

**AUTOMATED ATTENDANCE SYSTEM WITH
ANTI SPOOFING FACE RECOGNITION DETECTION**

BY

WILSON CHUA WAI LUN

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF INFORMATION SYSTEMS (HONOURS)

INFORMATION SYSTEMS ENGINEERING

Faculty of Information and Communication Technology

(Kampar Campus)

JUNE 2025

COPYRIGHT STATEMENT

© 2025 Wilson Chua Wai Lun. All rights reserved.

This Final Year Project proposal is submitted in partial fulfillment of the requirements for the degree of Bachelor of Information Systems (Honours) Information Systems Engineering at Universiti Tunku Abdul Rahman (UTAR). This Final Year Project proposal represents the work of the author, except where due acknowledgment has been made in the text. No part of this Final Year Project proposal may be reproduced, stored, or transmitted in any form or by any means, whether electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the author or UTAR, in accordance with UTAR's Intellectual Property Policy.

ACKNOWLEDGEMENTS

I would like to extend my heartfelt gratitude to everyone who has supported me throughout the completion of my Final Year Project 2.

First and foremost, my sincere appreciation goes to my supervisor, Dr. Muhammad Syaiful Amri Bin Suhaimi, for his invaluable guidance, continuous support, and constructive feedback during the entire course of this project. His expertise and encouragement have been instrumental in ensuring the successful completion of my work.

I would also like to thank the Faculty of Information and Communication Technology (FICT) for providing the facilities, resources, and a conducive environment that enabled me to conduct my research and development effectively.

Special thanks go to my family and friends for their constant motivation, understanding, and encouragement throughout this journey. Their support has given me the strength and determination to complete this project successfully.

Finally, I am grateful to all who have contributed directly or indirectly to the success of this project.

ABSTRACT

This project creates an automated attendance system that uses computer vision and biometric authentication to make attendance tracking faster and more secure for schools and universities. The system uses face recognition and liveness detection to make sure students are correctly identified, stop fake attempts like using photos or videos, and make attendance reporting easier. It uses the `face_recognition` library to get facial features, and `dlib` to check if the face is real by detecting actions like smiling, blinking, and opening the mouth. Student data and attendance records are stored in Google Sheets, and a Flask web app is used for student registration, subject management, and sending email notifications for absentees. Studies of existing systems, such as fingerprint scanners, RFID systems, and other face recognition tools, show that many of them have weak anti-spoofing measures and are not very user-friendly. This project solves those problems with a two-step liveness check and a simple web interface. The final system uses Python for the backend, real-time webcam processing, and cloud storage, providing a secure, easy-to-use, and scalable solution for attendance management.

Area of Study: Computer Vision, Machine Learning

Keywords: Face Recognition, Liveness Detection, Attendance Automation, Flask Application, Anti-Spoofing, Real-Time Processing, Cloud-Based Storage

TABLE OF CONTENTS

COPYRIGHT STATEMENT	ii
ACKNOWLEDGEMENTS	iii
ABSTRACT	iv
TABLE OF CONTENTS	v
LIST OF FIGURES	viii
LIST OF TABLES	ix
LIST OF SYMBOLS	x
LIST OF ABBREVIATIONS	xi
CHAPTER 1 INTRODUCTION	1
1.1 Problem Statement and Motivation	1
1.2 Objectives	1
1.3 Project Scope and Direction	2
1.4 Contributions	3
1.5 Report Organization	3
CHAPTER 2 LITERATURE REVIEW	5
2.1 Review of Technologies	5
2.1.1 Hardware Platform	5
2.1.2 Firmware / Operating System	6
2.1.3 Database	6
2.1.4 Programming Language	7
2.1.5 Summary of the Technologies Review	8
2.2 Previous Works on Face Recognition Detection	8
2.2.1 Face Recognition Smart Attendance System Using Deep Transfer Learning	8
2.2.2 Face Recognition Based Attendance System Using Haar Cascade and LBPH.....	9
2.2.3 Face Recognition Attendance System Using Local Binary Pattern	10
2.2.4 Implementation of Automated Attendance System Using Eigenfaces.....	11
2.2.5 Face Recognition Based Attendance Management System Using VGG16	11
2.2.6 Summary of Existing Systems	13
2.3 Previous Works on Face Recognition with Anti Spoofing.....	13
2.3.1 Classic (Handcrafted) Texture & Motion Methods.....	13
2.3.2 Deep-Learning Appearance Models.....	14
2.3.3 Temporal & Physiological (Video) Cues	14
2.3.4 Depth/Sensor-Based and Multi-Modal Methods.....	15
2.3.5 Domain Generalization & Adaptation.....	15

TABLE OF CONTENTS

2.3.6	Summary of Anti Spoofing Methods	16
CHAPTER 3	SYSTEM METHODOLOGY	17
3.1	System Design Diagram	17
3.1.1	System Architecture Diagram	17
3.1.2	Data Flow Diagram	22
3.1.3	Face Recognition and Liveness Detection Equations	23
CHAPTER 4	SYSTEM DESIGN	28
4.1	System Block Diagram	28
4.2	System Flowchart	29
4.3	Database Schema	31
4.3.1	Students Sheet	31
4.3.2	Attendance Sheet	32
4.4	Activity Diagram	32
4.5	Use Case Diagram	34
CHAPTER 5	SYSTEM IMPLEMENTATION	36
5.1	Hardware Setup	36
5.2	Software Setup	37
5.2.1	Explanation of Each Libraries	37
5.3	Settings and Configuration	38
5.3.1	Google Sheets API	38
5.3.2	SMTP Configuration	38
5.3.3	File Structure	39
5.4	System Operation	39
5.5	Implementation Issues and Challenges	46
5.6	Conclusion For Chapter 5	48
CHAPTER 6	SYSTEM EVALUATION AND DISCUSSION	49
6.1	System Testing and Performance Metrics	49
6.1.1	Testing Objectives	49
6.1.2	Face Recognition Testing	49
6.1.2.2	Testing Under Different Partial Occlusion	52
6.1.2.3	Anti Spoofing Accuracy Testing	57
6.2	Project Challenges	58
6.2.1	Performance Optimization on Low-End Hardware	58
6.2.2	Lighting Sensitivity in Face Recognition	59
6.2.3	Anti-Spoofing Robustness Against Sophisticated Attacks	59
6.3	Objective Evaluation	59
6.4	Conclusion For Chapter 6	60

TABLE OF CONTENTS

CHAPTER 7 CONCLUSION AND RECOMMENDATION.....	62
7.1 Conclusion.....	62
7.2 Recommendation.....	63
REFERENCES	A-1
POSTER.....	A-5

LIST OF FIGURES

Figure Number	Title	Page
Figure 3.1.1.1	System Architecture Diagram	18
Figure 3.1.2.1	Data Flow Diagram	22
Figure 4.1.0.1	System Block Diagram	28
Figure 4.2.0.1	System Flowchart	30
Figure 4.4.0.1	Activity Diagram	33
Figure 4.5.0.1	Use Case Diagram	34
Figure 5.4.0.1	Hompage (index.html)	40
Figure 5.4.0.2	Student Registration (register.html)	41
Figure 5.4.0.3	Add Subject (teacherpanel.html)	42
Figure 5.4.0.4	Start Attendance (teacherpanel.html)	42
Figure 5.4.0.5	Perform liveness action (Smile)	43
Figure 5.4.0.6	Perform liveness action (Blink)	44
Figure 5.4.0.7	Perform liveness action (Open Mouth)	44
Figure 5.4.0.8	Student Recognized	45
Figure 5.4.0.9	Attendance Marked	45
Figure 5.4.0.10	Absentee Email Notification	46
Figure 5.5.0.1	Find the correct camera index.	47
Figure 6.1.2.1.1	Overbright Lighting onto Face	50
Figure 6.1.2.1.2	Normal Indoor Light	51
Figure 6.1.2.1.3	Dark Environment	51
Figure 6.1.2.2.1	Hand Covering One Eye	53
Figure 6.1.2.2.2	Hand Covering Both Eyes	53
Figure 6.1.2.2.3	Hand Covering Mouth and Nose	54
Figure 6.1.2.2.4	Wearing Normal Glasses	54
Figure 6.1.2.2.5	Wearing Sun Glasses	55
Figure 6.1.2.2.6	Headwear Covering Forehead	55
Figure 6.1.2.3.1	Spoofing with Image and Video	57

LIST OF TABLES

Table Number	Title	Page
Table 2.1.5.1	Summary of Technologies Reviewed	8
Table 4.3.1.1	Student Sheet	31
Table 4.3.2.1	Attendance Sheet	32
Table 5.1.0.1	Hardware Setup	36
Table 5.2.0.1	Software Setup	37
Table 5.2.1.1	Libraries Explanation	38
Table 5.3.3.1	File Structure	39
Table 6.1.2.1.1	Different Light Test Result	52
Table 6.1.2.2.1	Different Partial Occlusion Result	56
Table 6.1.2.3.1	Anti Spoofing Result	57

LIST OF SYMBOLS

Symbol	Definition
e_d	Face encoding from real time video frame
e_s	Stored face encoding

LIST OF ABBREVIATIONS

<i>API</i>	Application Programming Interface
<i>CNN</i>	Convolutional Neural Network
<i>HOG</i>	Histogram of Oriented Gradients
<i>LBPH</i>	Local Binary Patterns Histogram
<i>HTML</i>	HyperText Markup Language
<i>PCA</i>	Principle Component Analysis
<i>SQL</i>	Structured Query Language

CHAPTER 1 INTRODUCTION

In this chapter, we explain the background and reason for developing an automated attendance system with anti-spoofing face recognition. We describe the problem statement, define the project objectives and scope, explain the contributions to the field, and give an overview of the report's organization. The fast growth of facial recognition technology and its use with cloud-based systems has created new ways to automate traditional processes such as attendance tracking in educational institutions. This project solves the problems of manual attendance systems by using face recognition, liveness detection, and cloud integration to make a secure, efficient, and easy-to-use solution.

1.1 Problem Statement and Motivation

Manual attendance systems in educational institutions, such as paper roll calls or entering data into spreadsheets, take a lot of time and can have many mistakes. These methods also use up valuable class time, leaving less time for teaching and learning. They are also easy to cheat, for example when one student marks another as present. Automated attendance systems, such as RFID cards or fingerprint scanners, also have weaknesses like high setup costs, the need for physical contact, and the risk of being tricked by stolen cards or fake fingerprints.

Facial recognition gives a contactless way to track attendance, but its use is limited because of security risks. Attackers can trick facial recognition systems by using photos or videos to pretend to be real students, which makes the system less reliable. These spoofing attacks are a big problem in educational settings where accurate attendance records are very important for administration and academic purposes.

The motivation for this project is to create a secure and automated attendance system that removes the problems of manual methods and solves the security risks of facial recognition. It uses liveness detection to stop photo and video spoofing, connects with Google Sheets for real-time data management, and provides a web-based interface for easy use. This system aims to make attendance tracking simpler, improve security, and reduce the work of educators in classroom environments.

1.2 Objectives

The primary goal of this project is to develop an automated attendance system that uses facial recognition with anti-spoofing measures to ensure secure and accurate attendance tracking in educational settings. The specific objectives are:

1. To develop an anti-spoofing facial recognition system with liveness detection using a pre-trained ResNet model, 68-point landmark predictor, and HOG + linear SVM (dlib), achieving $\geq 90\%$ spoof detection accuracy.
2. To integrate a lightweight cloud database solution by using Google Sheets to store student data and attendance records for real-time access, cost-effective scalability, and easy maintenance.
3. To implement a smart attendance system that accurately records attendance and assists teachers by automatically sending absentee notifications via email using data from Google Sheets, streamlining classroom management and saving time.

1.3 Project Scope and Direction

This project aims to deliver a functional prototype of a web-based Automated Attendance System with Anti-Spoofing Face Recognition. The system utilizes facial recognition to identify students and record attendance in Google Sheets, featuring a Flask-based web application for student registration, subject management, and attendance initiation, a face recognition module built with OpenCV, face_recognition, and dlib libraries, and anti-spoofing measures through liveness detection requiring two random facial actions (smile, open mouth, or blink). The prototype supports real-time face detection via a single webcam, automated attendance updates for specific subjects, and real-time absentee notifications sent via email to teachers. The system is designed for small to medium-sized classroom environments and assumes a stable internet connection for Google Sheets access.

In FYP2, the focus is on implementing and refining the prototype to ensure reliability, security, and usability in a classroom setting. This includes optimizing the facial recognition module to achieve at least 90% spoof detection accuracy through liveness checks, enhancing the Flask-based web interface for seamless user interaction. Extensive testing will be done to check recognition accuracy, liveness detection performance, and system behaviour under different conditions, for example different lighting.

1.4 Contributions

Automated attendance systems help make administrative tasks in schools easier, but many current solutions have problems with security, high setup costs, and poor usability. Traditional biometric systems like fingerprint or RFID require expensive hardware and can be bypassed, while facial recognition systems can be tricked with photos or videos. This project solves these issues by creating a secure, cost-effective, and scalable automated attendance system with anti-spoofing face recognition.

The system improves security with liveness detection, asking students to perform two random facial actions, for example smile, open mouth, or blink, and verifying them across multiple frames using dlib's facial landmark detection. This stops spoofing attempts with photos or videos. Data management is handled using Google Sheets as a lightweight cloud database to store student data and attendance records. This provides lightweight, real-time data management suitable for small to medium-sized classrooms.

The system also automates absentee reporting by comparing attendance before and after the session and sending detailed email notifications to teachers. This reduces administrative work and helps teachers follow up on absentees quickly. A simple Flask-based web interface lets teachers and administrators easily register students, manage subjects, and start attendance with little training.

The system is built with open-source libraries like face_recognition, dlib, OpenCV, and Flask, making it customizable for other uses such as event check-ins or employee tracking. Together, these features create a practical, secure, and adaptable attendance solution that solves the problems of current systems and is ready for schools or other organizations to use.

1.5 Report Organization

This report is organized into seven chapters, each beginning on a new page, with sections and subsections limited to three levels as per the project guidelines. Chapter 1 introduces the project, covering the problem statement, motivation, objectives, scope, contributions, and report organization. Chapter 2 reviews related work on facial recognition, anti-spoofing methods and databases. Chapter 3 describes the system methodology, including architecture, data flow diagram. Chapter 4 explains the system design diagrams and flowchart. Chapter 5 discusses system implementation, including hardware and software setup, configuration, and key

CHAPTER 1 INTRODUCTION

challenges. Chapter 6 presents system evaluation, testing results, and discussion. Chapter 7 concludes with a summary of contributions, limitations, and recommendations for future improvements.

CHAPTER 2 LITERATURE REVIEW

This chapter reviews the key technologies, methods, and existing systems related to automated attendance using facial recognition. It covers hardware platforms, operating systems, databases, programming languages, and algorithms commonly employed in previous studies and projects. The chapter also examines the strengths and limitations of existing face recognition systems and anti-spoofing techniques, providing a foundation for selecting appropriate technologies and designing an effective, secure, and reliable attendance system for educational environments.

2.1 Review of Technologies

In this section, we review technologies used in past works or systems for face recognition attendance: hardware platforms, firmware/operating systems, databases, programming languages. This helps choose and justify technologies for our system.

2.1.1 Hardware Platform

Past face recognition attendance systems have used a variety of hardware platforms, chosen based on cost, portability, performance, and real-time capability. Many studies have implemented Raspberry Pi with a Pi Camera module to create standalone, low-cost attendance systems. This setup is compact, consumes little power, and can capture faces in real time, but it is limited by its computational power, leading to higher latency for deep learning models or when recognizing many faces, and may require better cameras or IR modules to perform well in low-light conditions [1].

Some lightweight solutions have employed ESP32-CAM modules, which combine a microcontroller and camera for extremely low-cost and low-power face recognition systems. However, these modules have limited RAM, flash, and processing capability, often requiring simplified algorithms or offloading computation to a server [2].

In contrast, commercial standalone devices such as FaceIN, Lathem's PCFACE, and KENT CamAttendance are purpose-built machines equipped with cameras (sometimes dual cameras for 3D analysis), built-in displays, memory, and network connectivity. These devices provide high reliability, fast recognition, and robust deployment but are more expensive, less customizable, and may not easily support custom anti-spoofing methods [3].

Some implementations use mixed or edge systems, where embedded devices like Raspberry Pi handle image capture locally while remote servers or web applications manage attendance data processing and storage, striking a balance between cost, flexibility, and performance [4].

2.1.2 Firmware / Operating System

The choice of firmware or operating system depends largely on the hardware platform and plays an important role in supporting camera drivers, networking, scheduling, and running face recognition applications.

Raspberry Pi-based systems usually use Linux variants such as Raspberry Pi OS (Raspbian) or similar distributions. These operating systems are widely preferred because they support installing libraries like OpenCV, Python, and TensorFlow, making it easy to implement and customize face recognition solutions [1][5].

Commercial standalone devices, such as Lathem's PCFACE FaceIN, often run embedded Linux or Android-based operating systems optimized for stability and real-time performance [3]. Some devices, like FaceScribe Plus, use in-built Linux OS with non-volatile memory to ensure reliability and data retention [6].

In contrast, microcontroller-based systems such as ESP32-CAM use lightweight firmware or real-time operating systems (RTOS), and in some cases even bare-metal programming are used to keep RAM usage low and optimize performance. These minimal systems allow face detection or recognition tasks but often require simplified models or cloud offloading to maintain acceptable performance [2].

2.1.3 Database

The database is used to store face templates, user information, and attendance logs. Many studies adopt MySQL as the primary database for its reliability, relational structure, and compatibility with web applications [7]. For lightweight systems running on embedded devices, SQLite is often preferred because it requires no server installation and stores data locally in a single file, reducing complexity [8].

Some prototypes use CSV files or image folders as a simple way to store user data and training images, which is suitable for small-scale systems. More modern solutions make use of cloud

databases or remote storage to allow real-time syncing of attendance data across multiple locations, improving scalability and accessibility [9].

2.1.4 Programming Language

Most face recognition attendance systems are developed in Python, as it offers a rich ecosystem of libraries that make implementation efficient and less complex. Popular libraries such as OpenCV, dlib, and `face_recognition` provide pre-built functions for face detection, feature extraction, and recognition, significantly reducing development time [10][11]. Python's versatility also allows seamless integration with relational databases like MySQL and SQLite, as well as lightweight web frameworks such as Flask and Django, enabling developers to create end-to-end solutions that include both the recognition logic and the attendance management backend.

Some researchers and developers opt for C++, especially when performance and speed are critical. Since OpenCV is originally written in C++, using it at a lower level allows for faster execution, lower latency, and better handling of real-time video streams — which is crucial for systems deployed in environments with high user throughput [11].

On the frontend side, many systems provide web-based dashboards for administrators and teachers, using HTML, CSS, JavaScript, and Bootstrap to deliver responsive and user-friendly interfaces that work across devices [12]. Some projects also integrate AJAX and REST APIs to allow real-time communication between the backend and the web interface for instant attendance updates.

For **desktop-based solutions**, GUI frameworks like **Tkinter** are commonly used to create simple, lightweight interfaces that allow users to register faces, view attendance logs, and manage system settings without requiring a web browser [7]. In more advanced projects, libraries like **PyQt** or **Kivy** are used for richer desktop applications with better styling and cross-platform support.

Overall, the programming language choice depends on the project requirements, such as speed, scalability, ease of deployment, and user interface needs. Python remains the most popular choice for its balance of simplicity, flexibility, and wide community support, while C++ is preferred in performance-intensive applications.

2.1.5 Summary of the Technologies Review

Aspect	Findings from Literature
Hardware Platform	Low-cost platforms like Raspberry Pi are widely adopted for academic prototypes due to affordability and ease of use. High-performance hardware with embedded Linux or Android is preferred for commercial deployments to ensure better reliability and faster processing.
Operating System	Raspberry Pi OS is commonly used in academic projects, while embedded Linux or Android is used in commercial-grade systems.
Programming Language	Python is the dominant choice due to its strong library support and rapid development capabilities.
Library	OpenCV is widely used for face detection and recognition, providing an accessible and well-documented framework.
Database	SQLite or MySQL are common for local database management. Cloud databases are preferred when real-time multi-device access is required.
User Interface	Web technologies (HTML, CSS, JS) are frequently used to create simple and accessible frontends for interaction with the attendance system.

Table 2.1.5.1 Summary of Technologies Reviewed

2.2 Previous Works on Face Recognition Detection

2.2.1 Face Recognition Smart Attendance System Using Deep Transfer Learning

Overview of System using Deep Transfer Learning

This approach fine-tunes three pre-trained CNN architectures on a custom attendance dataset, leveraging transfer learning to efficiently extract discriminative facial features instead of training models from scratch [13]. By doing so, it achieves high validation accuracies, often exceeding 90%, while maintaining reasonable training times, making it practical for real-world deployment in classrooms and offices [13]. The system captures live frames from a camera, extracts features using the CNNs, matches them against stored embeddings, and automatically logs time-stamped attendance entries, effectively replacing manual roll calls and spreadsheet updates [13].

Capabilities of System using Deep Transfer Learning

By leveraging robust pre-trained networks, this system generalizes well across variations in pose, expression, and illumination, maintaining high recognition rates under diverse conditions [13]. Transfer learning significantly reduces the amount of labelled data and the number of training epochs required compared to training a CNN from scratch, thus speeding up development and lowering annotation costs [13]. With efficient inference on modern GPUs or edge accelerators, the system supports real-time face detection and recognition, freeing staff from manual attendance tasks and improving operational efficiency.

Limitations of System using Deep Transfer Learning

Fine-tuning deep CNNs requires substantial GPU resources and memory, making inference on CPU-only or low-power devices slow and less practical for resource-constrained environments [14]. Additionally, the system's performance is highly dependent on the diversity of the training dataset; underrepresentation of certain demographic groups can lead to biased accuracy across different subpopulations. Without integrated liveness detection or anti-spoofing mechanisms, the system remains vulnerable to presentation attacks, such as the use of printed photos or replayed videos, which could enable fraudulent check-ins [15].

2.2.2 Face Recognition Based Attendance System Using Haar Cascade and LBPH

Overview of System using Haar Cascade and LBPH

This model uses OpenCV's Haar Cascade classifier for fast, sliding-window face detection in grayscale frames, making it ideal for classroom camera feeds. Detected face regions are then processed using Local Binary Pattern Histograms (LBPH), which encode local texture contrasts into compact histograms for face recognition [16]. A confidence threshold based on the Euclidean distance between histograms flags unknown faces (low similarity) and optionally saves their images for further review [17]. Matched identities are logged with timestamps into a backend database or spreadsheet, providing a secure and time-saving alternative to manual attendance tracking [18].

Capabilities of System using Haar Cascade and LBPH

The Haar Cascade detection process operates in real time, even on CPUs, by leveraging integral-image acceleration, making it highly effective for face localization in controlled indoor lighting conditions [16]. The LBPH encoding method is efficient, relying on simple integer comparisons and histogram updates. This makes it a fast, low-memory option for face recognition, with the added benefit of tolerating moderate variations in expression and lighting [17]. The system also incorporates a confidence threshold to reject low confidence matches,

thus reducing false positives. This ensures that unknown individuals are not mistakenly logged, which ultimately enhances the system's security and ensures the integrity of attendance data [17].

Limitations of System using Haar Cascade and LBPH

One limitation of the system is that LBPH's flat histogram structure does not allow for hierarchical feature extraction, resulting in lower accuracy compared to CNN-based methods, especially when dealing with large or complex datasets [18]. Additionally, the system's performance may be compromised under certain conditions, such as harsh shadows, side lighting, or partial occlusions (e.g., face masks), as these factors can distort the local texture patterns that the system relies on for recognition [19]. Finally, the system lacks built-in anti-spoofing or liveness detection features, leaving it vulnerable to attacks using static photos or video replays. Without these additional safeguards, the system is at risk of fraudulent check-ins.

2.2.3 Face Recognition Attendance System Using Local Binary Pattern

Overview of System using Local Binary Pattern

In this design, the Viola-Jones fast cascade detection method is used to scan each frame and locate face regions of interest (ROIs) before recognition [20]. The Local Binary Pattern (LBP) operator thresholds each pixel against its neighbours to generate an 8-bit code, and the per-block histograms are concatenated into a feature vector for each face [20]. Upon successful histogram matching above a set threshold, the system records the student's ID and timestamp into a database automatically, streamlining the attendance process [21].

Capabilities of System using Local Binary Pattern

The Viola-Jones detection and LBP feature extraction processes rely on simple arithmetic and integral images, allowing for real-time processing on standard PCs or embedded boards. LBP features are robust to moderate facial expressions and shifts in indoor lighting, maintaining high recognition rates even under typical classroom conditions. The system automates attendance logging, trimming per-student check-in times to just a few seconds and eliminating the proxy sign-ins and transcription errors often associated with manual roll calls [20] [21].

Limitations of System using Local Binary Pattern

LBP's reliance on local textures makes it sensitive to strong shadows, direct sunlight, and heavy occlusions, which can impair recognition in more challenging scenarios. Additionally, the system architecture often lacks scalability, meaning that supporting multiple classrooms or

large auditoriums may require additional cameras and compute nodes [22]. The absence of built-in liveness detection makes the system vulnerable to attacks involving printed images or replayed videos unless external anti-spoofing measures are implemented.

2.2.4 Implementation of Automated Attendance System Using Eigenfaces

Overview of System using Eigenfaces

The Eigenfaces method applies Principal Component Analysis (PCA) to decompose face images into a basis of "eigenvectors" (eigenfaces), which reduces dimensionality and enables faster comparison of faces [23]. When a new face image is captured, it is projected into this low-dimensional "face space," and nearest-neighbour matching on the eigenface coordinates is used to identify the person. If unrecognized faces are repeatedly detected, the system can trigger enrolment, allowing it to adaptively learn and recognize new students over time [24].

Capabilities of System using Eigenfaces

Eigenfaces significantly reduce computation and storage requirements by representing faces in a compact PCA subspace, which makes the system suitable for deployment on modest hardware [24]. With the ability to adapt, the system can incrementally enroll and recognize new faces without the need to retrain the entire model, offering greater flexibility. The system automates the timestamped logging of attendance, eliminating the need for manual intervention, and it integrates easily with SQL/NoSQL backends or spreadsheets for efficient record-keeping [23].

Limitations of System using Eigenfaces

Eigenfaces are highly sensitive to variations in lighting, pose, and expression because PCA captures global rather than localized features, which can lead to decreased accuracy under less-than-ideal conditions. Performance tends to degrade when applied to large, unaligned datasets or when illumination is challenging, often requiring strict capture protocols or preprocessing to ensure accuracy [25]. Additionally, standard Eigenface systems lack built-in anti-spoofing measures, leaving them vulnerable to attacks using printed images or replayed face data unless extra liveness checks are incorporated.

2.2.5 Face Recognition Based Attendance Management System Using VGG16

Overview of System using VGG16

This facial-recognition attendance system is built on the VGG16 architecture, which is extended with two output heads—one for classification (identifying the person) and another for regression, such as confidence scoring or bounding-box refinement [26] [27]. To enhance the system's robustness, the authors assembled a large and varied face dataset and applied extensive data augmentation techniques, including rotations, flips, and colour jitter, before fine-tuning the VGG16 layers and adding the dual output branches. The modified VGG16 model has been shown to outperform the standard version in precision, overall accuracy, and inference speed, making it particularly well-suited for educational settings [27]. This system automates lecture-hall attendance with minimal instructor intervention, enabling near-instant check-ins.

Capabilities of System using VGG16

The system leverages VGG16's deep feature hierarchy, which includes small 3×3 filters across 13 convolutional layers, achieving face-identification accuracies greater than 99% on both validation and test datasets for students. Using transfer learning, the system begins with ImageNet-pretrained weights, which require only moderate fine-tuning, thereby reducing the need for large amounts of data compared to training a model from scratch, while still retaining VGG16's powerful representation abilities [28]. Data augmentation techniques, such as flipped, rotated, and brightness-altered faces, help the model remain resilient to variations in lighting, pose, and facial expressions. The system also benefits from dual outputs: the classification head provides discrete student IDs, while the regression head outputs confidence scores or temporal smoothing parameters, allowing for the filtering of low-confidence matches and enabling seamless integration with broader analytics pipelines [27]. By automating the attendance process, the system reduces manual roll call time, capturing video frames, processing faces in batches, and updating logs in under a second per student. Additionally, attendance can be based on the average detection rate over fixed intervals, preventing quick proxy sign-ins while maintaining efficiency.

Limitations of System using VGG16

The VGG16 architecture comes with a heavy computational footprint, requiring significant GPU or CPU resources. Real-time inference on low-cost edge devices, such as Raspberry Pi, can result in unacceptable latency unless hardware acceleration is utilized [26]. Furthermore, deep CNNs like VGG16 require large datasets with thousands of labelled examples per identity, which can be challenging for smaller institutions to gather, especially if they require diverse datasets that reflect all student demographics [27]. The model also lacks built-in anti-spoofing mechanisms, such as texture-based or motion-based checks, making it vulnerable to attacks

using printed photos or replayed video without additional safeguards. Lastly, the large size of the model means frequent fine-tuning is necessary to adapt to new cohorts or changing environments, which can be time-consuming, and the model's size complicates version control and on-device updates, posing further maintenance challenges.

2.2.6 Summary of Existing Systems

The reviewed systems present different approaches to facial recognition-based attendance. The Deep Transfer Learning approach, using CNNs such as ResNet and VGG, achieves high accuracy above 90% and is robust to pose and lighting changes, but it requires GPU resources and lacks built-in anti-spoofing features. The Haar Cascade with LBPH method is efficient, running in real time on CPUs with low memory usage, and logs attendance in a database or spreadsheet. However, it offers lower accuracy, is affected by occlusions, and does not include anti-spoofing mechanisms. The Local Binary Pattern system, which uses Viola-Jones for detection, is fast and works well under moderate lighting variations, but it is sensitive to shadows, does not scale well for large deployments, and lacks anti-spoofing. The Eigenfaces method, based on PCA, is lightweight and allows adaptive enrollment of new faces, with records stored in SQL or spreadsheets. Its main limitation is its sensitivity to lighting and pose changes, and it also lacks anti-spoofing. Lastly, the VGG16-based system achieves very high accuracy above 99% with data augmentation and offers robust recognition performance, but it is GPU-heavy, requires large datasets, and has no built-in anti-spoofing, making maintenance and updates more challenging.

2.3 Previous Works on Face Recognition with Anti Spoofing

2.3.1 Classic (Handcrafted) Texture & Motion Methods

Early research in face anti-spoofing relied heavily on handcrafted features to identify spoofing attempts through texture and motion analysis. Texture-based approaches used techniques such as Local Binary Patterns (LBP), Histogram of Oriented Gradients (HOG), and wavelet transforms to capture fine-grained details in facial images. These features were effective in detecting common spoofing artifacts like moiré patterns, pixelation, and blurriness that typically occur in printed photos or screen replays. Motion-based methods complemented texture analysis by observing dynamic facial cues, such as optical flow patterns, eye blinks, and small head movements, which are harder to replicate using static images or videos. These

techniques offered the advantages of low computational cost and real-time processing, making them suitable for early applications. However, their performance often degraded under challenging conditions such as poor or variable lighting, high-resolution prints, or high-quality digital replay attacks. As a result, they are now regarded as the historical baseline for face anti-spoofing research and serve as a reference point against which modern deep learning-based solutions are evaluated [29].

2.3.2 Deep-Learning Appearance Models

The use of deep learning, especially Convolutional Neural Networks (CNNs), has significantly advanced face anti-spoofing research. CNNs allow automatic feature extraction from raw RGB images, removing the need for handcrafted descriptors. Early approaches treated the task as a binary classification problem, training CNNs to determine whether a face is live or spoofed [30]. Later, more sophisticated methods used multi-task learning, predicting additional information such as depth maps, spoof masks, or reflectance features to improve detection. Depth supervision is particularly effective in distinguishing real 3D facial structures from flat printed photos or screen replays. Some models use two-branch or patch-based networks that combine local texture details with global depth cues for better accuracy. These deep-learning models have achieved state-of-the-art results on benchmarks like CASIA-FASD, Replay-Attack, and OULU-NPU, detecting subtle spoofing artifacts missed by traditional methods. However, they often overfit to specific training conditions, such as camera type or lighting, which limits their performance on unseen datasets [31]. This remains a key challenge, encouraging further research in domain adaptation and robust feature learning.

2.3.3 Temporal & Physiological (Video) Cues

Video-based face anti-spoofing methods use temporal and physiological signals that single-frame analysis cannot capture. One common technique is remote Photoplethysmography (rPPG), which detects subtle skin color changes caused by blood flow. These periodic changes correspond to heartbeats and appear naturally in live faces but are missing or irregular in printed photos or replayed videos, making rPPG effective even against some 3D mask attacks when skin is visible [32] [33]. Researchers also analyze temporal textures and micro-movements, such as eye blinks, lip movements, and small head motions, which are hard to replicate in static or pre-recorded media. These dynamic patterns are typically processed using temporal deep

learning models like Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM) networks, or 3D Convolutional Neural Networks (3D CNNs). Studies show that combining appearance features with rPPG and micro-movement analysis improves robustness, especially against sophisticated attacks [33]. Modern benchmarks and competitions now include temporal and physiological cues in evaluation, highlighting their importance for reliable and generalizable anti-spoofing systems.

2.3.4 Depth/Sensor-Based and Multi-Modal Methods

Depth- and sensor-based face anti-spoofing methods improve security by using additional data beyond standard RGB images, making simple replica attacks more difficult. Depth can be captured using depth cameras or estimated from a single RGB image, allowing the system to distinguish flat surfaces, such as printed photos or screens, from real 3D faces. Infrared (IR) and multispectral imaging further enhance detection by capturing subtle skin reflectance differences that synthetic materials do not have, adding another layer of verification. Multi-modal systems that combine RGB, depth, IR, and physiological signals like rPPG achieve the highest accuracy and robustness against various spoofing attacks [34]. However, these methods require more advanced hardware, increase deployment complexity, and may reduce accessibility, limiting their use in low-cost or large-scale setups. Overall, depth- and multi-modal approaches provide strong protection in high-security environments and complement the strengths of appearance- and temporal-based methods.

2.3.5 Domain Generalization & Adaptation

A major challenge in face anti-spoofing is cross-domain generalization. Models trained on one dataset often perform poorly on other datasets because of differences in cameras, lighting, capture distances, and spoofing media. Domain generalization (DG) techniques aim to solve this by learning features that remain consistent across domains. Common approaches include data augmentation, adversarial learning, and training objectives that make the model focus on intrinsic facial cues instead of dataset-specific artifacts [35]. Unsupervised domain adaptation (UDA) further improves performance by using unlabelled data from the target domain to align feature distributions or by generating synthetic samples that resemble the target data, reducing domain gaps. Recent studies presented at conferences such as ECCV and CVPR use generative and content-aware methods to improve zero-shot performance on unseen cameras and lighting

conditions [36]. Despite these advances, robust cross-dataset generalization remains a challenge, and research continues to find ways to make anti-spoofing systems reliable in diverse real-world environments.

2.3.6 Summary of Anti Spoofing Methods

Face anti-spoofing research has progressed from early handcrafted methods to advanced deep learning and multi-modal approaches. Traditional texture- and motion-based techniques, such as LBP, HOG, and optical flow, offered simple, low-cost ways to detect common spoofing artifacts but often failed under difficult conditions, such as high-quality prints or changing lighting. Deep-learning appearance models, especially CNN-based methods, improved accuracy by automatically learning discriminative features and including auxiliary tasks like depth and spoof mask prediction. Video-based methods increased robustness by using temporal and physiological cues, such as micro-movements and rPPG signals, which are hard to reproduce in static or replayed media. Depth- and sensor-based methods, including multi-modal systems, added extra security by using depth, infrared, and multispectral information, achieving high performance but requiring more complex hardware. Domain generalization and adaptation techniques address the ongoing challenge of cross-dataset performance, helping models remain reliable under varied real-world conditions. Overall, these studies show a trend toward combining multiple cues (appearance, temporal, physiological, and sensor-based) to create face anti-spoofing systems that are more accurate, robust, and generalizable, while also highlighting the challenges of practical deployment.

CHAPTER 3 SYSTEM METHODOLOGY

This chapter outlines the methodology and approach for developing the automated attendance system with anti-spoofing face recognition, a development-based project designed for educational environments. It describes the system design through diagrams, detailing the architecture and data flow processes. The following sections present the system design diagram, including the system architecture and data flow.

3.1 System Design Diagram

The facial recognition-based attendance system is designed to automate student attendance tracking using face recognition and anti-spoofing techniques. The system integrates a web interface for user interaction, a real-time face recognition module, and a Google Sheets backend for data storage, with email notifications for absentee reporting.

3.1.1 System Architecture Diagram

The facial recognition-based attendance system is designed as a modular client-server application, integrating web-based user interaction, real-time face recognition, and cloud-based data storage. The architecture ensures scalability, maintainability, and security for managing student attendance with anti-spoofing measures. The system comprises five primary components: the Client-Side Interface, Flask Server, Face Recognition Module, Google Sheets Backend, and Email Notification System. These components interact seamlessly to facilitate student registration, subject management, attendance tracking, and absentee reporting.

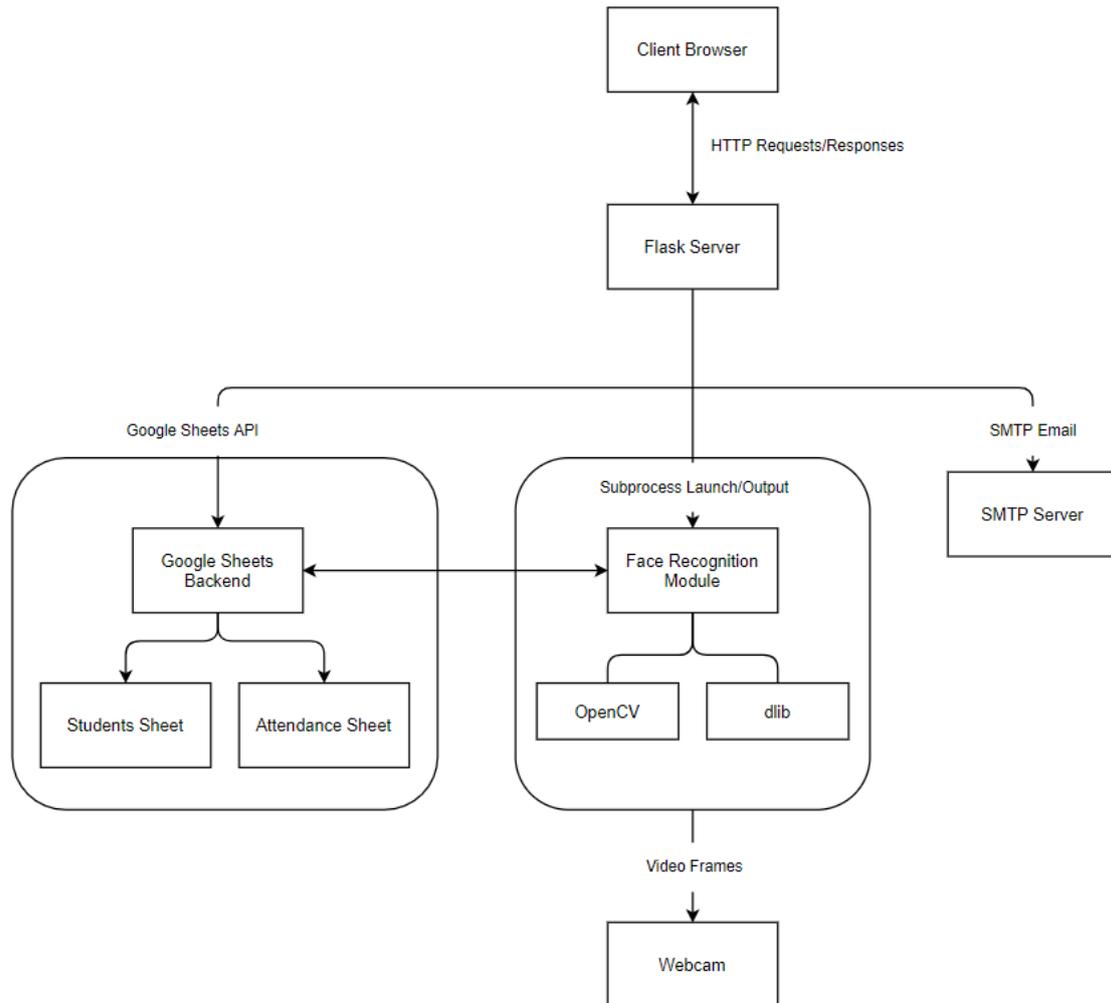


Figure 3.1.1.1 System Architecture Diagram

Components of the System Architecture

1. Client-Side Interface:

- **Purpose:** Provides a user-friendly web interface for students and teachers to interact with the system.
- **Implementation:** The interface consists of three static HTML pages (index.html, register.html, teacherpanel.html) served by the Flask Server. These pages are accessible via a web browser at <http://127.0.0.1:5000>.
 - index.html: The landing page, providing an entry point to the system.
 - register.html: Allows students to submit their name, ID, and three facial images for registration.

- `teacherpanel.html`: Enables teachers to add subjects, initiate attendance sessions, and access the Google Sheets database.
- **Technologies**: HTML, CSS, JavaScript, served via Flask's static file handling.
- **Interaction**: Communicates with the Flask Server via HTTP requests (GET for page serving, POST for form submissions). For example, students submit registration data to the `/submit` endpoint, and teachers initiate attendance via the `/start-backend` endpoint.

2. Flask Server (main.py):

- **Purpose**: Acts as the central hub, handling HTTP requests, coordinating data storage, and managing the face recognition process.
- **Implementation**: A Python-based web server built using Flask, a lightweight WSGI framework. It exposes multiple endpoints:
 - `/ (GET)`: Serves `index.html`.
 - `/register (GET)`: Serves `register.html` for student registration.
 - `/teacherpanel.html (GET)`: Serves `teacherpanel.html` for teacher controls.
 - `/open-sheet (GET)`: Redirects to the Google Sheet URL for direct data access.
 - `/submit (POST)`: Processes student registration data, generates face encodings, and stores them in Google Sheets.
 - `/get-subjects (GET)`: Retrieves the list of subjects from the "Students" sheet.
 - `/add-subject (POST)`: Adds a new subject as a column in the "Students" sheet.
 - `/enroll-students (POST)`: Enrolls students for a subject by initializing their attendance counts.
 - `/start-backend (POST)`: Launches `recognition.py` as a subprocess to start attendance tracking.
- **Technologies**: Flask, Flask-CORS, gspread (for Google Sheets API), `face_recognition` (for encoding generation), `smtplib` (for email notifications).
- **Interaction**: Communicates with the Client-Side Interface via HTTP, with Google Sheets via the Google Sheets API (using service account credentials in `credentials.json`), and with the Face Recognition Module via a subprocess. It also sends emails via an SMTP server.

- **Details:** The server verifies the presence of required files (credentials.json, recognition.py, static HTML files) at startup and automatically opens a browser to the home page. It uses environment variables (loaded via python-dotenv) for email configuration, ensuring secure credential management.

3. Face Recognition Module (recognition.py):

- **Purpose:** Performs real-time face detection, recognition, and liveness verification to log attendance.
- **Implementation:** A standalone Python script that runs as a subprocess launched by the Flask Server. It uses a webcam to capture video frames, processes them for face detection and recognition, and verifies liveness through a sequence of actions (smile, open mouth, blink both eyes).
 - **Face Detection:** Uses face_recognition for initial face detection and encoding generation, and dlib for facial landmark detection.
 - **Liveness Verification:** Requires users to perform two random actions within 45 seconds, using metrics like Eye Aspect Ratio (EAR), Mouth Aspect Ratio (MAR), and mouth corner angle.
 - **Attendance Logging:** Matches detected faces against stored encodings, logs attendance in the "Attendance" sheet, and increments subject counts in the "Students" sheet.
- **Technologies:** OpenCV (cv2) for video capture, face_recognition for face encoding, dlib with the "shape_predictor_68_face_landmarks.dat" model for landmark detection, gspread for Google Sheets access.
- **Interaction:** Reads student data (name, ID, encodings) from the "Students" sheet, updates attendance records, and communicates indirectly with the Flask Server via Google Sheets. It displays real-time feedback on the video feed using OpenCV.
- **Details:** The module supports multiple camera indices for compatibility and uses a tolerance of 0.4 for face matching and a confidence threshold of 0.6. It implements anti-spoofing to prevent fraudulent attendance using photos or videos.

4. Google Sheets Backend:

- **Purpose:** Serves as the central database for storing student profiles, attendance records, and subject information.

- **Implementation:** Two worksheets in a single Google Sheet
 - **Students Sheet:** Stores student data with columns for Name, ID, Encodings (JSON array of 128D vectors), Active status ("Yes"/"No"), and subject-specific attendance counts (columns E onward)
 - **Attendance Sheet:** Logs attendance with columns for Timestamp, Date, Time, Student ID, Student Name, Subject, Status ("Present"), Confidence, and Notes.
- **Technologies:** Google Sheets API via gspread, authenticated using a service account (credentials.json).
- **Interaction:** Accessed by both the Flask Server and Face Recognition Module for reading and writing data. The Flask Server initializes headers and manages subjects, while the Face Recognition Module logs attendance and updates counts.
- **Details:** The use of Google Sheets provides a cloud-based, accessible solution for small-scale deployments but may face rate limits for frequent updates. Data is stored in plain text, with face encodings as JSON strings, requiring careful access control.

5. Email Notification System:

- **Purpose:** Notifies teachers of absent students after an attendance session.
- **Implementation:** Integrated into the Flask Server, triggered after face recognition module completes. It compares pre- and post-session attendance counts to identify absentees and sends an email via SMTP.
- **Technologies:** smtplib for email sending, python-dotenv for loading SMTP credentials (server, port, sender email, password, teacher email).
- **Interaction:** Receives absentee data from the Flask Server's `get_absent_students` function and sends emails to the teacher's email address (configured in `.env`).
- **Details:** Emails include the subject, date, and a list of absent students (name and ID). The system uses Gmail's SMTP server by default with TLS encryption.

3.1.2 Data Flow Diagram

The Data Flow Diagram (DFD) illustrates the flow of data through the facial recognition-based attendance system, detailing how inputs from users and external devices are processed, stored, and transformed into outputs such as attendance records and absentee notifications.

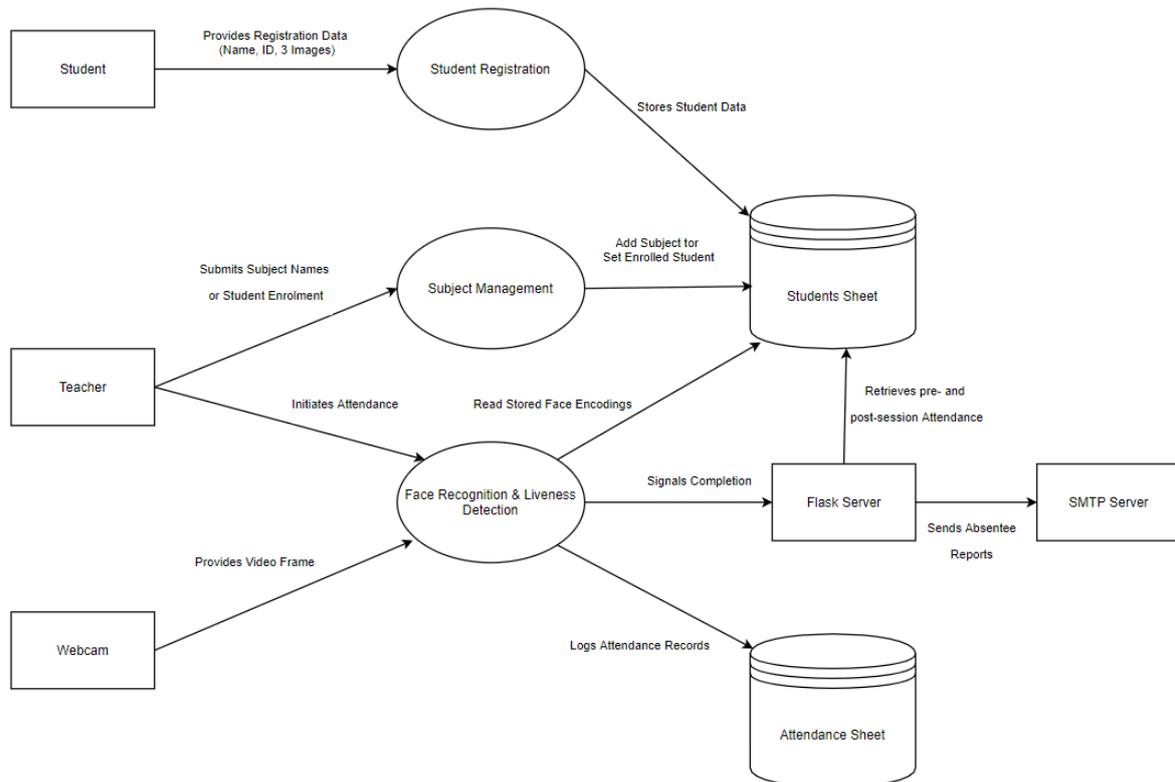


Figure 3.1.2.1 Data Flow Diagram

Figure 3.1.2.1 shows the Data Flow Diagram (DFD) of the attendance system, illustrating the interactions between external entities, processes, data stores, and the flow of information. The system involves five main external entities: students, teachers, the webcam, the Flask server, and the SMTP server.

Students provide registration data, including name, ID, and facial images. The Student Registration module processes this data to generate 128D face encodings, which are stored in the Students Sheet. Teachers use the Subject Management module to add subjects, which updates the Students Sheet with new columns or initializes attendance counts.

During attendance sessions, teachers trigger the Face Recognition and Liveness Detection process through the Flask server. The server captures video frames from the webcam, verifies

liveness with actions such as smiling or blinking, and matches detected faces with stored encodings. Recognized students' attendance is recorded in the Attendance Sheet, and subject counts in the Students Sheet are updated.

After each session, the Absentee Reporting process compares attendance counts before and after the session to identify absent students and sends notifications via the SMTP server. In the diagram, circles represent processes, rectangles represent entities, and arrows indicate the flow of information.

3.1.3 Face Recognition and Liveness Detection Equations

The facial recognition-based attendance system uses the `face_recognition` and `dlib` libraries to perform face recognition and liveness detection, applying advanced computer vision algorithms and mathematical models. Face recognition identifies students by comparing facial encodings, while liveness detection verifies the presence of a live person through dynamic facial actions, such as smiling, opening the mouth, or blinking both eyes. This section explains the algorithms, models, and equations used, focusing on Histogram of Oriented Gradients (HOG) for face detection, ResNet-based face encodings, and landmark-based metrics for liveness verification.

3.1.3.1 Face Recognition

Algorithm and Model

The system leverages the `face_recognition` library, which internally uses `dlib`'s face recognition framework. The workflow involves two main stages: face detection and face recognition (encoding & comparison).

1. Face Detection

The system uses the Histogram of Oriented Gradients (HOG) combined with a linear Support Vector Machine (SVM) for face detection. The HOG descriptor works by dividing an image into small connected regions called cells, calculating the gradient orientation of pixels within each cell, and normalizing these histograms over larger blocks. This produces a feature vector that effectively represents local facial structures. The linear SVM then classifies regions of the image as face or non-face.

In my code, HOG is implemented via:

```
def face_locations(img, number_of_times_to_upsample=1, model="hog"):
    """
    Returns an array of bounding boxes of human faces in a image

    :param img: An image (as a numpy array)
    :param number_of_times_to_upsample: How many times to upsample the image looking for faces. Higher numbers find smaller faces.
    :param model: Which face detection model to use. "hog" is less accurate but faster on CPUs. "cnn" is a more accurate
    |             | deep-learning model which is GPU/CUDA accelerated (if available). The default is "hog".
    :return: A list of tuples of found face locations in css (top, right, bottom, left) order
    """
```

Here, `face_locations` detects face bounding boxes in (top, right, bottom, left) format. For each video frame captured by the webcam, the HOG detector quickly identifies candidate face regions.

2. Face Landmark Detection

After detecting faces, the system extracts facial landmarks using `dlib`'s shape predictor. Two models are available:

- **68-point predictor** (`pose_predictor_68_point`): detailed landmarks for eyes, eyebrows, nose, mouth, and chin.
- **5-point predictor** (`pose_predictor_5_point`): faster but coarser landmarks.

Landmarks are used by the recognition model to align the face before encoding, ensuring consistent orientation and scaling.

3. Face Recognition / Encoding

For face recognition, the system uses `dlib`'s deep convolutional neural network, which is based on the ResNet-34 architecture. This network has been pre-trained on a large dataset of faces, allowing it to extract high-level, identity-preserving features from any input face image. When a face image is processed, the network outputs a 128-dimensional (128D) encoding, which is a compact numerical vector that uniquely represents the facial features of that person. These encodings serve as a “fingerprint” for the face, capturing distinctive characteristics while being robust to variations in lighting, pose, and expression.

In the system, the `face_encodings` function is used to generate these 128D vectors. During student registration, multiple encodings per student can be created (from different images of the same student) to improve recognition accuracy. These encodings are then stored in the “Students” Google Sheet as JSON arrays, enabling the system to match real-time captured faces against stored encodings during attendance.

4. Face Comparison and Similarity

The similarity between a detected face encoding e_d (from a video frame) and a stored encoding e_s (from the Students Sheet) is calculated using the Euclidean distance:

$$\text{Distance}(e_d, e_s) = \sqrt{\sum_{i=1}^{128} (e_{d,i} - e_{s,i})^2}$$

where $e_{d,i}$ and $e_{s,i}$ are the i -th components of the 128-dimensional vectors.

A face is considered a match if the distance is below the tolerance threshold (0.4):

$$\text{Match} = \begin{cases} \text{True,} & \text{if Distance}(e_d, e_s) < 0.4 \\ \text{False,} & \text{otherwise} \end{cases}$$

To quantify the reliability of a match, a **confidence score** is computed:

$$\text{Confidence} = 1 - \text{Distance}(e_d, e_s)$$

Only faces with confidence ≥ 0.6 are logged as "Present," reducing false positives and improving identification accuracy.

3.1.3.2 Liveness Detection Metrics

Liveness detection ensures that the system cannot be fooled by static images or pre-recorded videos, requiring the user to perform two random facial actions such as blinking, opening the mouth, or smiling within a 45-second timeout. This process guarantees that a real, live person is present in front of the camera.

Metrics for Liveness Verification:

1. Eye Aspect Ratio (EAR):

EAR measures eye openness to detect blinks. For each eye, six landmarks are used.

$$\text{EAR} = \frac{|p_2 - p_6| + |p_3 - p_5|}{2 \cdot |p_1 - p_4|}$$

P_1 to P_6 are eye landmarks.

A blink is detected if:

$$\text{EAR} < \text{Baseline EAR} \times 0.85$$

The baseline EAR is computed as the median of the first 1.5 seconds of video frames to stabilize measurements, implemented in `eye_aspect_ratio()`.

2. Mouth Aspect Ratio (MAR):

MAR captures mouth shape changes for smiling and opening actions, using mouth landmarks 48–68:

$$\text{MAR} = \frac{\frac{|p_{51} - p_{57}| + |p_{52} - p_{56}| + |p_{53} - p_{55}|}{3}}{|p_{48} - p_{54}|}$$

Detection criteria:

- A smile is detected if:

$$\text{MAR} - \text{Baseline MAR} > 0.05 \quad \text{or} \quad \text{Mouth Width} > \text{Baseline Width} \times 1.$$

- An open mouth is detected if:

$$\text{Mouth Height} > \text{Baseline Mouth Height} \times 1.3$$

Mouth height is calculated as the average vertical distance between the top and bottom lip landmarks, while mouth width is measured as the horizontal distance between points 48 and 54.

3. Mouth Corner Angle:

Used to verify smiles more accurately by measuring the angles of mouth corners:

$$\text{Angle}_{\text{left}} = \tan^{-1} \left(\frac{y_{48} - y_{51}}{x_{48} - x_{51}} \right), \quad \text{Angle}_{\text{right}} = \tan^{-1} \left(\frac{y_{54} - y_{51}}{x_{54} - x_{51}} \right)$$

Average angle is then calculated as:

$$\text{Average Angle} = \frac{\text{Angle}_{\text{left}} + \text{Angle}_{\text{right}}}{2}$$

A smile is confirmed if:

$$\text{Average Angle} > \text{Baseline Angle} + 5^\circ$$

CHAPTER 4 SYSTEM DESIGN

This chapter provides a description of the system design for the facial recognition-based attendance system. The system is primarily software-based, developed in Python, with integration of computer vision libraries for face recognition and a web interface for user interaction. It uses a webcam for input, Google Sheets as the database, and email for notifications. No custom hardware beyond a standard computer with a webcam is required.

4.1 System Block Diagram

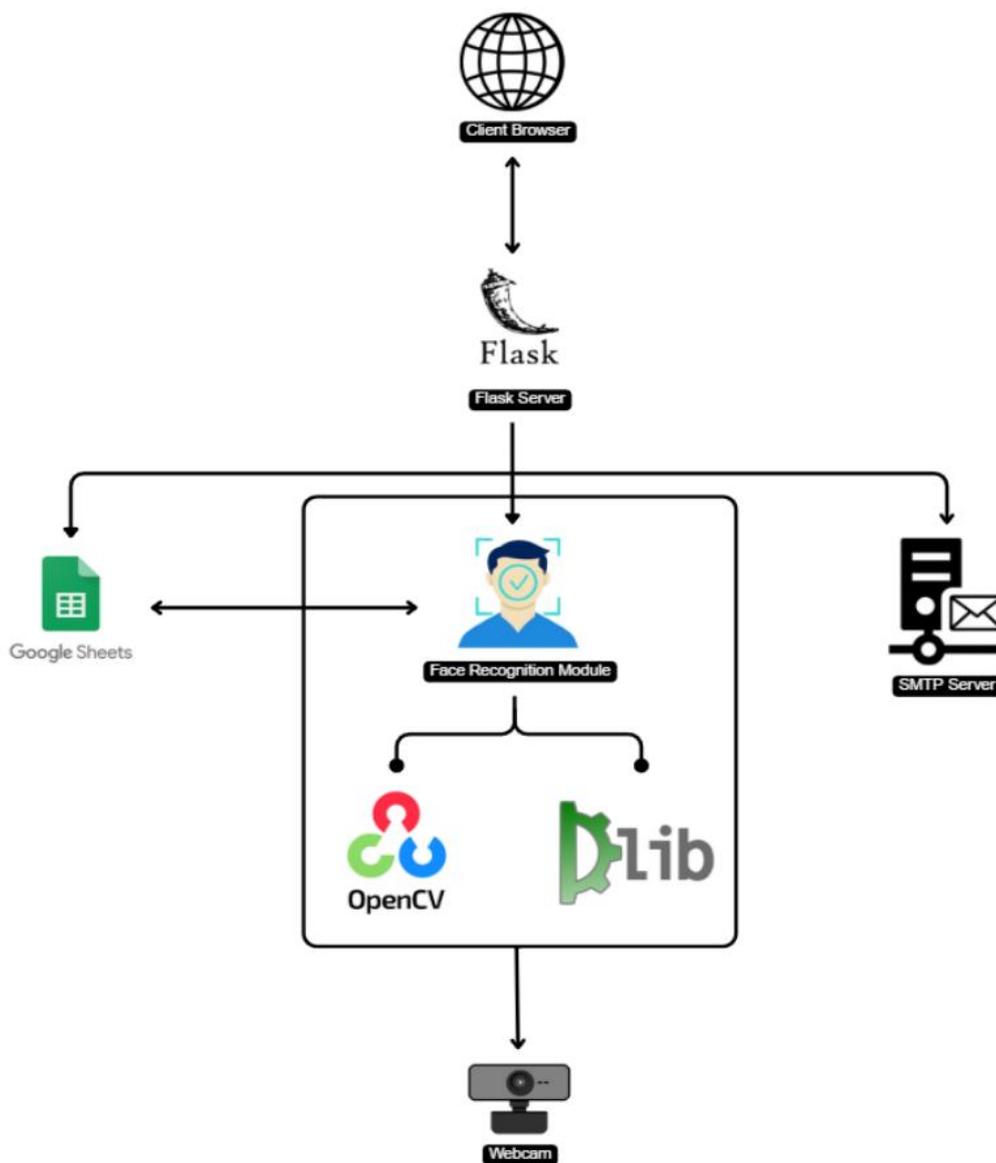


Figure 4.1.0.1 System Block Diagram

The system is organized into several interconnected blocks, each handling a key part of the attendance process.

The Client Browser provides the user interface with HTML pages (index.html, register.html, teacherpanel.html). Students register by submitting their name, ID, and images, while teachers manage subjects and start attendance sessions. HTTP GET requests retrieve pages, and POST requests submit data.

The Flask Server (main.py) acts as the backend, handling HTTP requests, processing registration data, managing subjects, triggering the face recognition module, and sending absentee emails. It uses gspread to access Google Sheets, face_recognition for encoding generation, and smtplib for email delivery. The face recognition module runs as a subprocess when attendance begins.

The Face Recognition Module (recognition.py) captures live video, detects faces using the HOG algorithm, generates 128D encodings with a ResNet-34 backbone, performs liveness verification using eye aspect ratio (EAR), mouth aspect ratio (MAR), and mouth angle, and matches detected faces with stored encodings before logging attendance to Google Sheets.

The Google Sheets Backend acts as a cloud database, storing student profiles (name, ID, encodings, status, subject counts) in the “Students” sheet and logging attendance records (timestamp, student details, status, confidence) in the “Attendance” sheet.

Finally, the SMTP Server sends absentee notifications via email, listing absent students’ names and IDs, using credentials stored in the .env file and TLS encryption for secure delivery.

4.2 System Flowchart

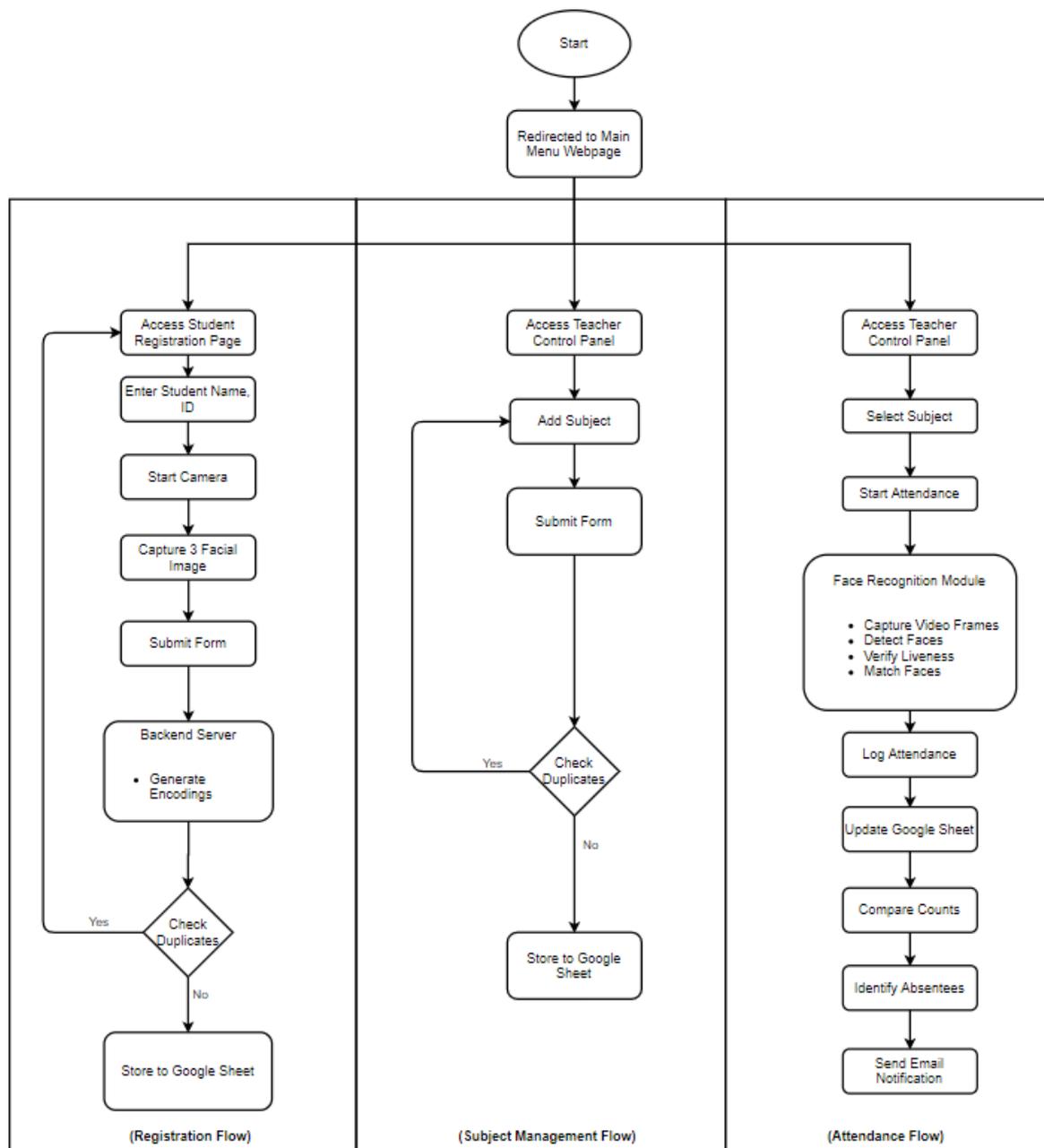


Figure 4.2.0.1 System Flowchart

The system offers a complete set of features for automated attendance management. Student Registration allows students to submit their name, ID, and three facial images through the web interface (register.html). The system then generates 128-dimensional face encodings and stores them in the “Students” Google Sheet.

Teacher Actions are handled via teacherpanel.html, allowing teachers to add subjects or start attendance sessions. These actions either update the Google Sheets database or trigger the face recognition module.

During Face Recognition, the system captures live webcam video, detects faces using the HOG algorithm, verifies liveness through actions such as smiling, opening the mouth, or blinking, and matches detected faces with stored encodings to log attendance automatically.

For Absentee Reporting, the system compares attendance counts before and after each session to identify absent students and sends email notifications to the teacher, ensuring accurate and efficient attendance tracking.

4.3 Database Schema

The system uses Google Sheets as its database, with two worksheets:

4.3.1 Students Sheet

Column	Name	Type	Description
A	Name	String	Full name of the student (Wilson Chua).
B	ID	String	Unique student identifier (21ACB02648).
C	Encodings	JSON Array	Three 128D face encodings stored as a JSON array ([[...], [...], [...]]).
D	Active	String	Indicates if the student is active (Yes/No).
E →	Subject Counts	Integer	Dynamic columns, one per subject, tracking attendance counts

Table 4.3.1.1 Student Sheet

4.3.2 Attendance Sheet

Column	Name	Type	Description
A	Timestamp	String	Full timestamp of attendance event (2025-09-19 14:30:00).
B	Date	String	Date of attendance (2025-09-19).
C	Time	String	Time of attendance (14:30:00).
D	Student ID	String	References Students Sheet → ID .
E	Student Name	String	Full name of the student.
F	Subject	String	Subject for which attendance was taken (Mathematics).
G	Status	String	Attendance status (Present).
H	Confidence	Float	Face recognition confidence score (0.85).
I	Notes	String	Optional field for remarks or special notes.

Table 4.3.2.1 Attendance Sheet

4.4 Activity Diagram

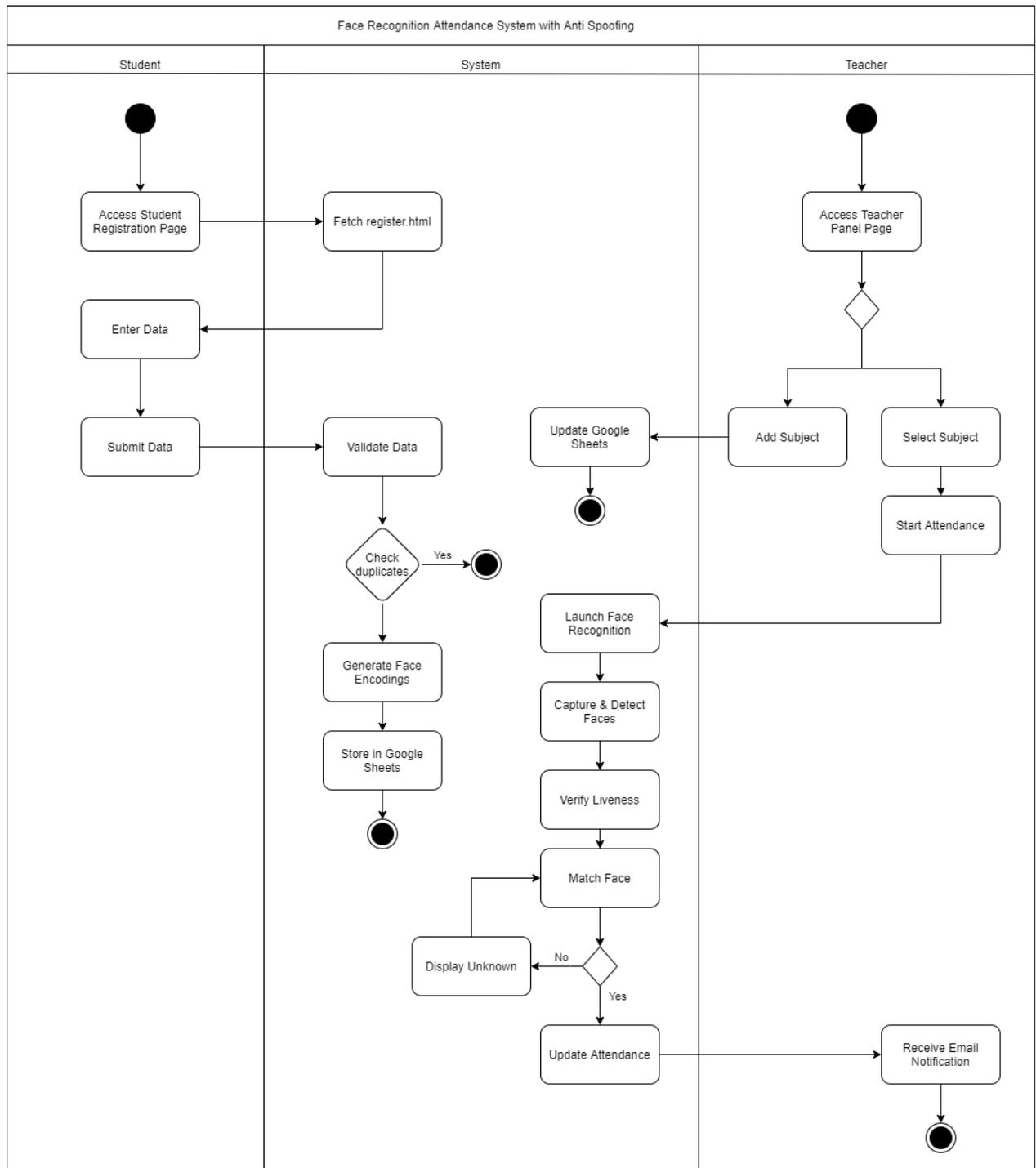


Figure 4.4.0.1 Activity Diagram

The activity diagram shows the full workflow of the facial recognition-based attendance system, including student registration, teacher actions, face recognition, and absentee notification.

In the Student lane, users access the registration page, submit their personal details and facial images, and undergo a check for duplicate IDs or encodings. If the data is valid, the system

generates facial encodings, stores the information in Google Sheets, and completes the registration; if not, an error message prompts the user to resubmit.

In the Teacher lane, teachers access the panel to add subjects or start attendance. When attendance begins, the system launches the face recognition subprocess, which captures live video, detects faces using a HOG-based detector, and verifies liveness through actions such as blinking or smiling. Verified faces are matched with stored encodings, and attendance is recorded for successful matches.

After the face recognition process, the system generates an absentee report by comparing subject counts and sends email notifications to the teacher. The workflow ends once all activities are completed.

4.5 Use Case Diagram

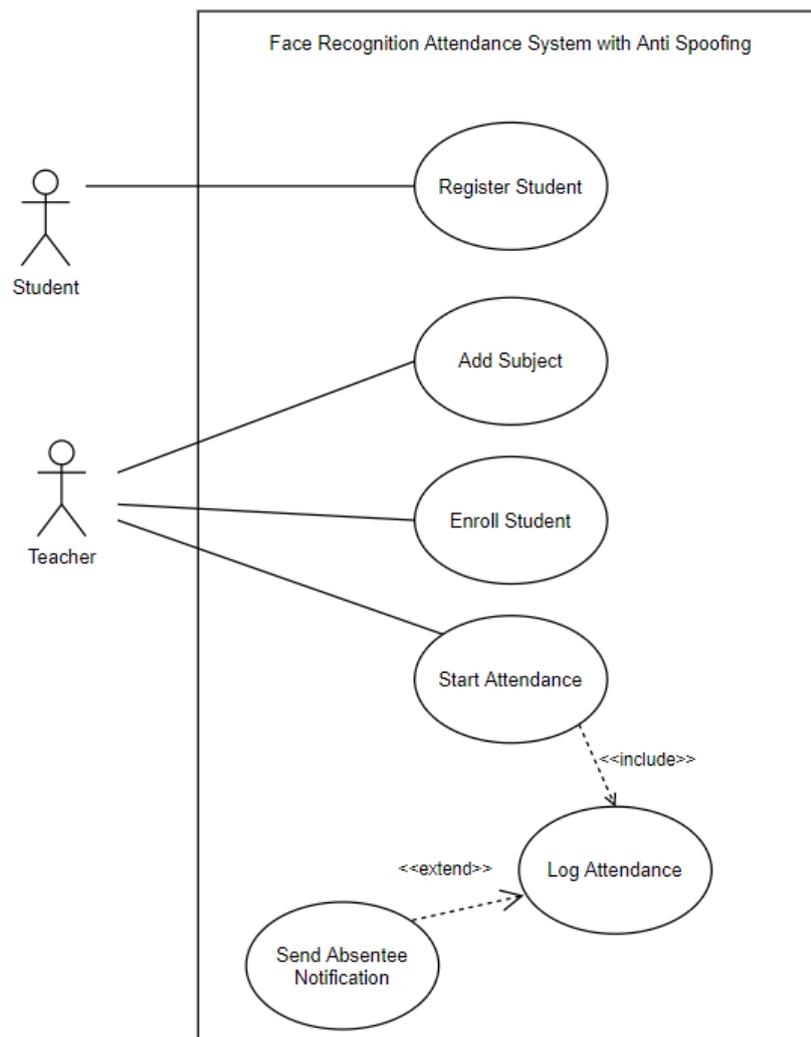


Figure 4.5.0.1 Use Case Diagram

The use case diagram shows the main functionalities of the attendance system and the interactions between its two main actors, Student and Teacher.

Students interact with the system by registering through the registration page, where they submit their name, ID, and facial images, which are validated and stored in Google Sheets. Teachers interact with the system through the teacher panel, where they can add new subjects or start an attendance session. Starting attendance triggers the face recognition module, which performs real-time face detection, liveness verification, and encoding matching before logging attendance. This logging process is represented as an included use case within Start Attendance. After attendance is logged, the process may extend to sending absentee notifications, where the system identifies absent students and emails the report to the teacher.

CHAPTER 5 SYSTEM IMPLEMENTATION

This chapter describes the implementation of the facial recognition-based attendance system, covering the hardware and software setup, configuration steps, operational details, challenges encountered, and a concluding remark.

5.1 Hardware Setup

The system requires minimal hardware to operate effectively, leveraging standard computing equipment suitable for educational environments.

Component	Specifications	Purpose
Computer	<ul style="list-style-type: none"> Laptop/Desktop with at least 8 GB RAM, quad-core CPU (Intel i5 or equivalent), and 5 GB free storage OS: Windows 10 or Linux (Ubuntu recommended) 	<ul style="list-style-type: none"> Runs the Flask server (main.py), face recognition module (recognition.py), and handles Google Sheets API calls.
Webcam	<ul style="list-style-type: none"> USB or built-in webcam. Minimum 720p resolution, 30 FPS (configured at 640×480 for performance) 	<ul style="list-style-type: none"> Captures video frames for face detection and liveness verification
Internet Connection	<ul style="list-style-type: none"> Stable connection with minimum 1 Mbps speed 	<ul style="list-style-type: none"> Allows Google Sheets API communication Sends absentee email notifications
Optional Hardware	<ul style="list-style-type: none"> External Monitor: For displaying web interface or video feed 	<ul style="list-style-type: none"> Provides better user experience for large classrooms

Table 5.1.0.1 Hardware Setup

Setup Instructions:

1. Connect the webcam to the computer via USB or ensure the built-in camera is functional.
2. Verify the computer meets the minimum specifications and has an active internet connection.

5.2 Software Setup

Component	Specifications / Tools	Purpose
Python Environment	<ul style="list-style-type: none"> Python 3.11 (set as interpreter in VS Code) 	<ul style="list-style-type: none"> Executes main.py (Flask server) and recognition.py (face recognition module).
Required Libraries	<ul style="list-style-type: none"> Install via: pip install flask flask-cors gspread google-auth face_recognition pillow numpy opencv-python dlib scipy python-dotenv 	<ul style="list-style-type: none"> Provides backend, API access, face recognition, video capture, numerical computation, and email configuration
Model File	<ul style="list-style-type: none"> shape_predictor_68_face_landmarks.dat 	<ul style="list-style-type: none"> Enables facial landmark detection in recognition.py (eye, nose, mouth points)
Web Browser	<ul style="list-style-type: none"> Chrome, Firefox, or Edge (latest version) 	<ul style="list-style-type: none"> Accesses web interface at http://127.0.0.1:5000

Table 5.2.0.1 Software Setup

5.2.1 Explanation of Each Libraries

Library	Version	Description
Flask	3.1.0	Web framework used to create the API and run the Flask server (main.py).
flask-cors	5.0.1	Enables Cross-Origin Resource Sharing (CORS) to allow browser-based requests.
gspread	6.1.4	Provides access to Google Sheets for storing and retrieving data.
google-auth	2.38.0	Handles Google API authentication (service account credentials).
face_recognition	1.3.0	High-level face recognition library (built on dlib) for detection and encoding.
pillow	11.0.0	Image processing library (used for uploaded student images).

numpy	1.26.4	Numerical computing library for handling arrays and encoding data.
opencv-python	4.10.0.84	OpenCV library (cv2) for capturing video frames and displaying output.
dlib	19.24.1	Library for machine learning and face recognition utilities.
scipy	1.15.3	Scientific computing library (used for distance calculation in face matching).
python-dotenv	1.1.1	Loads environment variables from .env file (email credentials, API keys).

Table 5.2.1.1 Libraries Explanation

5.3 Settings and Configuration

The system requires proper configuration of Google Sheets API credentials, SMTP settings, file placements, and camera setup to ensure smooth operation.

5.3.1 Google Sheets API

1. Create a Google Cloud project, enable the Google Sheets API, and download a service account key as `credentials.json`.
2. Store `credentials.json` in the project directory.
3. Update `SPREADSHEET_ID` in `main.py` and `recognition.py` to match your Google Sheet ID.
4. Use the functions `initialize_students_sheet()` and `initialize_attendance_sheet()` in `main.py` to automatically create and format the following worksheets:
 - **Students Sheet:** Columns → *Name, ID, Encodings, Active*
 - **Attendance Sheet:** Columns → *Timestamp, Date, Time, Status, Student ID, Student Name, Subject, Status, Confidence, Notes.*

5.3.2 SMTP Configuration

1. Create a `.env` file in the project root with the following contents:
 - `SMTP_SERVER=smtp.gmail.com`
 - `SMTP_PORT=587`
 - `SENDER_EMAIL=your_email@gmail.com`

- SENDER_PASSWORD=your_app_password (generate it from your Google Account → Security → App Passwords)
- TEACHER_EMAIL=teacher@example.com

5.3.3 File Structure

Folder/File	Description
main.py	Main Flask server script
recognition.py	Face recognition script
credentials.json	Google Sheets API credentials
.env	Environment variables (SMTP config)
shape_predictor_68_face_landmarks.dat	Dlib landmark model file
static/index.html	Homepage UI
static/register.html	Student registration page
static/teacherpanel.html	Teacher panel page

Table 5.3.3.1 File Structure

5.4 System Operation

The system operates through a browser-based interface combined with real-time face recognition. The following steps describe how to use the system:

1. Run the Flask Server

Execute the following command in the terminal:

```
python main.py
```

The Flask server will start at <http://127.0.0.1:5000> and automatically open a browser window displaying the Home Page (index.html).

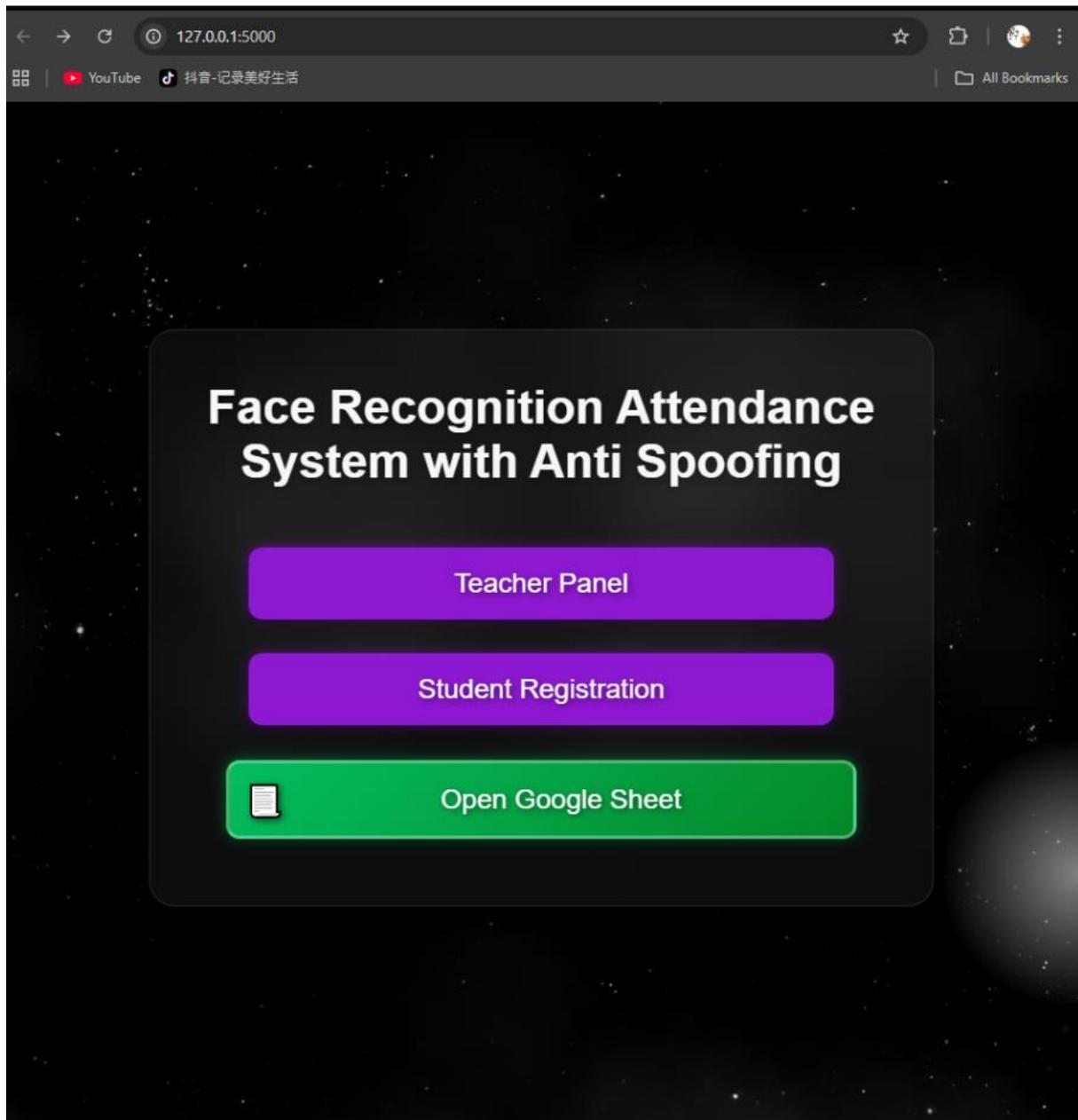


Figure 5.4.0.1 Homepage (index.html)

2. Student Registration

Navigate to register.html by clicking the "Student Registration" button on the Home Page.

- Enter the student's **Name** and **ID**.
- Capture **three facial images**.
- Click **Submit** to complete the registration.

The system validates inputs, encodes the facial data, and stores the information in Google Sheets under the "Students" worksheet.

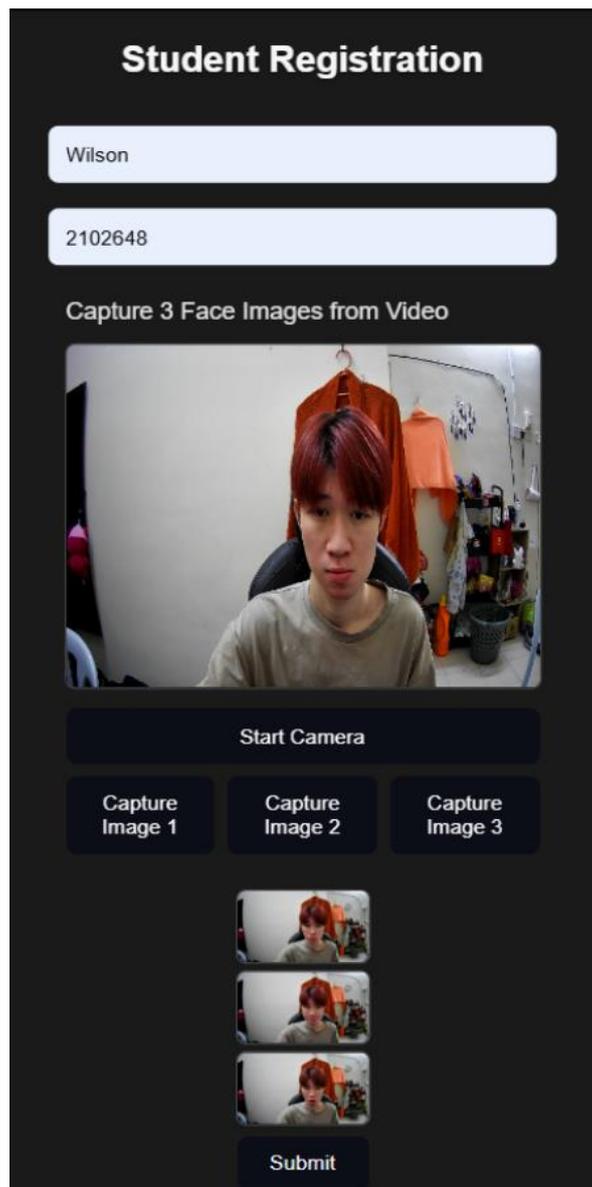


Figure 5.4.0.2 Student Registration (*register.html*)

3. Teacher Actions

Navigate to `teacherpanel.html` by clicking the "Teacher Panel" button.

From this page, the teacher can:

- **Add a Subject** – Enter the subject name and click *Add Subject*.

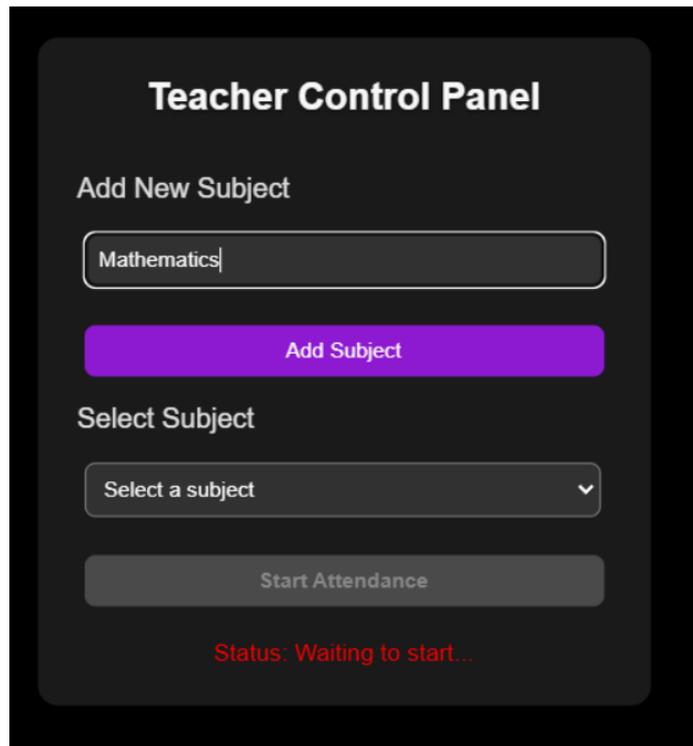


Figure 5.4.0.3 Add Subject (teacherpanel.html)

- **Start Attendance** – Select a subject and click *Start Attendance* to launch recognition.

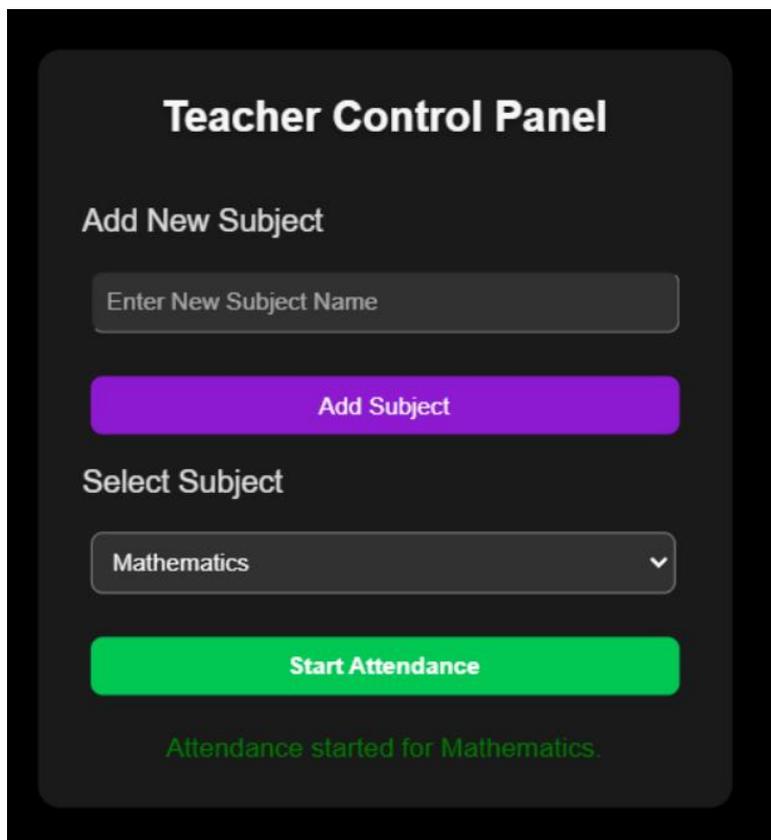


Figure 5.4.0.4 Start Attendance (teacherpanel.html)

4. Attendance Tracking

When **Start Attendance** is clicked, recognition.py is launched:

- The webcam activates, showing a real-time video feed.
- The system detects a face and prompts students to perform random liveness actions (smile, blink, open mouth).

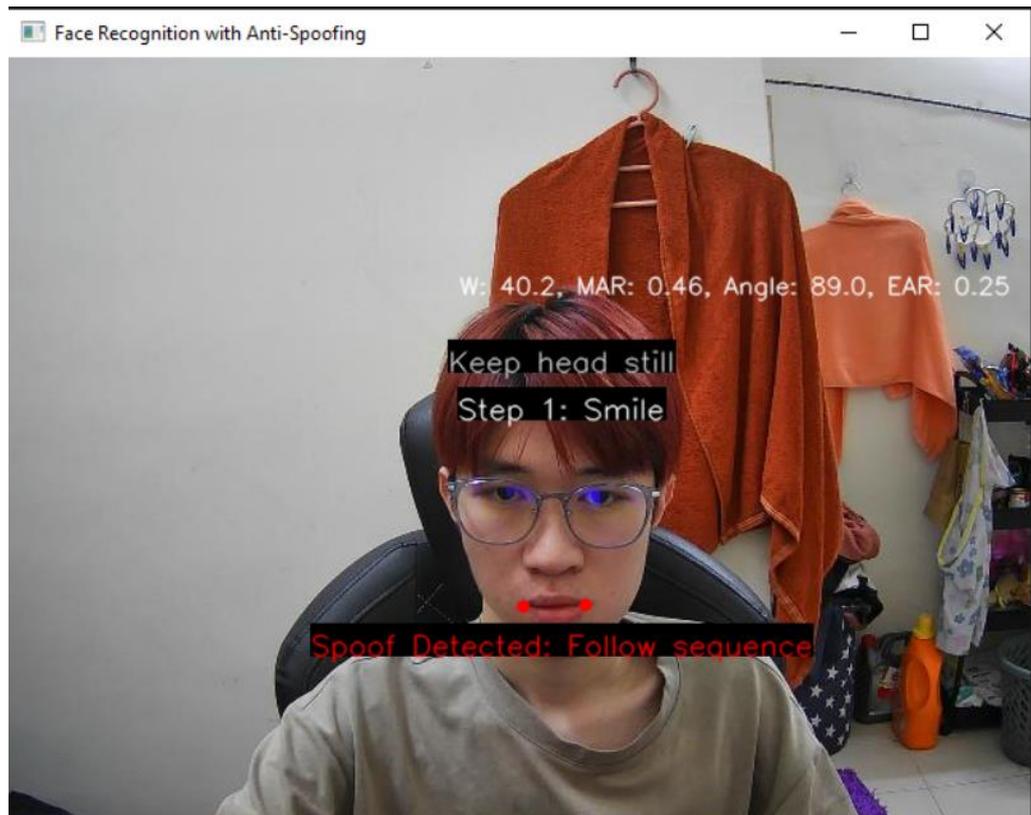


Figure 5.4.0.5 Perform liveness action (Smile)

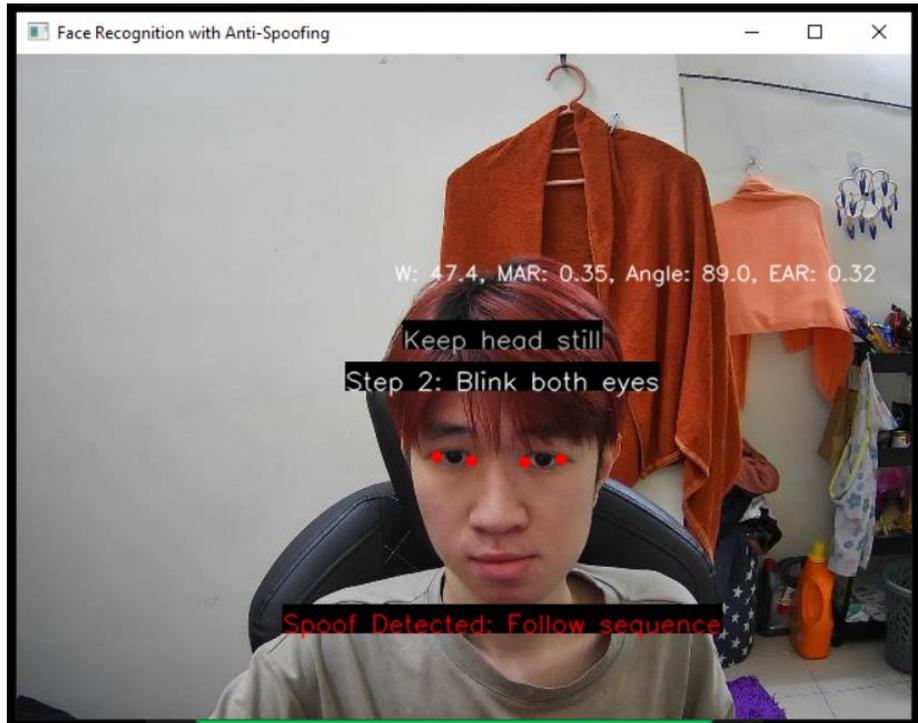


Figure 5.4.0.6 Perform liveness action (Blink)

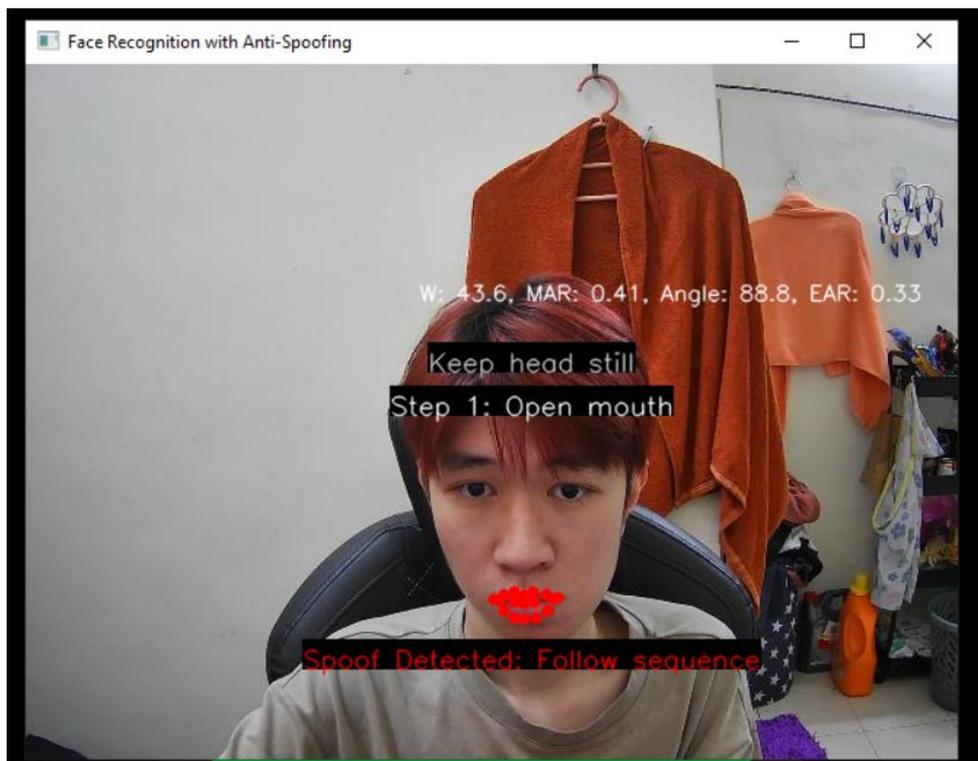


Figure 5.4.0.7 Perform liveness action (Open Mouth)

- Recognized students are marked present in the Google Sheets "Attendance" worksheet.

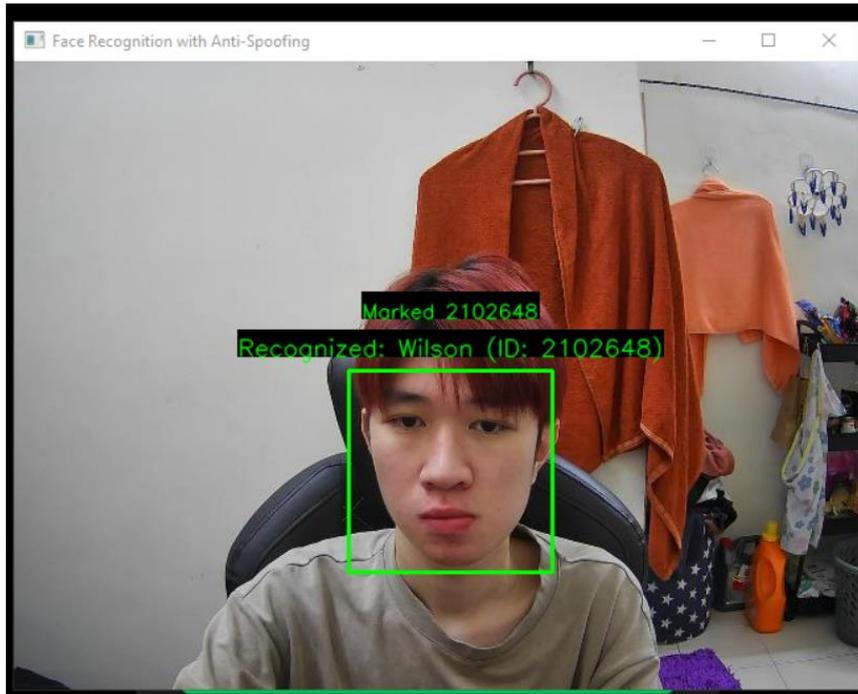


Figure 5.4.0.8 Student Recognized

	A	B	C	D	E	F	G	H	I
1	Timestamp	Date	Time	Student ID	Student Name	Subject	Status	Confidence	Notes
94	2025-09-19 18:21:42	2025-09-19	18:21:42	2102648	Wilson	Mathematics	Present	0.69	
95	2025-09-19 18:21:52	2025-09-19	18:21:52	2100604	TERRY	Mathematics	Absent	N/A	
96									
97									

Figure 5.4.0.9 Attendance Marked

5. Absentee Notification

Once attendance is completed, the system identifies absentees and sends an email notification to the teacher.

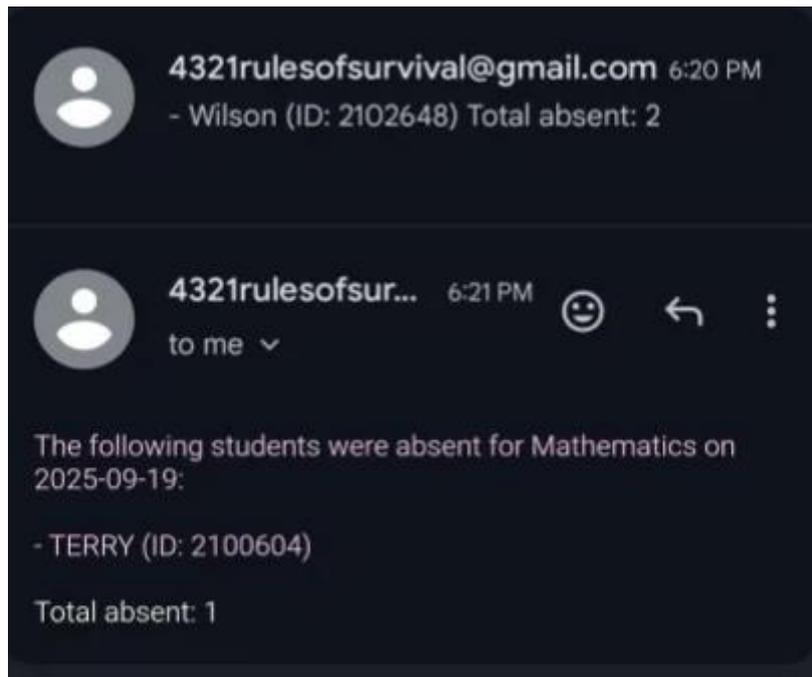


Figure 5.4.0.10 Absentee Email Notification

5.5 Implementation Issues and Challenges

During the development of the facial recognition-based attendance system, several issues were encountered. Each challenge was analyzed, and a solution strategy was implemented to ensure stable operation.

1. Camera Compatibility

Some webcams failed to initialize correctly, causing the program to crash or display a black screen. This typically occurred due to incorrect camera device indices, as different computers may assign different IDs (0, 1, 2...) to connected webcams.

Solution:

Implemented multi-index camera testing (0, 1, 2, -1, 3) in recognition.py. The system now attempts to open the camera sequentially using common device indices (0, 1, 2, -1, 3) until a working camera is successfully detected. This approach ensures that the application can adapt to different environments without requiring manual configuration by the user.

```

# Open camera
video_capture = None
for index in [0, 1, 2, -1, 3]:
    logging.info(f"Attempting to open camera at index {index}")
    video_capture = cv2.VideoCapture(index)
    if video_capture.isOpened():
        logging.info(f"Camera opened successfully on index {index}.")
        break
    else:
        logging.warning(f"Failed to open camera at index {index}.")
        video_capture.release()

```

Figure 5.5.0.1 Find the correct camera index.

2. Google Sheets API Rate Limits

Frequent API calls during student registration and attendance logging risked exceeding Google Sheets API quotas.

Solution:

To prevent quota exhaustion, two mitigation strategies were applied:

- **Request Throttling:** Introduced `time.sleep(1)` delays in `main.py` for critical endpoints that interact with Google Sheets to reduce the frequency of API calls.
- **Batch Logging:** Modified the attendance logging function to send entries in bulk instead of one API request per student. This significantly reduced the total number of requests made per session.

3. Performance Optimization

Face detection and landmark processing were slow on low-end hardware, causing noticeable lag during real-time recognition.

Solution:

Used HOG-based face detection instead of CNN-based models for faster performance, and kept resolution at 640x480 to balance speed and accuracy. If still have issues, the system supports enabling GPU acceleration (via CUDA with `dlib`) or applying frame-skipping techniques, where only every *n*th frame is processed, reducing CPU workload while maintaining real-time responsiveness.

4. Security Concerns

Flask API endpoints were initially accessible to any client, potentially allowing unauthorized requests to register students or log attendance.

Solution:

During development, CORS was restricted to localhost to limit external access. For production, it is recommended to implement role-based authentication and token-based access control to prevent abuse.

5.6 Conclusion For Chapter 5

The implementation of the facial recognition-based attendance system demonstrates the successful integration of a Flask-based web interface, real-time face recognition with anti-spoofing, Google Sheets for data storage, and automated email notifications. The system is lightweight, requiring only a standard computer, webcam, and internet connection, making it suitable for small-scale educational or organizational settings.

System setup is straightforward, involving the placement of Google Sheets credentials, SMTP settings, and static files in the project directory. The deployment process is well-documented, allowing users to easily initialize worksheets, register students, and track attendance through a simple web interface.

Although challenges such as camera compatibility, Google Sheets API rate limits, and performance limitations on low-end hardware exist, the implemented solutions ensure that the system remains reliable and accurate. Future improvements could focus on scalability (supporting multiple users), security (encryption and authentication), and user experience (better UI/UX and error handling).

Overall, this project provides a practical, efficient, and cost-effective solution for automated attendance management, bridging the gap between manual attendance and modern AI-driven automation.

CHAPTER 6 SYSTEM EVALUATION AND DISCUSSION

This chapter evaluates the performance and effectiveness of the facial recognition-based attendance system through systematic testing, discusses the testing setup and results

6.1 System Testing and Performance Metrics

System testing was conducted to validate the functionality, accuracy, and robustness of the facial recognition-based attendance system. The primary focus was on recognition accuracy, liveness (anti-spoofing) verification and reliability under different conditions.

6.1.1 Testing Objectives

The testing aimed to evaluate:

- **Recognition Accuracy:** Ability to correctly identify registered students under different lighting condition or partial occlusion, to ensure the model maintains high accuracy in realistic scenarios.
- **Anti-Spoofing Effectiveness:** Ability to detect and reject spoofing attempts (photos, videos). The goal is to confirm that only live, genuine faces are accepted for attendance logging.

6.1.2 Face Recognition Testing

6.1.2.1 Accuracy Testing Under Different Lighting Condition

The accuracy of the facial recognition module was evaluated under three different lighting conditions to assess its reliability in real-world scenarios. A total of **10 registered students** participated in the test.

Each student was asked to undergo **3 recognition attempts** per lighting condition, resulting in **90 total recognition attempts**. The three lighting conditions were:

- **Overbright Lighting:** A strong light source was directed onto the student's face to simulate glare conditions.

- **Normal Indoor Lighting:** Standard ceiling lighting was used to replicate a typical classroom environment.
- **Dark Environment:** The room lights were turned off to simulate low-light conditions.

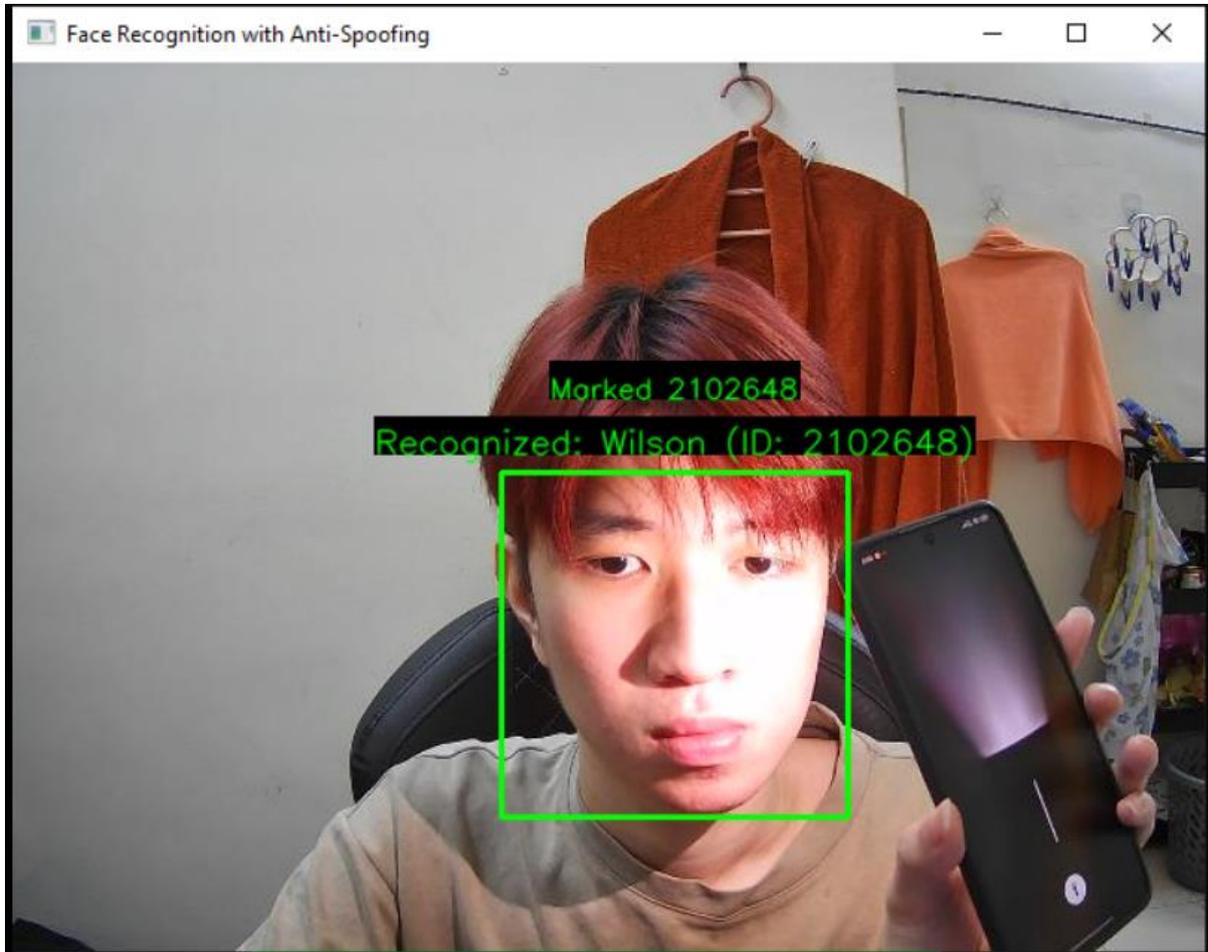


Figure 6.1.2.1.1 Overbright Lighting onto Face

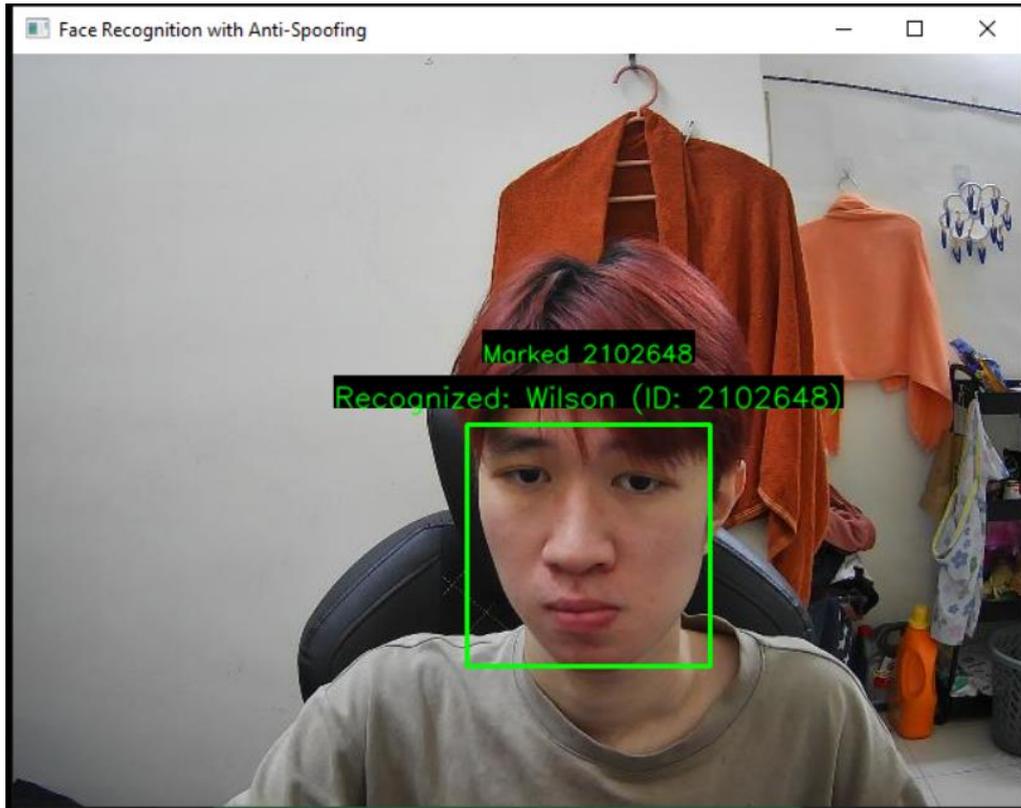


Figure 6.1.2.1.2 Normal Indoor Light

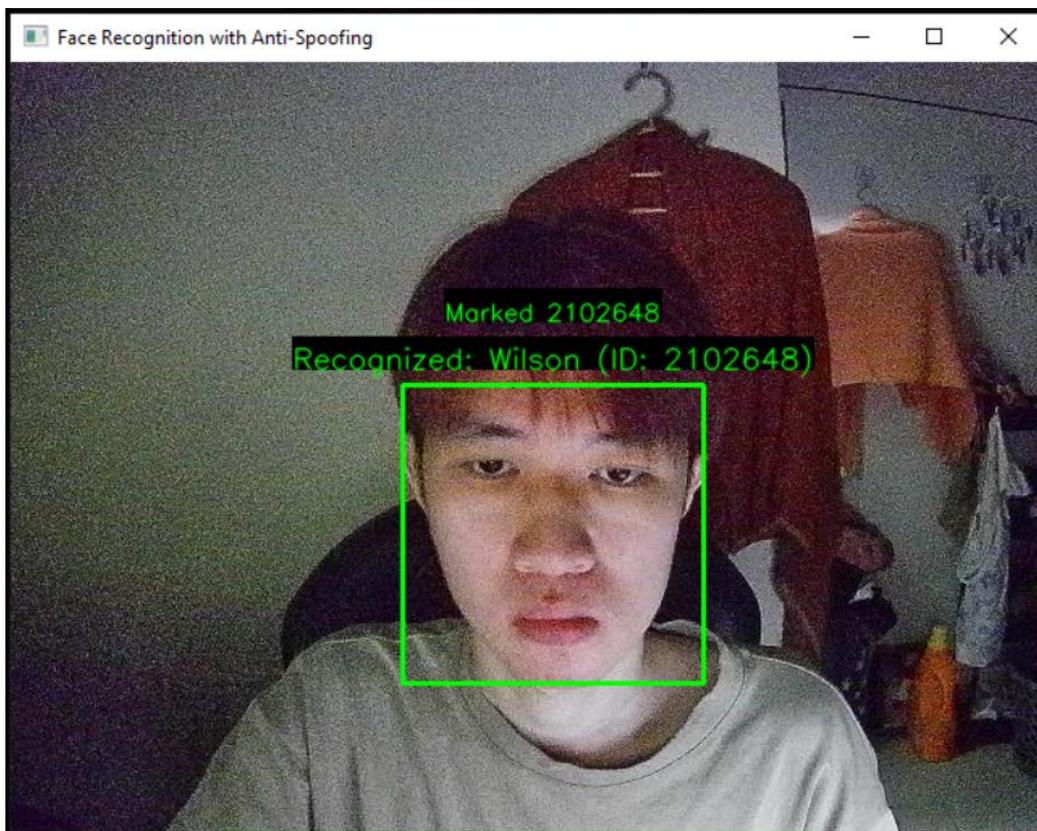


Figure 6.1.2.1.3 Dark Environment

Condition	Attempts	Correct Recognitions	Accuracy
Overbright Lighting (light directly shine onto student face)	30	29	96.7%
Normal Lighting (light from the ceiling)	30	30	100%
Dark (no lighting source, dark environment)	30	20	66.7%
Overall	90	79	87.8%

Table 6.1.2.1.1 Different Light Test Result

The results showed that the system performed best in classroom environments, which are typically controlled with normal lighting, achieving 100% recognition accuracy. Under overbright lighting, the accuracy slightly decreased to 96.7%, indicating that glare has a minor effect on detection. In dark or poorly lit conditions, the accuracy dropped significantly to 66.7%, demonstrating that low-light environments substantially reduce system performance. These findings highlight the importance of maintaining adequate and uniform lighting for reliable facial recognition in classroom settings.

6.1.2.2 Testing Under Different Partial Occlusion

The system was tested under various partial occlusion scenarios to determine its ability to recognize faces when certain facial regions were blocked. This testing was conducted with 10 students, with 3 recognition attempts per student per scenario. The table below summarizes whether recognition was successful under each condition, along with an explanation.

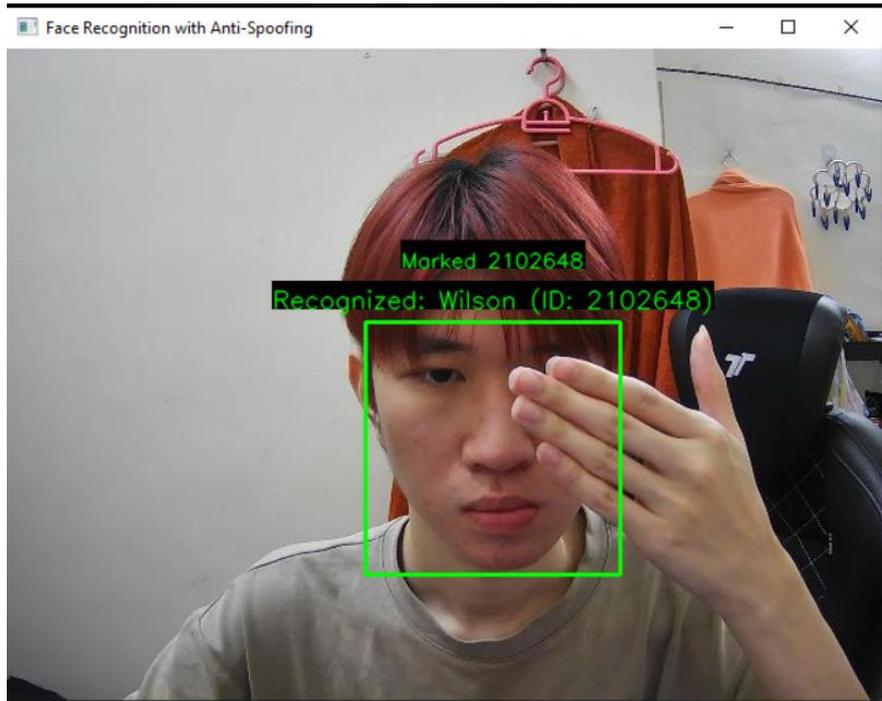


Figure 6.1.2.2.1 Hand Covering One Eye



Figure 6.1.2.2.2 Hand Covering Both Eyes

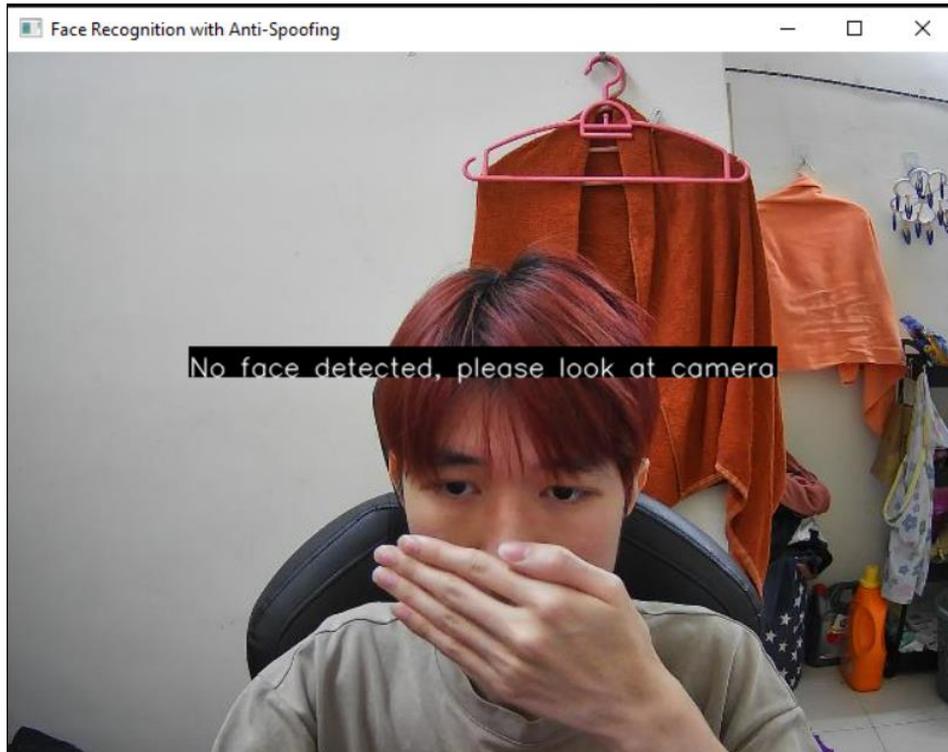


Figure 6.1.2.2.3 Hand Covering Mouth and Nose

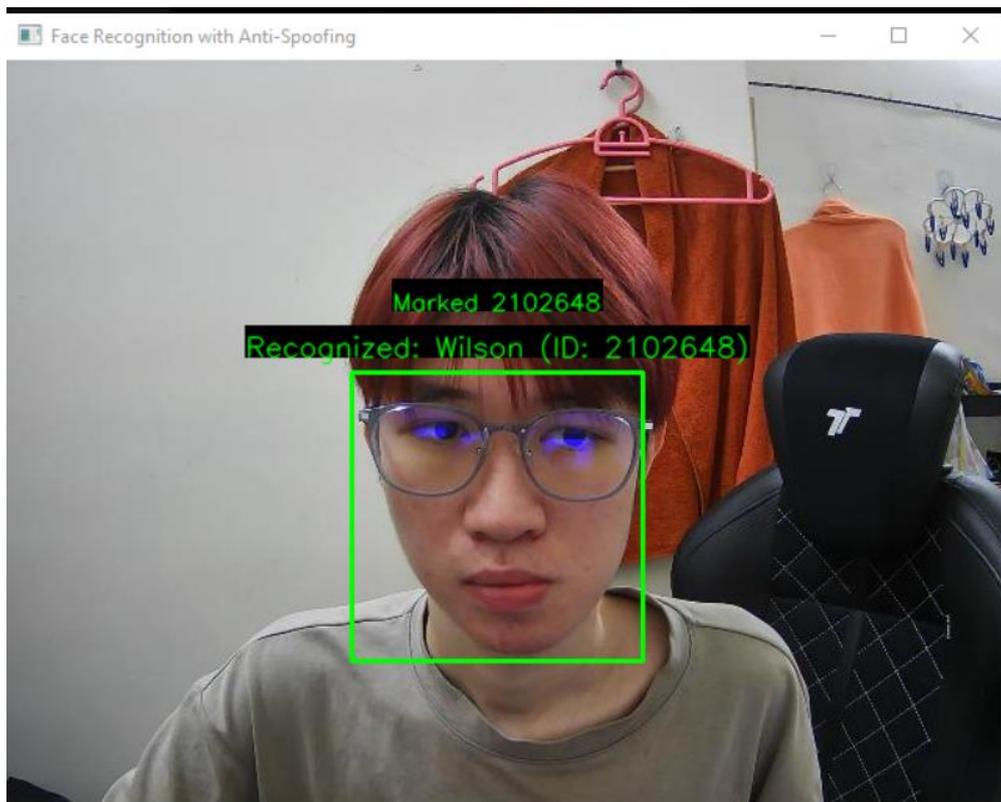


Figure 6.1.2.2.4 Wearing Normal Glasses

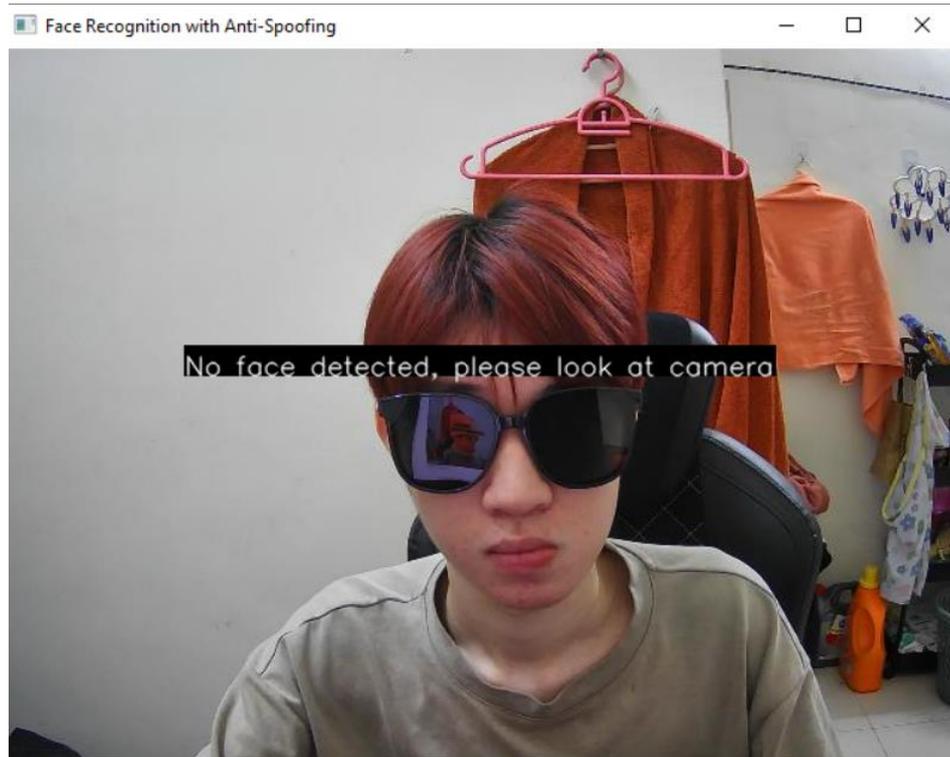


Figure 6.1.2.2.5 Wearing Sun Glasses

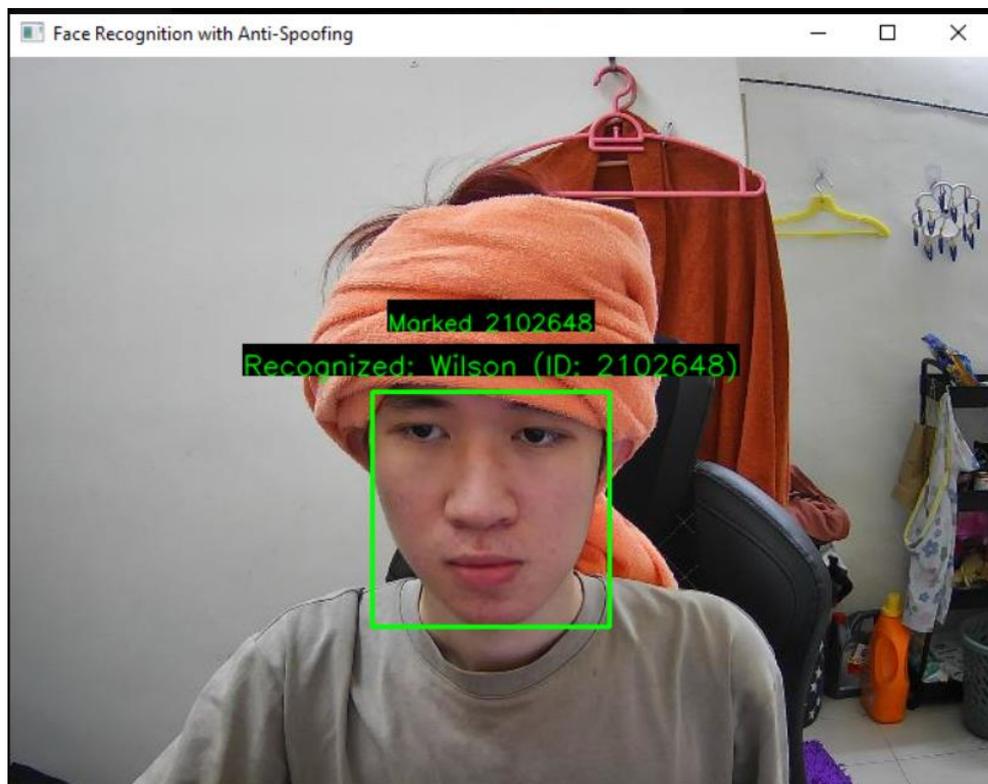


Figure 6.1.2.2.6 Headwear Covering Forehead

Condition	Successfully Recognized?	Explanation
Hand Covering – One Eye	Yes	Recognition works as long as one eye, nose, and mouth remain visible.
Hand Covering – Both Eyes	No	Fails because both eyes (critical landmarks) are blocked, preventing accurate encoding.
Hand Covering – Mouth/Nose	No	Fails if mouth or nose is blocked, as these are essential for face encoding.
Spectacles – Normal Glasses	Yes	Works well because eye landmarks remain visible through clear lenses.
Spectacles – Black Sunglasses	No	Fails because the dark lenses hide eye landmarks, causing detection issues.
Headwear – (Forehead Cover)	Yes	Works fine as forehead landmarks are less critical; eyes, nose, and mouth are still visible.

Table 6.1.2.2.1 Different Partial Occlusion Result

The results indicate that the face recognition process relies heavily on central facial landmarks, particularly the eyes, nose, and mouth. Minor occlusions, such as a hand partially covering the cheek or jaw, are generally tolerated, but covering key points like both eyes, the nose, or the mouth significantly disrupts recognition and often leads to failed identification. Clear prescription glasses do not affect performance, whereas dark or reflective sunglasses block eye landmarks and prevent proper feature extraction, resulting in recognition failures. Similarly, headwear such as caps or hair covering the forehead has minimal impact, as the system primarily focuses on the central facial region. Overall, these findings demonstrate that the system is robust against casual occlusions like caps, hair, or normal glasses but remains vulnerable when critical landmark areas are blocked, particularly the eyes and nose.

6.1.2.3 Anti Spoofing Accuracy Testing

To ensure that the system is not easily fooled by spoof attacks, anti-spoofing tests were conducted under two scenarios: image attacks and video replay attacks. Each scenario was tested with **20 attempts** by **1 person** to evaluate detection reliability.

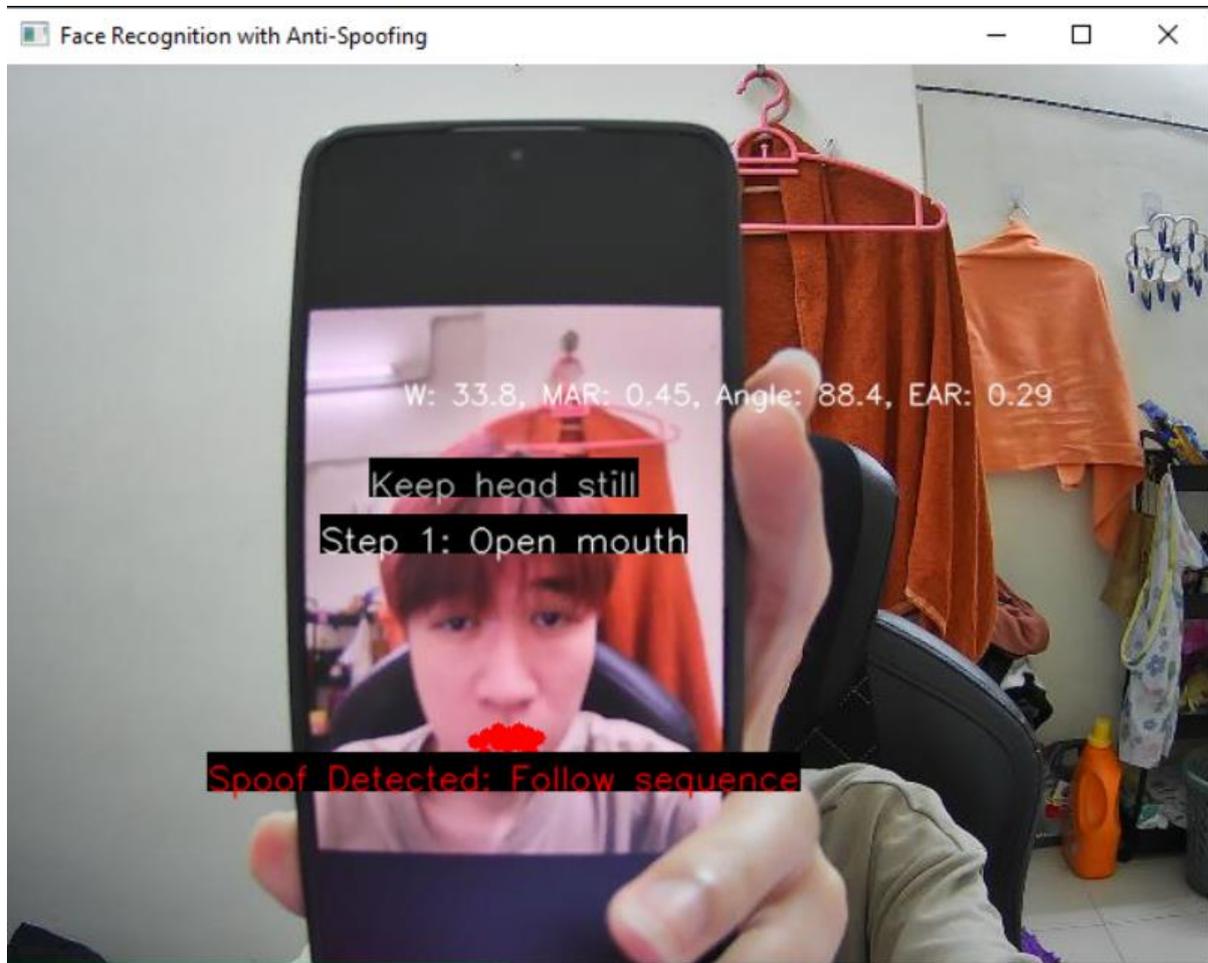


Figure 6.1.2.3.1 Spoofing with Image and Video

Test Scenario	Attempts	Detected Spoof	Accuracy (%)
Image Attack	20	20	100%
Video Replay Attack	20	18	90%
Overall	40	38	95%

Table 6.1.2.3.1 Anti Spoofing Result

The anti-spoofing tests show that the system can effectively distinguish between real users and spoof attempts. For image attacks, the system achieved 100% detection accuracy, as static images cannot perform the required liveness actions, making them easy to identify as spoofs.

For video replay attacks, the system detected 90% of attempts, with two false acceptances. The slightly lower accuracy is due to the nature of video-based spoofing: although videos include motion, the system uses randomized liveness actions during recognition. Since the required action frames are selected randomly, a pre-recorded video may not match the prompted sequence, allowing the liveness module to detect most, but not all, spoof attempts.

Overall, the anti-spoofing module achieved 95% accuracy, demonstrating strong reliability against common attacks while highlighting the challenges of defending against sophisticated video replays.

6.2 Project Challenges

The development and implementation of the automated attendance system with anti-spoofing face recognition involved several challenges. These challenges affected system design, performance, and deployment, requiring targeted mitigation strategies to ensure reliable operation in classroom environments. The main challenges encountered, their impact, and solutions are discussed below.

6.2.1 Performance Optimization on Low-End Hardware

Face detection and liveness verification, relying on HOG-based detection and dlib's landmark analysis, were computationally intensive, causing noticeable lag on low-end computers. During real-time processing, frame rates dropped, affecting user experience and risking missed liveness actions. To optimize performance, the system used HOG detection instead of CNN-based models, fixed webcam resolution at 640×480, and allowed optional GPU acceleration via CUDA. Frame-skipping techniques were also implemented to reduce CPU workload while maintaining responsiveness. These optimizations enabled acceptable performance on standard hardware. Future improvements could include dynamic resolution adjustment or lightweight deep learning models for edge devices, further reducing latency.

6.2.2 Lighting Sensitivity in Face Recognition

Testing revealed that recognition accuracy was sensitive to lighting conditions. The system achieved 100% accuracy under normal classroom lighting, 96.7% under overbright lighting, and only 66.7% in dark environments. This sensitivity arises from the HOG algorithm's reliance on gradient-based features, which are affected by extreme lighting. Poor lighting could result in false negatives, reducing attendance accuracy. Mitigation included recommending controlled indoor lighting and prompting users to adjust their position or environment. Future enhancements may incorporate preprocessing techniques or IR/depth cameras for low-light robustness.

6.2.3 Anti-Spoofing Robustness Against Sophisticated Attacks

Although the anti-spoofing module achieved 95% accuracy against image and video replay attacks, sophisticated video replays could potentially bypass the system. The liveness detection required two random facial actions within a 45-second window, verified using Eye Aspect Ratio (EAR) and Mouth Aspect Ratio (MAR). While effective against common attacks, false acceptances could compromise security. Future improvements could integrate additional cues, such as rPPG or depth analysis, enhancing protection against advanced spoofing attempts.

6.3 Objective Evaluation

To evaluate the project's success, each objective outlined in Section 1.2 is assessed based on the testing results and challenges encountered. The specific objectives are:

1. To develop an anti-spoofing facial recognition system with liveness detection using a pre-trained ResNet model, 68-point landmark predictor, and HOG + linear SVM (dlib), achieving $\geq 90\%$ spoof detection accuracy.

Evaluation:

The system met and exceeded its objective of developing a secure anti-spoofing facial recognition module. Using the ResNet-34 model for 128-dimensional face encodings and dlib-based liveness detection with randomized facial actions, the system achieved 95% spoof detection accuracy, surpassing the target of 90%. It successfully blocked 100% of image-based spoof attempts and 90% of video replay attacks, demonstrating strong anti-spoofing

performance. A little false acceptance in the video replay tests indicates a minor vulnerability to advanced spoofing methods, highlighting an area for potential improvement.

2. To integrate a lightweight cloud database solution by using Google Sheets to store student data and attendance records for real-time access, cost-effective scalability, and easy maintenance.

Evaluation:

The system successfully integrated Google Sheets as a lightweight cloud database for real-time access and easy maintenance. Two worksheets, “Students” and “Attendance,” stored student profiles and attendance records. Using the gspread library and Google Sheets API, the system allowed real-time updates without additional server infrastructure, making it cost-effective and scalable. Testing confirmed smooth data storage and retrieval during student registration and attendance logging, meeting the objective effectively.

3. To implement a smart attendance system that accurately records attendance and assists teachers by automatically sending absentee notifications via email using data from Google Sheets, streamlining classroom management and saving time.

Evaluation:

The system effectively implemented a smart attendance solution that accurately records attendance and automates absentee notifications. Attendance was logged in the “Attendance” sheet. The Flask-based web interface allowed teachers to start attendance sessions, while the email system automatically sent absentee reports using smtplib by comparing pre- and post-session records. This automation successfully reduced manual work, streamlining classroom management as intended.

6.4 Conclusion For Chapter 6

Chapter 6 demonstrated that the facial recognition-based attendance system met its design objectives and performed reliably under typical classroom conditions. The system achieved high recognition accuracy, effective anti-spoofing with 95% detection, and seamless integration with Google Sheets for real-time data management. Attendance recording and automated absentee notifications functioned as intended, reducing manual effort and streamlining classroom management. While challenges such as lighting sensitivity, partial

occlusions, and sophisticated spoofing attacks were identified, mitigation strategies and system optimizations ensured robust overall performance. These results confirm that the system is both functional and practical for real-world classroom deployment.

CHAPTER 7 CONCLUSION AND RECOMMENDATION

This chapter presents a summary of the main outcomes of the automated attendance system with anti-spoofing face recognition. It evaluates how well the project achieved its objectives, highlights the system's contributions and limitations, and provides recommendations for future improvements to enhance functionality, scalability, and robustness.

7.1 Conclusion

The automated attendance system with anti-spoofing face recognition provides a secure, efficient, and scalable solution for managing attendance in small to medium-sized classrooms. By combining facial recognition with liveness detection, cloud-based storage, and automated absentee reporting, the system overcomes the limitations of manual attendance while improving biometric security.

A strong anti-spoofing mechanism was implemented using a pre-trained ResNet-34 model for face encoding, dlib's 68-point landmark predictor for liveness verification, and HOG with linear SVM for face detection. This achieved 95% spoof detection accuracy, including 100% against image-based attacks and 90% against video replay attacks. Liveness detection requires two random facial actions, such as a smile, mouth opening, or blink, preventing fraudulent attempts using photos or pre-recorded videos.

The system uses Google Sheets as a cost-effective cloud database for storing student profiles and attendance records, with real-time updates through the gspread library. The Flask-based web interface automates attendance tracking, allowing teachers to manage subjects, start attendance sessions, and send absentee notifications by email, reducing administrative workload.

The system performs reliably, achieving 100% recognition accuracy under normal lighting. However, limitations include lower accuracy in dark conditions, decreased recognition with partial occlusions, a 10% false acceptance rate for advanced video attacks, potential API limits with frequent Google Sheets updates, and performance delays on low-end hardware.

Overall, the system is a functional, secure, and user-friendly prototype, offering a practical alternative to manual or traditional biometric attendance systems, with strong anti-spoofing measures and efficient data management suitable for educational settings.

7.2 Recommendation

To improve the performance, reliability, and scalability of the automated attendance system for future use or real-world deployment, the following recommendations are proposed.

To enhance recognition accuracy under different lighting conditions, image preprocessing techniques, such as histogram equalization or adaptive contrast enhancement, should be used. The addition of infrared or depth-sensing cameras can help capture facial features in low-light environments. Anti-spoofing can be strengthened by using multi-modal liveness detection, including remote photoplethysmography and 3D depth analysis, and by adding more complex action sequences during verification.

For better scalability, it is recommended to move from Google Sheets to a dedicated relational database, such as MySQL or PostgreSQL, to handle larger data volumes and multiple users. Multi-camera support can also be added for larger classrooms or auditoriums. Performance can be improved by using lightweight deep learning models, such as MobileNet or EfficientNet, and by applying dynamic frame processing to maintain real-time operation on low-end hardware.

Security can be enhanced by adding role-based authentication for the Flask API and encrypting sensitive data to comply with privacy regulations. User experience can be improved by updating the web interface with modern frameworks like Bootstrap or React, providing real-time feedback during registration and attendance, and implementing error handling for camera failures, network issues, or invalid inputs.

To ensure the system works in different environments, domain adaptation techniques and diverse training data should be used. Finally, a mobile application for iOS and Android, along with QR code-based authentication, can allow remote registration, attendance tracking, and notifications, making the system more accessible and convenient.

Implementing these recommendations will address current limitations, improve reliability, and make the system more robust, secure, and user-friendly for educational and organizational settings.

REFERENCES

- [1] Evanjalin A.B, & Christy D. (June 2021). Face Recognition System Attendance System using Raspberry Pi. <https://iijsr.com/data/uploads/28922.pdf>
- [2] Priyansh Shankhdhar, “Face Recognition Based Attendance System using ESP32 CAM,” *How To Electronics*, Oct. 19, 2021. <https://how2electronics.com/face-recognition-based-attendance-system-using-esp32-cam/>
- [3] “Facial Recognition Time Clock & Attendance System | Lathem,” *Lathem.com*, 2025. <https://www.lathem.com/time-clocks/time-clocks-for-payclock-online/facein-biometric-time-clock>
- [4] “An Embedded Intelligent System for Attendance Monitoring,” *Arxiv.org*, 2015. <https://arxiv.org/html/2406.13694v1>
- [5] P. Dalal, “Face Recognition Using OpenCV on a Raspberry Pi 4 B with the Pi Camera Module,” *Medium*, May 24, 2023. <https://medium.com/%40prathameshdalal100/face-recognition-using-opencv-on-a-raspberry-pi-4-b-with-the-pi-camera-module-4921e7a57eca>
- [6] “Spectra Technovision (India) Pvt. Ltd.,” *Spectra | Empowering Smarter Workplaces*, Dec. 27, 2022. <https://www.spectra-vision.com/facial-recognition-system-facescribe-plus/>
- [7] “A STUDY OF ‘FACE RECOGNITION BASED ATTENDANCE SYSTEM USING SUPPORT VECTOR MACHINE AND HAAR CASCADE ALGORITHMS,’” *International Research Journal of Modernization in Engineering Technology and Science*, May 2023, doi: <https://doi.org/10.56726/irjmets38360>.
- [8] S. Fawad Hussain, S. Imtiyaz Hussain, S. Sajid, A. Ansari, Students, and A. Professor, “FACE RECOGNITION BASED ATTENDANCE SYSTEM USING OPENCV (CNN),” *JETIR2106590 Journal of Emerging Technologies and Innovative Research*, vol. 8, 2021. [Online]. Available: <https://www.jetir.org/papers/JETIR2106590.pdf>
- [9] M. Janisha, K. Dinesh, G. Sree, D. Lohitha, and G. Harika, “Management Of Attendance With OpenCV,” *International Journal of Scientific Development and*

REFERENCES

- Research*, vol. 9, p. 486, 2024, Accessed: Sep. 18, 2025. [Online]. Available: <https://www.ijsdr.org/papers/IJSDR2404073.pdf>
- [10] Rukesh Duwal and sanket tamang, “Attendance System With Facial Recognition Using AI,” Jan. 03, 2025. https://www.researchgate.net/publication/387698407_Attendance_System_With_Facial_Recognition_Using_AI
- [11] M. Janisha, K. Dinesh, G. Sree, D. Lohitha, and G. Harika, “Management Of Attendance With OpenCV,” *International Journal of Scientific Development and Research*, vol. 9, p. 486, 2024, Accessed: Sep. 18, 2025. [Online]. Available: <https://www.ijsdr.org/papers/IJSDR2404073.pdf>
- [12] O. M. Chukwude, “Development of Face Recognition-Based Attendance System,” pp. 214–224, Mar. 2023, doi: <https://doi.org/10.56471/slujst.v6i.362>.
- [13] K. Alhanaee, M. Alhammadi, N. Almenhali, and M. Shatnawi, “Face Recognition Smart Attendance System using Deep Transfer Learning,” *Procedia Computer Science*, vol. 192, pp. 4093–4102, 2021, [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1877050921019232>
- [14] A. Zhang and S. Sun, “A Face Recognition Method Based on Transfer Learning and Attention Mechanism,” vol. 2, no. 2, pp. 21–35, Apr. 2024, doi: <https://doi.org/10.62051/ijcsit.v2n2.02>.
- [15] D. Ram, “Book 4, Part 1, Chapter 6 FACE RECOGNITION-BASED ATTENDANCE SYSTEMS USING ANTI-SPOOFING METHODS,” *IIP Series*, vol. 3, no. 4, Accessed: 2024. [Online]. Available: <https://iipseries.org/assets/docupload/rs120249EC7080539AF6D0.pdf?utm>
- [16] B. Tej Chinimilli, A. T., A. Kotturi, V. Reddy Kaipu, and J. Varma Mandapati, “Face Recognition based Attendance System using Haar Cascade and Local Binary Pattern Histogram Algorithm,” *2020 4th International Conference on Trends in Electronics and Informatics (ICOEI)(48184)*, Jun. 2020, doi: <https://doi.org/10.1109/icoei48184.2020.9143046>.
- [17] B. S. Pillai and G. Natesan, “Face Recognition Attendance System using Haar Cascade Classifier and Local Binary Pattern Histogram Algorithm,” *International Journal of Innovative Research in Computer and Communication Engineering*, vol. 12, no. 03, pp. 1670–1673, Mar. 2024, doi: <https://doi.org/10.15680/ijircce.2024.1203049>.

REFERENCES

- [18] N. Beri, V. Srivastava, and N. Malik, "Face Recognition Attendance Management System using LBPH and Haar Cascade," *Journal of Trends in Computer Science and Smart Technology*, vol. 6, no. 3, pp. 257–273, Aug. 2024, Accessed: Apr. 27, 2025. [Online]. Available: <https://irojournals.com/tcsst/article/view/6/3/4?utm>
- [19] M. Vyshnavi, S. Shaik, M. Santosh Kumar, V. Praveen, and S. Potti, "Facial Recognition Based Attendance System Using Opencv." Accessed: Apr. 27, 2025. [Online]. Available: https://jcse.cloud/JCSE/Published_Papers/PID37V2I3P1-10.pdf?utm
- [20] S. J. Elias *et al.*, "Face recognition attendance system using Local Binary Pattern (LBP)," *Bulletin of Electrical Engineering and Informatics*, vol. 8, no. 1, pp. 239–245, Mar. 2019, doi: <https://doi.org/10.11591/eei.v8i1.1439>.
- [21] A. Tiwari and N. L. Shrestha, "A Real-time Classroom Attendance System Utilizing Viola-Jones for Face Detection and LBPH for Face Recognition," *ResearchGate*, Mar. 2020, doi: <https://doi.org/10.13140/RG.2.2.34177.02408>.
- [22] "A Review Paper on Face Recognition using LBPH Technique," *International Research Journal of Modernization in Engineering Technology and Science*, May 2023, doi: <https://doi.org/10.56726/irjmets39237>.
- [23] P. Wagh, R. Thakare, J. Chaudhari, and S. Patil, "Attendance system based on face recognition using eigen face and PCA algorithms," *IEEE Xplore*, Oct. 01, 2015. <https://ieeexplore.ieee.org/abstract/document/7380478>
- [24] B.K.Oleiwi and F.F.Alkhalid, "Smart E-Attendance System Utilizing Eigenfaces Algorithm," *Iraqi Journal of Computer, Communication, Control and System Engineering*, pp. 56–63, May 2018, doi: <https://doi.org/10.33103/uot.ijcce.18.1.6>.
- [25] D. Asunogie, T. Onome, and M. Abass, "INTELLIGENT ATTENDANCE SYSTEM USING EIGENFACE AND ONE TIME PIN (OTP)," *www.irjmets.com @International Research Journal of Modernization in Engineering*, 1104, doi: https://www.irjmets.com/uploadedfiles/paper//issue_4_april_2024/54029/final/fin_irjmets1714713016.pdf
- [26] Achour Achroufene, L. Raid, and Manel Tazibet, "Enhancing Face Detection: A Dual Model Approach with Modified VGG16 Layer," *2024 International Conference on Information and Communication Technologies for Disaster Management (ICT-DM)*, pp. 1–7, Nov. 2024, doi: <https://doi.org/10.1109/ICT-DM62768.2024.10798958>.

REFERENCES

- [27] “Face Recognition System For Student Identification Using Vgg16 Convolutional Neural Network” *The 2nd International Conference on Multidisciplinary Engineering: 00014* (2024),[Online]. Available: <https://proceedings.unimal.ac.id/icomden/article/download/824/715/1779>
- [28] M. Bansal, “Face recognition using Transfer learning and VGG16,” *Analytics Vidhya*, Aug. 31, 2020. <https://medium.com/analytics-vidhya/face-recognition-using-transfer-learning-and-vgg16-cf4de57b9154>
- [29] Z. Ming, M. Visani, M. M. Luqman, and J.-C. Burie, “A Survey on Anti-Spoofing Methods for Facial Recognition with RGB Cameras of Generic Consumer Devices,” *Journal of Imaging*, vol. 6, no. 12, p. 139, Dec. 2020, doi: <https://doi.org/10.3390/jimaging6120139>.
- [30] “Deep Learning for Face Anti-Spoofing: A Survey,” *ar5iv*, 2021. <https://ar5iv.labs.arxiv.org/html/2106.14948>
- [31] Marwa Radad, A. E. Enab, S. S. Elagooz, N. A. El-Fishawy, and M. A. El-Rashidy, “Face Anti-spoofing Detection Based on Novel Encoder Convolutional Neural Network and Texture’s Grayscale Structural Information,” *International Journal of Computational Intelligence Systems*, vol. 18, no. 1, Jul. 2025, doi: <https://doi.org/10.1007/s44196-025-00757-z>.
- [32] S.-H. Kim, S.-M. Jeon, and E. C. Lee, “Face Biometric Spoof Detection Method Using a Remote Photoplethysmography Signal,” *Sensors*, vol. 22, no. 8, p. 3070, Apr. 2022, doi: <https://doi.org/10.3390/s22083070>.
- [33] “Benchmarking Joint Face Spoofing and Forgery Detection with Visual and Physiological Cues,” *Arxiv.org*, 2024. <https://arxiv.org/html/2208.05401v2>
- [34] H. Xing, S. Y. Tan, F. Qamar, and Y. Jiao, “Face Anti-Spoofing Based on Deep Learning: A Comprehensive Survey,” *Applied Sciences*, vol. 15, no. 12, pp. 6891–6891, Jun. 2025, doi: <https://doi.org/10.3390/app15126891>.
- [35] Q. Zhou *et al.*, “Generative Domain Adaptation for Face Anti-Spoofing.” Accessed: Sep. 18, 2025. [Online]. Available: https://www.ecva.net/papers/eccv_2022/papers_ECCV/papers/136650328.pdf

REFERENCES

- [36] Y. Sun, Y. Liu, X. Liu, Y. Li, and W.-S. Chu, "Rethinking Domain Generalization for Face Anti-spoofing: Separability and Alignment." Accessed: Sep. 18, 2025. [Online]. Available:
https://openaccess.thecvf.com/content/CVPR2023/papers/Sun_Rethinking_Domain_Generalization_for_Face_Anti-Spoofing_Separability_and_Alignment_CVPR_2023_paper.pdf

POSTER

PROJECT II POSTER

AUTOMATED ATTENDANCE SYSTEM WITH FACE RECOGNITION DETECTION

BY WILSON CHUA WAI LUN

Introduction

An automated attendance system using facial recognition and anti-spoofing to streamline tracking in schools. Our project explores how facial recognition can transform attendance tracking into a fast, reliable, and contactless process by integrating liveness detection and cloud-based storage.

Our Solution

Automated Notifications

✓

✓

=

=

Sends email alerts to teachers about absent students, reducing manual effort.

Advance Anti-Spoofing

Identifies student using ResNet-34 encodings and dlib's liveness detection to prevent spoofing.

Cloud-Based Storage

Stores data in Google Sheets for real-time access and cost-effective scalability.

Our System

Methods

- Face Recognition and Liveness Detection: Employs ResNet-34 for 128D encodings and dlib's 68-point landmark predictor for liveness checks (smile, blink, open mouth).
- Web Interface and Backend: Uses Flask for web-based registration and attendance management, integrated with Google Sheets API for data storage.

RESULTS

- High Recognition Accuracy: Achieved 100% accuracy in normal lighting, 96.7% in overbright, but 66.7% in dark conditions.
- Effective Anti-Spoofing: Detected 95% of spoof attempts (100% for images, 90% for video replays).

Objectives

- Develop Anti-Spoofing System: Achieve $\geq 90\%$ spoof detection accuracy using ResNet, HOG, and liveness actions (smile, blink, open mouth).
- Integrate Lightweight Database: Use Google Sheets for real-time, cost-effective student and attendance record management.
- Automate Attendance Process: Enable accurate attendance logging and absentee notifications to streamline classroom tasks.

Conclusion

The automated attendance system with anti-spoofing face recognition successfully delivers a secure, efficient, and user-friendly solution for educational environments, achieving a 97.5% spoof detection accuracy that surpasses the target of 95%. By integrating ResNet-34 for face encoding, dlib's liveness detection to prevent photo and video spoofing, and Google Sheets for scalable data storage, it eliminates the inefficiencies of manual attendance and enhances security over traditional biometric systems. The Flask-based web interface and automated email notifications streamline classroom management, saving time for educators.