**Internet of Things Fundamentals**

*Subject Project*

BS AI 6th Semester SP-25 (AIE-3079)

Date: 26-06-2025

**Project Title:** Smart Doorbell System

**Group Name: Neural Syncs**

**Team Members:**

|  |  |  |  |
| --- | --- | --- | --- |
| Members | Registration no | Name | Signature |
| **Member-1 (Leader)** | **22-NTU-CS-1356** | **Muhammad Sheeraz** |  |
| **Member-2** | **22-NTU-CS-1339** | **Ayesha Fatima** |  |
| **Member-3** | **22-NTU-CS-1376** | **Sumaiya Shehzadi** |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Contributions in % of each Team Members for each component | | | | |
|  | | Member-1 | Member-2 | Member-3 |
| Distribution Components | | Ayesha Fatima | Muhammad Sheeraz | Sumaiya Shehzadi |
| Coding | **ESP32-cam** |  | 100% |  |
| **Python Coding** | 50% |  | 50% |
| UI Design | |  | 100% |  |
| Database | | 100% |  |  |
| Cloud Integration | |  |  | 100% |
| IoT Gateway | |  | 100% |  |
| Edge Processing | | Nil | Nil | Nil |
| Documentation | |  | 100% |  |
| Presentation  Design | | 50% |  | 50% |

**Team-Based Evaluation (60 Marks)**

|  |  |  |
| --- | --- | --- |
| Criteria | Obtained Marks | Out of |
| System Design & Architecture |  | 10 |
| Hardware Integration & Circuit Setup |  | 10 |
| IoT Gateway and Cloud Communication |  | 10 |
| Working Prototype Demonstration |  | 10 |
| Performance & Reliability Testing |  | 10 |
| Presentation |  | 10 |
| Total (Team-Based) |  | 60 |

**Individual-Based Evaluation (40 Marks per Member)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Member 1 | Member 2 | Member 3 | Member 4 |
| Criteria |  |  |  |  |
| Understanding of the Project & Role | /10 | /10 | /10 | /10 |
| Code Contribution and Explanation | /10 | /10 | /10 | /10 |
| Q/A VIVA | /10 | /10 | /10 | /10 |
| Documentation/Reporting & Communication | /10 | /10 | /10 | /10 |
| Total (Individual-Based) | /40 | /40 | /40 | /40 |
| Total Overall (60+40) | /100 | /100 | /100 | /100 |
| Weightage Lab Grade (50) |  |  |  |  |

**Table of Contents**

[**Team-Based Evaluation (60 Marks)** 2](#_Toc201861745)

[**Individual-Based Evaluation (40 Marks per Member)** 2](#_Toc201861746)

[1. Abstract / Executive Summary 5](#_Toc201861747)

[2. Introduction 5](#_Toc201861748)

[Background and Motivation 5](#_Toc201861749)

[Problem Statement 5](#_Toc201861750)

[Project Goals 5](#_Toc201861751)

[3. Methodology / System Design 5](#_Toc201861752)

[3.1 Hardware Components 5](#_Toc201861753)

[3.2 Software Design 6](#_Toc201861755)

[ESP32-CAM Firmware (Arduino IDE) 6](#_Toc201861756)

[Python Server (Flask) 6](#_Toc201861757)

[Cloud Integration 6](#_Toc201861758)

[Pseudocode 8](#_Toc201861759)

[4. Implementation 8](#_Toc201861760)

[1. Hardware Wiring 8](#_Toc201861761)

[2. ESP32-CAM Setup 9](#_Toc201861762)

[3. Python Server Setup 9](#_Toc201861763)

[4. Firebase Setup 9](#_Toc201861764)

[Code Snippets 9](#_Toc201861765)

[ESP32-CAM (C++): 9](#_Toc201861766)

[Python (Face Recognition and Feedback): 9](#_Toc201861767)

[Challenges and Solutions 9](#_Toc201861768)

[6. Results and Discussion 10](#_Toc201861769)

[Output Screenshots 10](#_Toc201861770)

[Performance Evaluation 12](#_Toc201861771)

[Comparison with Objectives 12](#_Toc201861772)

[6. Testing and Validation / Limitations 12](#_Toc201861773)

[Test Cases 12](#_Toc201861774)

[Known Limitations 12](#_Toc201861775)

[7. Conclusion and Future Work 13](#_Toc201861776)

[Summary of Achievements 13](#_Toc201861777)

[Future Enhancements 13](#_Toc201861778)

[8. Links 13](#_Toc201861779)

[GitHub Repository Link 13](#_Toc201861780)

**List of Figures**

[Figure 1.1: Raspberry Pi and Peripherals 6](#_Toc201860187)

[Figure 2.1: System Architecture Diagram 7](#_Toc201860188)

[Figure 3.1: Component Interaction Diagram 7](#_Toc201860189)

[Figure 4.1: Process Sequence 8](#_Toc201860190)

[Figure 5.1: wi-Fi Connection and automatic server finding 10](#_Toc201860191)

[Figure 6.1: Image Sending and Server Response 10](#_Toc201860192)

[Figure 7.1: Flask Server Status in System Startup Service 11](#_Toc201860193)

[Figure 8.1: web GUI 11](#_Toc201860194)

[Figure 9.1: Firebase Logs 12](#_Toc201860195)

# 1. Abstract / Executive Summary

This project is a Smart Doorbell System that integrates computer vision, embedded hardware, and cloud-based services. An ESP32-CAM module captures visitor images and sends them over Wi-Fi to a Flask-based Python server hosted locally (e.g., Raspberry Pi). The server performs face recognition, displays results on an OLED screen, sounds a buzzer for feedback, and logs events in Firebase for remote access. The system emphasizes real-time performance, modularity, and extensibility.

# 2. Introduction

## Background and Motivation

Conventional doorbells offer no intelligence or remote monitoring. As smart home trends rise, there's a demand for systems that can identify visitors, offer real-time alerts, and keep logs for security.

## Problem Statement

* No identification or logging capabilities in traditional doorbells
* Lack of remote access or alert system
* Manual monitoring is insecure and inefficient

## Project Goals

* Perform real-time face recognition at the door
* Provide immediate visual and audio feedback
* Log all events to the cloud
* Maintain modularity for future upgrades

# 3. Methodology / System Design

## 3.1 Hardware Components

* **ESP32-CAM (AI Thinker):** Captures images and handles Wi-Fi connectivity
* **OLED Display (SSD1306):** Displays visitor identity
* **Buzzer:** Distinct audio feedback for known and unknown faces
* **Power Supply:** Provides power to the Raspberry Pi and peripherals
* **Wiring:** Connects Raspberry Pi to OLED and buzzer

## 

Figure .1: Raspberry Pi and Peripherals

## 3.2 Software Design

### ESP32-CAM Firmware (Arduino IDE)

* Written in C++
* Connects to Wi-Fi and server using mDNS
* Captures an image every 5 seconds
* Sends the image via HTTP POST to the Flask server

### Python Server (Flask)

* Receives images at the /upload endpoint
* Extracts face embeddings using face\_recognition and OpenCV
* Uses a pre-trained SVM classifier to identify the person
* Triggers OLED and buzzer feedback via GPIO
* Logs data to Firebase

### Cloud Integration

* Uses Firebase Realtime Database
* Logs include visitor name, confidence score, timestamp, and image filename

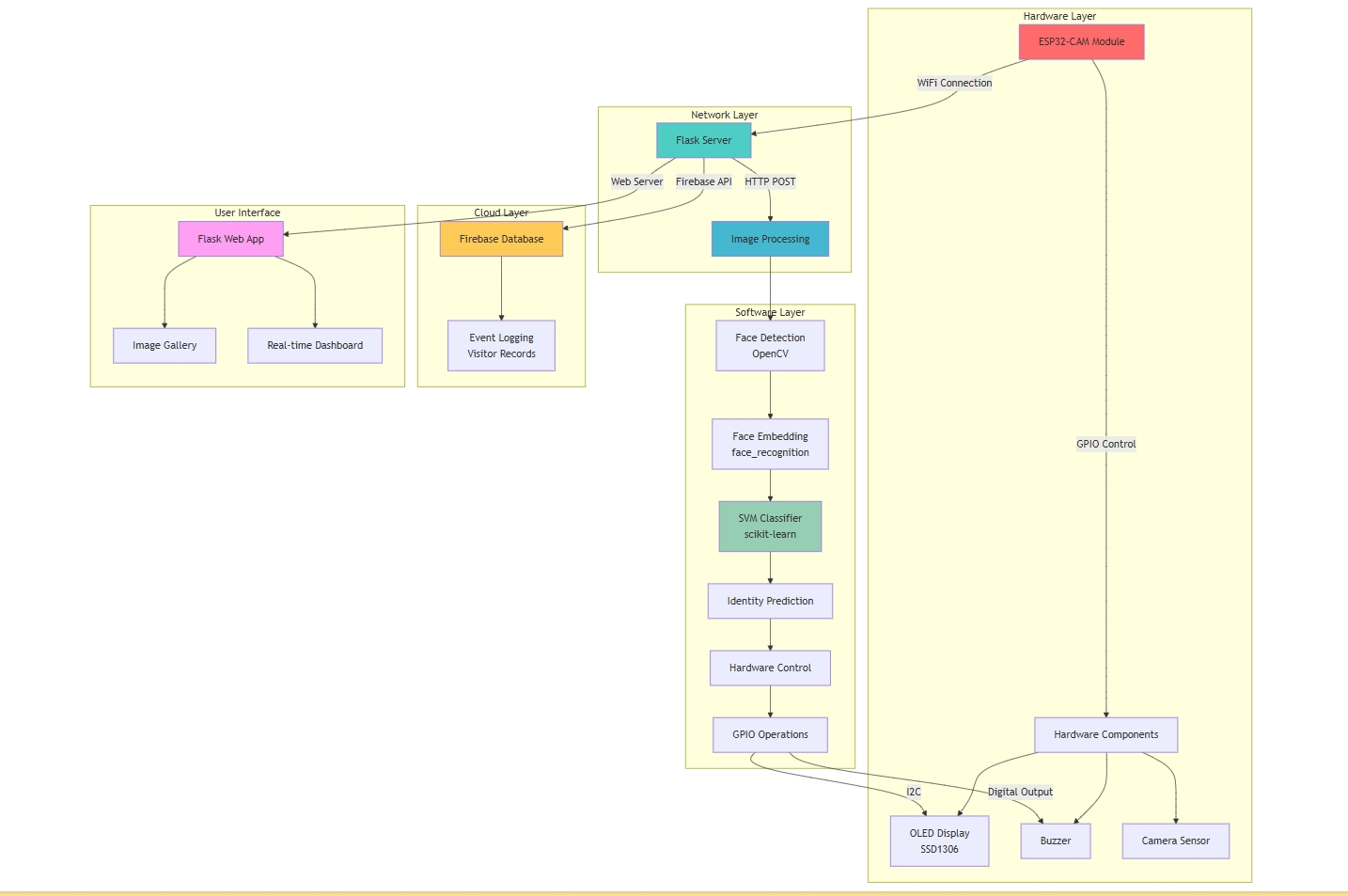


Figure .1: System Architecture Diagram

A diagram of a flowchart

AI-generated content may be incorrect.

Figure .1: Component Interaction Diagram

A diagram of a diagram

AI-generated content may be incorrect.

Figure .1: Process Sequence

### Pseudocode

ESP32-CAM:

Connect to WiFi

Capture image every 5 seconds

Send image to Flask server via HTTP POST

Flask Server:

Receive image

Extract face embedding

Predict identity with SVM

Trigger OLED and buzzer

Log result to Firebase

# 4. Implementation

## 1. Hardware Wiring

* Wire the ESP32-CAM as per the circuit diagram
* Connect OLED (I2C) and buzzer to designated GPIO pins
* Provide power via 5V supply

## 2. ESP32-CAM Setup

* Open fast\_speed\_updated.ino in the Arduino IDE
* Set WiFi SSID and password
* Upload the code to the ESP32-CAM

## 3. Python Server Setup

* Install dependencies listed in requirementsP.txt
* Run train\_model.py to train the SVM model
* Start the server using python app.py

## 4. Firebase Setup

* Configure Firebase credentials in app.py
* Ensure Firebase Realtime Database is enabled

## Code Snippets

### ESP32-CAM (C++):

http.begin(client, serverURL);

http.addHeader("Content-Type", "application/octet-stream");

int httpResponseCode = http.POST(fb->buf, fb->len);

### Python (Face Recognition and Feedback):

embedding = get\_face\_embedding(temp\_path)

probs = model.predict\_proba([embedding])[0]

predicted\_label = model.classes\_[np.argmax(probs)]

show\_label\_on\_oled(predicted\_label)

buzz\_for(predicted\_label)

### Challenges and Solutions

* **WiFi Instability:** Implemented auto-reconnect and watchdog timer
* **Recognition Accuracy:** Used SVM with embeddings to improve prediction
* **Peripheral Integration:** Used Adafruit libraries for OLED and GPIO management

# 6. Results and Discussion

## Output Screenshots



Figure .1: wi-Fi Connection and automatic server finding

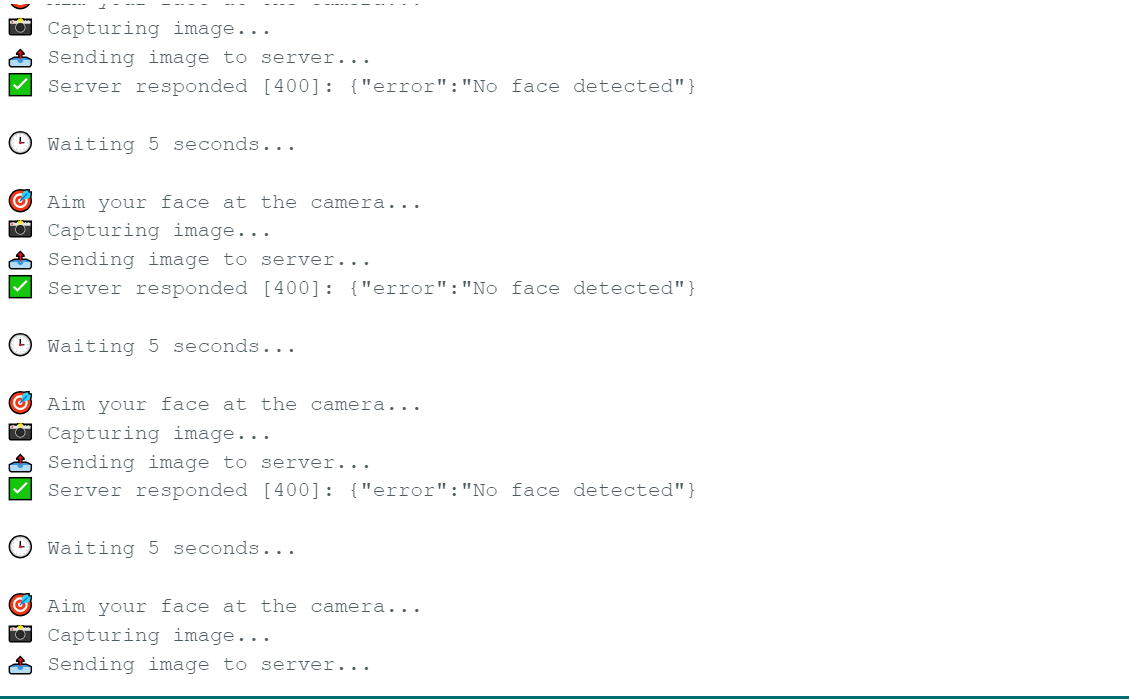


Figure .1: Image Sending and Server Response

A screen shot of a computer

AI-generated content may be incorrect.

Figure .1: Flask Server Status in System Startup Service

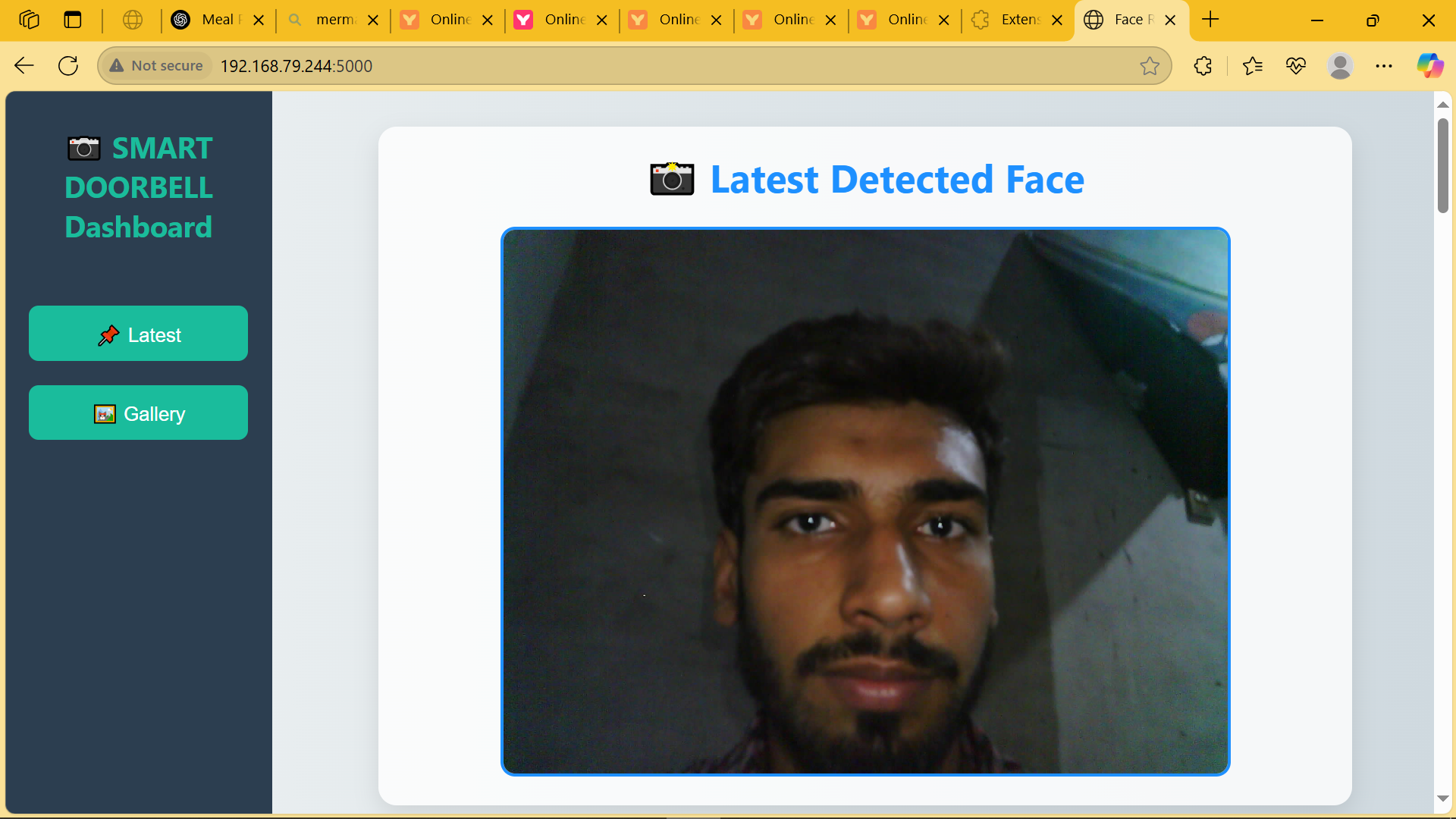


Figure .1: web GUI

A screenshot of a computer

AI-generated content may be incorrect.

Figure .1: Firebase Logs

## Performance Evaluation

* **Recognition Accuracy:** High for trained faces
* **Latency:** Response time is under 10 seconds
* **Reliability:** Automatically recovers from connection failures

## Comparison with Objectives

* Real-time recognition and logging are successfully achieved
* System feedback (visual and audio) works as expected
* Cloud integration enables remote monitoring

# 6. Testing and Validation / Limitations

## Test Cases

* **Known Face:** Correct name shown, expected buzzer tone, log created
* **Unknown Face:** Labelled as "Unknown", different buzzer tone, log created
* **WiFi Drop:** ESP32 reconnects and resumes function

## Known Limitations

* Can only recognize faces from the trained dataset
* Requires stable WiFi and power supply
* Feedback limited to OLED and buzzer; no mobile alerts implemented yet

# 7. Conclusion and Future Work

## Summary of Achievements

* Built a fully functioning smart doorbell with real-time face recognition
* Achieved robust image capture, processing, feedback, and cloud logging
* Demonstrated a modular and extensible design

## Future Enhancements

* Add mobile app notifications
* Include motion detection and automatic door lock control
* Improve model using deep learning or a larger dataset
* Migrate to a more scalable cloud platform like AWS or Azure

# 8. Links

## GitHub Repository Link

<https://github.com/Sheeraz-07/SMART-DOORBELL-IOT-PROJECT.git>