**Software Requirements Specification for SmartAttendX: An Intelligent Attendance Management System**

**Version 1.0**

**Prepared by: Pranav Lanka**

**Date Created: July 28, 2025**

**Table of Contents**

1. **Introduction**

1.1. Purpose

1.2. Document Conventions

1.3. Intended Audience and Reading Suggestions

1.4. Product Scope

1.5. References

1. **Overall Description**

2.1. Product Perspective

2.2. Product Functions

2.3. User Classes and Characteristics

2.4. Operating Environment

2.5. Design and Implementation Constraints

2.6. User Documentation

2.7. Assumptions and Dependencies

1. **External Interface Requirements**

3.1. User Interfaces

3.2. Hardware Interfaces

3.3. Software Interfaces

3.4. Communications Interfaces

1. **System Features**

4.1. Feature 1: User Authentication & Management

4.2. Feature 2: Role-Based Access Control (RBAC)

4.3. Feature 3: Face Recognition Attendance

4.4. Feature 4: QR Code Attendance

4.5. Feature 5: Geo-Fencing Constraint

4.6. Feature 6: Dashboard and Data Visualization

4.7. Feature 7: Report Generation

4.8. Feature 8: ML-Based Anomaly Detection

4.9. Feature 9: Real-time Notifications

1. **Other Nonfunctional Requirements**

5.1. Performance Requirements

5.2. Safety Requirements

5.3. Security Requirements

5.4. Software Quality Attributes

5.5. Business Rules

**1. Introduction**

**1.1 Purpose**

This Software Requirements Specification (SRS) document provides a complete description of the requirements for the **"SmartAttendX: An Intelligent Attendance Management System."** This document outlines the system's purpose, features, functionalities, interfaces, and performance criteria. It is intended to serve as a foundational guide for the project's design, development, and testing phases, ensuring that the final product meets all specified stakeholder needs.

**1.2 Document Conventions**

* **Requirement IDs**: All functional requirements are uniquely identified with the prefix 'FR' followed by a number (e.g., FR-1.1). Nonfunctional requirements use prefixes like 'PERF', 'SEC', etc.
* **TBD**: The phrase "To Be Determined" (TBD) is used to mark areas where decisions are yet to be finalized.
* **Emphasis**: Key terms and concepts are highlighted in **bold**.
* **Priority**: Requirement priorities are stated as High, Medium, or Low.

**1.3 Intended Audience and Reading Suggestions**

This document is intended for a diverse audience, including:

* **Project Managers**: For project planning, scheduling, and tracking. (Sections 1, 2, 4)
* **Software Developers**: For understanding the system's functionalities and constraints to guide the implementation. (Sections 2, 3, 4, 5)
* **QA/Testing Team**: For creating test plans, test cases, and validation scripts. (Sections 3, 4, 5)
* **System Administrators**: For understanding deployment and operational environment requirements. (Sections 2.4, 3.2, 5.3)
* **Stakeholders (Faculty/HR)**: To review and confirm that the system aligns with business objectives. (Sections 1.4, 2.2, 4)

**1.4 Product Scope**

**SmartAttendX** is a next-generation attendance system designed to automate and secure attendance tracking using modern technologies. The system aims to replace traditional, manual attendance methods which are prone to errors and proxy marking.

**Key objectives and benefits include:**

* **Automation**: Automate attendance tracking with minimal human interaction using biometrics (face recognition) and other digital methods (QR codes, GPS).
* **Accuracy & Integrity**: Prevent proxy or fraudulent attendance through multi-factor verification, including face matching and geo-fencing.
* **Efficiency**: Provide real-time dashboards, analytics, and instant report generation, saving significant administrative time.
* **Insight**: Leverage Machine Learning (ML) to provide predictive insights and detect anomalies in attendance patterns.
* **Scalability**: A secure and scalable cloud-based architecture that can support a growing number of users and integrate with other institutional systems.

**1.5 References**

This SRS refers to the following documents and resources:

1. *Research Papers on Face Recognition for Attendance Systems* - For algorithmic and theoretical foundations.
2. *OpenCV Library Documentation* - For image processing functionalities.
3. *Google Maps API Documentation* - For implementing geo-fencing.
4. *Firebase/MongoDB Official Documentation* - For database schema design and queries.
5. *TensorFlow/Scikit-learn Guides* - For the implementation of machine learning models.
6. *Twilio API for SMS Documentation* - For implementing SMS notifications.

**2. Overall Description**

**2.1 Product Perspective**

SmartAttendX is a new, self-contained product designed to function as a standalone attendance management solution. It is also designed with interoperability in mind, capable of being integrated into a larger ecosystem, such as a university's Learning Management System (LMS) or a company's Human Resource Management (HRM) system, via APIs. It serves as a modern replacement for existing manual or legacy digital attendance systems.

**2.2 Product Functions**

The major functions provided by SmartAttendX are summarized below:

* **Multi-Role Access**: Secure, role-based access for Administrators (Faculty/HR) and Users (Students/Employees).
* **Attendance Dashboard**: A comprehensive dashboard showing daily, weekly, and monthly attendance statistics in a visually intuitive format.
* **Multi-Modal Attendance Marking**:
  + Face Recognition
  + QR Code Scanning
  + Geo-location validation
  + NFC/RFID card support (as an extension)
* **Report Generation**: Admins can generate and export detailed attendance reports in various formats (PDF, Excel).
* **Real-time Notifications**: Automated alerts via SMS/Email for absenteeism or other configured events.
* **Predictive Analytics**: An ML-powered module to detect unusual attendance patterns and predict future absenteeism.

**2.3 User Classes and Characteristics**

The system will be used by two primary classes of users:

| User Class | Characteristics | Responsibilities |
| --- | --- | --- |
| **Administrator** (Faculty/HR) | Tech-savvy, responsible for managing courses/teams and official records. Authorized to configure system settings. | - Manage user accounts. - Configure attendance sessions and rules (e.g., geo-fence). - View aggregate dashboard and analytics. - Generate and analyze official reports. - Review and act on flagged anomalies. |
| **User** (Student/Employee) | General users with smartphones. Varying levels of technical expertise. Primary goal is to mark attendance quickly and view their own history. | - Register and manage their profile (including face data). - Mark attendance using available modes. - View their personal attendance history and statistics. - Receive notifications. |

**2.4 Operating Environment**

* **Server-Side**: The backend application will be deployed on a cloud platform (e.g., AWS, GCP, Azure) running a Linux-based OS. It requires a Node.js or Python runtime environment.
* **Client-Side (Web)**: The web application will be accessible through modern web browsers such as Google Chrome, Mozilla Firefox, Safari, and Microsoft Edge.
* **Client-Side (Mobile)**: The mobile application or responsive web app will run on recent versions of Android and iOS, requiring access to the device's camera and GPS hardware.
* **Database**: The system will utilize a cloud-based NoSQL (Firebase, MongoDB) or SQL (MySQL) database for data persistence.

**2.5 Design and Implementation Constraints**

* **Technology Stack**: The system must be developed using the specified technology stack: Python/Flask or Node.js for the backend, React/Vue.js for the frontend, and TensorFlow/Scikit-learn for the ML model.
* **Third-Party APIs**: The system is dependent on external services like Google Maps API for geo-fencing and Twilio for SMS notifications. System availability may be affected by the uptime of these services.
* **Hardware Dependency**: Face recognition and QR scanning features require a functional webcam or smartphone camera. Geo-fencing requires an active GPS module.
* **Data Privacy**: All handling of Personally Identifiable Information (PII) and biometric data (face templates) must comply with relevant data protection regulations (e.g., GDPR, CCPA). Biometric data must be stored as encrypted templates, not raw images.
* **Internet Connectivity**: Real-time synchronization, notifications, and attendance marking require a stable internet connection on both client and server sides.

**2.6 User Documentation**

The following user documentation will be delivered with the software:

* **Admin User Manual**: A guide for administrators on how to configure the system, manage users, and generate reports.
* **End-User Guide**: A simple guide for students/employees on how to register, enroll their face, and mark their attendance.
* **Online Help**: Context-sensitive help and FAQs integrated within the web/mobile application.

**2.7 Assumptions and Dependencies**

* **Assumptions**:
  + Users will have access to a compatible device (PC with a webcam or a smartphone).
  + The environment for face recognition will have adequate lighting conditions.
  + Users consent to providing their location and biometric data for attendance purposes.
  + GPS accuracy is sufficient within the designated campus/office area.
* **Dependencies**:
  + The project depends on the continued availability and pricing models of third-party APIs (Google Maps, Twilio).
  + The performance of ML features is dependent on the quality and quantity of training data.
  + The project relies on open-source libraries like OpenCV, TensorFlow, and others, which are subject to their own licensing and maintenance schedules.

**3. External Interface Requirements**

**3.1 User Interfaces**

The system will feature a clean, modern, and responsive user interface.

* **Login Screen**: A secure entry point for both Admins and Users with fields for username, password, and an OTP/2FA prompt.
* **Admin Dashboard**: A primary view for Admins showing aggregate attendance statistics, live counts, a list of active sessions, and links to management modules. It will feature graphs, charts, and key performance indicators.
* **User Dashboard**: A personalized view for Users showing their own attendance percentage, a calendar heatmap of their presence/absence, and a log of their recent check-ins.
* **Attendance Marking Interface (Mobile/Web)**: An interface that activates the device camera for face recognition or QR scanning. It will provide real-time feedback (e.g., "Success," "Outside Geo-Fence," "Face Not Matched").
* **Report Generation Screen**: An Admin-only interface with filters (date range, user, course) to generate and download reports.

**3.2 Hardware Interfaces**

* **Webcam/Smartphone Camera**: The system will interface with standard webcams and smartphone cameras (front and back) to capture images for face recognition and videos for QR code scanning.
* **GPS Module**: The system will interface with the GPS hardware on users' smartphones to obtain precise location data for geo-fencing validation.
* **NFC/RFID Reader (Optional)**: For future enhancements, the system may interface with standard NFC/RFID card readers connected via USB or integrated into a dedicated terminal.

**3.3 Software Interfaces**

* **Database Interface**: The application will interface with a Firebase/MongoDB/MySQL database using a standard database driver or ORM/ODM to perform CRUD (Create, Read, Update, Delete) operations.
* **Google Maps API**: The system will make API calls to the Google Maps service to define and validate geographical boundaries (geofences).
* **Twilio SMS API**: The system will interface with the Twilio REST API to send SMS notifications for events like absenteeism.
* **OpenCV Library**: The system will utilize the OpenCV library for all image processing tasks, including face detection, image normalization, and feature extraction.
* **TensorFlow/Scikit-learn**: The system will interface with these ML libraries to load trained models and perform predictions for face recognition and anomaly detection.

**3.4 Communications Interfaces**

* **Protocol**: All communication between the client (web/mobile app) and the server will use the **HTTPS** protocol to ensure data encryption and security.
* **API Format**: The backend will expose a **RESTful API** using JSON (JavaScript Object Notation) as the data interchange format.
* **Email Notifications**: The system will use the **SMTP** (Simple Mail Transfer Protocol) to send email notifications.
* **Push Notifications**: Mobile applications will use platform-specific push notification services (e.g., Firebase Cloud Messaging) to deliver real-time alerts.

**4. System Features (Functional Requirements)**

**4.1 Feature 1: User Authentication & Management**

* **4.1.1 Description and Priority**: Provides secure mechanisms for users to register, log in, and manage their profiles. **Priority: High.**
* **4.1.2 Stimulus/Response Sequences**:
  + *Stimulus*: A new user submits their registration details.
  + *Response*: The system validates the details, creates a new user account, and sends a verification email.
  + *Stimulus*: A user enters their credentials and a valid OTP.
  + *Response*: The system authenticates the user and grants access to their respective dashboard.
* **4.1.3 Functional Requirements**:
  + **FR-1.1**: The system shall allow new users to register by providing a name, email, role, and password.
  + **FR-1.2**: The system shall enforce password complexity rules (e.g., minimum length, character types).
  + **FR-1.3**: The system shall support secure login using email/password and a second factor (OTP via email or an authenticator app).
  + **FR-1.4**: The system shall provide a "Forgot Password" functionality that allows users to reset their password securely.
  + **FR-1.5**: Users shall be able to view and edit their own profile information (name, contact details).
  + **FR-1.6**: Admins shall have the ability to create, update, and deactivate user accounts.

**4.2 Feature 2: Role-Based Access Control (RBAC)**

* **4.2.1 Description and Priority**: Ensures that users can only access features and data appropriate for their role. **Priority: High.**
* **4.2.2 Stimulus/Response Sequences**:
  + *Stimulus*: An authenticated 'User' attempts to access the Admin report generation page.
  + *Response*: The system denies access and shows an "Unauthorized" error message.
* **4.2.3 Functional Requirements**:
  + **FR-2.1**: The system shall define two roles: 'Admin' and 'User'.
  + **FR-2.2**: The 'Admin' role shall have permissions to access all system functionalities, including user management, system configuration, and aggregate reporting.
  + **FR-2.3**: The 'User' role shall have permissions limited to marking their own attendance, viewing their personal history/dashboard, and managing their own profile.
  + **FR-2.4**: System APIs shall be protected to prevent unauthorized access based on user roles.

**4.3 Feature 3: Face Recognition Attendance**

* **4.3.1 Description and Priority**: Allows users to mark their attendance by using their face as a biometric identifier. **Priority: High.**
* **4.3.2 Stimulus/Response Sequences**:
  + *Stimulus*: A user initiates the face recognition process from their device's camera.
  + *Response*: The system captures the user's face, compares it against their enrolled biometric template, verifies liveness, and if successful, records the attendance with a timestamp.
* **4.3.3 Functional Requirements**:
  + **FR-3.1**: The system shall allow a user to enroll their face by capturing multiple images from different angles during a one-time setup process.
  + **FR-3.2**: The system shall store the facial data as a secure, encrypted biometric template, not as a raw image.
  + **FR-3.3**: The system shall use the device camera to perform live face detection.
  + **FR-3.4**: The system shall implement a liveness detection mechanism (e.g., blink detection, head movement challenge) to prevent spoofing attacks using static photos.
  + **FR-3.5**: The face matching algorithm shall have an accuracy of at least 98%.
  + **FR-3.6**: The system shall provide clear feedback to the user on success or failure of the recognition process.
* **4.3.4 Liveness Detection & Enrollment (Added v1.1)**  
  • Enrollment Procedure: Capture at least 5 face images at varying angles.  
  • Liveness Mechanism: Randomized challenge (blink, head-turn, smile).  
  • Failure Handling: 3 fails → require re-enrollment or fallback to QR.  
  • Data Retention: Delete raw images after template creation.

**4.4 Feature 4: QR Code Attendance**

* **4.4.1 Description and Priority**: Allows attendance to be marked by scanning a unique QR code. **Priority: High.**
* **4.4.2 Stimulus/Response Sequences**:
  + *Stimulus*: A user scans a QR code displayed by the Admin for a specific session.
  + *Response*: The system validates the QR code's authenticity and timeliness, and if valid, records the user's attendance.
* **4.4.3 Functional Requirements**:
  + **FR-4.1**: Admins shall be able to generate unique QR codes for a specific attendance session.
  + **FR-4.2**: The system shall support both static (reusable) and dynamic (single-use or time-limited) QR codes.
  + **FR-4.3**: Users shall be able to use their mobile app to scan the QR code.
  + **FR-4.4**: The system must validate that the QR code is valid for the current session and has not expired or been used before (in the case of dynamic codes).

**4.5 Feature 5: Geo-Fencing Constraint**

* **4.5.1 Description and Priority**: Restricts attendance marking to a predefined geographical boundary. **Priority: High.**
* **4.5.2 Stimulus/Response Sequences**:
  + *Stimulus*: A user attempts to mark attendance.
  + *Response*: The system fetches the user's current GPS coordinates. It compares the location with the predefined geofence. If the user is inside the boundary, the process continues; otherwise, it is blocked with a notification.
* **4.5.3 Functional Requirements**:
  + **FR-5.1**: Admins shall be able to define a geographical boundary (geofence) on a map for each attendance session (e.g., a campus, a specific building).
  + **FR-5.2**: Before allowing any attendance marking (Face, QR), the system must request and receive the user's current GPS location.
  + **FR-5.3**: The system must verify that the user's coordinates fall within the active session's geofence.
  + **FR-5.4**: The system shall deny the attendance request if the user is outside the geofence.

**4.6 Feature 6: Dashboard and Data Visualization**

* **4.6.1 Description and Priority**: Provides visual representations of attendance data for easy interpretation. **Priority: High.**
* **4.6.2 Stimulus/Response Sequences**:
  + *Stimulus*: An Admin logs in and navigates to the dashboard.
  + *Response*: The system retrieves and displays aggregate attendance data, such as overall percentage, absentee count for the day, and trends.
* **4.6.3 Functional Requirements**:
  + **FR-6.1**: The Admin dashboard shall display system-wide analytics, including overall attendance rates, most frequent absentees, and a comparison across different groups/courses.
  + **FR-6.2**: The User dashboard shall display personal analytics, including the user's individual attendance percentage and a log of their check-in/check-out times.
  + **FR-6.3**: The system shall display attendance data using a calendar heatmap, where days are colored based on presence or absence.
  + **FR-6.4**: The system shall use pie charts and bar graphs to visualize attendance data (e.g., present vs. absent ratio).

**4.7 Feature 7: Report Generation**

* **4.7.1 Description and Priority**: Allows Admins to generate and export formal attendance reports. **Priority: High.**
* **4.7.2 Stimulus/Response Sequences**:
  + *Stimulus*: An Admin selects a date range and a user group, then clicks "Generate Report."
  + *Response*: The system compiles the relevant attendance data and provides a downloadable file in the selected format (PDF/Excel).
* **4.7.3 Functional Requirements**:
  + **FR-7.1**: Admins shall be able to generate reports for specific date ranges.
  + **FR-7.2**: Admins shall be able to filter reports by user, course, department, or other custom tags.
  + **FR-7.3**: The system must be able to export reports in **PDF** format for official documentation.
  + **FR-7.4**: The system must be able to export reports in **CSV/Excel** format for further data analysis.

**4.8 Feature 8: ML-Based Anomaly Detection**

* **4.8.1 Description and Priority**: Uses machine learning to identify suspicious or unusual attendance patterns that might indicate proxy attempts or other issues. **Priority: Medium.**
* **4.8.2 Stimulus/Response Sequences**:
  + *Stimulus*: The system runs a scheduled batch job over the day's attendance logs.
  + *Response*: The ML model flags a user's attendance as an anomaly (e.g., logged in from two different locations in an impossible timeframe). The system alerts the Admin.
* **4.8.3 Functional Requirements**:
  + **FR-8.1**: The system shall use historical attendance data to train an ML model to understand a user's normal attendance pattern (e.g., typical time, location).
  + **FR-8.2**: The system shall analyze new attendance records against the trained model to detect deviations.
  + **FR-8.3**: Anomalies to be detected shall include, but are not limited to, impossible travel between check-ins, consistently late check-ins, and abnormal attendance frequency.
  + **FR-8.4**: The system shall provide a dashboard for Admins to review flagged anomalies and take action (e.g., approve, reject, investigate).

**4.9 Feature 9: Real-time Notifications**

* **4.9.1 Description and Priority**: Proactively informs stakeholders (e.g., parents, managers) about important attendance events. **Priority: Medium.**
* **4.9.2 Stimulus/Response Sequences**:
  + *Stimulus*: At the end of an attendance session, the system identifies a user as absent.
  + *Response*: The system triggers an API call to Twilio to send a pre-formatted SMS message to the user's registered emergency contact.
* **4.9.3 Functional Requirements**:
  + **FR-9.1**: Admins shall be able to enable or disable notifications for absenteeism.
  + **FR-9.2**: The system shall send real-time notifications via **SMS** and/or **Email** to a designated contact when a user is marked absent.
  + **FR-9.3**: Notification templates shall be customizable by the Admin.
  + **FR-9.4**: The system shall also send notifications to users for events like password resets or profile changes.

**5. Other Nonfunctional Requirements**

**5.1 Performance Requirements**

* **PERF-1 (Response Time)**: Web application pages must load in under 3 seconds. API responses for typical requests must be completed in under 500ms.
* **PERF-2 (Biometric Speed)**: The face recognition process (capture, match, response) should complete within 2 seconds under optimal conditions.
* **PERF-3 (Availability)**: The system shall have a service availability of **99.5%** or higher, excluding planned maintenance windows.
* **PERF-4 (Scalability)**: The system architecture must be capable of supporting 1,000 concurrent users with a response time degradation of no more than 15%.
* **PERF-5 (Capacity)**: The system must be able to store and process attendance records for at least 10,000 users over a 5-year period without significant performance degradation.

**5.2 Safety Requirements**

* **SAFE-1**: The system is a software-only application and has no direct safety requirements related to causing physical harm. Data safety is covered under Security Requirements.

**5.3 Security Requirements**

* **SEC-1 (Data Encryption)**: All data, especially PII and biometric templates, must be encrypted both at rest (using AES-256 or equivalent) and in transit (using TLS 1.2 or higher).
* **SEC-2 (Authentication)**: The system must enforce strong password policies and support Two-Factor Authentication (2FA) via OTP.
* **SEC-3 (Authorization)**: Access to all system resources (APIs, data) must be strictly controlled through a Role-Based Access Control (RBAC) mechanism.
* **SEC-4 (Data Privacy)**: Biometric face data must be stored as irreversible, encrypted mathematical templates. Raw images must be deleted immediately after processing for enrollment or authentication.
* **SEC-5 (Vulnerability Prevention)**: The application must be protected against common web vulnerabilities, including but not limited to the OWASP Top 10 (e.g., SQL Injection, Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF)).
* **SEC-6 (Audit Trails)**: The system must maintain an immutable log of all critical actions, such as login attempts, changes to user permissions, and report generation.

**5.4 Software Quality Attributes**

* **USABILITY-1**: The user interface shall be intuitive and require minimal training for both user classes. A new user should be able to enroll and mark attendance within 5 minutes.
* **RELIABILITY-1**: The system must handle errors gracefully and provide informative error messages to the user. The Mean Time Between Failures (MTBF) should be greater than 200 hours of continuous operation.
* **MAINTAINABILITY-1**: The source code shall be modular, well-commented, and adhere to established coding standards for the chosen programming language to facilitate easy debugging and future enhancements.
* **PORTABILITY-1**: The web application must be compatible with the latest two versions of all major web browsers. The backend server application should be containerized (e.g., using Docker) for easy deployment across different environments.
* **INTEROPERABILITY-1**: The system shall provide a well-documented REST API to allow for future integration with third-party systems like Learning Management Systems (LMS) or Enterprise Resource Planning (ERP) software.

**5.5 Business Rules**

* **BR-1**: An "attendance session" is defined by a start time, end time, and an optional geofence boundary, all configured by an Admin.
* **BR-2**: A user can be marked as "Present" only once per defined session.
* **BR-3**: Attendance marking is only permitted within the active time window of a session.
* **BR-4**: Admin-level privileges are required to modify any system configuration, user role, geofence, or to manually override an attendance record.
* **BR-5**: An account will be locked for a predefined period (e.g., 15 minutes) after a specified number (e.g., 5) of failed login attempts.