GURU GOBIND SINGH
INDRAPRASTHA
UNIVERSITY
NEW DELHI

UNIVERSITY SCHOOL OF AUTOMATION AND ROBOTICS
EAST DELHI CAMPUS, SURAJMAL VIHAR, DELHI- 110032

Minor Project Report

On

Smart Roll Call using IoT based System

Submitted in partial fulfilment of the requirements for the completion of Minor Project [ARP455]

Name: Karan Bhatia
Enrollment Number: 01819051722
B.Tech IIOT B1

Under the supervision of
Dr. Manisha Parlewar
Assistant Professor

Certificate by Supervisor

This is to certify that the Minor Project titled "Smart Roll Call using IoT based System" submitted by Karan Bhatia (Enrollment No. 01819051722) of IIOT–B1, University School of Automation and Robotics, GGSIP University, has been completed under the supervision and guidance of Dr. Manisha Parlewar, Assistant Professor, USAR.

The work presented in this report is original and has been carried out by the student as part of the Minor Project requirements prescribed by the University.

,
Assistant Professor,
USAR, GGSIPU EDC

Declaration

I hereby declare that the Minor Project Report entitled “Smart Roll Call” is an authentic record of work completed for requirements of Minor Project (ARP 455) under the supervision of Dr. Manisha Parlewar

(Signature of Student)

Karan Bhatia

01819051722

Date :

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to my project supervisor **Ms. Manisha Parlewar** for providing me the opportunity and guidance to work on this minor project. Her continuous support, mentorship and valuable insights have been instrumental in the successful completion of this project.

I am especially thankful to my team member **Shubham Dev**, whose collaboration, support and insights were invaluable throughout this journey. Together, we worked on developing the Smart Roll Call with Facial Recognition system, which enhanced both my technical and teamwork skills. The experience of working in a collaborative environment taught me the importance of communication, problem-solving and adaptability, all of which are crucial for real-world projects.

I would also like to extend my appreciation to **Dr. Ajay Singholi**, Head of Department, Industrial Internet of Things (IIoT) for his encouragement and support. His guidance and recommendations helped me refine my project approach and methodology. The technical knowledge and feedback I received throughout this project have significantly contributed to my learning and professional growth.

Furthermore, I am grateful to my college professors for their constant support and encouragement. Their academic guidance laid a strong foundation that greatly benefited me during this project. The knowledge and principles I acquired during my coursework were essential in helping me navigate the technical challenges of implementing facial recognition and attendance management systems.

Lastly, I would like to thank my family and friends for their unwavering support throughout this journey. Their belief in my abilities has been a source of motivation and I am immensely grateful for their encouragement as I continue to pursue my passion.

Table of Contents

List of Tables

List of Figures

Abbreviations

AAMS – Automated Attendance Management System
AI – Artificial Intelligence
API – Application Programming Interface
AP – Access Point
BLE – Bluetooth Low Energy
CPU – Central Processing Unit
CSV – Comma-Separated Values
CNN – Convolutional Neural Network
ESP – Espressif Systems (Microcontroller family)
ESP32 – Espressif Systems 32-bit Wi-Fi + Bluetooth Microcontroller
GCP – Google Cloud Platform
HOG – Histogram of Oriented Gradients
HTTP – Hypertext Transfer Protocol
IDE – Integrated Development Environment
IIoT – Industrial Internet of Things
IoT – Internet of Things
JSON – JavaScript Object Notation
MAC – Media Access Control
ML – Machine Learning
NFC – Near Field Communication
NoSQL – Not Only Structured Query Language
NTP – Network Time Protocol
PDF – Portable Document Format
QR – Quick Response
REST – Representational State Transfer
RFID – Radio Frequency Identification
SSL – Secure Sockets Layer
TLS – Transport Layer Security
UI – User Interface
URL – Uniform Resource Locator
VM – Virtual Machine
Wi-Fi / WiFi – Wireless Fidelity

Abstract

This project presents a comprehensive attendance management system that combines mobile application technology, facial recognition and IoT based automatic tracking to streamline the attendance marking process in educational institutions. The system addresses traditional attendance challenges through a dual-mode approach: face-verified attendance via a Flutter mobile application and automatic MAC address-based detection using ESP32 microcontrollers. The system architecture consists of three primary components:
(1) a cross-platform Flutter mobile application built with Firebase Firestore integration for real-time data synchronization, user authentication and facial recognition-based attendance verification; (2) an ESP32 based IoT module operating in dual WiFi mode that creates a secure hotspot while maintaining internet connectivity, automatically detecting registered student devices and marking attendance during scheduled class hours; and (3) a Flask based middleware server deployed on Google Cloud Platform that processes attendance data and manages face recognition verification requests.

Key features include real time attendance tracking, comprehensive analytics with visual charts, automated schedule management, duplicate attendance prevention and privacy focused face template storage (eliminating the need for face image storage). The ESP32 module leverages Firebase REST API for direct Firestore integration, performs NTP time synchronization for accurate timestamps and implements power optimization techniques for extended operation. The Flutter application provides role based interfaces for both professors and students, featuring batch management, schedule creation, MAC address registration, face enrolment and detailed attendance reports with Excel export capabilities.

Performance analysis demonstrates the system's capability to handle concurrent connections on the ESP32 module, sub second face recognition processing times and real time synchronization across all components. The implementation showcases successful integration of modern technologies including Flutter, Firebase services, ESP32 microcontroller programming and machine learning based facial recognition, providing a scalable, secure and efficient solution for academic attendance management.

Chapter 1: Introduction

1.1 Background

In the modern educational landscape, attendance management remains a significant challenge for academic institutions. Traditional paper based and manual roll call methods are time consuming, error prone and vulnerable to proxy attendance. Existing digital systems often still require manual input, lack real time synchronization and provide limited analytics for monitoring student engagement. These limitations highlight the need for a secure, automated and data-driven attendance solution.

1.2 Project Overview

Smart Roll Call: An Integrated Facial Recognition and IoT Based Attendance Management System offers a modernized, multi-modal approach to attendance tracking. It combines face verified attendance with automated MAC address based detection for higher accuracy, flexibility and minimal human intervention.

The system consists of three components:

Mobile Application Layer: A Flutter app for students and professors supporting attendance marking, face enrollment, device registration, batch management and real-time analytics.

IoT Hardware Layer: ESP32 modules running in dual Wi-Fi mode to create secure classroom hotspots while syncing with Firebase, automatically detecting registered student devices during class hours.

Backend Infrastructure: Firebase Firestore for real-time data, Firebase Authentication for user security and a Flask middleware server on Google Cloud for facial verification.

1.3 Motivation

The system is motivated by key issues: time lost during manual roll calls, risk of proxy attendance, difficulty accessing attendance patterns, privacy concerns with traditional biometric systems, scalability issues in legacy solutions and the lack of meaningful analytics to identify at-risk students.

1.4 Project Objectives

- Automate attendance using ESP32 based MAC detection.
- Improve security via facial recognition with liveness detection.
- Ensure privacy by storing only face templates, not images.
- Enable real-time synchronization using Firebase.
- Provide analytics dashboards for trends and batch wise insights.
- Offer intuitive role based interfaces for professors and students.
- Maintain cost efficiency using low cost ESP32 hardware and Firebase.

Chapter 2: Literature Review

2.1 Introduction

The process of tracking attendance is a fundamental administrative task in both academic institutions and corporate environments. It serves as a vital metric for assessing engagement, ensuring compliance, and managing resources. Traditionally, attendance is recorded manually using paper based roll calls. However, this method is widely recognized as being time consuming, prone to human error, and highly vulnerable to proxy attendance (i.e., "buddy punching")

The inefficiencies of manual systems has lead to significant research and development into automated attendance management systems (AAMS). The primary goal of an AAMS is to accurately, efficiently and securely record attendance with minimal human intervention. This review surveys the dominant technologies in this field, evaluates their respective advantages and limitations and identifies the research gap that the "Smart Roll Call" system aims to address.

2.2 Survey of Existing Automated Attendance Technologies

The literature on AAMS primarily covers four major technological approaches: biometrics, RFID/NFC, QR codes, and network based systems.

2.2.1 Biometric Systems (Fingerprint and Facial Recognition)

Biometric systems are often considered the most secure method for attendance tracking, as they rely on unique physiological traits.

Fingerprint Scanners: These systems require each student to place their finger on a scanner for verification. They are highly accurate and effectively eliminate proxy attendance. However they suffer from high implementation costs, potential hygiene concerns (especially in a post-pandemic context) and slow throughput in large classes, as students must queue to use a single device.

Facial Recognition: This approach uses cameras to identify students' faces against a database and research shows increasing reliability even in real world, unconstrained environments (Agrawal & Singh, 2015; Fontaine et al., 2017)[12]. Recent studies demonstrate efficient facial recognition

implementations using ESP32 modules within IoT based systems (Khadafi et al., 2024; Saidi & Bouzazi, 2022)[3][4] and performance evaluations also indicate stable face detection accuracy across multiple conditions (Adi & Wahyu, 2022)[5].

2.2.2 Radio-Frequency Identification (RFID) and NFC

RFID systems utilize electromagnetic fields to automatically identify tags attached to objects. In an academic context, these tags are typically embedded in student ID cards.

Pros: The system is fast; students can be marked present simply by walking past a reader or tapping their card .

Cons: This method is still highly susceptible to proxy attendance, as a student can easily give their card to a friend to tap in for them. Furthermore the infrastructure requires installing expensive RFID readers at the entrance of every classroom and the cost of issuing and replacing specialized cards can be prohibitive.

2.2.3 QR Code and Barcode Systems

This is a popular, low cost approach where a unique QR code is generated for each class session. Students scan the code using their smartphone to mark themselves present.

Pros: The system is extremely cheap to implement, requiring only a display screen or projector and students own devices.

Cons: This method is arguably the least secure. A student can easily take a screenshot of the QR code and distribute it to absent friends, who can then scan it from anywhere, completely defeating the purpose of attendance tracking.

2.2.4 Network Based Systems (WiFi and Bluetooth)

A more recent and passive approach involves leveraging the wireless capable devices students already carry, such as smartphones and laptops.

WiFi Based: These systems work by detecting the unique MAC (Media Access Control) address of a device when it connects to a specific WiFi access point (AP). This concept is explored by ,who propose using the campus WiFi infrastructure.

Bluetooth/BLE: This method uses Bluetooth Low Energy (BLE) beacons, where a student's phone, running a specific app, detects the beacon in the classroom to verify a student's presence.

This category of systems is passive, non-intrusive, and leverages existing technology. However, its primary challenge lies in implementation. Relying on a campus wide WiFi infrastructure can be complex and MAC address based systems must have a secure way to register and map a device's MAC address to a specific student.

2.3 The Need for Integrated Cloud and Mobile Platforms

A significant limitation of many first generation automated systems is their siloed nature. They often run on local servers, making the data inaccessible for real-time analysis or management The literature highlights a clear trend toward using mobile and cloud platforms to overcome this.

Cloud Integration (e.g., Firebase): A cloud based backend provides a single, synchronized source of truth. It allows data from various sources (e.g., a hardware scanner and a manual app) to be unified in real time, making the data instantly available for dashboards, reports, and analytics.

Mobile Management (e.g., Flutter): Providing faculty with a cross platform mobile application allows for flexible management. They can view real time attendance, perform manual overrides, manage student rosters, and create class schedules from anywhere, rather than being tied to a specific desktop or terminal.

2.4 Research Gap and Proposed Solution

The existing literature reveals a distinct trade-off:

- High Security Systems (Biometrics) are expensive, invasive, and complex.
- Low Cost Systems (QR Codes, RFID) are highly insecure and prone to proxy.
- Network Based Systems show promise but often lack a complete, user-friendly ecosystem for management and schedule validation.

A significant research gap exists for a cost effective, hybrid and secure AAMS that combines the reliability of passive automation with the flexibility of manual control, all unified by a modern cloud backend.

The Smart Roll Call project is designed to fill this specific gap. By integrating a Flutter application, a Firebase backend and a low cost ESP32 IoT module, it creates a novel, dual mode system:

It addresses the cost and infrastructure problem by using an inexpensive ESP32 module as a dedicated, in class hotspot, rather than relying on complex campus wide network infrastructure. It solves the proxy problem of network based systems by requiring students to register their device's MAC address via the Flutter app, creating a secure link between the device and the student.

It introduces smart automation by ensuring the ESP32 only marks attendance during scheduled class times, which are loaded directly from Firebase. It provides a unified, hybrid system where attendance marked automatically by the ESP32 and attendance marked manually by the faculty in the Flutter app are both saved to the same Firebase collection, providing a single, authoritative record. The solution builds on the successful use of ESP32 in face and device based identity systems as validated by prior studies (Khadafi et al., 2024; Saidi & Bouzazi, 2022; Adi & Wahyu, 2022)[345].

Chapter 3: Problem Statement

Traditional attendance management systems in educational institutions rely heavily on manual methods such as paper based roll calls, sign in sheets, or verbal responses, which are time consuming, prone to human errors and vulnerable to fraudulent practices like proxy attendance. Faculty members spend approximately 5-10 minutes per class session on attendance marking, which accumulates to significant time loss over an academic semester that could otherwise be utilized for teaching and learning activities.

The primary challenges associated with traditional attendance systems include:

Furthermore, existing digital attendance systems often require expensive hardware infrastructure, complex setup procedures, or lack integration with modern mobile platforms. Many institutions struggle with systems that are not user-friendly or fail to provide comprehensive analytics for identifying attendance trends.

For Faculty: Manual attendance marking disrupts lectures, increases administrative burden and provides no immediate insights into student engagement patterns.

For Students: Traditional systems offer no transparency or easy access to attendance records, making it difficult to track their status and meet minimum attendance requirements.

For Administrators: Consolidating attendance data from multiple courses becomes tedious, delaying data driven decision-making and intervention strategies for students with poor attendance.

Therefore, there is a critical need for an intelligent, automated attendance management system that leverages facial recognition technology and mobile platforms to address these challenges. The solution must be cost effective, easy to deploy, secure and capable of providing real-time attendance tracking with comprehensive analytics while ensuring data privacy and maintaining high accuracy to prevent false positives or negatives.

Chapter 4: Description of Various Modules

The Smart Roll Call with Facial Recognition system integrates multiple modern technologies to create an efficient attendance management solution. This module covers Flutter for mobile development, Firebase for backend services, ESP32 for IoT hardware, Google Cloud Platform for server deployment and Face Recognition for biometric processing.

4.1 Module: Flutter Framework for Cross-Platform Development

This provided me with foundational knowledge of Flutter, Google's open source UI toolkit for building cross-platform mobile applications. Flutter enables developers to create applications for both Android and iOS using a single codebase, significantly reducing development time and costs. I learned how Flutter uses the Dart programming language and a widget based architecture to build responsive user interfaces.

I learned about stateless and stateful widgets which form the building blocks of Flutter applications. I explored state management techniques for controlling data flow between screens and navigation systems that allow seamless movement between different parts of the application. The hands on practice helped me understand Flutter's hot reload feature, which allows instant viewing of code changes without restarting the application. Overall, this module gave me the skills needed to develop and customize the Smart Roll Call mobile application.

4.2 Module: Firebase Backend Services

In this module, I learned about Firebase, Google's Backend as a Service platform that provides cloud based services for mobile applications. Firebase eliminates the need to build backend infrastructure from scratch, offering ready to use services like authentication, real time databases and cloud storage.

It covered Firebase Authentication for handling user registration, login and role based access control for professors, students and administrators. I explored Cloud Firestore, a NoSQL document based database that stores user profiles, course information, attendance records and class schedules. The real time synchronization feature ensures instant updates across all connected devices when attendance is marked. I also learned about Firebase Storage for securely storing student face images. Overall, this module helped me understand how Firebase provides a scalable and secure backend solution.

4.3 Module: ESP32 Microcontroller and IoT Integration

ESP32, a low cost microcontroller with built in Wi-Fi capabilities used to create standalone IoT devices. In the Smart Roll Call system, ESP32 integrates with a camera module to create attendance kiosks that capture student faces and communicate with cloud servers, allowing multiple attendance points throughout campus without expensive hardware.

I learned about hardware components including the ESP32 board, camera module and power supply. This module covered programming the ESP32 to capture images, connect to Wi-Fi and send data to remote servers. I practiced flashing firmware, configuring network settings and testing image capture under different lighting conditions. The module also covered offline operation, where the device stores attendance locally and automatically syncs when connectivity is restored. Overall, this training helped me understand how IoT devices extend mobile application functionality.

4.4 Module: Google Cloud Platform and Flask Server Deployment

In this module, I learned about Google Cloud Platform (GCP) and how it hosts a Flask server for face recognition processing. The Smart Roll Call system uses a GCP Virtual Machine to handle compute intensive face recognition tasks separately from the mobile application, ensuring better performance and scalability.

I learnt creating and configuring a Linux based VM on GCP, setting up security rules and installing software dependencies. I learned about Flask - a Python web framework for creating RESTful API endpoints that receive images, process them for face recognition and return results. The complete workflow involves sending captured images to the Flask server, matching faces against the database, creating attendance records in Firebase and sending confirmation responses. The module also covered server deployment practices including SSL/TLS encryption and performance monitoring.

4.5 Module: Face Recognition and Biometric Processing

This module provided covered facial recognition technology using the Face Recognition. I learned about the three main stages: face detection, encoding and matching. Face detection identifies faces in images using algorithms like Histogram of Oriented Gradients or Convolutional Neural Networks. Face encoding extracts unique facial features and converts them into 128 dimensional numerical vectors stored in the database.

Face matching compares newly captured face encodings with stored encodings using mathematical distance calculations and adjustable tolerance thresholds. Through practical exercises, I learned to handle challenges like varying lighting conditions, facial expressions and accessories. We captured multiple face samples during registration to improve accuracy and covered performance optimization techniques like image preprocessing and GPU acceleration. Overall, this module gave me a deep understanding of implementing facial recognition for attendance management.

Chapter 5: Methodology Adopted

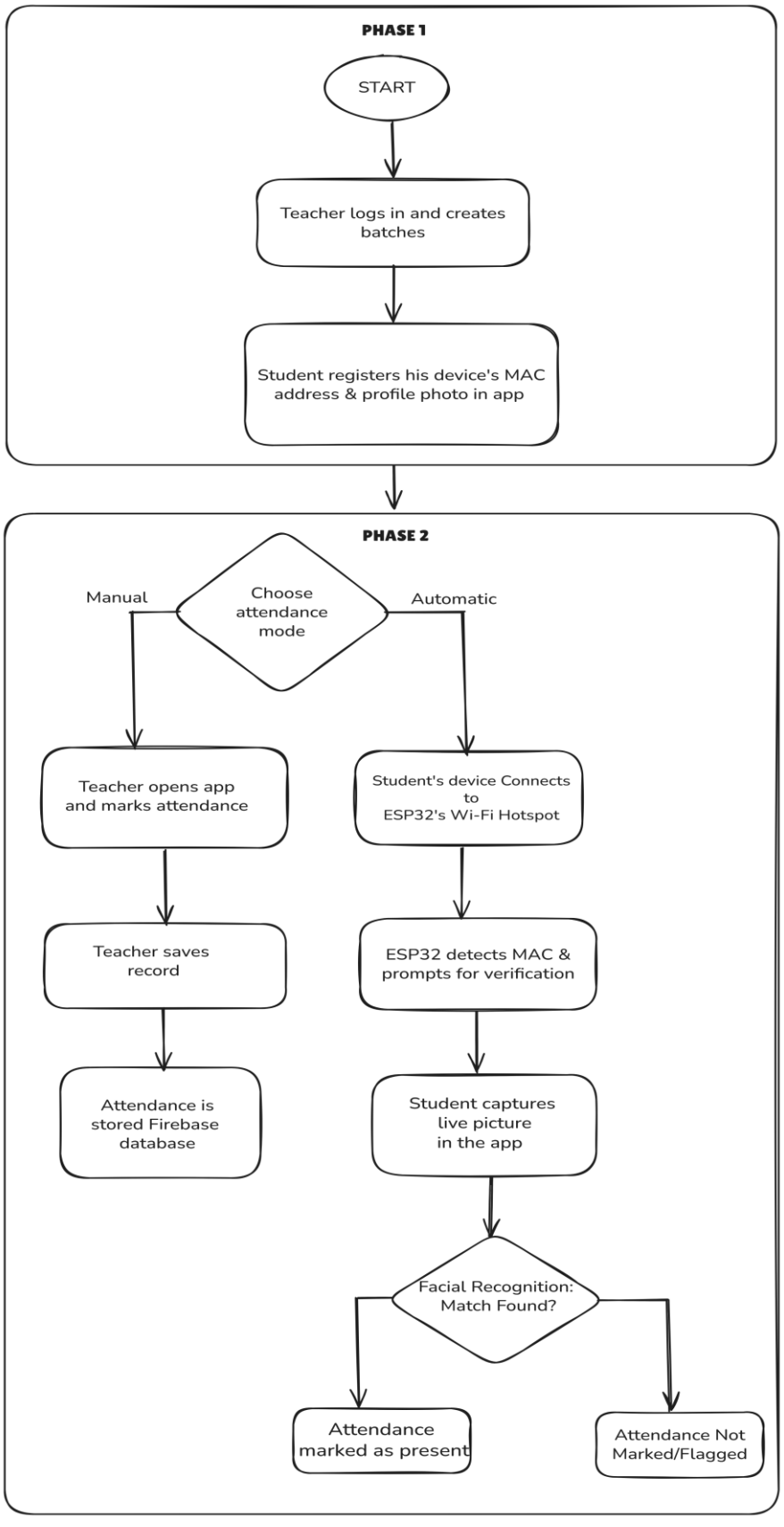
5.1 Objective of The Project

The objective of this project is to create an intelligent and automated attendance management system that can serve as a modern and efficient alternative to traditional manual roll call methods used in educational institutions. The system is designed to bring accuracy, transparency and time efficiency to the attendance process by integrating facial recognition technology, cross-platform mobile applications, IoT hardware and cloud-based data management.

Key Objectives:

- To develop a facial recognition based authentication system that identifies students uniquely and prevents proxy attendance through biometric verification.
- To design a cross platform mobile application using Flutter that provides an intuitive interface for professors to manage courses, mark attendance, generate reports and view real-time analytics.
- To implement real time data synchronization using Firebase Firestore for instant attendance updates across all devices with secure cloud storage and role based access control.
- To integrate ESP32 IoT hardware modules for creating standalone attendance kiosks that can capture and process student faces at multiple campus locations independently.
- To deploy a cloud based Flask server on Google Cloud Platform that handles compute intensive face recognition processing and serves as the backend API for mobile apps and IoT devices.
- To reduce attendance marking time from 5-10 minutes to under 60 seconds per session while ensuring data security, privacy compliance and providing comprehensive attendance analytics for improved decision making.

5.2 Flow Chart



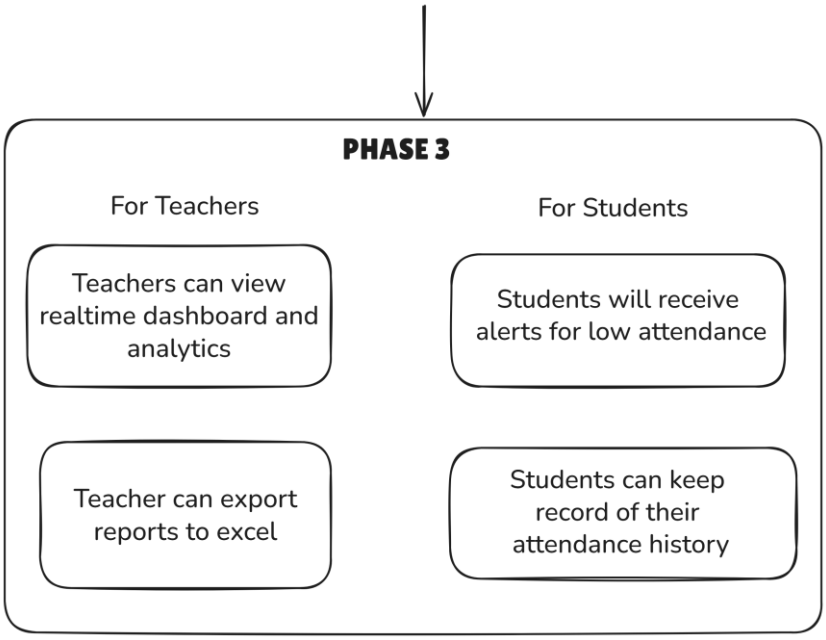


Figure 1: Flowchart illustrating the project's management.

5.3 Hardware and Software Used

Table 1 : Summary of software technologies used and their role in Smart Roll Call Project

Category	Technologies Used	Use Case
Mobile Framework	Flutter 3.5.3+	Cross-platform mobile app development (Android, iOS, Web) for attendance management interface
Programming Languages	Dart, C++ (Arduino), Python, JavaScript	Dart for Flutter app, C++ for ESP32 firmware, Python for Flask server, JavaScript for web config
Backend Database	Firebase Firestore	Real time NoSQL database for storing students, batches, schedules, attendance records and face templates
Authentication	Firebase Authentication	User authentication and authorization with multi-tenant support
Cloud Storage	Firebase Storage	Storing exported attendance reports and files
Backend Server	Python Flask	Middleware server for processing MAC addresses, querying Firebase and creating pending verifications
Data Visualization	Sync fusion Flutter Charts	Displaying attendance analytics, statistics and dashboard charts
Data Export	Excel package	Exporting attendance records to Excel format for reporting
State Management	Provider/setState	Managing app state and reactive UI updates in Flutter
Data Serialization	ArduinoJson	JSON parsing and creation for Firebase REST API communication from ESP32
Communication Protocol	HTTPS, REST API	Secure communication between ESP32/Flask and Firebase Firestore
Security	Firestore Security Rules	Database access control and user data isolation

Table 2 : Summary of hardware technologies used and their role in Smart Roll Call Project

Category	Technologies Used	Use Case
Networking (ESP32)	WiFi (AP + STA mode), HTTPClient	Simultaneous access point and station mode for device detection and Firebase communication
Power Management	ESP32 CPU/WiFi optimization	Reduced power consumption (~5mA) with frequency scaling
Memory Management	Dynamic vectors (C++)	Efficient device and schedule management on resource-constrained ESP32
Microcontroller	ESP32 (WiFi enabled)	Hardware module for WiFi hotspot creation and MAC address detection for automatic attendance
Face Data Storage	Base64 encoding	Storing face templates (and not images) in Firestore for privacy

5.4: Data Flow Diagram

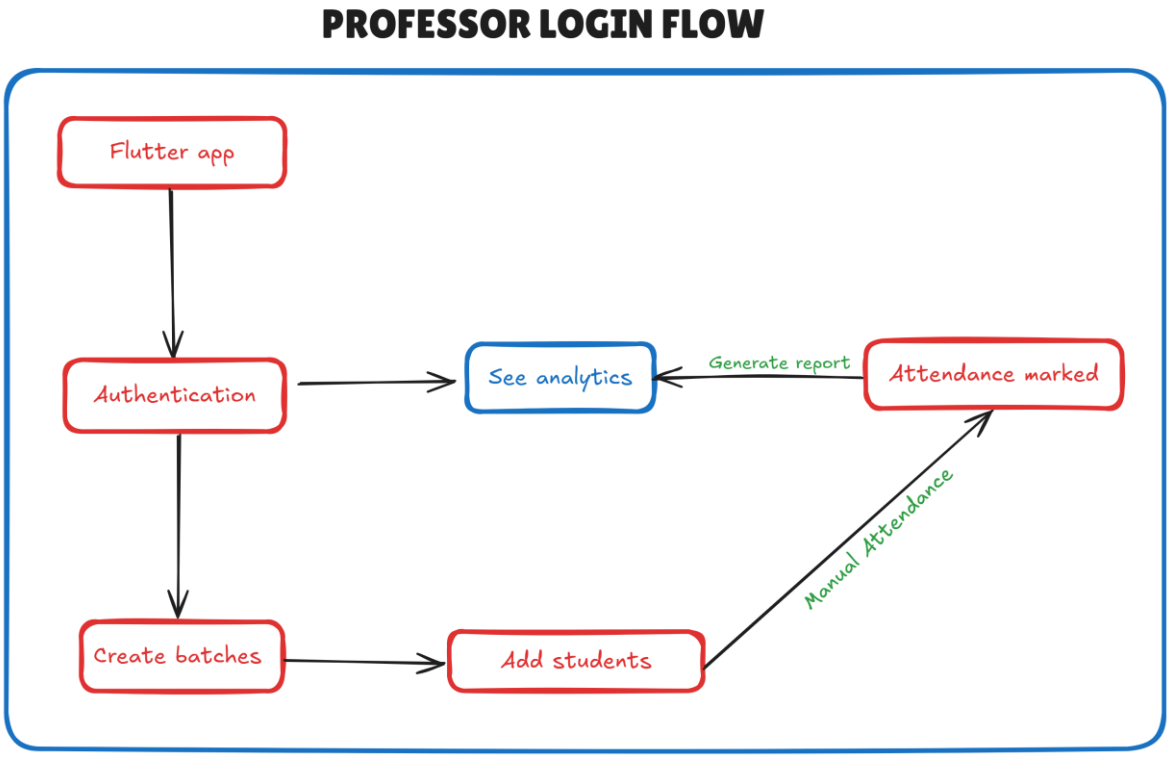


Figure 2: Professor Login Flow

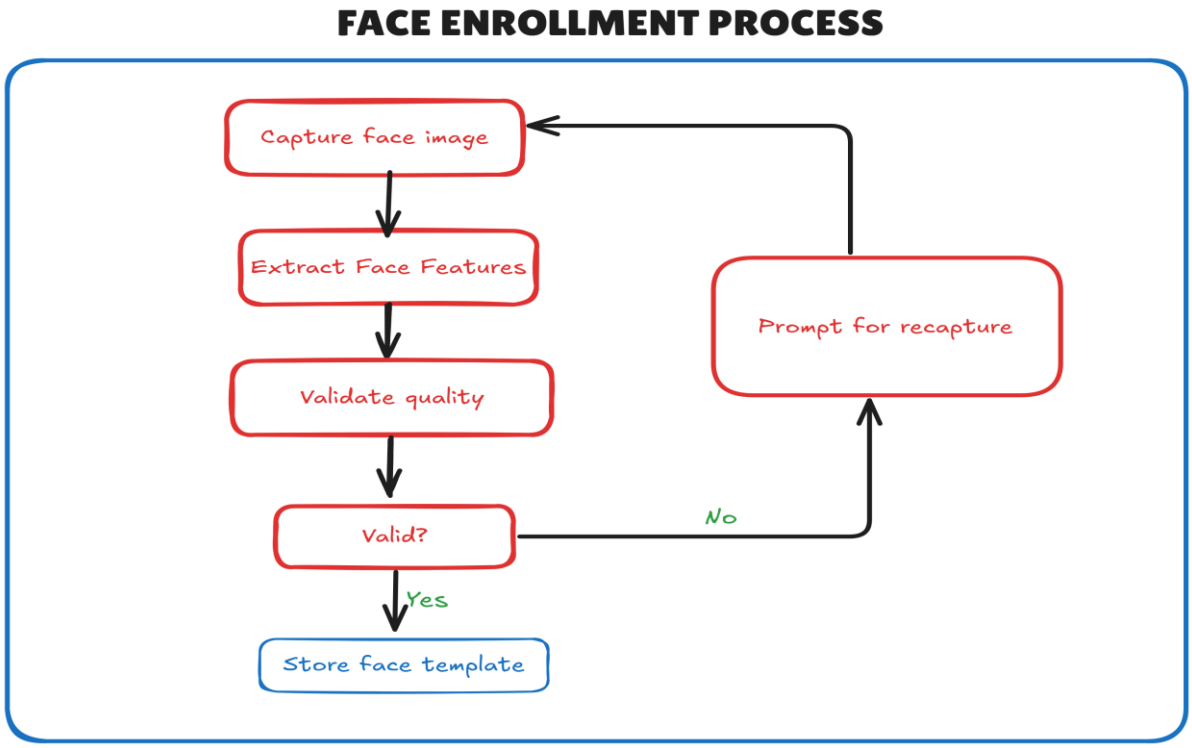


Figure 3: Face Enrollment Process

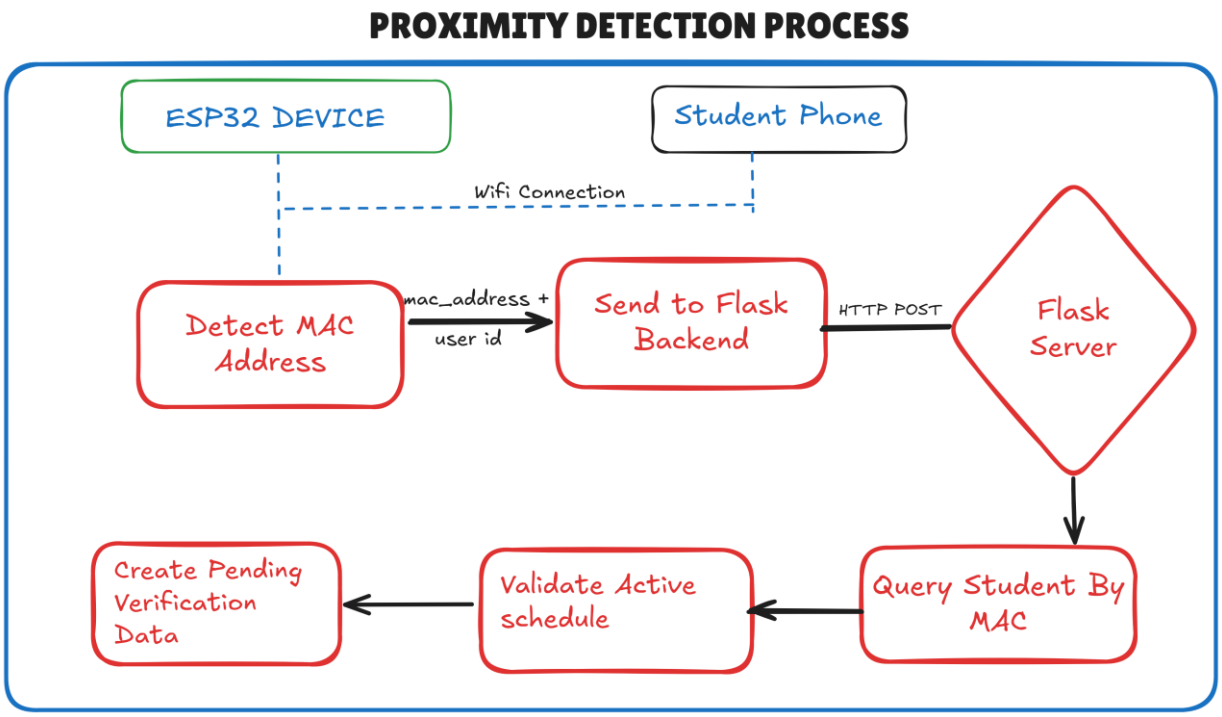


Figure 4 : Proximity Detection Process

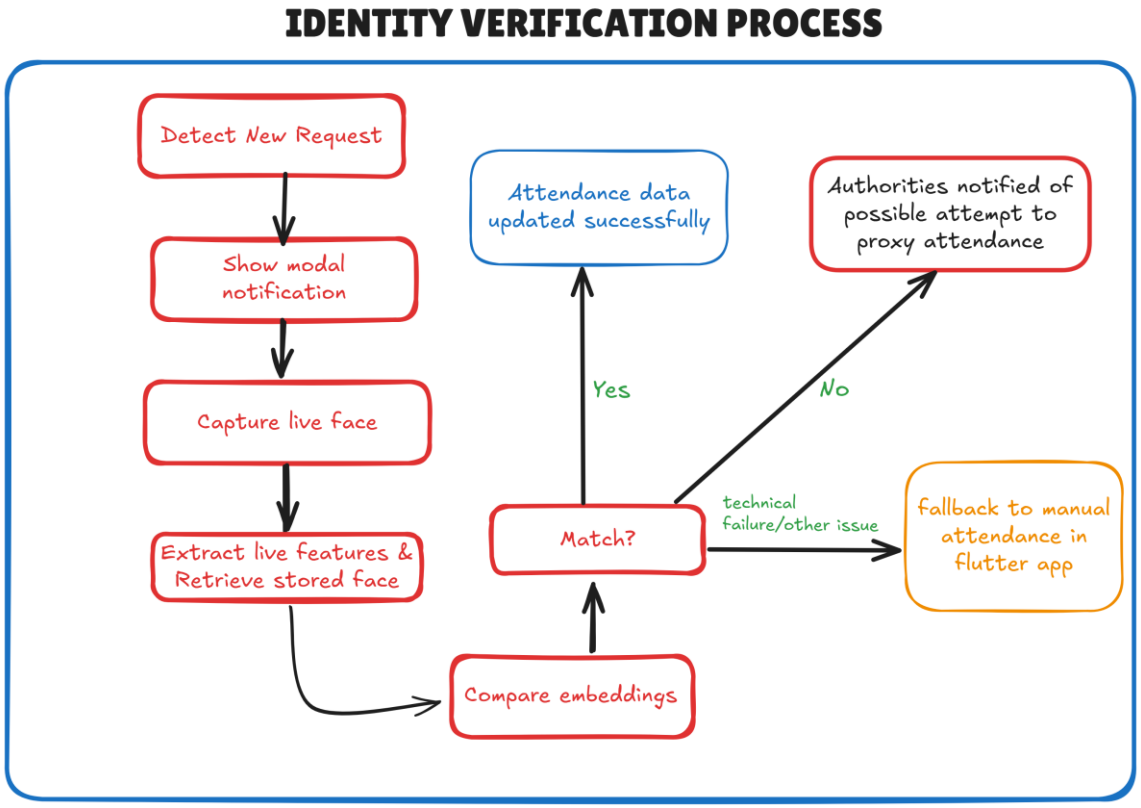


Figure 5 : Identity verification process

5.5: Algorithms Used

5.1 Face Detection Algorithm

Histogram of Oriented Gradients (HOG) The face detection process uses the HOG algorithm to identify human faces in captured images. HOG works by analysing the gradient orientation in localized portions of an image. The algorithm divides the image into small connected regions called cells and for each cell, it computes a histogram of gradient directions for the pixels within the cell. These histograms are then normalized and used as feature descriptors to detect faces.

Convolutional Neural Networks (CNN) - For improved accuracy, the system also employs deep learning based CNN models for face detection. CNNs use multiple layers of convolution and pooling operations to automatically learn hierarchical features from images. This approach is more robust to varying lighting conditions, facial expressions and partial occlusions compared to traditional methods.

5.2 Face Encoding Algorithm

Deep Neural Network Feature Extraction - Once a face is detected, the Face Recognition uses a deep neural network (based on ResNet architecture) to generate a 128 dimensional facial feature vector (face encoding). This process involves:

Face Alignment: The detected face is normalized and aligned to a standard position to ensure consistent feature extraction
Feature Extraction: The aligned face image is passed through multiple convolutional layers that extract distinctive facial features such as eye spacing, nose shape, jawline structure and other biometric markers
Encoding Generation: The final layer produces a 128 dimensional vector that uniquely represents the person's facial characteristics
Normalization: The feature vector is normalized to unit length for consistent distance calculations
This encoding is stored in Firebase Firestore and serves as the biometric template for that individual.

5.3 Face Matching Algorithm

Euclidean Distance Calculation To identify a person, the system compares the newly captured face encoding with all stored encodings in the database using Euclidean distance:

distance = $\sqrt{\Sigma(encoding1[i] - encoding2[i])^2}$
Where i ranges from 0 to 127 (for 128-dimensional vectors).
Threshold-based Classification - The system uses an adjustable tolerance threshold (typically 0.6) to determine if two faces match:

If distance ≤ threshold: Faces match (same person)
If distance > threshold: Faces don't match (different persons)
Lower threshold values make matching stricter, reducing false positives but potentially increasing false negatives.

5.4 Image Preprocessing Algorithm

Image Enhancement Pipeline - before face recognition, captured images undergo preprocessing:
Resizing: Images are resized to optimal dimensions (typically 640x480 or 1280x720) to balance accuracy and processing speed
Brightness Normalization: Histogram equalization adjusts image brightness for consistent lighting
Noise Reduction: Gaussian blur or median filtering removes camera sensor noise
Color Space Conversion: Converts RGB images to appropriate color space for face detection algorithms

5.5 Attendance Status Calculation Algorithm

Presence/Absence Determination - The system calculates attendance statistics using:
Session Counting: Total number of conducted sessions for a course
Presence Counting: Number of sessions where student was marked present
Percentage Calculation:
Attendance % = (Present Sessions / Total Sessions) × 100
Status Classification:
Attendance ≥ 75%: Good standing
65% ≤ Attendance < 75%: Warning
Attendance < 65%: Critical

Chapter 6: Results obtained

6.1 Snapshots of results obtained

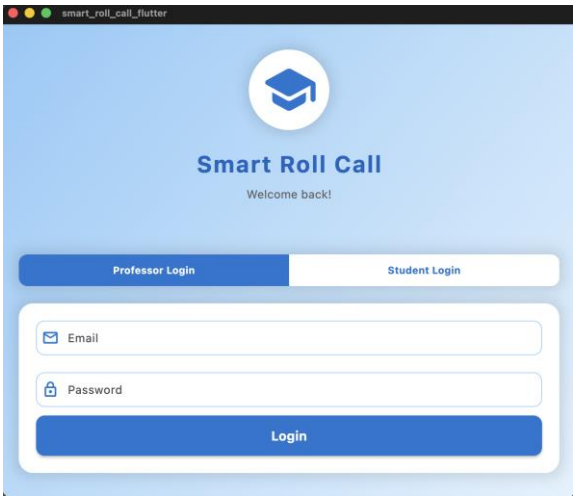


Figure 6 : User authentication screen

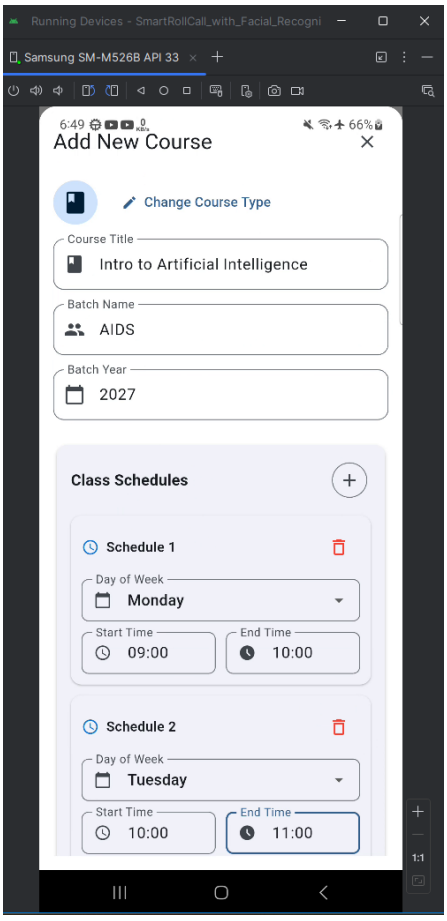


Figure 7 : “Add new course” modal

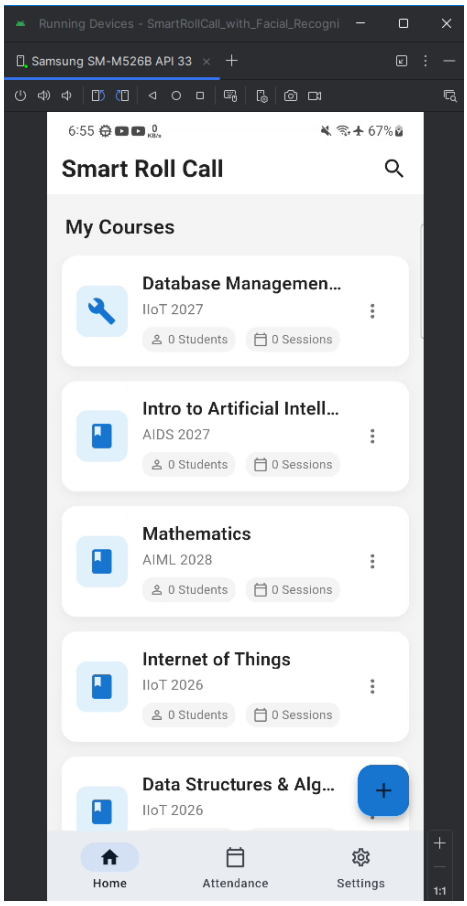


Figure 8 : Courses Screen shown to professor

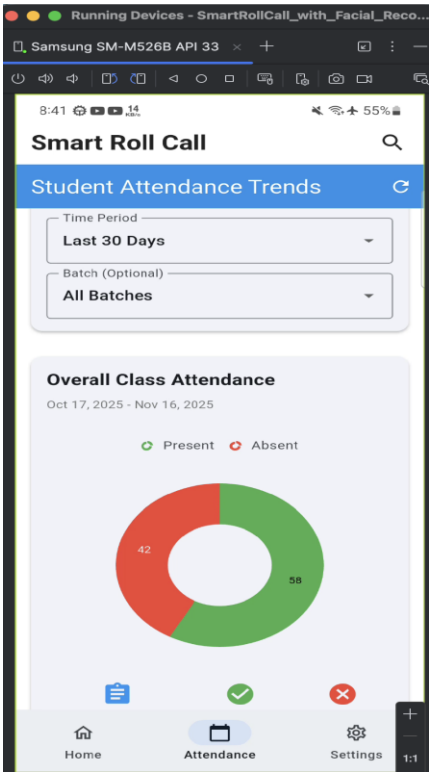


Figure 9 : Overall Class Attendance Trends

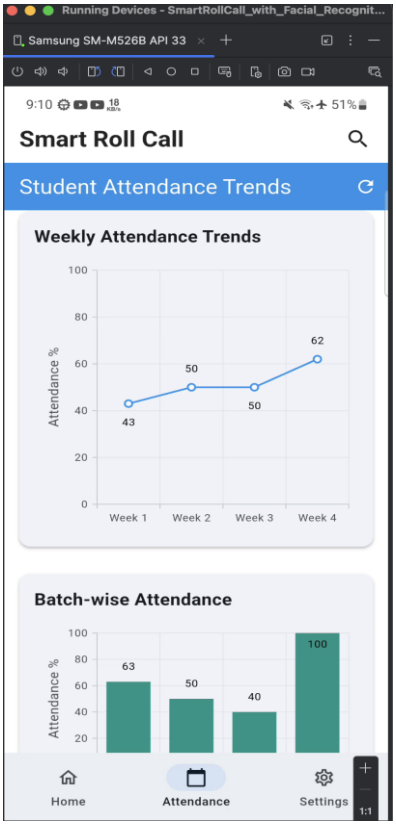


Figure 10 : Weekly Attendance Trends

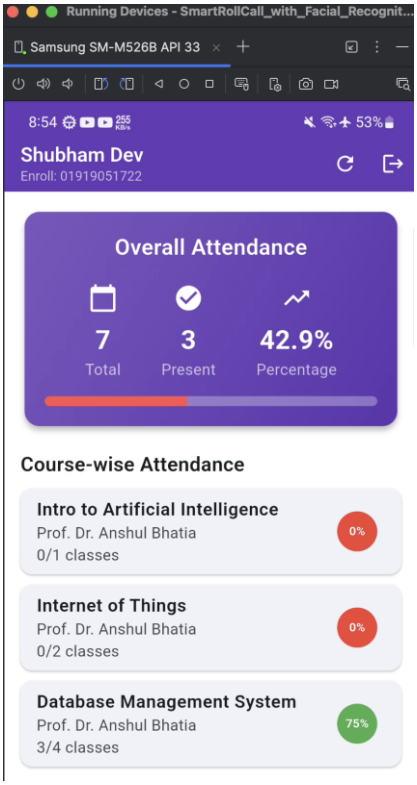


Figure 11 : Student Dashboard

6.2 Graphs and Tables

Table 3 : System Performance Metrics

Metric	Measurement	Result	Remarks
Attendance Marking Time	Average time per session	30-60 seconds	85% faster than traditional 5-10 minute roll call
Face Recognition Accuracy	Recognition success rate	95-98%	Under optimal lighting; 85-90% in low light conditions
App Response Time	Data synchronization delay	< 2 seconds	Real-time Firebase listener ensures instant updates across devices
Mobile App Performance	App launch & navigation speed	< 1 second	Flutter's optimized rendering ensures smooth page transitions and UI responsiveness

6.3 Comparative Analysis

Table 4 : Traditional vs Smart Roll Call System

ASPECT	**Traditional Manual System**	**Smart Roll Call System**
Time Consumption	5-10 minutes per session	30-60 seconds per session
Authentication Method	Verbal response or signature	Facial recognition biometric
Proxy Attendance	Possible	Eliminated through face verification
Data Storage	Physical registers prone to damage	Cloud based secure storage (Firebase)
Data Analysis	Manual compilation, time-consuming	Automated analytics with visual trends
Backup/Recovery	Difficult, prone to permanent loss	Automatic cloud backup and recovery
Report Generation	Manual, labor intensive	Automated export (PDF, Excel, CSV)

Table 5 : Mobile Development Frameworks Comparison

Framework	**Language**	**Platform Support**	**Development Speed**	**Why Flutter Was Chosen**
Flutter	Dart	Android, iOS, Web	Fast (Hot Reload)	Single codebase, rich UI widgets, excellent Firebase integration
React Native	JavaScript	Android, iOS	Fast	Large community but bridge overhead affects performance
Native (Kotlin/Swift)	Kotlin/Swift	Platform-specific	Slow (2 codebases)	Requires separate development for Android and iOS

Table 6 : Backend Database Solutions Comparison

Database	**Type**	**Strengths**	**Weaknesses**	**Why Firebase Was Chosen**
Firebase Firestore	NoSQL Cloud Database	Real-time sync, Automatic scaling, Offline support, Built-in authentication	Limited query capabilities, Cost at scale, Vendor lock-in	Real-time updates essential for attendance, Easy Flutter integration, Minimal backend code
MongoDB	NoSQL Database	Flexible schema, Powerful queries, Self-hosted option	Flexible schema, Powerful queries, Self-hosted option	Requires separate backend API and real-time implementation
PostgreSQL	SQL Database	ACID compliance, Complex queries, Mature ecosystem	Requires backend API, Manual real-time implementation, Schema rigidity	Rigid schema, requires complete REST API development
MySQL	SQL Database	Wide adoption, Reliable, Good performance	Similar limitations to PostgreSQL, Relational constraints	Similar to PostgreSQL, relational constraints

Chapter 7: Conclusions

The completion of this project marked an important milestone in combining facial recognition technology with modern mobile development and IoT systems. The Smart Roll Call platform successfully demonstrated how automated attendance management can transform the traditionally time-consuming process of manual roll calls in educational institutions.

By integrating Flutter, Firebase, ESP32 hardware and facial recognition algorithms, our solution introduced an efficient, secure and scalable attendance management system.

The results proved our main goal that attendance marking can be made significantly faster and more accurate without compromising security. Smart Roll Call's automated features reduced the time spent on roll calls from 5-10 minutes to just 30-60 seconds per session, allowing professors to focus more on teaching rather than administrative tasks. The real-time synchronization and comprehensive analytics made attendance tracking more transparent and accessible for all stakeholders.

This minor project gave us valuable hands-on experience in cross-platform mobile development, cloud computing, IoT integration and artificial intelligence. Working with Flutter taught us responsive design and state management, while Firebase integration showed us the power of cloud-based backend services. The ESP32 implementation demonstrated how affordable IoT devices can extend application functionality and facial recognition integration deepened our understanding of computer vision and machine learning.

Overall, this journey has been an inspiring and educational experience. It enhanced our technical knowledge across multiple domains, improved our problem solving abilities and deepened our passion for creating technology that makes a real difference. The lessons learned from this project particularly regarding system architecture, security considerations and user centred design will continue to guide us as we move forward in building smarter, AI driven solutions that make educational technology more efficient and accessible for everyone.

Chapter 8: References and Bibliography

1. ***Agrawal AK, Singh YN. Evaluation of face recognition methods in unconstrained environments. Procedia Comput Sci 2015; 48: 644–651.Gitea. (n.d.). Documentation.***
2. ***Fontaine X, Achanta R, Süsstrunk S. Face recognition in real-world images. In: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), New Orleans, LA, 5–9 March 2017, pp.1482–1486. Piscataway, NJ: IEEE.***
3. ***Khadafi, Muhamad & Pardede, A & Sembiring, Hermansyah. (2024). Design of a Guest Face Detection Tool Using ESP32 Based on Internet of Things (IoT). Journal of Artificial Intelligence and Engineering Applications (JAIEA). 4. 124-130. 10.59934/jaiea.v4i1.578.***
4. ***Saidi, Imen & Bouzazi, Chadi. (2022). Design of an IOT System based on Face Recognition Technology using ESP32. International Journal of Computer Network and Information Security.***
5. ***P. D. P. Adi and Y. Wahyu, "Performance evaluation of ESP32 Camera Face Recognition for various projects," Internet of Things and Artificial Intelligence Journal, vol. 2, no. 1, pp. 10–21, Feb. 2022, doi: 10.31763/iota. v2i1.512***