Software Requirements Specification

for

Face Recognition Based Attendance System

Version 1.0 approved

Prepared by

23th January 2024

Table of Contents

Table of Contents ii

Revision History ii

1. Introduction 1

1.1 Purpose 1

1.2 Document Conventions 1

1.3 Intended Audience and Reading Suggestions 1

1.4 Product Scope 2

1.5 References 2

2. Overall Description 3

2.1 Product Perspective 3

2.2 Product Functions 3

2.3 User Classes and Characteristics 3

2.4 Operating Environment 4

2.5 Design and Implementation Constraints 4

2.6 User Documentation 4

2.7 Assumptions and Dependencies 4

3. External Interface Requirements 5

3.1 User Interfaces 5

3.2 Hardware Interfaces 5

3.3 Software Interfaces 5

3.4 Communications Interfaces 5

4. System Features 6

4.1 System Feature 1 6

4.2 System Feature 2 (and so on) 6

5. Other Nonfunctional Requirements 7

5.1 Performance Requirements 7

5.2 Safety Requirements 7

5.3 Security Requirements 7

5.4 Software Quality Attributes 7

5.5 Business Rules 7

6. Other Requirements 8

Appendix A: Glossary 8

Appendix B: Analysis Models 8

Appendix C: To Be Determined List 8

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
|  |  |  |  |
|  |  |  |  |

# Introduction

## Purpose

In today's fast-paced academic environment, maintaining accurate records of student attendance is a crucial part of any institution’s administrative responsibilities. Traditional attendance methods such as manual sign-in sheets, roll calls, or biometric fingerprinting are often time-consuming, error-prone, and susceptible to manipulation. In response to these challenges, this Software Requirements Specification (SRS) outlines the functional and non-functional specifications of a robust, intelligent, and secure **Student Attendance System** powered by **AI-based face recognition technology**.

The main purpose of this SRS is to document a complete and clear understanding between stakeholders and developers regarding the functionalities, design considerations, and constraints of the proposed attendance system. It sets the groundwork for system analysis, design, implementation, and validation by defining all necessary specifications.

The system aims to automate the attendance process using facial recognition algorithms to improve reliability, reduce administrative burden, and ensure a proxy-free environment. By capturing student facial features in real-time using a webcam and storing these features in a centralized MySQL database, the system will facilitate daily attendance marking in classrooms with high accuracy and speed.

This SRS also serves as a reference for further software maintenance, future upgrades, and system enhancement opportunities. It is a comprehensive document that includes system behavior, external interfaces, design constraints, and the expected user interaction with the product.

## Document Conventions

To ensure clarity and consistency throughout the document, the following conventions have been adopted:

* Section headings are denoted by hierarchical decimal numbering (e.g., 1.1, 2.3.1) for ease of navigation and referencing.
* **Bold** text is used for section headers and important terms that require emphasis.
* *Italicized* text is used for book titles, module references, or external systems.
* Monospaced font is used to represent database queries, code snippets, and file names.
* Lists are used to enumerate features, requirements, and steps in a structured manner.
* Diagrams and tables (to be added in appendices or relevant sections) are referred to using appropriate figure/table numbers.

## Intended Audience and Reading Suggestions

This document is primarily intended for the following audiences:

* **Software Developers and Engineers**: To understand the system architecture, required algorithms, programming languages (Python), libraries (OpenCV, Flask), and database structures to be used in development.
* **Project Managers and Academic Administrators**: For planning and monitoring development phases, ensuring milestones are met, and aligning the product with institutional goals.
* **Test Engineers**: To derive relevant test cases from functional and non-functional requirements and perform verification and validation testing accordingly.
* **Database Administrators (DBAs)**: To manage, query, and maintain MySQL databases and provide backup and restore strategies.
* **End Users (Faculty, Admin Staff)**: To understand the system interface and learn how to interact with the system for attendance marking, report generation, and student management.
* **Students**: As the beneficiaries of the system, their interaction is mostly indirect, but the accuracy and transparency offered by the system directly impact their academic record.

**Reading Suggestions:**

* Sections 1 and 2 offer a high-level overview and should be read first by all stakeholders.
* Developers and testers should pay close attention to Sections 3 and 4, which detail the interface requirements and system features.
* Project managers may focus on Sections 1, 2, and 5 to evaluate system quality and constraints.
* Appendices can be referred to for technical references, terminologies, and pending decisions.

## Product Scope

The **Student Attendance System** is a standalone, web-based software solution integrated with AI-based facial recognition capabilities. The system is designed to modernize the attendance process in schools, colleges, and universities by replacing manual or semi-digital methods with a contactless, real-time solution.

This system consists of multiple modules such as:

* **Admin Authentication Module** – Ensures only authorized staff can access administrative features such as student registration, attendance overview, and data export.
* **Face Registration Module** – Captures multiple facial images of each student using a webcam and stores them in a dataset folder mapped to their Student ID.
* **Face Recognition Module** – During lectures, the system uses a real-time webcam feed to identify registered faces from the class and automatically logs attendance in the MySQL database.
* **Manual Attendance Entry** – For edge cases where a student’s face is not recognized, teachers can manually mark attendance through a secure panel.
* **Report Generation Module** – Attendance reports can be generated in Excel or PDF formats and filtered by date, subject, or student.
* **Notification and Logging System** – Logs all attendance activities and optionally sends alerts if certain thresholds are missed (e.g., low attendance).

**Key benefits of the system include:**

* Eliminates proxy attendance and impersonation.
* Increases teacher productivity.
* Maintains error-free digital records.
* Ensures transparency for academic audit purposes.

The product is scalable and can be deployed across various classrooms, departments, or even entire educational campuses.

## Reference

The following sources and tools were referred to during the design and development of this system:

| **Reference** | **Description** |
| --- | --- |
| IEEE SRS Template | IEEE Standard 830-1998 guidelines followed for document structure. |
| Python 3.10 | Programming language used for backend development and AI integration. |
| OpenCV Library | Used for real-time image processing and facial recognition. |
| Haarcascade Frontalface Classifier | Pre-trained model used to detect faces in image frames. |
| Flask Microframework | Lightweight Python web framework used to build the application. |
| MySQL Database | Relational database used to store student details and attendance records. |
| VS Code | Integrated Development Environment (IDE) used for code development. |
| W3Schools/GeeksforGeeks | Referenced for HTML, CSS, and Python integration techniques. |
| GitHub | Used for version control and project repository management. |

# Overall Description

## Product Perspective

The **Student Attendance System using Face Recognition** is an independent web-based application that functions as a modern alternative to traditional attendance recording systems. Although it can be integrated with existing school or college management systems, it is primarily designed to function autonomously.

The system uses a combination of **Python**, **OpenCV**, and **MySQL** for the backend operations and **HTML/CSS** for a user-friendly frontend interface. The use of **Flask** as a lightweight web framework ensures seamless communication between the backend and frontend.

From a technical standpoint, the system consists of the following layers:

1. **Presentation Layer (Frontend)**:
   * Responsible for user interaction.
   * Developed using HTML and CSS to offer an intuitive GUI for admin login, student registration, and attendance overview.
2. **Application Layer (Backend)**:
   * Built using Python and Flask.
   * Handles routing, database communication, face recognition logic, and session control.
3. **Database Layer**:
   * MySQL database manages persistent storage.
   * Stores student records, attendance logs, authentication data, and class schedules.
4. **Face Recognition Engine**:
   * Uses Haar Cascade classifiers and the face\_recognition Python library.
   * Encodes facial features and matches them in real time against stored data.

This modular approach allows for future scalability and easier maintenance. Whether for a single classroom or an entire institution, the system can be expanded with minimal architectural changes.

## Product Functions

The system performs the following key functions:

* **User Authentication**:
  + Allows only authorized admins or faculty members to log in using username and MPIN credentials.
* **Student Registration**:
  + Enables the admin to register new students by entering their details and capturing multiple facial images through the webcam.
  + Facial data is stored in a dataset folder for model training.
* **Model Training**:
  + Converts the dataset into encodings using machine learning.
  + Trained models are saved and used during attendance sessions.
* **Automated Attendance Marking**:
  + On starting a session, the webcam scans student faces in real time.
  + Matches detected faces with stored encodings and marks their attendance automatically in the database.
* **Attendance Report Generation**:
  + Generates daily/monthly/yearly attendance reports.
  + Filters available by student name, subject, or date.
  + Exports reports in Excel or PDF format.
* **Manual Override Feature**:
  + Allows teachers to manually update attendance in case of errors or unrecognized faces.

## User Classes and Characteristics

The system is designed for three major user types:

1. **Administrator / Faculty**:
   * Responsible for registering students, managing class schedules, and reviewing attendance reports.
   * Requires basic computer literacy.
   * Has full access to all modules and system settings.
2. **Students**:
   * The primary subject of attendance records.
   * Not required to interact directly with the system.
   * Their facial data is used for recognition and attendance logging.
3. **Database Manager / Developer (Optional Role)**:
   * Handles system installation, configuration, and maintenance.
   * Updates software modules or database structures when needed.

Each user class has a different level of access and interaction, ensuring data integrity and role-based security.

## Operating Environment

The proposed system is intended to run in the following environment:

* **Operating System**: Windows 10/11, macOS, or any Linux-based OS.
* **Python Version**: 3.9 or higher.
* **Database**: MySQL Server (Running on Port 3307).
* **Web Framework**: Flask (with support for routing and session management).
* **Development Tools**: Visual Studio Code (VS Code), XAMPP for MySQL management (phpMyAdmin).
* **Hardware Requirements**:
  + Minimum 4 GB RAM.
  + Webcam (built-in or external) with at least 720p resolution.
  + Dual-core processor or better.

All operations are performed on the local network or system, though it can be configured for cloud-based deployment with minimal changes.

## Design and Implementation Constraints

The system is developed under the following constraints:

* **Platform Dependency**: The facial recognition model is optimized for Python and OpenCV. Porting to other platforms (like Android or iOS) may require separate frameworks like TensorFlow Lite or OpenCV Mobile.
* **Real-time Processing**: The system must operate with minimal delay (< 2 seconds per recognition).
* **Security Requirements**: MPIN-based login is enforced to avoid unauthorized access.
* **Data Storage**: Image data and attendance logs must be securely stored to avoid loss during power failure or system crashes.
* **Model Training Time**: Depending on the dataset size, model training can take several minutes, so this should be done only after major data updates.

## User Documentation

* + **User Manual:** Instructions for teachers and administrators.
  + **Online Help**: Contextual help within the application.
  + **Tutorials:** Step-by-step guides for using the system.

## Assumptions and Dependencies

**Assumptions made during system development:**

* All students will register their faces clearly under proper lighting conditions.
* The webcam used is functional and of reasonable quality.
* Faculty or admin personnel are trained to operate the software.
* MySQL Server is installed, running, and accessible at port 3307.
* The system will primarily be accessed from a single machine or LAN environment.

**Dependencies:**

* Python libraries such as opencv-python, face\_recognition, numpy, and flask must be pre-installed.
* The Haar cascade file haarcascade\_frontalface\_default.xml must be present in the specified directory.
* Localhost server must be running for the Flask app to be accessible through a browser.

# External Interface Requirements

This section outlines the **interactions between the Student Attendance System and external components**, including users, hardware devices (like webcams), databases, and software interfaces. These requirements ensure seamless communication and functionality between the system modules and the external environment.

## User Interfaces

The system offers a **Graphical User Interface (GUI)** that allows users to interact with the system efficiently. The interface is designed to be **simple, clean, and intuitive**, ensuring ease of use even for non-technical users.

### Login Interface

**Fields**: Username and MPIN (numeric PIN).

**Functionality**: Verifies login credentials from the MySQL database.

**Design**: Clean login form with centered layout and attractive CSS styles,including animations and hover effects.

**Validation**: Displays error messages for incorrect credentials.

### Admin Dashboard

**Purpose**: Main control panel after login.

**Functions Available**:

Register new students.

View and edit student data.

Train model for face recognition.

Launch face recognition attendance.

Generate reports.

**Design**: User-friendly cards or buttons for each operation, with icons for quick recognition.

### Student Registration Page

**Fields**:

Full Name

Roll Number

Class/Division

Subject (Optional)

**Face Capture**:

Uses the system webcam.

Captures multiple face images per student.

**Feedback**: Shows real-time camera feed with bounding box.

### Attendance Interface

**Real-Time Camera Feed**:

Automatically detects and identifies faces.

Displays the student name and status ("Present") on screen.

**Manual Override**:

Faculty can manually mark or edit entries if face is not recognized.

### Report Interface

**Features**:

Filter by date, student, or subject.

Export as Excel/PDF.

**Design**: Tabular layout with search, pagination, and sorting features.

## Hardware Interfaces

The system requires minimal hardware integration but relies on key components to function correctly.

**3.2.1 Webcam**

**Type**: Internal or external webcam with at least 720p resolution.

**Usage**:

Student face registration.

Real-time face recognition during attendance.

**Interface**:

Interacts with Python using OpenCV (cv2.VideoCapture()).

Streams video to the recognition module for processing.

**3.2.2 Computer System**

**Requirements**:

Minimum 4 GB RAM, i3 processor or better.

USB ports for external webcams (if needed).

**Display**: 15.6" screen preferred for better UI visibility during attendance.

## Software Interfaces

The system interacts with several external libraries and platforms to ensure smooth operation.

**3.3.1 Python Libraries**

* **OpenCV** (cv2): For accessing the webcam, detecting and processing facial images.
* **face\_recognition**: Main facial recognition library used to encode and compare facial data.
* **NumPy**: For handling numerical arrays used in image processing.
* **Flask**: Backend web framework used to serve HTML pages and APIs.
* **Pandas/XlsxWriter**: For generating Excel reports.

**3.3.2 MySQL**

* **Database Software**: XAMPP with MySQL and phpMyAdmin.
* **Port**: Runs on port 3307 (custom configuration).
* **Communication**:

Python connects using mysql.connector.

Stores login details, student information, and attendance logs.

**3.3.3 Web Browsers**

* **Supported Browsers**:

Google Chrome (preferred)

Microsoft Edge

Mozilla Firefox

* **Role**:

Used to access the frontend GUI served via Flask.

Admin interacts with the system through browser-based interfaces.

## Communications Interfaces

The system uses local communication protocols for all its components since it's designed to run primarily on a local machine or local network.

### HTTP Protocol

* The web interface uses **HTTP requests** to interact with the Flask server.
* Communication is routed through localhost:5000 by default.
* All pages and form submissions are managed through GET/POST requests.

### Database Communication

* Communication between the backend and MySQL occurs through the mysql-connector-python driver.
* All queries, including SELECT, INSERT, UPDATE, and DELETE operations, are executed securely.

### File System Interface

**The system stores:**

* Facial image data in the dataset/ folder.
* Trained model files in trainer/.
* Exported reports in the exports/ folder (Excel/PDF).
* The Python backend has direct file system access to these directories for reading and writing operations.

# System Features

## Feature 1: Face Recognition Attendance System

**4.1.1 Description and Priority**

The **Face Recognition Attendance System** is the primary feature of this software. It allows automatic identification of registered students using a webcam and records their attendance in real-time without requiring manual input from the user.

**Priority**: High (Core Functionality)

**User Role**: Faculty/Admin

**Dependencies**: OpenCV, Haarcascade, face\_recognition library, webcam hardware.

**4.1.2 Stimulus/Response Sequences**

| **Stimulus** | **System Response** |
| --- | --- |
| Faculty clicks “Start Attendance” | System opens webcam and begins live face detection. |
| A student stands in front of camera | The system detects and encodes the face in real-time. |
| A match is found in the database | System logs attendance in the MySQL database and overlays “Present” on screen. |
| No match is found | System notifies: “Face not recognized. Please try again or use manual entry.” |
| Session ends | System saves all attendance records and optionally exports a session report. |

**4.1.3 Functional Requirements**

**REQ-FR-001**: System must open webcam and initiate video capture within 3 seconds of the “Start” command.

**REQ-FR-002**: Detected faces must be matched against stored encodings with at least 70% confidence.

**REQ-FR-003**: Attendance entries must include timestamp, student ID, and subject name.

**REQ-FR-004**: Duplicate entries must be avoided in a single session.

**REQ-FR-005**: Unrecognized faces must trigger an alert and log the attempt.

## Feature 2: Manual Attendance Entry Panel

### Description and Priority

This feature allows faculty to **manually mark attendance** for students whose faces were not detected or recognized during the automated process.

**Priority**: Medium

**User Role**: Admin/Faculty

**Dependencies**: GUI Form, database connection

### Stimulus/Response Sequences

| **Stimulus** | **System Response** |
| --- | --- |
| Faculty clicks “Manual Entry” | A form opens to enter Student ID and attendance status. |
| ID is entered and submitted | System verifies student existence and logs attendance if valid. |
| Invalid ID is entered | System shows error: “Student not found.” |

### Functional Requirements

**REQ-FR-006**: System must validate student ID before updating attendance.

**REQ-FR-007**: Manual entries must not overwrite automatic entries unless authorized.

**REQ-FR-008**: Manual logs should be flagged separately in the database for reporting.

## Feature 3: Student Registration & Face Capture

### Description and Priority

This module allows faculty to **register a new student** into the system by inputting personal information and capturing multiple face images using the webcam.

**Priority**: High

**User Role**: Admin

**Dependencies**: GUI Form, Webcam, File System

### Stimulus/Response Sequences

| **Stimulus** | **System Response** |
| --- | --- |
| Student details are filled | System validates all fields. |
| Webcam starts and captures images | Images are saved to dataset/ directory under folder named by Student ID. |

### Functional Requirements

**REQ-FR-009**: Admin must input at least 5 fields: Name, Roll Number, Class, Subject, and Unique ID.

**REQ-FR-010**: System must capture and save at least 20 face images per student.

**REQ-FR-011**: All data must be stored in the database and associated with image datasets.

## Feature 4: Face Recognition Model Training

### Description and Priority

After registering students, the system must **train a machine learning model** to recognize each face from the dataset.

**Priority**: High

**User Role**: Admin/Developer

**Dependencies**: face\_recognition library, OS file system

### Stimulus/Response Sequences

| **Stimulus** | **System Response** |
| --- | --- |
| Admin clicks “Train Model” | System loads all images from the dataset folder. |
| System encodes facial data | Encoded data is mapped to student IDs and stored. |
| Model is saved | Encoded data is stored in a pickle or .yml file for recognition use. |

### Functional Requirements

**REQ-FR-012**: Training module must scan all dataset folders recursively.

**REQ-FR-013**: Encodings must be stored in structured format with student metadata.

**REQ-FR-014**: Model must be updated each time a new student is added.

## Feature 5: Report Generation and Export

### Description and Priority

This feature enables the admin or faculty to **generate attendance reports** that can be filtered and exported as Excel or PDF files.

**Priority**: High

**User Role**: Admin

**Dependencies**: Pandas, XlsxWriter, database

### Stimulus/Response Sequences

| **Stimulus** | **System Response** |
| --- | --- |
| Admin clicks “Generate Report” | System displays filter options (date, subject, student). |
| Filter is applied | Filtered attendance data is retrieved from the database. |
| “Export to Excel” clicked | Report is downloaded as .xlsx file. |

### Functional Requirements

**REQ-FR-015**: User must be able to generate reports for specific time periods or students.

**REQ-FR-016**: Exported reports must contain headers, total present/absent count, and percentage.

**REQ-FR-017**: Export options must support both Excel (.xlsx) and PDF format.

# Other Nonfunctional Requirements

authorized users. This section outlines the **nonfunctional aspects** of the Student Attendance System that define how the system operates, rather than what it does. These requirements include system performance, security, reliability, usability, and more — essential for a robust and maintainable product.

## Performance Requirements

The performance of the Student Attendance System is critical, especially during live class sessions where multiple faces must be recognized and attendance must be marked in real time. The following performance metrics and benchmarks are required:

### Face Recognition Speed

* The system must recognize a student’s face and mark attendance within **2 seconds** of detection.
* The model must support real-time face detection and identification with minimal latency.

### Concurrency

* The system must handle **simultaneous recognition of up to 10 students** appearing in a single frame.The
* backend should be capable of processing **100 concurrent recognition attempts per minute** without crashing.

### Response Time

| **Task** | **Expected Time** |
| --- | --- |
| Load dashboard after login | ≤ 2 seconds |
| Start face recognition | ≤ 3 seconds |
| Generate Excel/PDF report | ≤ 4 seconds (for 100+ rows) |

### Uptime and Availability

* The system must ensure **99.9% uptime** during academic working hours (typically 8 AM to 5 PM).
* Recovery from system failure should occur within **10 minutes**.

## Safety Requirements

Although the system doesn’t pose physical danger, **data safety and privacy** must be prioritized. The following safety features are required:

### Data Integrity

* All student records, facial data, and attendance logs must be stored in a **tamper-proof** format.
* Any deletion or alteration of attendance should be logged with timestamp and user details.

### Secure Storage

* Facial images and model encodings must be stored in protected directories with limited access.
* Database should be protected by **role-based access permissions**.

### Access Control

Only authenticated users (faculty/admins) should be allowed to perform high-level operations such as training the model or exporting data.

### Redundancy and Backup

* Automated **nightly backups** should be taken to prevent data loss.
* Backups should include the MySQL database and image dataset.

## Security Requirements

Given the sensitivity of biometric data (faces) and student records, strong security protocols must be enforced.

### Authentication

All users must log in using a valid **Username and MPIN**.

Future enhancements may support **Two-Factor Authentication (2FA)** using OTP or email.

### Authorization

Access levels must be defined:

Admin: Full access

Faculty: Limited access

Guests/Students: Read-only (optional future feature)

### Data Encryption

All communication between client and server should be done via **HTTPS**.

User credentials and facial encodings should be hashed/encrypted using algorithms like SHA-256 or AES.

### Threat Prevention

Implement basic **firewall rules** or security filters to prevent:

* SQL Injection
* Cross-site scripting (XSS)
* Brute-force login attacks

Monitor and log unauthorized access attempts.

## Maintainability and Support

Maintainability ensures that the system is easy to update, debug, and extend as future requirements evolve.

### Modular Design

All major components (registration, recognition, reporting) should be developed as independent modules for easier maintenance.

### Code Documentation

Every module must be thoroughly documented with inline comments and developer guidelines.

### Support Structure

Admins should be able to access a “Support” section in the dashboard containing:

* Help manuals
* Contact forms
* Tutorial videos

### Update Mechanism

System should allow for **safe software updates** without disrupting current data.

Future versioning control (GitHub) should be enabled for easy rollback.

## Software Quality Attributes

To ensure high usability, reliability, and adaptability, the system should fulfill the following quality metrics:

| **Attribute** | **Specification** |
| --- | --- |
| **Reliability** | Must function correctly in ≥99% of all use cases; should not crash under load. |
| **Usability** | GUI must be intuitive for users with basic computer knowledge. |
| **Portability** | System should be operable on Windows, Linux, and macOS. |
| **Scalability** | Should support increasing number of students without performance loss. |
| **Interoperability** | Can be integrated with existing LMS or ERP systems in future versions. |
| **Robustness** | System should handle invalid inputs without crashing or corrupting data. |
| **Reusability** | Code and components should be modular and reusable in other academic systems. |
| **Testability** | System must be easy to test using unit testing and real-time test cases. |

# Other Requirements

* The system should support exporting attendance records to CSV format.
* The system should allow teachers to view attendance statistics.

Appendix A: Glossary

* + LBPH: Local Binary Pattern Histogram (face recognition algorithm).
  + Haar Cascade: A machine learning-based approach for object detection.
  + CSV: Comma-Separated Values (file format for storing data).

Appendix B: Analysis Models

* Data flow diagrams and class diagrams can be included to visualize the system's architecture.
* - Use case diagrams, flowcharts, and ER diagrams for the system.

Appendix C: To Be Determined List

* TBD-1: Define the maximum number of students the system can handle.
* TBD-2: Determine the minimum hardware requirements for optimal performance.