. Future Work

While the initial system is designed to meet the immediate needs of attendance management, future iterations and expansions are anticipated to enhance the system further:

- **Integration with Institution ERP Systems**: Seamlessly link attendance records to broader academic management systems to automate grading or participation metrics.
- **Advanced Facial Recognition**: Implement AI enhancements for improved recognition accuracy under different lighting or environmental conditions.
- **Offline Functionality**: Enable offline attendance logging that automatically syncs once internet connectivity is restored.
- **Behavioral Analytics**: Use attendance patterns and data analytics to predict student performance risks and intervene early.
- **Cross-Platform Expansion**: Develop a web-based administrative panel and tablet-specific applications for broader institutional flexibility.
- **Blockchain Integration for Attendance Records**: In the long-term, secure attendance records using blockchain technology to ensure tamper-proof data.
- **Multi-Factor Authentication (MFA)**: Strengthen security by combining facial recognition with OTP (One Time Password) codes or biometrics.
- **Multi-Institutional Deployment**: Expand the system to support multiple universities, colleges, or training centers under a centralized administrative panel.

10. Conclusion

This Software Requirements Specification (SRS) outlines the development of the Mobile-Based Attendance Management System using Geofencing and Facial Recognition. The system aims to solve issues found in manual attendance methods, such as inaccuracies, impersonation, and time inefficiencies, by offering a secure, automated, and user-friendly solution.

Key features include geolocation-based attendance validation, facial recognition login, real-time monitoring, error dispute handling, and role-based access control. Technologies such as React Native, Node.js, and PostgreSQL have been selected to ensure scalability, performance, and reliability. Non-functional requirements like security, cost-efficiency, fault tolerance, and auditability have also been prioritized.

This document serves as a complete guide for building the system according to stakeholder needs and sets the foundation for a structured and efficient development process.