May 5, 2025

UNIVERSITY OF BUEA
Buea, South West Region
Cameroon
P.O. Box 63,
Tel: (237) 3332 21 34/3332 26 90
Fax: (237) 3332 22 72

REPUBLIC OF CAMEROON

PEACE-WORK-FATHERLAND

FACULTY OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER ENGINEERING

NAME	MATRICULE
BILLA SOPHIA	FE22A176
EKANE METUGE AKAME FAVOUR	FE22A199
EYONG GODWILL NGANG	FE22A214
NEBOTA ISMAEL OWAMBA	FE22A256
ONYA MARTHA . O	FE22A292

Task 3

Course Master:
Dr. Nkemeni Valery
2024/2025

REQUIREMENT ANALYSIS AND SOFTWARE REQUIREMENTS SPECIFICATION (SRS)

Mobile-Based Attendance Management System using Geofencing and Facial Recognition

Group: 24
CEF440 - Internet Programming and Mobile Programming
Version: 2.0

TABLE OF CONTENTS

1. Introduction

1.1 Purpose

This Software Requirements Specification (SRS) document provides a comprehensive and detailed description of the functional and non-functional requirements for the Mobile-Based Attendance Management System. It serves as the definitive guide for the system's design, development, testing, and deployment, ensuring a common understanding among all stakeholders regarding the system's capabilities and limitations within the University of Buea context.

1.2 Scope

The system encompasses a cross-platform mobile application (Flutter for Android/iOS) for students and instructors, interacting with a dedicated backend server and database. The primary goal is to automate and secure student attendance tracking using facial recognition for identity verification and geofencing for location validation.

- **In Scope:** Secure user management, facial enrollment, geofence configuration, location-validated facial recognition check-in, real-time attendance monitoring, student attendance history, instructor/admin reporting, manual override capability, offline check-in support, audit logging.
- **Out of Scope (Version 1.0):** Direct integration with existing University ERP/SIS systems, advanced predictive analytics, parental access features, support for platforms other than Android/iOS mobile devices, automated timetable integration for geofence scheduling.

1.3 Definitions, Acronyms, and Abbreviations

Term	Definition
Admin	System Administrator
API	Application Programming Interface
CRUD	Create, Read, Update, Delete
CSV	Comma-Separated Values
ERP	Enterprise Resource Planning
FAR	False Acceptance Rate (Security metric for biometrics)
FR	Functional Requirement
Geofence	A virtual geographic boundary (latitude, longitude, radius) used for location validation.
GPS	Global Positioning System
HTTPS	Hypertext Transfer Protocol Secure
iOS	Apple's mobile operating system
JSON	JavaScript Object Notation
JWT	JSON Web Token (Standard for creating access tokens)
ML	Machine Learning
ML Kit	Google's mobile Machine Learning SDK
MoSCoW	Prioritization method: Must Have, Should Have, Could Have, Won't Have

MVP	Minimum Viable Product
NFR	Non-Functional Requirement
PDF	Portable Document Format
PostgreSQL	Open-source relational database system
P1, P2, P3	Priority Levels (P1=Must Have, P2=Should Have, P3=Could Have)
RESTful	Representational State Transfer (API architectural style)
RPO	Recovery Point Objective (Maximum acceptable data loss duration)
RTM	Requirements Traceability Matrix
RTO	Recovery Time Objective (Maximum acceptable downtime duration)
SIS	Student Information System
SRS	Software Requirements Specification
TAR	True Acceptance Rate (Security metric for biometrics)
TBD	To Be Determined (Requires further clarification or decision during design/implementation)
TLS	Transport Layer Security (Cryptographic protocol for HTTPS)
UI	User Interface
UX	User Experience
WCAG	Web Content Accessibility Guidelines
XAF	Central African CFA franc

1.4 References

1. CEF440 Course Outline & Project Descriptions
2. Group 24 Task 1 Report: Deep Analysis of Mobile App Development
3. Group 24 Task 2 Report: Requirement Gathering
4. IEEE Std 830-1998 - IEEE Recommended Practice for Software Requirements Specifications (Structure Influence)

1.5 Overview

This document details the system requirements. Section 2 provides an overall description of the product, its users, constraints, and assumptions. Section 3 contains the specific functional, non-functional, interface, data, and privacy requirements, which form the core technical specification. Section 4 outlines the prioritization of these requirements. Section 5 describes the approach planned for validating the requirements. Section 6 is reserved for appendices.

2. Overall Description

2.1 Product Perspective

The Mobile-Based Attendance Management System is a new, independent client-server application designed for the University of Buea. The client component is a mobile application developed using the Flutter framework for cross-platform compatibility (Android/iOS). The server component comprises a backend Application Programming Interface (API) (specific technology stack TBD, e.g., Node.js/Express) responsible for business logic and data processing, and a relational database (e.g., PostgreSQL) for persistent storage. The system interfaces with mobile device hardware (camera, GPS receiver) and utilizes operating system services (location, camera access) and specialized libraries (e.g., Machine Learning for facial recognition). It aims to modernize and improve the accuracy and efficiency of the existing attendance tracking processes.

2.2 Product Functions (Summary)

The system will provide the following core capabilities:

- Secure user authentication and role-based access control.
- Student enrollment process including capture and secure storage of facial biometric data.
- Administrative tools for managing users, courses, and geofence boundaries for classrooms.
- Student attendance check-in mechanism combining real-time facial recognition and geofence validation.
- Real-time view for instructors to monitor attendance during class sessions.
- Access for students to view their personal attendance history.
- Generation of attendance reports for instructors and administrators.
- System auditing capabilities.
- Support for offline check-in scenarios.

2.3 User Characteristics

User Role	Description	Technical Expertise	Key Needs
Student	University students enrolled in courses; varying levels of tech-savviness but generally familiar with mobile apps.	Low to Medium	Fast (<5s), reliable, and straightforward check-in process; assurance of data privacy; easy access to personal attendance records; clear system feedback.
Instructor	University teaching staff responsible for classes; comfortable with using standard software applications.	Medium	Accurate, real-time attendance information; simple tools for session management and monitoring; reliable reporting; mechanism for handling exceptions (e.g., manual override).
Administrator	Staff responsible for overall system management, configuration, and user support. Requires system oversight privileges.	Medium to High	Efficient tools for managing users, courses, and geofences; system configuration options; comprehensive reporting capabilities; access to audit trails for monitoring and accountability.
IT Department	(Implicit Stakeholder) University IT staff supporting infrastructure, deployment, security, and maintenance.	High	System security compliance; scalability to handle user load; maintainability for updates and fixes; reliability and uptime; adequate monitoring tools; clear technical documentation.

2.4 General Constraints

1. **Platform Constraint:** The primary user interface shall be a mobile application developed using Flutter, ensuring compatibility with specified minimum versions of Android and iOS operating systems (specific versions TBD).
2. **Core Technology Constraint:** Facial recognition functionality shall leverage Google ML Kit libraries (or an equivalent, pre-approved ML library). Geofencing shall utilize standard operating system location services accessed via a suitable Flutter plugin (e.g., `geolocator`).
3. **Performance Constraint:** The end-to-end attendance check-in process (from initiation by the student to confirmation) must achieve an average completion time of less than 5 seconds under defined typical network and device conditions.
4. **Hardware Constraint:** The system relies on users possessing and utilizing smartphones equipped with a functional front-facing camera and GPS/location capabilities.
5. **Network Constraint:** Real-time system functionalities depend on the availability of reliable internet connectivity (Campus Wi-Fi or Mobile Data). An offline check-in capability is required to mitigate temporary network outages.
6. **Development Constraint:** The project scope, timeline, and deliverables are constrained by the requirements and schedule of the CEF440 course.
7. **Data Privacy Constraint:** The system must strictly adhere to the data protection policies established by the University of Buea and comply with all relevant Cameroonian laws and regulations concerning the collection, processing, and storage of personal and biometric data.
8. **Accuracy Constraint:** The system requires high accuracy for facial recognition to ensure correct identification and functional accuracy for geofencing to reliably determine presence within designated areas, while acknowledging the inherent limitations of GPS technology, especially indoors.

2.5 Assumptions and Dependencies

1. **Device Availability Assumption:** It is assumed that the vast majority of students possess and will carry compatible smartphones (meeting minimum OS and hardware requirements) to their classes.
2. **User Cooperation Assumption:** It is assumed that users (students) will grant the necessary application permissions (Camera, Precise Location) upon request and will consent to the facial enrollment process after being adequately informed.
3. **Location Signal Assumption:** It is assumed that adequate GPS or other location signals are generally available within the designated geofenced areas (classrooms, labs). Mitigation strategies or clear communication regarding limitations in known poor-signal zones may be necessary.
4. **Network Reliability Assumption:** It is assumed that the campus network infrastructure (Wi-Fi and potentially cellular coverage) provides reasonably stable connectivity to support the system's real-time operations during peak usage times.
5. **Biometric Stability Assumption:** It is assumed that the facial features captured during enrollment remain sufficiently stable for recognition during check-in. A re-enrollment process is planned to handle significant appearance changes.
6. **Infrastructure Availability Assumption:** It is assumed that a suitable, secure, and reliable server hosting environment (either on-premise or cloud-based) will be available for deploying the backend system and database.

7. **Library Performance Assumption:** It is assumed that the chosen third-party libraries (ML Kit, location plugins) will perform adequately regarding speed and accuracy across the expected range of student devices.
8. **Configuration Data Assumption:** It is assumed that accurate and up-to-date information regarding users, course enrollments, and classroom locations (for geofence configuration) will be provided and maintained by authorized administrators.

3. Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces (UI)

- **UI.1 General:** The UI shall be designed to be intuitive, easy to navigate, and visually appealing, providing a consistent user experience across both Android and iOS platforms. All text shall be in English. Responsiveness to various standard smartphone screen sizes is required. Clear visual feedback shall be provided for user actions and system status changes.
- **UI.2 Student Application Interface:**
    - *Login Screen:* Fields for Matricule Number/Email and Password, Login Button, "Forgot Password?" link.
    - *Dashboard:* Primary view after login. Shall display relevant information like upcoming or currently active classes for check-in. A prominent "Check-in" button shall be visible when applicable. Quick access to Attendance History.
    - *Facial Enrollment Interface:* A guided, step-by-step process with clear instructions (textual and potentially graphical). Includes a live camera preview with guides (e.g., face outline). Provides real-time feedback on image quality (e.g., "Too dark," "Move closer," "Hold still"). Clear indication of success or failure upon completion.
    - *Check-in Interface:* Activated during the check-in process. Displays a live camera preview. Provides dynamic status updates overlayed or adjacent to the preview (e.g., "Acquiring location...", "Validating location...", "Please position your face within the frame", "Processing...", "Check-in Successful!", "Error: [Specific Reason]").
    - *Attendance History Screen:* Allows viewing of past attendance records. Shall support filtering by course and/or date range. Displays records in a clear list or calendar format.
    - *Profile Screen:* Displays basic user information (Name, Matricule, Email). Provides access to initiate facial re-enrollment. May include links to help/support or privacy policy.
- **UI.3 Instructor View Interface (May be integrated into the same app with role switching or a separate interface):**
    - *Login Screen:* Similar to student login.
    - *Dashboard/Course List:* Displays courses assigned to the instructor. Allows selection of a course to manage sessions.
    - *Session Management Interface:* Option to "Start Session" for a selected course (activating check-in for students). Displays active session status and provides an "End Session" button. May display the associated geofence details.
    - *Real-time Attendance Monitor Screen:* Displays a list of students enrolled in the currently active session. For each student: Name, Matricule Number, real-

    time Attendance Status (e.g., "Present" with timestamp, "Absent"). Includes summary counts (e.g., X Present / Y Total). Requires a mechanism for updating (automatic push/poll or manual refresh).
    - *Manual Override Interface:* Accessible from the monitor screen. Allows selection of a student, changing their status (Present/Absent), requires entry into a mandatory justification text field, and confirmation.
    - *Reporting Interface:* Allows selection of course(s) and date range. Provides options to generate and export attendance reports.
- **UI.4 Admin Interface (Web-based interface is strongly recommended for administrative tasks):**
    - *Secure Login Screen.*
    - *Dashboard:* Provides an overview of system status, key statistics (e.g., total users, active sessions), and quick links to management sections.
    - *User Management Section:* Interface for searching, viewing, creating, editing, and deactivating/activating user accounts (Students, Instructors, Admins). Role assignment capability.
    - *Course Management Section:* Interface for CRUD operations on courses. Interface for managing student enrollments within courses (individual assignment and bulk upload/management - P2).
    - *Geofence Management Section:* Interface for CRUD operations on geofences. Should ideally include an interactive map for visual placement/adjustment (P2) alongside manual coordinate/radius input. List view of existing geofences.
    - *Reporting Section:* Interface for generating comprehensive system-wide reports with advanced filtering capabilities. Export functionality.
    - *Audit Log Viewer Section:* Interface for viewing system audit logs with searching and filtering capabilities.
    - *System Settings Section:* Interface for viewing and modifying configurable system parameters (e.g., thresholds, timeouts).

3.1.2 Hardware Interfaces

- **HW.1 Camera Interface:** The mobile application shall access the device's front-facing camera hardware via standard operating system APIs to capture images or video streams necessary for facial enrollment and real-time recognition during check-in.
- **HW.2 Location Services Interface:** The mobile application shall interface with the device's location services subsystem via operating system APIs to obtain geographical coordinates (latitude, longitude, accuracy estimate). This requires access to GPS, Wi-Fi positioning, and cellular positioning data as available and permitted by the user ('Precise Location' permission is required).

3.1.3 Software Interfaces

- **SW.1 Target Operating Systems:** The mobile application must be compatible with Android API Level 23 (Android 6.0 Marshmallow) and higher, and iOS Version 12.0 and higher. (Final minimum versions subject to confirmation based on dependencies).
- **SW.2 Key Frameworks/Libraries:** The application shall be developed using the Flutter SDK (latest stable version recommended at project initiation). It will utilize the Google ML Kit Face Detection library (or equivalent approved library) for facial

feature processing. It will utilize the Flutter `geolocator` plugin (or equivalent approved plugin) for accessing device location services. Standard Flutter/Dart libraries for HTTP communication, state management, and UI components will be used.

- **SW.3 Backend API Interface:** The mobile application shall communicate with the backend server exclusively through a well-defined RESTful API. The API specification (e.g., using OpenAPI/Swagger format) must be documented.
- **SW.4 Database Interface:** The backend server component shall interface with a PostgreSQL relational database (Version 12 or higher recommended) for data persistence, using appropriate database drivers or Object-Relational Mappers (ORMs).

3.1.4 Communication Interfaces

- **COM.1 Network Transport Protocol:** All communication between the mobile client application and the backend server shall utilize the TCP/IP protocol suite.
- **COM.2 Secure Communication Protocol:** Application-level communication over the network must be secured using HTTPS, enforcing the use of Transport Layer Security (TLS) version 1.2 or higher with strong, currently recommended cipher suites. Unencrypted HTTP communication is prohibited.
- **COM.3 Data Interchange Format:** The standard format for data exchange in API request/response bodies between the client and server shall be JSON (JavaScript Object Notation).
- **COM.4 API Authentication Mechanism:** API endpoints requiring user authentication shall be protected using a token-based authentication scheme. JWT (JSON Web Tokens) are recommended. Tokens shall be passed securely via the HTTP Authorization header (Bearer scheme).

3.2 Functional Requirements

3.2.1 User Authentication and Profile Management

- **FR.1.1 (P1):** The system shall provide functionality within the Admin Interface for authorized administrators to create, view, update, and deactivate/activate user accounts for students and instructors. Mandatory information includes Full Name, unique identifier (Matricule Number for students, Staff ID for instructors), unique Email address, initial password (securely generated or set), and assigned Role (Student/Instructor/Admin).
- **FR.1.2 (P1):** Users shall authenticate to the system (mobile app or Admin Interface) by providing their registered unique identifier (Matricule/Staff ID or Email) and their corresponding password. The system shall verify these credentials against stored records.
- **FR.1.3 (P2):** The system shall offer a secure self-service password reset mechanism. Users initiating this process (e.g., via a "Forgot Password" link on the login screen) shall receive instructions or a secure reset link/code via their registered email address to set a new password.
- **FR.1.4 (P1):** Once authenticated, users shall be able to view their own basic profile information within their respective interfaces (e.g., Name, ID, Email, Role). Editing capabilities shall be restricted (e.g., students cannot change their Matricule Number).
- **FR.1.5 (P1):** The system shall enforce minimum password complexity requirements during account creation and password changes (e.g., minimum length, inclusion of

different character types like uppercase, lowercase, numbers, symbols). These rules shall be clearly communicated to the user during password setting. Administrator configurability of these rules is a P2 requirement (FR.8.5).

- **FR.1.6 (P2):** The system shall implement automatic session termination due to inactivity. If a user's session remains idle for a configurable duration (e.g., 30 minutes), they shall be automatically logged out and required to re-authenticate.

3.2.2 Facial Data Enrollment

- **FR.2.1 (P1):** The system shall automatically check if a student user logging in for the first time has completed facial enrollment. If not, the user shall be prompted and guided to initiate the enrollment process. Students shall also be able to initiate enrollment or re-enrollment manually via their profile screen.
- **FR.2.2 (P1):** The enrollment interface within the mobile app must provide clear, user-friendly instructions (textual and potentially graphical/animations) and real-time visual feedback (e.g., face outline guide, quality indicators) to assist the student in capturing a high-quality facial image suitable for template extraction (e.g., adequate lighting, face centered and fully visible, neutral expression, no obstructions like sunglasses).
- **FR.2.3 (P1):** The system shall utilize the device's front-facing camera to capture one or more images of the student's face during the guided enrollment process. The system may employ auto-capture logic when optimal conditions are detected or require explicit user action. Capturing multiple images or a short video snippet to generate a more robust template is a P2 enhancement consideration.
- **FR.2.4 (P1):** The process of extracting the facial feature vector (biometric template) from the captured image(s) shall occur directly on the student's mobile device using the integrated ML library. This enhances user privacy and reduces processing latency.
- **FR.2.5 (P1):** Only the extracted facial feature vector(s), not the raw facial image(s), shall be transmitted securely (using HTTPS) to the backend server. The server shall store this template in an encrypted format (see NFR.2.3), associating it with the student's unique identifier.
- **FR.2.6 (P1):** The mobile app must provide immediate and unambiguous feedback to the student upon completion of the enrollment attempt, clearly indicating success or failure. In case of failure, a specific reason should be provided where possible (e.g., "Image quality too low, please ensure good lighting," "Face not detected clearly," "Enrollment failed, please try again").
- **FR.2.7 (P2):** The system shall allow students to initiate a facial re-enrollment process through their profile settings. Depending on university policy, this action might require approval from an administrator. Successful re-enrollment shall securely replace the previously stored template.

3.2.3 Geofence Management

- **FR.3.1 (P1):** Authorized administrators shall have the capability to define, view, update, and delete geofences (representing physical classroom or lecture hall boundaries) using the Admin Interface.
- **FR.3.2 (P1):** Each geofence definition must include a unique, descriptive name (e.g., "FET Amphitheater 101"), its precise geographical center coordinates (latitude and longitude), and a radius specified in meters, defining the circular boundary.

- **FR.3.3 (P2):** The Admin Interface for geofence management should ideally feature an interactive map display (e.g., using Google Maps API or OpenStreetMap) allowing administrators to visually select locations, draw boundaries, and verify the defined areas. Functionality for direct input/editing of coordinates and radius must also be provided.
- **FR.3.4 (P1):** Administrators or potentially instructors (TBD based on operational model) must be able to associate specific, defined geofences with scheduled courses or individual class sessions. This linkage determines which geofence is active for validation during check-in attempts.
- **FR.3.5 (P1):** The system must maintain the status of each geofence definition (e.g., Active/Inactive). Only geofences marked as 'Active' and associated with the specific, currently ongoing class session shall be used for location validation during student check-in attempts.

3.2.4 Attendance Check-in (Student)

- **FR.4.1 (P1):** When a student is logged into the mobile app, the main dashboard or relevant screen shall clearly display any ongoing class sessions for which they are enrolled and for which the designated check-in window is currently active. A clear call-to-action (e.g., "Check-in Now" button) shall be presented for these sessions.
- **FR.4.2 (P1):** Upon the student initiating the check-in action for a specific session, the application must immediately attempt to acquire the device's current geographical location using the most accurate available method (prioritizing GPS). The application must handle scenarios where location permissions are denied by the user or location services are disabled on the device, preventing the check-in and providing informative feedback to the user.
- **FR.4.3 (P1):** The application shall query the backend server to retrieve the active geofence parameters (center coordinates, radius) specifically associated with the selected class session.
- **FR.4.4 (P1):** The application shall perform a calculation to determine the distance between the acquired device location and the center of the active geofence. This distance shall be compared against the defined geofence radius. A configurable tolerance buffer (e.g., +/- 10 meters, see NFR.7.2) should be incorporated to account for inherent GPS inaccuracies.
- **FR.4.5 (P1):** If the device's location is determined to be outside the geofence boundary (considering the tolerance buffer), the check-in process must be immediately terminated. The application shall display a clear message to the student explaining the reason (e.g., "Check-in Failed: You must be inside the designated area for [Classroom Name].").
- **FR.4.6 (P1):** If the device's location is successfully validated as being inside the geofence boundary, the application shall then proceed to activate the front-facing camera interface for facial recognition. The application must handle scenarios where camera permission is denied by the user, preventing check-in and informing the user.
- **FR.4.7 (P1):** The application shall capture the student's face from the live camera feed and perform on-device facial feature extraction using the integrated ML library to generate a temporary feature vector.
- **FR.4.8 (P1):** The application shall securely transmit this newly extracted feature vector to the backend server.
- **FR.4.9 (P1):** The backend server shall compare the received feature vector against the student's securely stored, encrypted template using a defined similarity algorithm

(e.g., cosine similarity). If the calculated similarity score meets or exceeds a predefined, configurable threshold (see NFR.7.1), the facial recognition step is considered successful.

- **FR.4.10 (P1):** If, and only if, both the geofence validation (FR.4.4) and the facial recognition (FR.4.9) steps are successful, the backend server shall immediately create an attendance record in the database. This record must include, at minimum: the Student's unique ID, the relevant Course/Session unique ID, a precise Timestamp of the successful check-in, and an attendance Status marker ("Present").
- **FR.4.11 (P1):** The mobile application must display immediate, clear, and unambiguous feedback to the student confirming the outcome of the check-in attempt. This includes a prominent "Check-in Successful!" message upon success, or a specific, user-friendly error message upon failure (e.g., "Check-in Failed: Face not recognized. Please try again.", "Check-in Failed: Network connection lost.", "Check-in Failed: Check-in window is closed.").
- **FR.4.12 (P2):** The system shall enforce a configurable time window during which check-in is permitted for each class session (e.g., starting 10 minutes before the scheduled class start time and ending 15 minutes after). Check-in attempts outside this active window shall be automatically denied by the system (either client-side or server-side).
- **FR.4.13 (P2):** The system shall implement a policy for handling repeated consecutive facial recognition failures during a single check-in attempt. For example, after 3 unsuccessful attempts, the check-in function for that specific session might be temporarily locked for that student, and the app should advise them to seek assistance from the instructor or administrator. The number of allowed attempts shall be configurable.
- **FR.4.14 (P1):** The entire check-in process, encompassing location acquisition and validation, camera activation, facial capture and recognition, server communication, and feedback display (steps FR.4.2 through FR.4.11), must be optimized to meet the average performance target specified in NFR.1.3 (< 5 seconds).

3.2.5 Real-time Attendance Monitoring (Instructor)

- **FR.5.1 (P1):** Authenticated instructors shall be able to access a list of courses they are assigned to teach within the instructor interface (mobile app or web admin).
- **FR.5.2 (P1):** For a selected course, instructors shall be able to initiate an "Active Session," effectively opening the check-in window and activating the associated geofence for students enrolled in that course.
- **FR.5.3 (P1):** Once a session is active, the instructor must be able to navigate to a dedicated monitoring screen that displays a list of all students officially enrolled in that specific course/session.
- **FR.5.4 (P1):** The monitoring screen shall display, for each listed student, their Full Name, Matricule Number (or relevant ID), and their current Attendance Status for that session (e.g., "Present," "Absent"). For students marked as "Present," the timestamp of their successful check-in must also be displayed.
- **FR.5.5 (P1):** The attendance status information displayed on the monitoring screen must update automatically or semi-automatically (e.g., via polling or WebSocket connection) to reflect successful student check-ins in near real-time, meeting the latency requirement specified in NFR.1.4. A manual refresh option shall also be available as a fallback.

- **FR.5.6 (P2):** The monitoring screen should provide a real-time summary count, clearly displaying the number of students currently marked as "Present" versus the total number of students enrolled in the session (e.g., "Attendance: 45 / 50").
- **FR.5.7 (P2):** The system shall provide instructors with the functionality to manually override a student's automatically recorded attendance status for the currently active session (i.e., change "Absent" to "Present" or "Present" to "Absent"). This action should be accessible directly from the monitoring screen.
- **FR.5.8 (P2):** Performing a manual override (as per FR.5.7) must be contingent upon the instructor providing a mandatory textual justification explaining the reason for the change. This justification, along with details of the override (Instructor ID, Student ID, Session ID, timestamp, previous status, new status), must be recorded immutably in the system's audit log (FR.10).
- **FR.5.9 (P1):** Instructors must have the ability to explicitly "End Session" via the interface. Ending the session shall close the check-in window for any remaining students and finalize the attendance state for that session.

3.2.6 Attendance History Viewing (Student)

- **FR.6.1 (P1):** Authenticated students must be able to access a dedicated section within the mobile application that allows them to view their complete personal attendance history recorded by the system.
- **FR.6.2 (P1):** The attendance history interface shall provide filtering capabilities, allowing students to narrow down the displayed records by selecting a specific Course and/or a custom Date Range.
- **FR.6.3 (P1):** For each attendance record displayed within the filtered history, the following information must be clearly presented: Date of the session, Course Code and/or Name, Session Time (if applicable/available), the recorded Attendance Status ("Present" or "Absent"), and the specific Check-in Timestamp (if the status is "Present").
- **FR.6.4 (P2):** The attendance history interface may optionally calculate and display a summary attendance percentage for the student based on the currently applied filters (e.g., "Attendance for [Course Name] between [Start Date] and [End Date]: 85%").

3.2.7 Attendance Reporting

- **FR.7.1 (P1):** Authenticated instructors shall be able to generate attendance reports for the courses they manage. Report generation shall be possible via the instructor interface (mobile or web admin) and must allow filtering by date range and potentially by specific class session(s).
- **FR.7.2 (P1):** Authenticated administrators shall have access to generate more comprehensive, system-wide attendance reports via the Admin Interface. Administrative reports must support filtering by various criteria, including Course, Instructor, individual Student, Date Range, and potentially by Attendance Status.
- **FR.7.3 (P2):** Generated attendance reports (both instructor and admin) must contain detailed information for each relevant attendance record, including: Student Full Name, Student Matricule Number, Date, Course Code/Name, Session Identifier (if applicable), final Attendance Status, and Check-in Timestamp (if Present). Reports covering multiple sessions for a student should include calculated summary statistics like overall attendance percentage for the filtered period.

- **FR.7.4 (P2):** The system must provide functionality to export the generated reports into standard, widely usable file formats. At minimum, export to CSV (Comma-Separated Values) for data analysis and PDF (Portable Document Format) for easy sharing and printing must be supported.

3.2.8 System Administration (Admin Interface)

- **FR.8.1 (P1):** User Account Management: Administrators shall have full CRUD (Create, Read, Update, Delete/Deactivate) capabilities for all user accounts (Students, Instructors, other Administrators) via the Admin Interface. This includes assigning appropriate roles and managing account status (active/inactive). Functionality to reset user passwords shall also be available.
- **FR.8.2 (P1):** Course Management: Administrators shall have full CRUD capabilities for course records within the system (e.g., Course Code, Course Name, Description).
- **FR.8.3 (P1):** Student Enrollment Management: Administrators shall manage the enrollment of students into specific courses. This includes adding individual students to courses and removing them. A mechanism for bulk enrollment (e.g., uploading a CSV file of student IDs and course codes) is a P2 requirement.
- **FR.8.4 (P1):** Geofence Configuration Management: Administrators shall have full CRUD capabilities for geofence definitions as specified in FR.3, accessible via the Admin Interface.
- **FR.8.5 (P2):** System Settings Configuration: The Admin Interface shall provide a dedicated section allowing authorized administrators to view and modify key operational parameters of the system. Configurable parameters should include, but are not limited to: facial recognition similarity threshold, check-in window start/end offsets relative to class time, password complexity rules, session inactivity timeout duration, offline check-in data validity period, default geofence tolerance buffer.
- **FR.8.6 (P2):** Audit Log Access: Administrators shall be able to access, view, search, and filter the system audit logs (as defined in FR.10) via the Admin Interface to monitor system activity and investigate incidents.

3.2.9 Offline Functionality

- **FR.9.1 (P2):** In scenarios where the mobile application successfully completes both the local geofence validation (FR.4.4) and the local facial recognition feature extraction (FR.4.7) but subsequently fails to communicate with the backend server to record the attendance (e.g., due to temporary loss of network connectivity), the application must securely store the essential details of the pending check-in record locally on the device's persistent storage.
- **FR.9.2 (P2):** The locally stored offline record must contain sufficient information to allow successful processing when connectivity is restored, including at minimum: Student ID, Session ID, the precise Timestamp of the successful local validation, and potentially a cryptographic proof or the securely handled feature vector. This locally stored data must be encrypted.
- **FR.9.3 (P2):** The mobile application must implement a robust background synchronization mechanism that periodically checks for network connectivity and automatically attempts to upload any pending offline check-in records to the backend server. Successful upload must result in the secure deletion of the local record. Failed uploads should be retried according to a defined backoff strategy.

- **FR.9.4 (P2):** The application's UI should provide a clear visual indicator to the user when there are pending offline records awaiting synchronization. Feedback should also be provided upon successful synchronization.
- **FR.9.5 (P2):** Core application functions that rely on previously synchronized data, such as viewing the attendance history, should remain accessible and functional even when the device is offline. Real-time functions like initiating a new check-in or viewing live monitoring data naturally require network connectivity. Facial enrollment also requires connectivity.

3.2.10 Audit Logging

- **FR.10.1 (P2):** The backend system must implement a comprehensive audit logging mechanism that securely records significant events related to system usage, security, and data modification. Logs should be tamper-resistant or tamper-evident.
- **FR.10.2 (P2):** The audit log must capture sufficient detail for each event, including: Timestamp, User ID performing the action (or system identifier), Type of event, Affected resource/entity (e.g., User ID, Course ID, Session ID), Outcome (Success/Failure), and relevant context (e.g., IP address for logins, justification text for overrides). Key events to log include:
    - User login attempts (success/failure).
    - Password change/reset events.
    - Facial enrollment attempts (success/failure).
    - Attendance check-in attempts (success/failure).
    - Instructor manual attendance overrides (including justification).
    - Administrative CRUD operations on Users, Courses, Geofences.
    - Changes to critical System Settings.
- **FR.10.3 (P2):** Access to view the audit logs shall be strictly restricted to users with the Administrator role, accessible via a dedicated interface (FR.8.6). Functionality for searching and filtering logs by time range, user, event type, etc., must be provided.
- **FR.10.4 (P2):** Audit logs shall be retained for a defined period according to university policy or regulatory requirements (linked to NFR.8.4), after which they may be archived or securely purged.

3.3 Non-Functional Requirements

3.3.1 Performance Requirements

- **NFR.1.1 (P1):** Facial Recognition Component Latency: The average time elapsed from capturing the face image to receiving a match/no-match decision (including on-device processing and any necessary server communication) shall be less than 3 seconds on representative mid-range target devices under typical conditions.
- **NFR.1.2 (P1):** Location Validation Latency: The average time required to acquire a sufficiently accurate location fix and validate it against the active geofence shall be less than 2 seconds, acknowledging variability due to GPS signal strength and availability.
- **NFR.1.3 (P1):** End-to-End Check-in Time: The average total time experienced by the student from initiating the check-in action to receiving visual confirmation of success or failure on the app screen shall be less than 5 seconds. This assumes typical campus network conditions (stable connection >1 Mbps, latency <100ms).

- **NFR.1.4 (P1):** Real-time Monitor Update Latency: Updates to the instructor's attendance monitoring screen reflecting new student check-ins shall appear within 5 seconds of the check-in being successfully recorded by the backend server.
- **NFR.1.5 (P2):** Backend Concurrency Handling: The backend system architecture must be designed to handle a peak load of at least 100 concurrent check-in requests per second without significant degradation in performance (e.g., maintaining average API response times below 500ms).
- **NFR.1.6 (P2):** Reporting Generation Time: The time required to generate typical reports shall be acceptable: less than 10 seconds for a standard single-class instructor report (e.g., <100 students, one month); less than 60 seconds for larger administrative reports (e.g., <1000 students, one semester).
- **NFR.1.7 (P1):** Mobile Application Responsiveness: UI interactions within the mobile app (e.g., navigating between screens, responding to button taps) shall feel immediate, with visual feedback occurring within 200 milliseconds. The average cold start launch time of the application shall be less than 3 seconds.

3.3.2 Security Requirements

- **NFR.2.1 (P1):** Secure Communication Channel: All network communication between the mobile client application and the backend server must be encrypted using HTTPS, enforcing TLS version 1.2 or higher and utilizing strong, industry-standard cipher suites.
- **NFR.2.2 (P1):** Secure Password Management: User passwords must never be stored in plain text. They shall be securely hashed using a modern, adaptive hashing algorithm that incorporates a unique salt per user (e.g., bcrypt with appropriate cost factor, or Argon2). Password transmission must always occur over the encrypted channel (HTTPS).
- **NFR.2.3 (P1):** Secure Biometric Data Handling: Facial feature vectors (templates) must be treated as highly sensitive data. They must be encrypted both at rest within the database (e.g., using AES-256) and during transmission (covered by HTTPS). Secure key management practices are essential. Raw facial images must not be persisted on the server after enrollment processing.
- **NFR.2.4 (P1):** Secure Authentication and Session Management: The system shall employ a secure mechanism for user authentication and subsequent session management, preferably using industry-standard tokens (e.g., JWT). Tokens must have appropriate expiration times, be transmitted securely, and stored securely on the client device (e.g., utilizing platform-specific secure storage like Android Keystore or iOS Keychain). Measures against token hijacking (e.g., HTTPS enforcement) are required.
- **NFR.2.5 (P1):** Strict Authorization and Access Control: Role-Based Access Control (RBAC) must be rigorously implemented and enforced on the backend server for every API endpoint. Users must only be granted access to the data and functionalities explicitly permitted by their assigned role (Student, Instructor, Admin). Principle of least privilege must be applied.
- **NFR.2.6 (P1):** Minimal Permissions: The mobile application must request only the absolute minimum set of device permissions required for its core functionality (specifically, Camera access for face capture and Precise Location access for geofencing). Permissions should be requested contextually (just-in-time) with clear explanations provided to the user regarding why they are needed.

- **NFR.2.7 (P1):** Input Validation and Sanitization: All data received from clients (mobile app, web admin interface) must be strictly validated and sanitized on the backend server before processing or storage. This is critical to prevent common web application vulnerabilities, including but not limited to SQL Injection, Cross-Site Scripting (XSS - especially for web admin), command injection, and parameter tampering. Input length and type constraints must be enforced.
- **NFR.2.8 (P2):** Presentation Attack Detection (Liveness): To mitigate the risk of spoofing attacks using static photos or videos during facial recognition check-in, the system should incorporate basic liveness detection mechanisms. This could involve analyzing subtle cues like eye blinking or head movements, potentially leveraging features available within the chosen ML library (e.g., ML Kit's face contour detection), or implementing a simple challenge-response interaction if feasible.
- **NFR.2.9 (P2):** Secure Audit Logging: The audit logging mechanism (FR.10) must ensure the integrity and non-repudiation of log entries. Logs should be protected from unauthorized modification or deletion, potentially through write-only permissions, cryptographic hashing, or forwarding to a separate secure log management system.
- **NFR.2.10 (P2):** Dependency Security Management: A process must be established for regularly scanning all third-party libraries and dependencies used in both the mobile application and the backend system for known security vulnerabilities (e.g., using tools like `npm audit`, `flutter pub outdated`, OWASP Dependency-Check). Identified critical vulnerabilities must be addressed promptly through patching or library updates.
- **NFR.2.11 (P2):** API Security Hardening: Standard API security best practices should be implemented on the backend, such as rate limiting to prevent abuse, request size limits, appropriate HTTP security headers (e.g., `Strict-Transport-Security`, `Content-Security-Policy` for web admin), and protection against common API attacks.

3.3.3 Reliability Requirements

- **NFR.3.1 (P1):** System Availability Target: The core backend services (API, database) required for real-time check-in and monitoring must achieve an availability of 99.5% or higher during the University's primary operational hours (defined, e.g., Monday-Friday, 7:00 AM - 7:00 PM WAT).
- **NFR.3.2 (P1):** Data Integrity Assurance: The system design, including database schema (with constraints) and application logic (using transactions where appropriate), must prevent the loss or corruption of critical data (user accounts, enrollment data, attendance records) during normal operation and anticipated error conditions.
- **NFR.3.3 (P2):** Data Backup and Recovery Strategy: Automated database backups must be performed regularly (at least daily). Backups must be stored securely in a separate physical or logical location from the primary database server. A Recovery Point Objective (RPO) of 24 hours (maximum acceptable data loss) and a Recovery Time Objective (RTO) of 4 hours (maximum acceptable time to restore service after a major failure) must be targeted. Backup integrity and restoration procedures must be documented and tested periodically (e.g., semi-annually).
- **NFR.3.4 (P1):** Graceful Fault Handling: The mobile application and backend system must handle anticipated runtime errors and exceptions gracefully (e.g., network timeouts, API errors, invalid responses, permission denials, hardware unavailability). The system should provide informative error messages to the user where appropriate,

log errors for diagnosis, and avoid crashing or entering an unstable state. The offline functionality (FR.9) serves as a key mechanism for network fault tolerance during check-in.

3.3.4 Usability Requirements

- **NFR.4.1 (P1):** Ease of Learning: The system interfaces (especially the student mobile app) must be designed such that a typical first-time user can understand the main purpose and successfully complete core tasks (e.g., facial enrollment, attendance check-in) with minimal or no external guidance (target: < 5 minutes for first successful check-in after enrollment).
- **NFR.4.2 (P1):** Task Efficiency: The number of user interactions (taps, swipes, inputs) required to perform frequent core tasks like initiating and completing an attendance check-in must be minimized to ensure speed and reduce user effort.
- **NFR.4.3 (P1):** Clarity of Feedback: The system must provide clear, concise, timely, and easily understandable feedback for all significant user actions and system events, particularly regarding the status and outcome of enrollment and check-in attempts. Error messages must be informative and suggest corrective actions where possible.
- **NFR.4.4 (P1):** Interface Consistency: The design language, terminology, layout patterns, navigation methods, and interaction behaviors must be consistent throughout the mobile application and, where applicable, the administrative web interface, to promote predictability and ease of use.
- **NFR.4.5 (P3):** Accessibility Considerations: The design should consider basic mobile accessibility principles to accommodate users with disabilities where feasible within the project scope. This includes ensuring sufficient color contrast ratios (aiming for WCAG AA), providing adequate touch target sizes for interactive elements, and supporting dynamic font sizing if possible without breaking layouts.

3.3.5 Maintainability Requirements

- **NFR.5.1 (P1):** Code Quality and Readability: All source code produced (Flutter/Dart for mobile, backend language) must adhere to established, documented coding standards and best practices (e.g., Effective Dart guidelines). Code must be well-structured, appropriately commented (especially for complex logic), and easily understandable by other developers. Meaningful naming conventions are required.
- **NFR.5.2 (P1):** Architectural Design: The software architecture for both the mobile application (e.g., using MVVM, Bloc, or Clean Architecture) and the backend system (e.g., layered architecture, microservices if appropriate) must promote modularity, separation of concerns, low coupling, and high cohesion. This facilitates easier understanding, modification, testing, and debugging.
- **NFR.5.3 (P2):** Configurability: System parameters that are likely to change or need adjustment during deployment or operation (e.g., API endpoints, external service keys, facial recognition thresholds, timeout values, feature flags) should be externalized into configuration files or managed via the administrative interface, avoiding hardcoding within the source code.
- **NFR.5.4 (P2):** Testability and Automation: The codebase must be designed with testability in mind. A comprehensive suite of automated tests must be developed, including unit tests for individual functions/classes, widget tests for UI components (Flutter), and integration tests for API endpoints and key workflows. High test

coverage targets (e.g., >80% for critical logic) should be established and tracked. A Continuous Integration (CI) process to run tests automatically is recommended.

- **NFR.5.5 (P1):** Technical Documentation: Essential technical documentation must be created and maintained alongside the code. This includes a high-level System Architecture description, detailed API specifications (e.g., using OpenAPI/Swagger), Database Schema diagrams and descriptions, and clear Deployment/Setup instructions. Inline code comments should explain the 'why' behind complex logic.

3.3.6 Portability Requirements

- **NFR.6.1 (P1):** Mobile Platform Compatibility: The mobile application, developed using Flutter, must be compilable and function correctly, providing a consistent user experience without platform-specific defects, on the defined minimum supported versions of the Android and iOS operating systems.
- **NFR.6.2 (P1):** Backend Deployment Environment: The backend server application must be deployable on standard, widely available Linux-based server environments (e.g., Ubuntu LTS, CentOS). Containerization using Docker is strongly recommended (P2) to encapsulate dependencies, ensure consistency between development and production environments, and simplify the deployment process.

3.3.7 Accuracy Requirements

- **NFR.7.1 (P1):** Facial Recognition System Accuracy: The facial recognition component must achieve a high level of accuracy under typical operating conditions (e.g., varying indoor lighting, frontal face pose). The target accuracy is a True Acceptance Rate (TAR) of 98% or higher, with a False Acceptance Rate (FAR) of 1% or lower, measured against a representative test dataset. The similarity threshold used for matching shall be configurable to allow tuning based on operational experience.
- **NFR.7.2 (P1):** Geofencing Location Accuracy: The system must reliably determine whether the user's device is inside or outside the specified geofence boundary. Given the inherent inaccuracies of GPS, especially indoors, the system shall operate with a configurable tolerance buffer (e.g., +/- 10 meters added to the defined radius). The system's behavior in marginal/boundary cases should prioritize preventing false check-ins (i.e., err towards denying check-in if location is highly uncertain or right on the edge). Limitations in areas with poor GPS reception must be recognized.
- **NFR.7.3 (P1):** Data Recording and Calculation Accuracy: All data recorded by the system, particularly attendance statuses and timestamps, must be accurate and reflect the true state determined by the validation logic. Any calculations performed based on this data (e.g., attendance percentage reporting) must be mathematically correct according to defined formulas.

3.4 Data Management Requirements

- **NFR.8.1 (P1):** Database System Choice: The system shall utilize a robust, open-source relational database management system (RDBMS) capable of handling the expected data volume and transaction load. PostgreSQL (Version 12 or higher) is the recommended choice.
- **NFR.8.2 (P1):** Database Schema Design: A well-defined logical and physical database schema must be designed and documented. The schema must enforce data integrity using appropriate mechanisms such as primary keys, foreign keys, unique

constraints, data type constraints, and potentially check constraints where applicable. Relationships between entities must be clearly defined.

- **NFR.8.3 (P1):** Adherence to Privacy Requirements: All aspects of data management, including storage, access, modification, and deletion, must strictly comply with the privacy requirements outlined in section NFR.9.
- **NFR.8.4 (P2):** Data Retention and Purging Policy: A clear, configurable policy for data retention must be implemented. This policy shall define the duration for which different types of data (e.g., detailed attendance logs, system audit trails, inactive user accounts, expired biometric templates) are kept within the live system before being securely archived or purged. Example policy durations (subject to final confirmation based on University regulations): Attendance logs - 2 academic years; Audit logs - 1 year. Biometric templates must be purged upon user account deletion or successful re-enrollment.
- **NFR.8.5 (P1):** Data Backup Strategy: Regular, automated data backups must be performed according to the strategy defined in NFR.3.3 to prevent data loss.

3.5 Privacy Requirements

- **NFR.9.1 (P1):** Transparency and Informed Consent: Prior to initiating the facial enrollment process, students must be presented with a clear, concise, and easily understandable privacy notice. This notice must explicitly state: what personal data is being collected (identifiers, location during check-in, facial template), the specific purpose of collection (attendance verification), how the data will be stored and secured, the data retention period, and the user's rights regarding their data. Explicit, affirmative consent must be obtained from the student before any biometric data is captured or processed.
- **NFR.9.2 (P1):** Purpose Limitation: All personal data collected, especially sensitive biometric and location data, shall be processed strictly for the purpose of managing and verifying student attendance as described in the privacy notice and system documentation. The data must not be used for any other purpose (e.g., general surveillance, unrelated tracking) without separate, explicit consent.
- **NFR.9.3 (P1):** Data Minimization: The system shall only collect and retain the minimum amount of personal data necessary to achieve the specified purpose. On-device processing for facial feature extraction is mandated to minimize transmission of raw biometric data. Precise location data should only be accessed during the active check-in attempt and not stored long-term unless required for audit purposes (and stated in the privacy notice). Raw facial images must not be stored server-side.
- **NFR.9.4 (P1):** Security Safeguards: Robust technical and organizational security measures, as detailed in NFR.2, must be implemented and maintained to protect all personal data (especially biometric templates) against unauthorized access, accidental loss, destruction, alteration, or disclosure.
- **NFR.9.5 (P1):** Access Control Enforcement: Access to personal data within the system, particularly sensitive biometric templates and detailed attendance logs, must be strictly controlled based on user roles and granted only on a legitimate need-to-know basis (applying the principle of least privilege). Administrators' access should also be logged.
- **NFR.9.6 (P2):** User Data Rights: The system must provide clear procedures and contact information for students to exercise their data rights, which may include requesting access to their stored personal data and attendance records, requesting correction of inaccurate data, and requesting the deletion of their account and

associated data (including the biometric template), subject to overriding University policies or legal obligations regarding record retention.

- **NFR.9.7 (P1):** Legal and Policy Compliance: The design, implementation, and operation of the system must fully comply with all applicable data protection laws and regulations within Cameroon, as well as the specific data privacy and security policies established by the University of Buea.

4. Requirements Prioritization

The requirements specified in this document have been prioritized to guide development focus and manage project scope. The MoSCoW method (Must Have, Should Have, Could Have, Won't Have) is used, reflecting the essentiality of each requirement for achieving the core project objectives.

- **P1 - Must Have:** These requirements represent the Minimum Viable Product (MVP) and are absolutely essential for the system to function and meet its primary goal of automated, secure attendance tracking using facial recognition and geofencing. The system is not considered operational without these features. This category includes core functionalities like user login, facial enrollment, geofence definition, the combined check-in process, basic attendance viewing for students and instructors, fundamental administrative capabilities, and critical non-functional aspects like security foundations, basic reliability, core performance targets, and privacy compliance. *(Specific P1 requirements listed include: FR.1.1-1.2, 1.4-1.5; FR.2.1-2.6; FR.3.1-3.2, 3.4-3.5; FR.4.1-4.11, 4.14; FR.5.1-5.5, 5.9; FR.6.1-6.3; FR.7.1-7.2; FR.8.1-8.4; NFR.1.1-1.4, 1.7; NFR.2.1-2.7; NFR.3.1-3.2, 3.4; NFR.4.1-4.4; NFR.5.1-5.2, 5.5; NFR.6.1-6.2; NFR.7.1-7.3; NFR.8.1-8.3, 8.5; NFR.9.1-9.5, 9.7).*
- **P2 - Should Have:** These requirements are important and add significant value, usability, robustness, or administrative control to the system. While the system could technically function without them at a very basic level, their inclusion greatly enhances its effectiveness and user satisfaction. Examples include offline functionality, instructor manual override, detailed reporting with export, password reset, audit logging, liveness detection, and specific backup/recovery targets. Development should strongly aim to include these after P1 requirements are met. *(Specific P2 requirements listed include: FR.1.3, 1.6; FR.2.7; FR.3.3; FR.4.12-4.13; FR.5.6-5.8; FR.6.4; FR.7.3-7.4; FR.8.3 [bulk], FR.8.5-8.6; FR.9.1-9.5; FR.10.1-10.4; NFR.1.5-1.6; NFR.2.8-2.11; NFR.3.3; NFR.5.3-5.4; NFR.6.2 [Docker]; NFR.8.4; NFR.9.6).*
- **P3 - Could Have:** These requirements are desirable enhancements but are less critical than P1 and P2. They represent features that would be "nice to have" if time and resources permit after higher-priority items are completed. Examples include advanced accessibility features, more sophisticated analytics, or push notifications. *(Specific P3 requirements listed include: NFR.4.5).*
- **Won't Have (Version 1.0):** These are features explicitly identified as being outside the scope of the initial project release, often due to complexity, time constraints, or strategic decisions. Examples include integration with external university systems (ERP/SIS), a dedicated portal for parents, or support for platforms beyond Android/iOS mobile devices. These might be considered for future versions.

This prioritization framework allows for iterative development, ensuring that the most critical aspects of the system are delivered first.

5. Requirements Validation Approach

To ensure the requirements specified in this document are accurate, complete, unambiguous, consistent, and feasible, a structured validation process will be employed. This process involves multiple techniques and stakeholder participation:

1. **Stakeholder Reviews:** This SRS document will be formally reviewed by key stakeholders, including the Course Master (Dr. Nkemeni), representatives from the target user groups (students, instructors), system administrators, and potentially IT department personnel. Feedback will be solicited through review meetings, questionnaires, and direct comments to ensure requirements align with operational needs and expectations.
2. **Prototyping and Usability Evaluation:** Interactive prototypes or mockups demonstrating key user interface workflows (especially student enrollment and check-in, instructor monitoring) will be developed. These prototypes will be used in usability testing sessions with representative end-users to validate workflow logic, UI clarity, ease of use, and gather feedback on the overall user experience.
3. **Technical Feasibility Assessment:** The development team will conduct thorough technical reviews of the specified requirements, particularly focusing on challenging aspects like achieving the performance targets, ensuring facial recognition accuracy and robustness, handling indoor geofencing limitations, and implementing security measures. This review aims to identify potential risks and confirm the feasibility within the chosen technology stack and project constraints.
4. **Requirements Traceability:** A Requirements Traceability Matrix (RTM) will be created and maintained throughout the project lifecycle. This matrix will link each requirement specified in this SRS back to its origin (e.g., project objective, stakeholder input) and forward to corresponding design elements, implementation modules, and test cases. This ensures that all requirements are addressed and verifiable.
5. **Formal Inspections:** Structured walkthroughs or formal inspections of this SRS document will be conducted involving representatives from different stakeholder groups and the development team. The goal is to systematically identify any remaining ambiguities, inconsistencies, omissions, or errors before the requirements are formally baselined.
6. **Feedback Incorporation:** All feedback gathered through reviews, prototyping, technical assessments, and inspections will be documented, discussed, and used to refine this SRS document. Changes will be managed through version control, and significant modifications may require re-validation.

This multi-faceted validation approach aims to build confidence in the requirements specification and minimize the risk of errors or misunderstandings propagating into later stages of the project.

6. Appendices

(This section is reserved for supplementary materials such as Use Case Diagrams, Data Models, UI Mockups, etc.