Software Requirements Specification (SRS)

Project: Smart Attendance Management System with Face Recognition

1. Introduction

1.1 Purpose

The purpose of this project is to design and develop a **Smart Attendance Management System** that uses **Face Recognition Technology** to automate attendance recording in educational institutions. This system aims to replace manual methods, reduce errors, and save time by providing a **fast, secure, and reliable solution** for attendance management.

1.2 Scope

The system will capture student images using a webcam or camera-enabled device, recognize their faces using machine learning algorithms (OpenCV/Deep Learning), and mark attendance in the database.

- **Users:** Admin, Teachers, Students.
- **Key Functions:** Face registration, face recognition, attendance marking, report generation, role-based access.
- **Exclusions:** This system will not handle payroll, fee management, or non-academic data.

1.3 Definitions, Acronyms, and Abbreviations

- **SRS** – Software Requirements Specification
- **DBMS** – Database Management System
- **FR** – Functional Requirement
- **NFR** – Non-Functional Requirement
- **GUI** – Graphical User Interface

1.4 References

- IEEE Standard 830-1998 for SRS documentation
- OpenCV documentation (https://opencv.org)
- Spring Boot official documentation (https://spring.io/projects/spring-boot)

1.5 Overview

This document specifies the complete requirements for the Smart Attendance System. It covers system features, constraints, interfaces, and performance requirements.

2. Overall Description

2.1 Product Perspective

The Smart Attendance Management System is a **standalone application** but can be integrated with an institution's ERP. It requires a camera-enabled device, database (MySQL), and backend processing (Java + Spring Boot).

2.2 Product Features

- Automated attendance using face recognition.
- Role-based login (Admin, Teacher, Student).
- Real-time attendance recording.
- Attendance reports (daily, monthly, semester-wise).
- Secure database storage.

2.3 User Characteristics

- **Admin:** Manages student/teacher records, system settings.
- **Teacher:** Views attendance reports, verifies attendance.
- **Student:** Views personal attendance records.

2.4 Constraints

- Requires stable internet connection.
- Works only with camera-enabled devices.
- Limited to facial recognition accuracy based on dataset quality.

2.5 Assumptions and Dependencies

- All users must be pre-registered in the system.

- Users must cooperate during face registration.
- The system depends on MySQL database and Java environment.

3. System Features (Functional Requirements)

- **FR1:** System shall allow Admin to register students and capture face data.
- **FR2:** System shall detect and recognize student faces in real-time.
- **FR3:** System shall mark attendance automatically in the database.
- **FR4:** System shall generate reports (daily, monthly, semester-wise).
- **FR5:** System shall allow Teachers to view and verify attendance.
- **FR6:** System shall allow Students to view personal attendance records.
- **FR7:** System shall provide a secure login system with different roles.

---

4. External Interface Requirements

4.1 User Interfaces

- **Admin Dashboard:** Manage users, view reports.
- **Teacher Dashboard:** View attendance, download reports.
- **Student Panel:** Check attendance status.

4.2 Hardware Interfaces

- Camera/webcam for image capturing.
- Server/PC for processing and storage.

4.3 Software Interfaces

- **Frontend:** HTML, CSS, JavaScript.
- **Backend:** Java, Spring Boot, Java Servlets.
- **Database:** MySQL.
- **Libraries:** OpenCV (for face recognition).

4.4 Communication Interfaces

- HTTP/HTTPS protocols for data transmission.
- Local network/internet required.

---

5. Non-Functional Requirements

- **Performance:** System shall mark attendance within 2 seconds per student.
- **Security:** Only authenticated users can access data, passwords encrypted.
- **Usability:** User-friendly GUI for non-technical staff.
- **Reliability:** At least 99% uptime with fault tolerance.
- **Scalability:** Should support up to 5000 students simultaneously.
- **Portability:** Should run on Windows/Linux environments.

6. Other Requirements

- Must comply with **Data Privacy Regulations** for biometric data.
- Backup mechanism for database.
- System logs for all activities.

---

7. Appendices

7.1 Glossary

- **Face Recognition:** Technology that identifies or verifies a person using their face.
- **Dataset:** A collection of stored images used for training and testing recognition models.

📡 **Protocols Used in Smart Attendance Management System with Face Recognition**

🔹 **1. HTTP / HTTPS (HyperText Transfer Protocol / Secure)**

📌 **Purpose:**

- Used for **communication between client (browser)** and **server (backend)**.

📌 **Where it is used:**

- When user logs in, submits face data, views attendance, or requests reports.

- All frontend (HTML/JS) requests to backend (Java/Spring Boot) use HTTP/HTTPS.

✅ **HTTPS is preferred over HTTP because:**

- It uses **SSL/TLS encryption**.
- Provides **data confidentiality and integrity**.
- Protects user credentials and biometric data.

🔹 **2. RESTful API Protocol (Representational State Transfer)**

📌 **Purpose:**

- Used for **data exchange between frontend and backend**.
- Follows REST architecture using HTTP methods:
  - GET – retrieve data (attendance records)
  - POST – send data (register face data, mark attendance)
  - PUT – update records (student info)
  - DELETE – delete data (remove user)

📌 **Where it is used:**

- Spring Boot backend exposes **REST APIs** that are consumed by frontend or Android app.

🔹 **3. JDBC Protocol (Java Database Connectivity)**

📌 **Purpose:**

- Used by the **Java backend** to communicate with the **MySQL database**.

📌 **Where it is used:**

- To insert, update, delete, and retrieve student and attendance records.
- JDBC acts as a **bridge between Java and SQL**.

🔹 **4. TCP/IP (Transmission Control Protocol / Internet Protocol)**

📌 **Purpose:**

- Provides **low-level network communication**.
- Ensures reliable transmission of data packets between client, server, and database.

📌 **Where it is used:**

- Underlies HTTP, HTTPS, and REST.
- Used when sending attendance data from client to server over LAN/Wi-Fi.

🔹 **5. SSL/TLS (Secure Socket Layer / Transport Layer Security)**

📌 **Purpose:**

- Provides **encryption and authentication** for HTTPS connections.

📌 **Where it is used:**

- When students/teachers log in.
- When biometric data is sent to server.
- Ensures **data security and privacy**.

🔹 **6. JSON (JavaScript Object Notation)**

📌 **Purpose:**

- Lightweight data exchange format used in REST APIs.

📌 **Where it is used:**

- Backend sends/receives data in JSON format:

📌 **Why These Protocols Are Important in This Project:**

1. 🛡️ **Security** → HTTPS + SSL/TLS protect user credentials & biometric data.
2. 📡 **Reliable Communication** → TCP/IP ensures all requests/responses are delivered correctly.
3. ⚙️ **Interoperability** → REST + JSON allow integration with mobile apps or third-party systems.
4. 💾 **Database Access** → JDBC ensures backend can store and retrieve attendance data efficiently.

5. 📷 (Optional) Protocol Architecture Diagram

6. You can include a simple architecture diagram in your project report like this:

```
[Web Browser / Android App]
          |
      HTTPS/REST
          |
[Spring Boot Backend (Java)]
          |
       JDBC Protocol
```

```
14.          |
15.    [MySQL Database]
```

1. ER Diagram (Entity Relationship Diagram)

Entities and their relationships in the database.

Entities:

- **Student (Student_ID, Name, Roll_No, Email, FaceData)**
- **Teacher (Teacher_ID, Name, Email, Subject)**
- **Admin (Admin_ID, Name, Email, Role)**
- **Attendance (Attendance_ID, Date, Time, Status, Student_ID, Course_ID)**
- **Course (Course_ID, Course_Name, Teacher_ID)**

Relationships:

- A **Student** is enrolled in one or many **Courses**.
- A **Course** is taught by one **Teacher**.
- **Attendance** is recorded for each **Student** in a **Course**.
- **Admin** manages Students, Teachers, and Attendance.

📌 **ER Diagram (Text Sketch):**

Student ----------< Attendance >---------- Course >----- Teacher

| (Student_ID PK) (Attendance_ID PK) | (Teacher_ID PK)

| Name Date | Name

| Roll_No Time | Email

| Email Status | Subject

| FaceData Student_ID (FK) | Course_ID (FK) |

|

Admin

|

(Manages Students, Teachers, Courses)

---

2. Use Case Diagram

Actors: **Admin, Teacher, Student, Camera System (AI model)**

Use Cases:

- Admin → Register Student, Manage Database, Generate Reports
- Teacher → View Attendance, Validate Attendance
- Student → View Attendance Status
- Camera System → Detect & Recognize Faces, Mark Attendance

📌 **Use Case (Text Sketch):**

[Admin] ---------- (Register Student)

| ---------- (Generate Reports)

| ---------- (Manage Database)

[Teacher] ---------- (View Attendance)

| ---------- (Validate Attendance)

[Student] ---------- (View Attendance Status)

[Camera System] -------- (Detect Face) ----> (Mark Attendance in DB)

---

3. Data Flow Diagram (DFD)

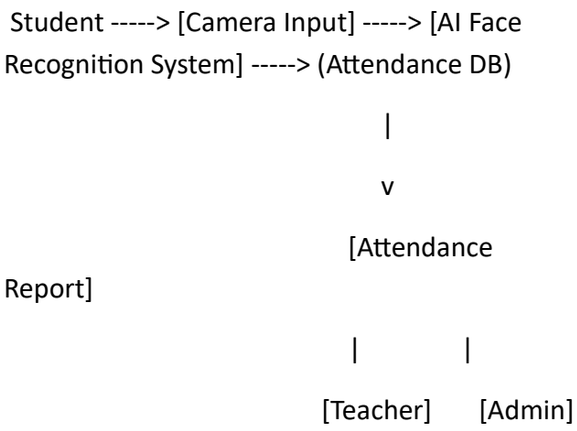
Level 0 (Context Level):

- Camera → Face Recognition System → Database → Reports

Level 1:

- Student Face Data captured → AI Model → Attendance Marked → Stored in DB
- Teacher retrieves report from DB
- Admin manages student/course data

📌 **DFD (Text Sketch):**



4. **File Structure Diagram**

```
SmartAttendanceSystem/
|
├── backend/
|   ├── src/
|   |   ├── controllers/        # Java classes for handling HTTP requests
|   |   ├── models/             # Java POJOs representing DB entities
|   |   ├── services/           # Business logic and face recognition service
|   |   ├── repositories/       # Database access logic
|   |   └── utils/              # Utilities such as image processing
|   ├── resources/
|   |   ├── application.properties # DB config, server properties
|   └── pom.xml or build.gradle   # Build configuration
|
├── frontend/
|   ├── css/
|   ├── js/
|   ├── images/
|   ├── index.html            # Student login page
|   ├── faculty.html          # Faculty dashboard
|   ├── attendance.html       # Attendance marking page
|
├── database/
|   └── schema.sql            # SQL script to create DB schema
|
└── documentation/
    ├── project-report.docx     # Full documentation report
    ├── module-diagrams.png     # Architecture and module diagrams
    └── user-manual.pdf         # Instructions for users and faculty
```

5. Sequence Diagram

📌 Shows the step-by-step flow of attendance marking.