CHAPTER 3 – DESIGN DOCUMENTS

3.1 Purpose of the Document

This design document provides a complete technical blueprint for the Smart Face Recognition Attendance System. It describes the system architecture, design decisions, modules, data structures, algorithms, workflow, and experimental setup. The intended audience includes developers, project guides, faculty evaluators, and future maintainers.

3.2 Background & Motivation

Traditional attendance systems such as manual registers, ID cards, and biometric devices are time-consuming, error-prone, and vulnerable to proxy attendance. Recent studies emphasize integrating facial recognition with GPS-based geofencing and time validations to improve security and reliability.

The proposed system addresses these limitations by combining:

- Facial recognition powered by AI
- Geolocation-based attendance validation
- Time-window restricted attendance marking
- Automatic notifications and cloud synchronization

3.3 Scope of the Project

In Scope:

- Android-based attendance application
- Facial recognition with live-image capture
- GPS-based geofencing
- Time-window validation
- Notification services
- Cloud database and real-time analytics dashboard

Out of Scope:

- Integration with ERP/LMS systems
- Multi-campus deployment
- Offline attendance marking

3.4. Problem Statement

1. Clear articulation of the research problem

Most institutions still use manual or basic digital attendance methods such as roll calls, paper registers, ID cards, or fingerprints. These methods cannot reliably confirm a student's identity or ensure that they are physically present during class. This leads to proxy attendance, inaccurate records, and difficulty in maintaining proper documentation.

2. Current limitations in existing systems or knowledge

Existing attendance systems lack key features such as real-time identity verification, GPS-based location checking, and time-window validation. Many systems also do not store attendance securely in the cloud or provide role-based access for students, teachers, and administrators. As a result, these systems become unreliable and time-consuming to manage.

3. Expected outcomes

The project aims to deliver a Smart Face Recognition Attendance System that provides secure facial verification, location validation, and time-based attendance marking. The expected outcome is a fast, accurate, and fraud-free attendance process with cloud-based storage and easy reporting, reducing manual work and improving transparency.

3.5. Research Objectives

1. Main Objective

To design and implement a secure, real-time Smart Face Recognition Attendance System using facial recognition, geofencing, and time validation.

2. Specific Objectives

- Implement AI-based automatic face recognition
- Validate location using GPS geofencing
- Allow attendance only during predefined time windows
- Enable liveness detection via live-camera capture
- Send automated notifications to absent students
- Store data securely in a cloud database

3. Research Questions or Hypotheses

• Research Question 1:
Can facial recognition eliminate proxy attendance more effectively than traditional methods?

• Research Question 2:
Does GPS geofencing improve accuracy by ensuring that attendance is marked only from authorized locations?

• Research Question 3:
Will a combined system of face verification, location check, and time validation provide a more secure and efficient attendance process?

• Hypothesis:
If facial recognition, GPS validation, and time-window rules are integrated into a single system, then attendance can be recorded more accurately, securely, and faster compared to existing manual methods.

3.6. System Overview

3.6.1 System Description

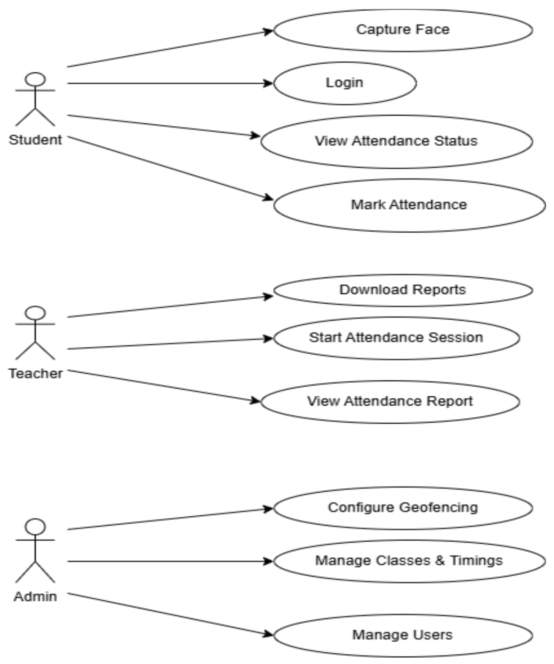
The system captures a live image of the student, verifies their identity using AI-based facial recognition, checks geolocation boundaries (geofence), validates session timing, and records attendance in Firebase. Faculty can view analytics through a dashboard.

3.6.2 Architecture Style

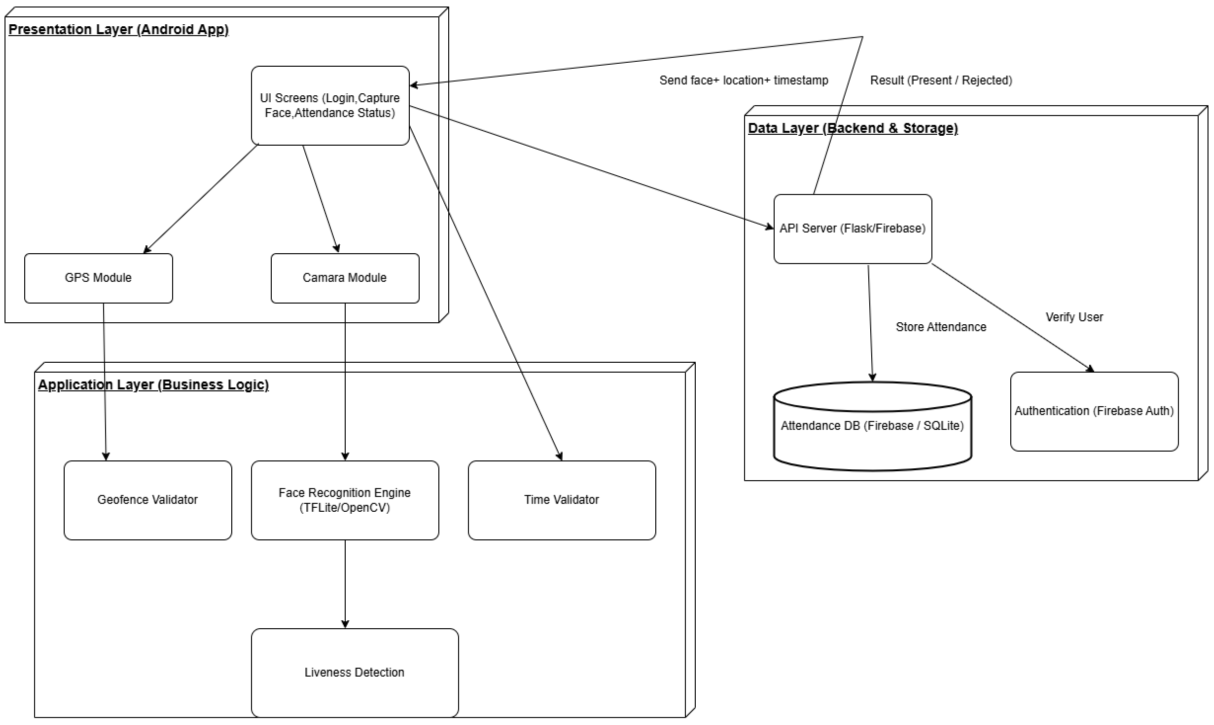
- **Client–Server Architecture:**
  A model where the Android app acts as the client sending requests, and Firebase acts as the server managing data and authentication.
- **Layered Architecture (Presentation → Logic → Data):**
  A structured design that separates the user interface, application logic, and database operations into independent layers.
- **AI Processing Pipeline (Capture → Detect → Recognize → Validate → Log):**
  A step-by-step flow where the system captures a live image, detects the face, recognizes the student, validates location/time, and logs attendance.

3.6.3 High-Level Diagram

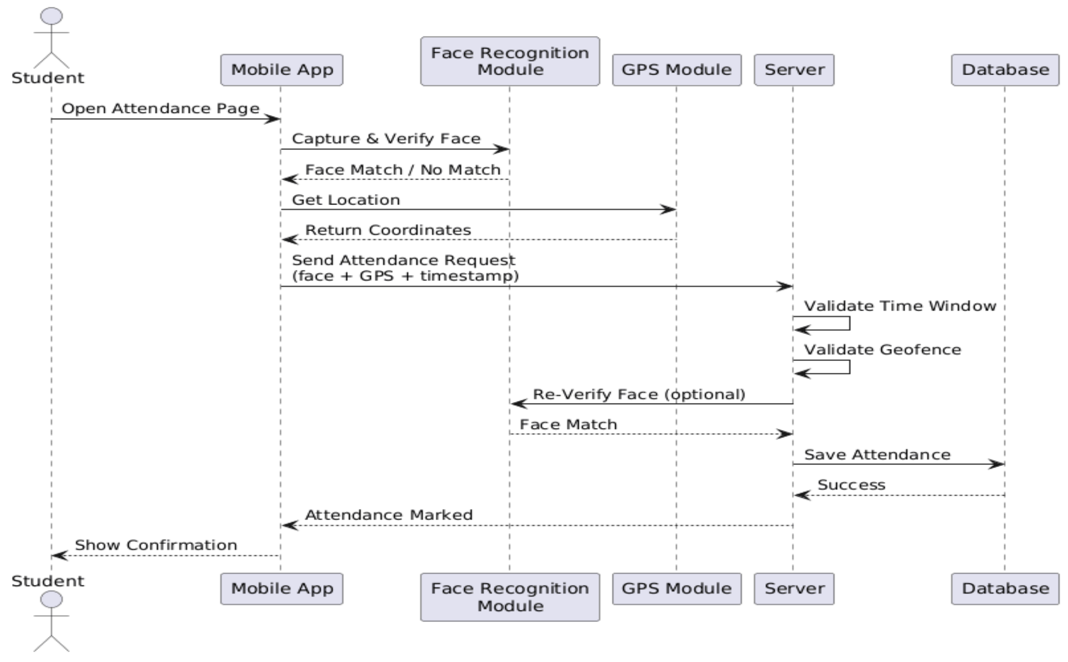
Use-case diagram



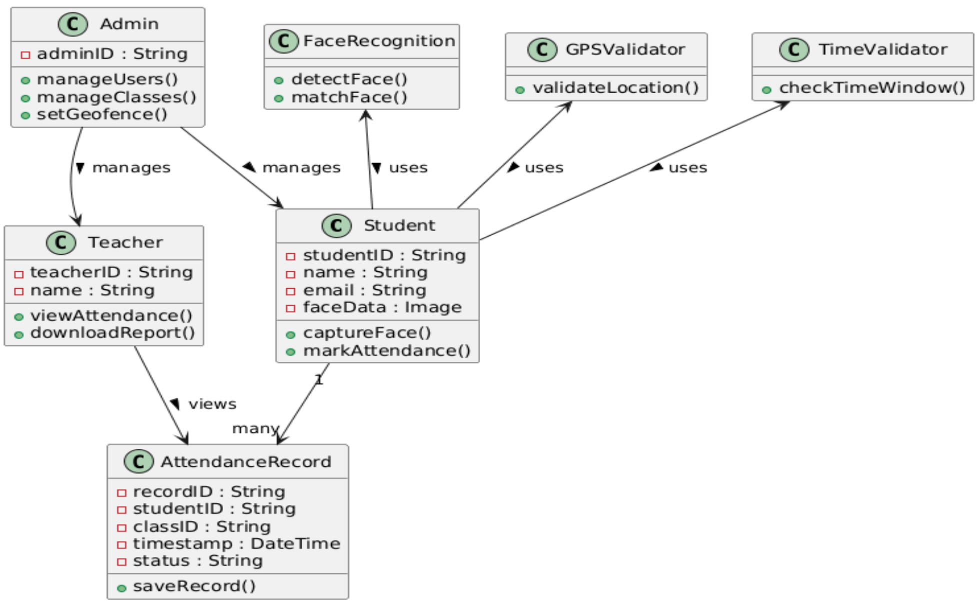
Architecture diagram


Presentation Layer (Android App)
UI Screens (Login,Capture Face,Attendance Status)
GPS Module
Camara Module
Send face+ location+ timestamp
Result (Present / Rejected)
Data Layer (Backend & Storage)
API Server (Flask/Firebase)
Store Attendance
Verify User
Attendance DB (Firebase / SQLite)
Authentication (Firebase Auth)
Application Layer (Business Logic)
Geofence Validator
Face Recognition Engine (TFLite/OpenCV)
Time Validator
Liveness Detection

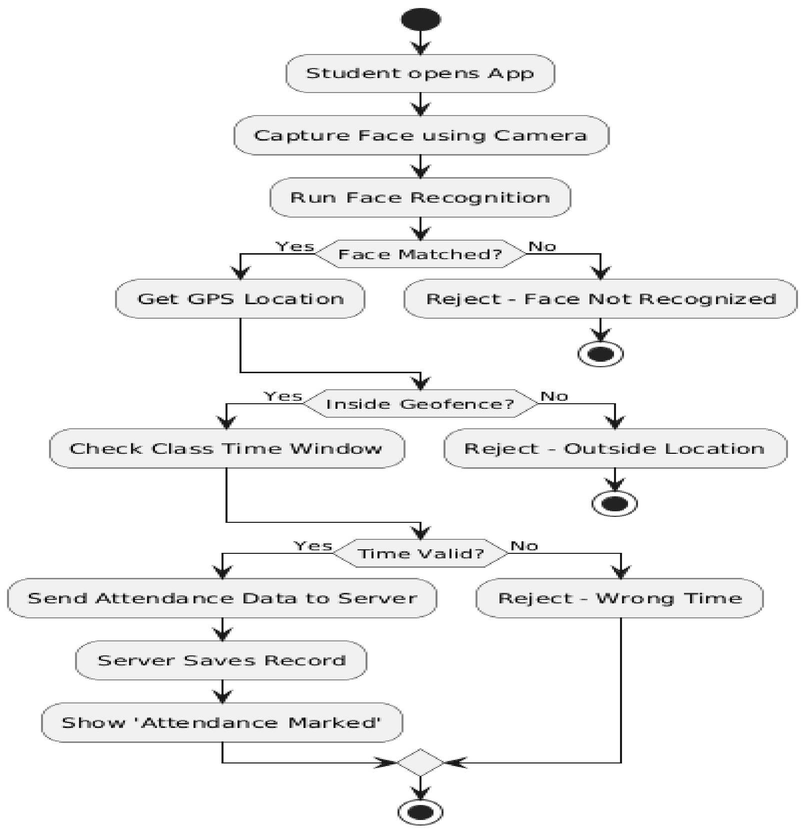
Sequence Diagram



Class Diagrams



Workflow diagram



3.7. Detailed Design

3.7.1 Data Design

Data Schema

The system uses a combination of Firebase Realtime Database and SQLite. The primary entities include Users, Attendance, and Class Schedule.

1.User Table

Field	Type	Description
user_id	String	Unique ID assigned by Firebase
name	String	Student/Teacher name
email	String	Registered login email
role	String	student / teacher / admin
face_encoding	Array(Float)	Numerical vector for face recognition
phone	String	Optional contact number
created_at	Timestamp	Registration date

2.Attendance Table

Field	Type	Description
attendance_id	String	Unique primary key
user_id	String	Foreign key for Users
class_id	String	Foreign key for Class Schedule
timestamp	DateTime	Date & time of marking attendance
latitude	Float	Captured device latitude
longitude	Float	Captured device longitude
status	String	Present / Absent
verified	Boolean	True if face + location + time valid

3.Class Schedule Table

Field	Type	Description
class_id	String	Unique class identifier
subject	String	Subject name
start_time	Time	Attendance window start
end_time	Time	Attendance window end
geofence_lat	Float	Latitude of classroom
geofence_long	Float	Longitude of classroom
radius	Integer	Allowed geofence radius

Preprocessing Steps

Preprocessing ensures clean and consistent inputs for face recognition.

1. Image Preprocessing

- Convert to RGB format
- Resize to required model dimensions (e.g., 112×112 px)
- Normalize pixel values
- Apply histogram equalization (optional)
- Detect face region and crop

2. Facial Embedding Preprocessing

- Extract facial landmarks
- Align face using eye-center alignment
- Generate 128-D or 512-D embedding vector
- Store vector securely in Firebase

3. GPS Preprocessing

- Fetch GPS coordinates from Location API
- Filter inaccurate readings
- Compute Haversine distance for geofence check
- Pass coordinates to attendance validation module

4. Time Preprocessing

- Sync device time with internet clock
- Convert to 24-hour format
- Compare with class schedule window

3.7.2 Algorithm Design

Algorithms used

1. Facial Detection Algorithm (OpenCV Haar Cascade / DNN)

Detects faces in the live camera frame by identifying facial features such as eyes, nose, and face outline.

2. Facial Recognition Algorithm (MobileFaceNet / TensorFlow Lite)

Generates a numerical face embedding and compares it with stored student embeddings to verify identity.

3. Liveness Detection Algorithm

Ensures the captured image is live by checking real-time camera capture, blink/motion, or frame change to prevent spoofing.

4. GPS Geofencing Algorithm

Validates whether the student is physically present inside the predefined geolocation boundary before marking attendance.

5. Time-Window Validation Algorithm

Checks if the current time falls within the scheduled attendance session window (start time → end time).

6. Attendance Logging Algorithm

Records valid attendance data (user ID, timestamp, GPS, status) into the Firebase Realtime Database after all checks pass.

7. Notification Algorithm (Firebase Cloud Messaging)

Automatically sends attendance/absence alerts to students using push notifications once the attendance session ends.

Mathematical model

1.Facial Recognition

Each face is represented as a vector $F = (f_1, f_2, \ldots, f_n)$
Using Euclidean distance:

$$D = \|F_{input} - F_{stored}\|$$

If

$$D \leq \theta$$

(face matches threshold), recognition = True.

2.Geofencing

Let:

- L_s = Student's location (lat_1, lon_1)
- L_c = Classroom location (lat_2, lon_2)
- R = Allowed radius (e.g., 50 meters)

Distance between two coordinates:

$$d = 6371 \times \cos^{-1}(\cos(lat1)\cos(lat2)\cos(lon2 - lon1) + \sin(lat1)\sin(lat2))$$

Condition:

$$d \leq R$$

3..Time Validation

Let:

- T_now = current time
- T_start, T_end = class start & end time

Attendance allowed if:

$$Tstart \leq Tnow \leq Tend$$

Pseudocode

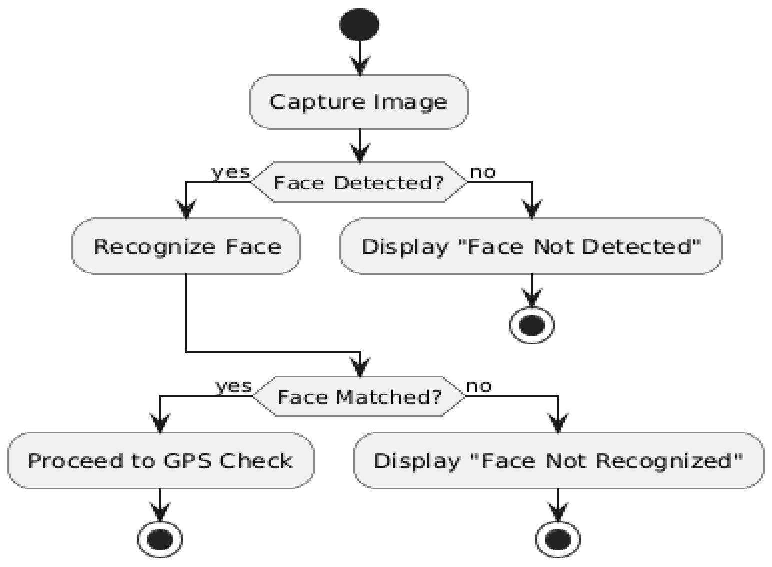
```
START

// Step 1: Capture Image

capture_image = openCamera()

IF capture_image is NULL THEN

   DISPLAY "Unable to capture image"

   STOP

END IF


// Step 2: Face Detection

face_found = detectFace(capture_image)

IF face_found == FALSE THEN

   DISPLAY "Face not detected. Try again."

STOP

END IF


// Step 3: Liveness Check
```

```
is_live = checkLiveness(capture_image)
IF is_live == FALSE THEN
   DISPLAY "Liveness check failed. Please use live camera."
   STOP
END IF

// Step 4: Face Recognition
recognized_user = matchFace(capture_image)
IF recognized_user == UNKNOWN THEN
   DISPLAY "Face not recognized"
   STOP
END IF

// Step 5: GPS Geofencing
current_location = getGPSLocation()
IF current_location NOT IN allowed_geofence THEN
   DISPLAY "You are outside the allowed location"
   STOP
END IF

// Step 6: Time Validation
current_time = getCurrentTime()
IF current_time NOT IN class_time_window THEN
   DISPLAY "Attendance window closed"
   STOP
```

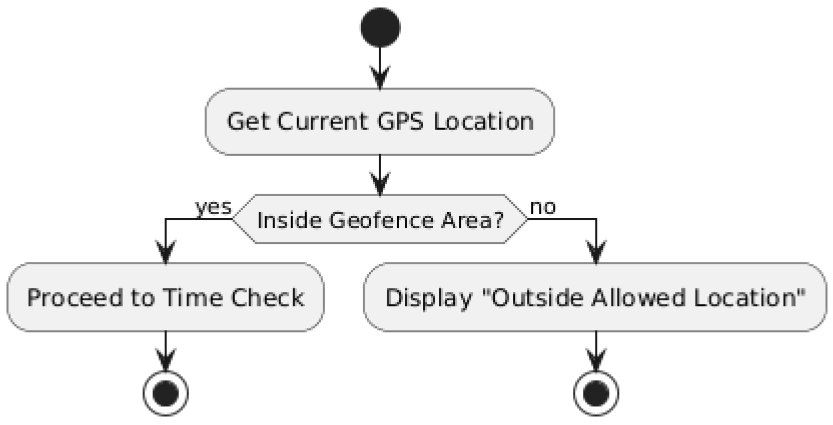
```
END IF

// Step 7: Mark Attendance
attendance_status = "Present"
saveAttendance(recognized_user, current_time, current_location, attendance_status)
DISPLAY "Attendance marked successfully"

// Step 8: Trigger Notifications (Optional)
sendNotificationToStudent(recognized_user, "Attendance Marked")
END
```

Flowcharts

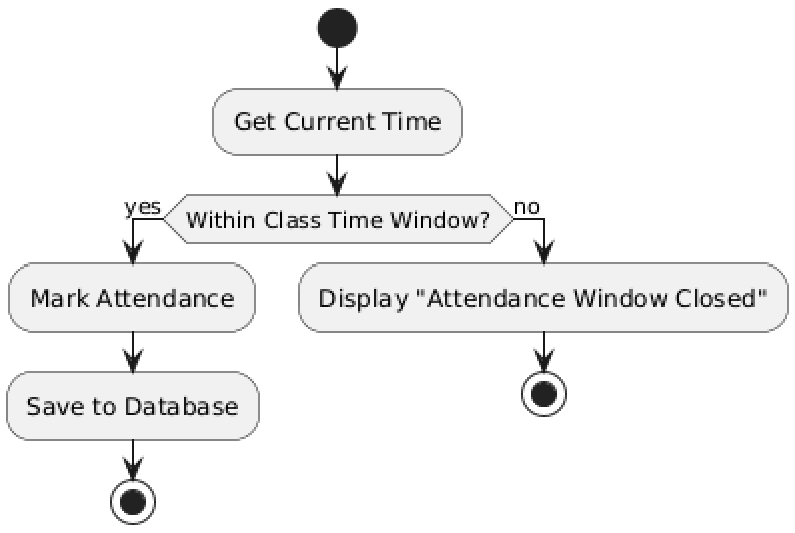
Face Recognition Flowchart



GPS Geofencing Flowchart



Time Validation Flowchart



3.7.3 Module-wise Design

1. Student Module

Purpose:

- To allow students to mark attendance automatically and securely using face recognition and geolocation verification.

Inputs:

- Live image captured from the device camera
- GPS location
- Current time

Outputs:

- Attendance status (Present / Absent / Late / Rejected)

Internal Logic:

- Capture live image from the student's device.
- Run face recognition algorithm (compare with registered student images).
- Validate GPS location against the pre-defined geofence.
- Verify if the attendance is being marked within the valid class time window.
- Update attendance status in the database.

Constraints:

- Device must have a functional camera and GPS.
- Face recognition accuracy depends on lighting, pose, and image quality.
- Student must be within geofence boundaries.
- Attendance can only be marked during allowed class time.

2. Teacher Module

Purpose:

- To monitor and manage student attendance, generate reports, and communicate with students.

Inputs:

- Attendance data from student module
- Teacher commands (view reports, send notifications)

Outputs:

- Real-time attendance view
- Attendance reports (daily/weekly/monthly)
- Notifications to students (e.g., reminders)

Internal Logic:

- Retrieve attendance records from the database.
- Generate visual reports (charts, tables).
- Allow teacher to send notifications to students regarding attendance issues.
- Provide filtering and sorting options for better analysis.

Constraints:

- Requires stable network connection to fetch real-time data.
- Notifications depend on student device connectivity.
- Access control: only assigned teachers can view their class attendance.

3. Admin Module

Purpose:

- To manage system settings, users, and analytics for the attendance system.

Inputs:

- Admin commands (add/update students/teachers, set class times, define geofence)
- Attendance data for analytics

Outputs:

- Updated user database (students & teachers)
- Class schedule and geofence configuration
- Attendance analytics dashboards

Internal Logic:

- Manage student and teacher records (CRUD operations).
- Set and update class schedules.
- Define geofence areas for each class/location.
- Analyze attendance patterns and generate reports/graphs.

Constraints:

- Only authorized admins can access this module.
- Changes in class time or geofence must be validated for conflicts.
- System performance may degrade with very large datasets if not optimized.

3.8. Experimental Design

3.8.1 Experiment Setup

Hardware Requirements:

For Students (Mobile App Testing):

- Android smartphone (Android 9.0 or above)
- Minimum 3 GB RAM
- Quad/Octa-core CPU
- Front camera (5–8 MP or higher)

- GPS sensor with high accuracy
- Stable Wi-Fi / 4G / 5G connection

For Developer / Testing Environment:

- Laptop/PC with minimum Intel i3 processor
- 4 GB RAM (recommended 8 GB)
- Webcam for face recognition tests (optional)
- 10 GB free disk space

Software Requirements:

1. Programming Languages:
   - Java/Kotlin (Android app)
   - Python 3.11+ (model training + recognition testing)
2. Libraries & Frameworks:
   - OpenCV → face detection
   - TensorFlow Lite / MobileFaceNet → face recognition
   - NumPy, Pandas → dataset handling
   - Firebase Authentication + Firebase Realtime Database
   - Android Location API → GPS & geofencing
   - FCM → push notifications
3. Development Tools:
   - Android Studio (primary development)
   - VS Code / PyCharm (backend Python code)
   - Postman (optional API testing)

Dataset details

The system uses a custom dataset collected specifically for research on mobile-based face recognition attendance.

Dataset Composition

- **Source:** Real students or volunteer participants
- **Images per person:** 20–30 facial images
- **Variations included:**
  - Different lighting conditions
  - Multiple angles (left, right, front)
  - Different expressions
  - Indoor/outdoor test images

Dataset Structure

Folder	Description
/dataset/person_01/	Cleaned face images of student 1
/dataset/person_02/	Face images of student 2
/test_images/	Images for testing recognition accuracy
/liveness_images/	Spoof attempts (photos / screenshots)

Additional Data Used

- GPS coordinates (latitude, longitude)
- Timestamp of attendance attempt
- User metadata (Student ID, class information)

Dataset Size

- ~500–800 total face images
- Real-time collected images used for evaluation
- Liveness dataset includes printed photo, screen image, and replay attack samples

Preprocessing steps

Preprocessing ensures high-quality input for both face recognition and attendance validation.

1. Image Preprocessing

- Convert images to RGB
- Resize to model-compatible dimensions (e.g., 112×112 for MobileFaceNet)
- Normalize pixel values (0–1 range)
- Detect and crop the face region using Haar Cascade / DNN
- Remove blurry or low-quality images

2. Face Embedding Extraction

- Use MobileFaceNet/TFLite to convert face images into 128-dimensional embeddings
- Store embeddings in a secure database for matching

3. Data Augmentation (Optional for Research)

- Slight rotation (±10°)

- Brightness adjustment
- Horizontal flip
- Helps improve robustness

4. GPS & Time Preprocessing

- Filter invalid GPS signals (accuracy > 100 m ignored)
- Store geofence coordinates
- Convert timestamps to standardized format (HH:MM:SS)

5. Liveness Detection Preprocessing

- Extract eye/mouth landmarks
- Compute blink ratio / motion difference
- Validate whether input is live or spoofed

6. Attendance Rules Preprocessing

- Define allowed time window
- Set geofence radius (e.g., 50–100 meters)
- Map student IDs to face embeddings

3.8.2 Methodology

Experimental procedure

The research methodology describes how the Smart Face Recognition Attendance System was designed, tested, and validated. The procedure follows a structured research workflow similar to the previous project (Sustainable Product System) but adapted for biometric and GPS-based attendance verification.

1. **Dataset Preparation**
    - Collected student face images under different lighting and angles.
    - Preprocessed images (cropping, resizing, normalization).
    - Used real-time camera captures to ensure dataset authenticity.
2. **Feature Extraction**
    - Applied OpenCV's Haar Cascade / SSD to detect the face region.
    - Used MobileFaceNet / TensorFlow Lite to extract facial embeddings.
    - GPS coordinates and timestamps were also recorded as additional features.

3. **Model Training & System Setup**
    - Trained the facial recognition model using 80% of collected images.
    - Tested using the remaining 20% to measure accuracy.
    - Integrated GPS geofencing logic and time-window validation.
    - Implemented liveness checks (blink/motion detection).
4. **Application Integration**
    - Integrated face recognition model into the Android app.
    - Embedded GPS tracking, geofence validation, and Firebase sync.
    - Built attendance marking flow:
      Face Match → Inside Geofence → Time Check → Mark Attendance
5. **System Testing**
    - Performed controlled tests with multiple students.
    - Tested under various lighting, distances, and network conditions.
    - Collected performance data:
        - Face recognition accuracy
        - GPS precision
        - Time validation accuracy
        - Liveness detection success rate
6. **Deployment & Validation**
    - Deployed prototype on actual Android phones.
    - Conducted classroom simulation to validate system behaviour.
    - Analysed false positives, false negatives, and attendance errors.

Control vs. experimental conditions

Control Condition (Without Additional Security Layers)

- Only face recognition used.
- No GPS geofencing.
- No time restrictions.
- No liveness detection.

This allowed measurement of baseline accuracy and behaviour.

Experimental Condition (Full System Activated)

- Face recognition + liveness detection
- GPS geofence validation (inside campus only)
- Time-based attendance window
- Real-time Firebase sync
- Auto-notification on absence

Tools used (Python, MATLAB, R, etc.)

- **Programming:** Java/Kotlin, Python
- **Libraries:**
    - OpenCV (face detection, image processing)
    - TensorFlow Lite, MobileFaceNet (face recognition)
    - Android Location API (GPS tracking)
- **Database**: Firebase Realtime Database + SQLite
- **Development Environment**: Android Studio, VS Code
- **Testing Tools:** Emulator + physical Android device
- **Cloud Services:** Firebase Authentication, Firebase Cloud Messaging (notifications)

3.8.3 Evaluation Metrics

To evaluate the performance, reliability, and effectiveness of the Smart Face Recognition Attendance System, both quantitative and qualitative evaluation metrics were used. These metrics assess the accuracy of face recognition, the efficiency of geofencing and time validation, and the overall user experience of the system.

1. Classification Metrics (Face Recognition Performance)

These metrics evaluate how accurately the system identifies and verifies student faces.

Accuracy

Measures overall correctness of the face recognition model.

$$Accuracy = \frac{Correct\ Predictions}{Total\ Predictions}$$

Precision

Indicates how many of the faces the system recognized as “correct” were actually correct. Useful to measure false positives (wrong face accepted).

Recall

Measures how well the system detects the correct face when it is present.
Important for reducing false negatives (correct student rejected).

F1-Score

A balanced metric combining precision and recall.
Useful when face datasets are imbalanced.

2. Error Metrics for Facial Embeddings (Optional Research Metrics)

These metrics help evaluate the numerical difference between expected and predicted facial embeddings.

RMSE – Root Mean Squared Error

Shows how close the predicted embedding distance is to actual values.
Lower RMSE = better model stability.

MAE – Mean Absolute Error

Indicates average error in embedding comparison.
Useful for comparing different recognition thresholds.

3. GPS & Geofencing Evaluation Metrics

Location Accuracy

Measures how correctly the system identifies whether a student is inside the geofenced area.
Tested by varying distance from classroom (e.g., 5m, 10m, 20m).

False Acceptance Rate (FAR) for GPS

Percentage of cases where students outside geofence are incorrectly marked as inside.

False Rejection Rate (FRR) for GPS

Cases where students inside geofence are incorrectly rejected.

4. Time Validation Metrics

Execution Time for Validation

Measured time taken for:

- Reading system time
- Matching it with allowed attendance window

- Returning response

Expected output: < 1 second

5. Liveness Detection Metrics

Spoof Detection Rate

Percentage of successful detection when students attempted to use:

- Mobile photos
- Printed photos
- Video replay attacks

Liveness Accuracy

Measures how accurately the system distinguishes live faces from spoofed ones.

6. System Performance Metrics

Execution Time / Response Time

How long the system takes to complete:

- Face detection
- Face recognition
- GPS check
- Time validation
- Attendance marking

Expected total duration: 2–3 seconds per student

Throughput

Maximum number of students the system can process per minute.

Cloud Sync Delay

Time taken to upload attendance to Firebase.
Expected: < 2 seconds under stable network conditions.

7. Reliability & Error Rates

False Acceptance Rate (FAR)

Cases where the system incorrectly marks an unknown or wrong face as present.

False Rejection Rate (FRR)

Cases where the system rejects the correct student’s face.

System Uptime

Percentage of time the application remains functional without crashes.

8. Qualitative Metrics

User Experience Feedback

Collected from students and teachers to evaluate:

- Simplicity of app interface
- Ease of marking attendance
- Clarity of notifications
- Overall satisfaction

Interpretability & Transparency

Measures how easily users understand:

- Why attendance was accepted or rejected
- GPS boundary messages
- Time window restrictions

Usability Score

Assesses navigation ease, button placement, and clarity of instructions.

9. Comparative Metrics (Control vs Experimental)

Used to compare system behavior:

- Without geofencing & liveness (control)
- With full security (experimental)

Measured improvements in:

- Accuracy
- Proxy attendance prevention
- Reliability
- User trust