# HOME SECURITY SYSTEM

**A Mini-Project Report submitted in partial fulfillment of the requirements for the award of the degree of,**

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE AND ENGINEERING (IOT)**

Submitted by:

| | |
|---|---|
| **G. ROHITH** | **BU22CSEN0600072** |
| **P. SURENDRA** | **BU22CSEN0600049** |
| **G. RANJITH KUMAR** | **BU22CSEN0600060** |
| **S. B. JEEVAN** | **BU22CSEN0600197** |

**Subject Code:** CSEN2091P

**Subject Name:** OOSE-Based Application Development Lab

**Department of Computer Science & Engineering,**

**GITAM SCHOOL OF TECHNOLOGY**

**GANDHI INSTITUTE OF TECHNOLOGY AND MANAGEMENT**

**(Deemed to be University)**

**Bengaluru Campus.**

**April 2025**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**GITAM SCHOOL OF TECHNOLOGY**

**GITAM**

**(Deemed to be University)**

## CERTIFICATE

This is to certify that the project report entitled **"HOME SECURITY SYSTEM"** is a Bonafide record of work carried out by **G. ROHITH (BU22CSEN0600072), P. SURENDRA (BU22CSEN0600049), and G. RANJITH KUMAR (BU22CSEN0600060), S.B. JEEVAN (BU22CSEN06000197)** submitted in partial fulfillment of the requirement for the award of the degree of **Bachelors of Technology in Computer Science and Engineering**.

**SIGNATURE OF THE CHARGE**

# CHAPTER-1

# 1. INTRODUCTION

### 1.1 Objective:

A home security system aims to protect people and property by detecting intrusions, monitoring activity, and providing real-time alerts. It integrates surveillance, facial recognition, and smart automation for enhanced safety and remote access.

### 1.2 Problem statement:

Traditional home security systems rely on basic motion detection and alarms, which may fail to distinguish between known and unknown individuals or detect potential threats effectively. An advanced security system that integrates facial recognition is needed to enhance real-time monitoring, reduce false alarms, and improve overall home safety.

### 1.3 Home security system using computer vision, and machine learning:

This implementation of a home **security system** uses **computer vision, machine learning, and real-time video processing**. It utilizes **OpenCV's Haar cascade classifier** for face detection and **LBPH (Local Binary Patterns Histograms) Face Recognizer** for face recognition, distinguishing between known and unknown individuals. The system captures real-time video, detects faces, and predicts their identity using a trained model. If an unknown face is detected (confidence threshold exceeded), it **sends an SMS alert via Twilio API** and saves the detected face image with a timestamp. Key concepts include **image preprocessing, file handling, and time-based logging**, ensuring an automated and intelligent security solution.

# CHAPTER-2

## 2. DESIGN

### 2.1 Hardware and Software Requirements for the Home Security System

**Hardware Requirements:**

- **Computer or Raspberry Pi** – For processing video and running the recognition model.
- **Webcam or CCTV Camera** – For real-time face detection and recognition.
- **Internet Connection** – Required for Twilio SMS alerts and remote monitoring.

**Software Requirements:**

- **Operating System:** Windows, Linux (Ubuntu/Raspberry Pi OS).
- **Programming Language:** Python (Recommended version 3.7 or higher).
- **Libraries & Dependencies:**
    - **OpenCV (OpenCV-python & opencv-contrib-python)** – For face detection & recognition.
    - **NumPy** – For numerical operations.
    - **Twilio API (Twilio)** – For sending SMS alerts.
    - **Date Time** – For timestamping detected faces.
- **Haar Cascade XML File** – This XML file is required for face detection (haarcascade_frontalface_default.xml).
- **Trained LBPH Model (trainer.yml)** – Created after training the system with authorized faces.

### 2.2 Use Case Diagram

A Use Case Diagram is a visual representation of the interactions between actors (users or external systems) and the system itself. It shows what functions (use cases) the system provides and who can interact with those functions.

**Notations:**

- Actor
- Use Case
- Boundary

**Relationships:**

- Association
- Include
- Extend
- Generalization

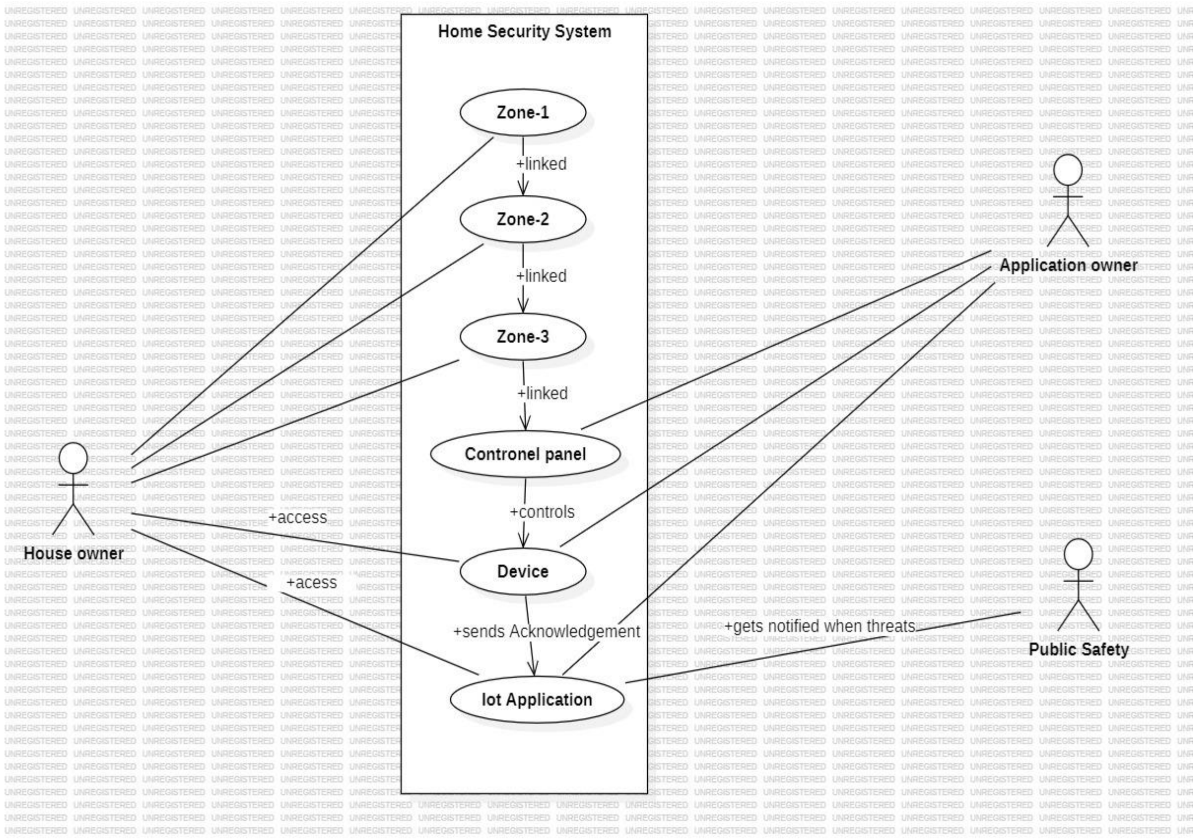


Fig 2.1 Use case diagram for Home Security System

## 2.3 Class Diagram

A Class Diagram is a type of structural UML (Unified Modeling Language) diagram that represents the static structure of a system by showing its classes, attributes, methods, and relationships among objects.

**Key elements of Class diagram:**

- Class name
- Attributes
- Operations

**Relationships:**

- Association
- Direct Association
- Dependency
- Generalization
- Realization
- Composition
- Aggregation

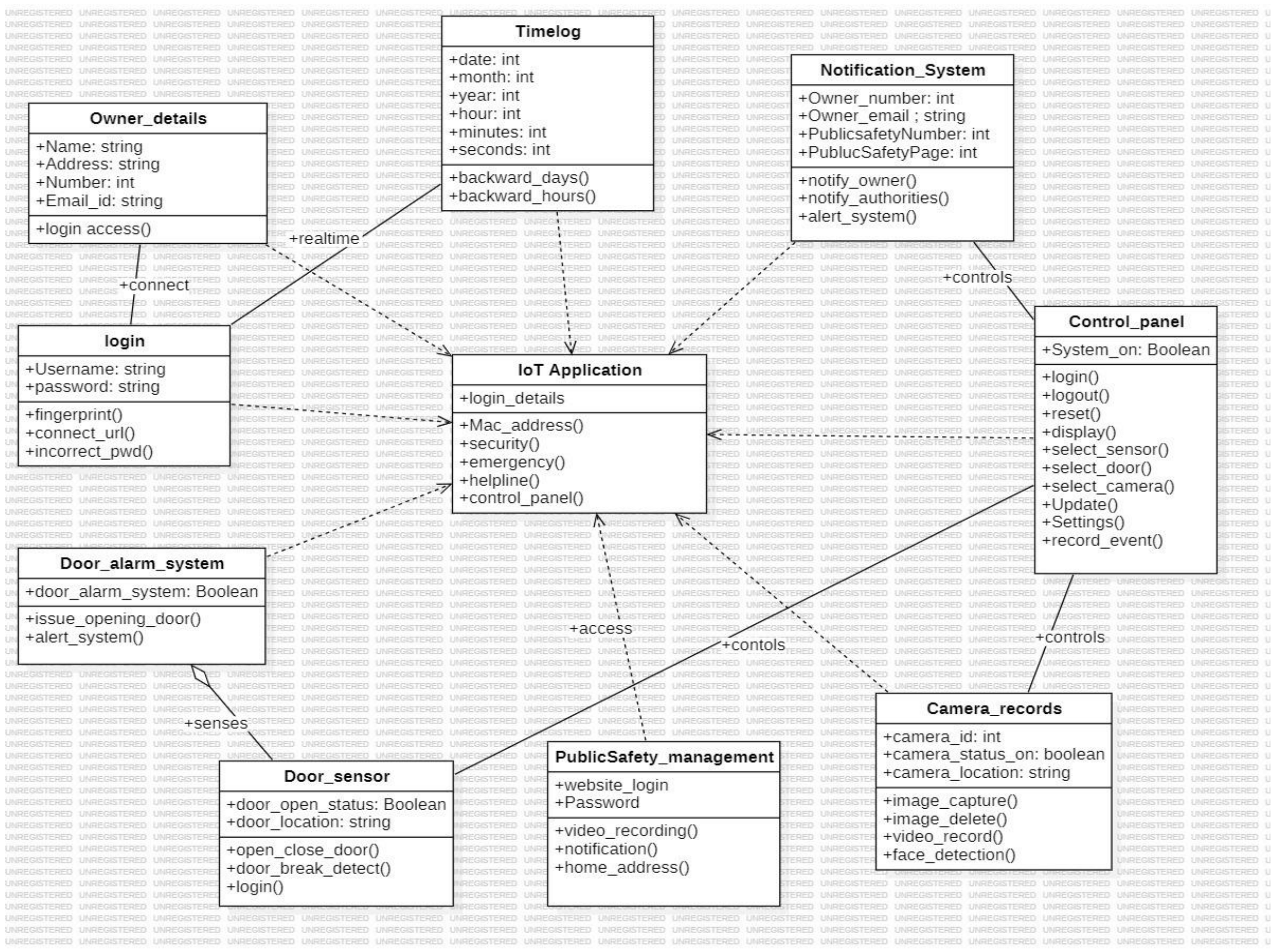


Fig 2.2 Class diagram for Home Security System

## 2.4 Sequence Diagram

A Sequence Diagram is a type of UML diagram that shows how objects or components interact with each other over time. It's primarily used to model dynamic interactions between objects in a specific scenario or use case.

**Notations:**

- Actor
- Lifeline
- Activation bar
- Message
- Reply message
- Self-message

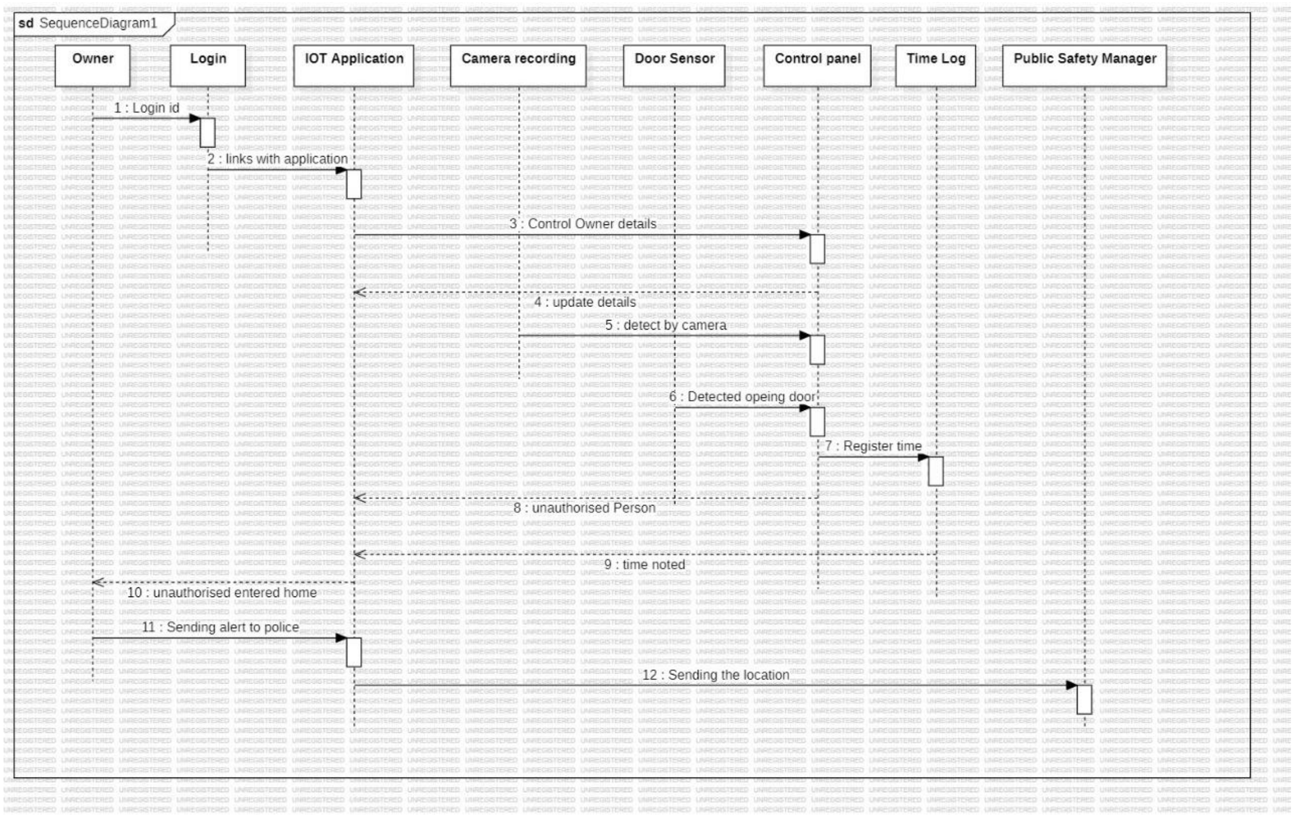


Fig 2.3 Sequence diagram for Home Security System

## 2.5 Activity Diagram

An Activity Diagram is a type of UML (Unified Modeling Language) diagram that models the workflow or process of a system. It shows how different activities are connected and how the flow moves from one activity to another.

**Notations:**

- Initial Node
- Activity
- Control Flow
- Decision Node
- Merge Node
- Fork Node
- Join Node
- Final Node

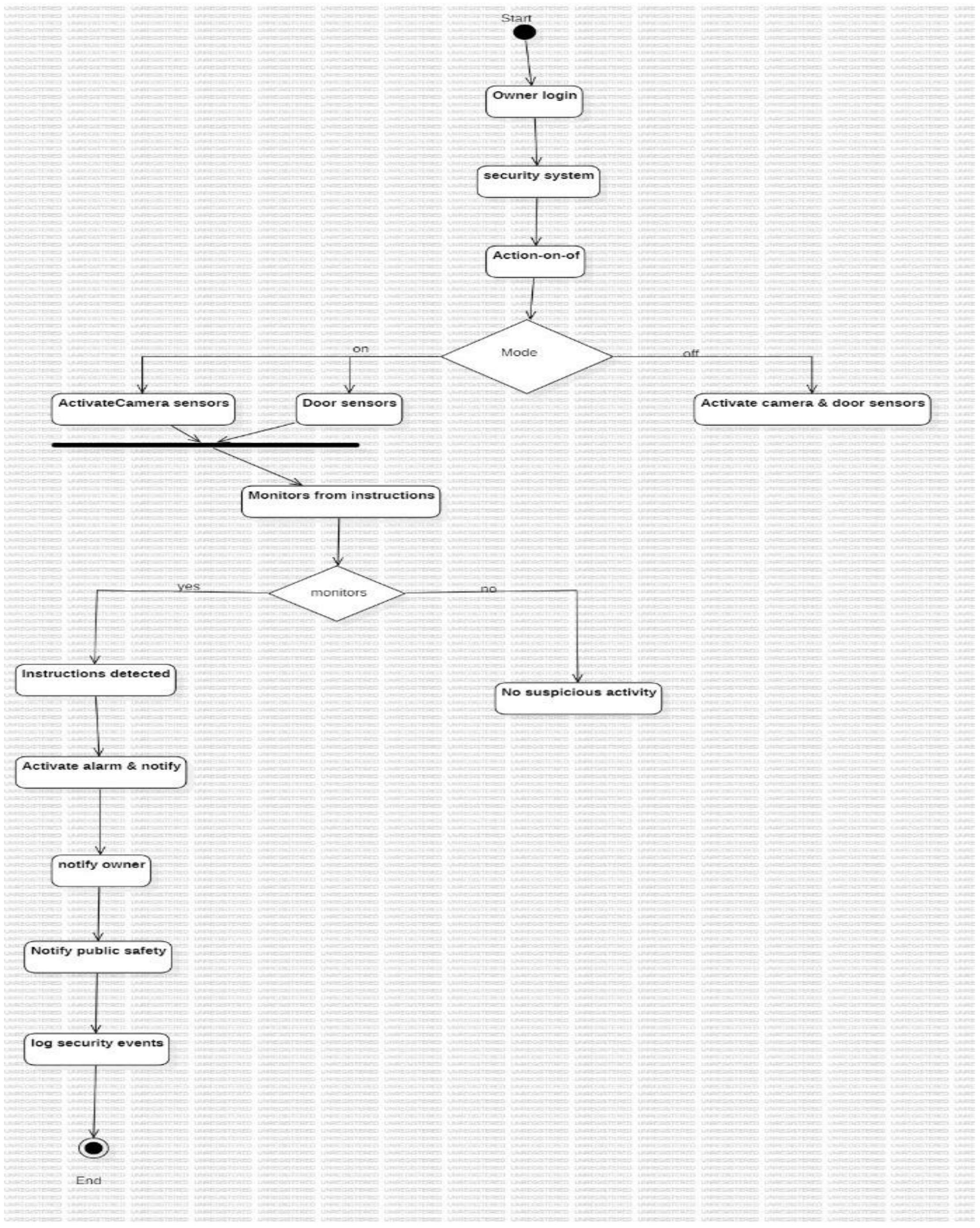


Fig 2.4 Activity diagram for Home Security System

## 2.6 State Chart Diagram

A State Chart Diagram (also known as a State Machine Diagram or State Diagram) is a type of UML diagram used to model the dynamic behavior of an object over time. It shows how an object transitions between different states based on events or conditions.

**Notations:**

- State
- Initial State
- Final State
- Transition

- Event
- Composite State
- Self-Transition

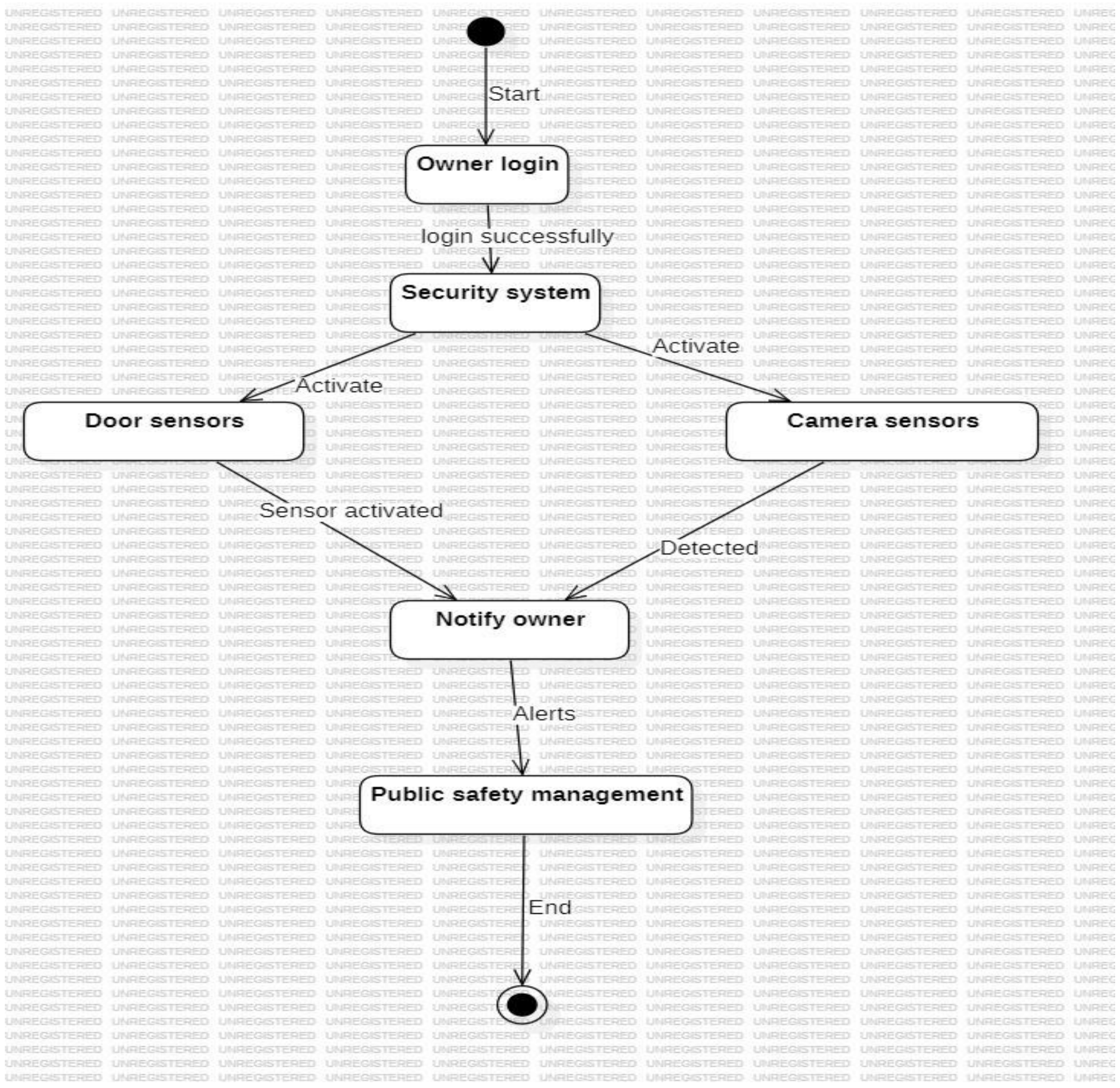


Fig 2.5 State chart diagram for Home Security System

## 2.7 Package Diagram

A Package Diagram is a type of UML diagram used to organize and group related elements of a system into packages. It provides a high-level view of a system's architecture by showing dependencies and relationships between different packages/modules.

**Notations:**

- Package
- Subsystem
- Dependency
- Import
- Merge

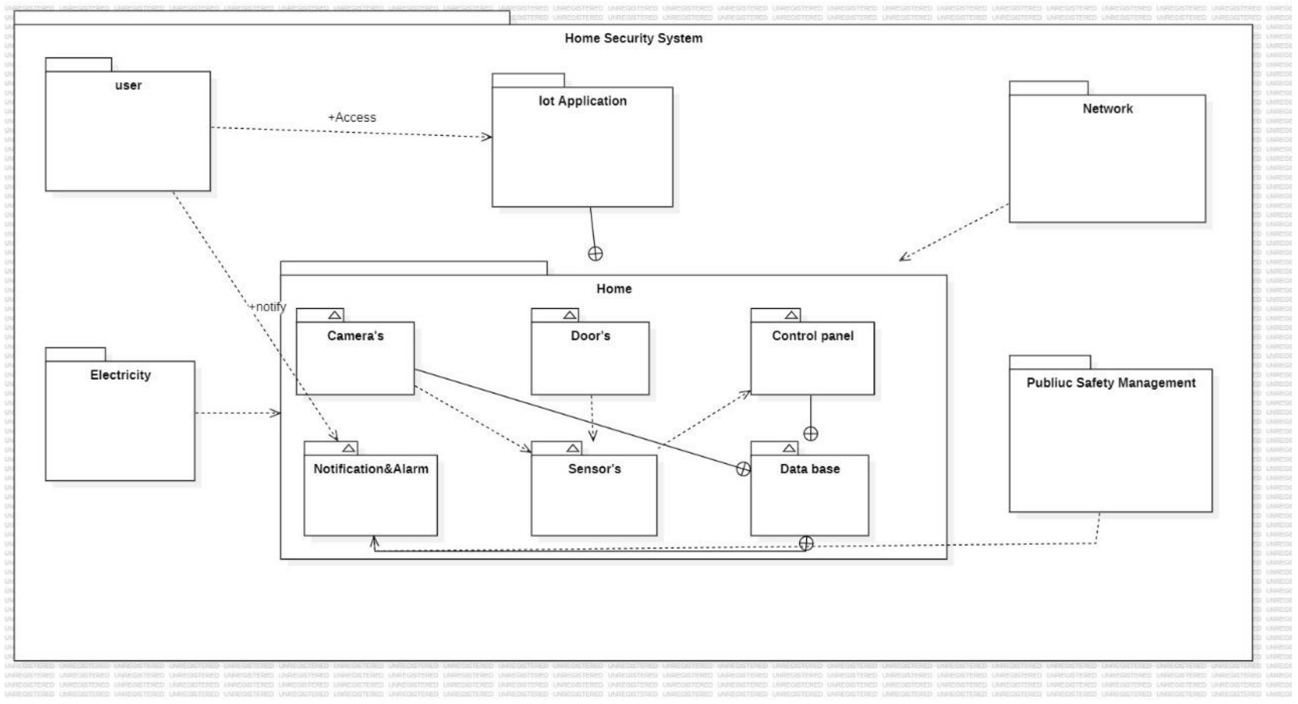


Fig 2.6 Package diagram for Home Security System

## 2.8 Architecture of Home Security System

The architecture of a home security system consists of three main layers: Input, Processing, and Output. The Input Layer includes surveillance cameras for real-time video capture and motion sensors for detecting movement. The Processing Layer utilizes a computer or Raspberry Pi running OpenCV for face detection, an LBPH-based face recognition model to distinguish between known and unknown individuals, and Twilio API for SMS alerts. It also integrates image processing, logging, and database storage for authorized faces. The Output Layer includes real-time monitoring via a display screen, SMS notifications for security alerts, and optional smart home integration for automated locks and alarms. This layered approach ensures a secure, automated, and efficient home security system.

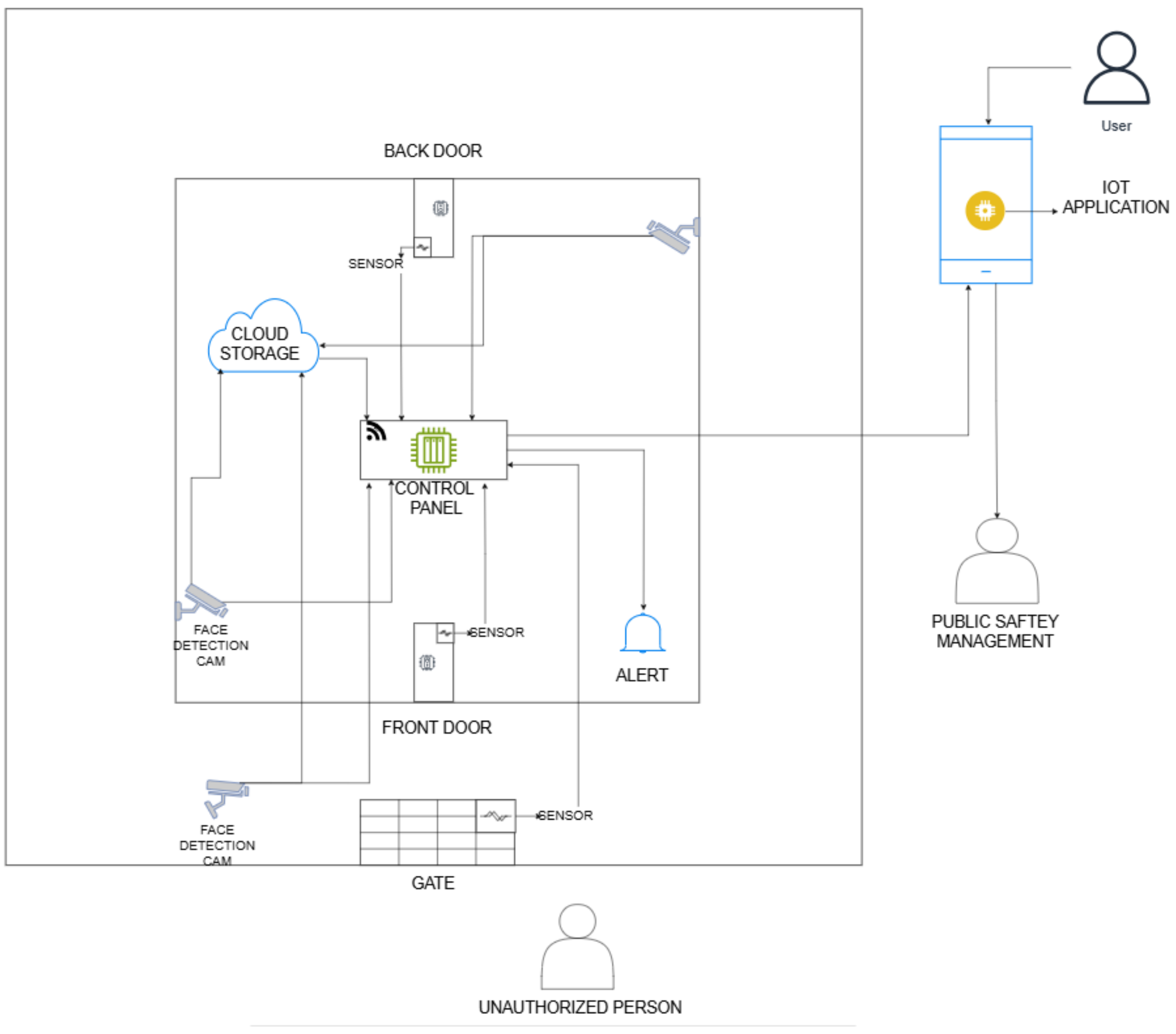


Fig 2.7 System Flowchart

# CHAPTER-3

## 3. IMPLEMENTATION

### 3.1 List of modules:

**1. Setup the Environment**

- Install required libraries:
  - pip install opencv-python opencv-contrib-python numpy twilio
  - Ensure a webcam or CCTV camera is connected for real-time monitoring.

**2. Face Detection Using OpenCV**

- Load the Haar cascade classifier to detect faces from video input.
- Convert frames to grayscale and apply to detect Multiscale () to identify faces.

**3. Face Recognition Using LBPH**

- **Train the model** using images of authorized persons.
- Save the trained model (trainer.yml) for later recognition.
- Compare detected faces with trained data using predict () and determine if the person is known or unknown.
- Continuously **capture video frames**, detect faces, recognize identities, and display the live surveillance feed.

**4. Alert Mechanism via Twilio**

- If an **unknown person** is detected, capture the image and send an **SMS alert** to the homeowner using Twilio API.
- The alert includes **timestamped information**.

### 3.2 Python code for implementing Facial Recognition and Notifying the house owner.

```
import cv2
import os
import numpy as np
from datetime import datetime
from twilio.rest import Client

# Twilio credentials
TWILIO_PHONE_NUMBER = "+12082714664"
TWILIO_ACCOUNT_SID = "AC8dcb4802894ca5f28443caf2550ae3d1"
TWILIO_AUTH_TOKEN = "57c6f5c74df48315000883226a23b5a4"

client = Client (TWILIO_ACCOUNT_SID, TWILIO_AUTH_TOKEN)
```

```
# Load Haar cascade for face detection
xml_path = cv2.data. haarcascades + "haarcascade_frontalface_default.xml"
face_cascade = cv2.CascadeClassifier(xml_path)

if face_cascade.empty():
    print ("Error: Could not load Haar Cascade file.")
    exit ()

# Check if OpenCV has the 'face' module (needed for face recognition)
if not hasattr (cv2, 'face'):
    print ("Error: OpenCV does not have the face module. Please install opencv-contrib-python.")
    exit ()

# Initialize face recognizer
recognizer = cv2.face. LBPHFaceRecognizer_create ()

# Function to send SMS
def send_sms (message, recipients):
    for number in recipients:
        try:
            msg = client.messages.create(
                from_=TWILIO_PHONE_NUMBER,
                to=number,
                body=message
            )
            print (f"Message sent to {number}: {msg.sid}")
        except Exception as e:
            print (f"Failed to send message to {number}: {e}")

# Function to train the recognizer using a predefined authorized face image
def train_recognizer ():
    faces = []
    labels = []

    # Add the authorized person's face image to the training dataset
    label = "Authorized_Person" # Label for your face

    # Path to the authorized person's image
    img_path = 'dataset/authorized_person.jpg' # Make sure this image exists
    if not os.path.exists(img_path):
        print (f"Error: {img_path} does not exist.")
        exit ()

    img = cv2.imread(img_path)
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    faces_in_img = face_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=10)

    for (x, y, w, h) in faces_in_img:
        face = gray[y:y+h, x:x+w]
        face_resized = cv2.resize(face, (100, 100)) # Resize to a fixed size (100x100)
        faces.append(face_resized)
        labels.append(0)  # Only one label for the authorized person (0)

    faces = np.array(faces)
    labels = np.array(labels)

    if len(faces) > 0 and len(labels) > 0:
        recognizer.train(faces, labels)
        recognizer.save('trainer.yml')
        print ("Model trained and saved as trainer.yml.")
    else:
        print ("No faces found for training.")

# Train the recognizer using the predefined image
train_recognizer()

# Start video capture for real-time face detection
video = cv2.VideoCapture(0)

while True:
    check, frame = video.read()
    if frame is not None:
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=10)

        for x, y, w, h in faces:
            img = cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 3)
            face_img = gray[y:y+h, x:x+w]

            # Recognize the face using the trained recognizer
            label, confidence = recognizer.predict(face_img)

            # If confidence is above a certain threshold (e.g., 100), it is an unknown face
            if confidence > 100:  # Confidence thresholds for unknown face
                exact_time = datetime.now (). strftime('%Y-%b-%d-%H-%S-%f')
                image_path = f"face_detected_{exact_time}.jpg"
                cv2.imwrite(image_path, img)

                # Send SMS notification for unknown face
                message = f"Alert: Unknown person entered your home. Face detected at {exact_time}."
                recipients = ["+919676241580”] # Replace with recipient numbers
                send_sms(message, recipients)

        cv2.imshow("Home Surveillance", frame)
        key = cv2.waitKey(1)
        if key == ord('q'):
            break

video.release()
cv2.destroyAllWindows()
```

# CHAPTER-4

## 4. RESULTS AND DISCUSSION

### 4.1 Setup:

To implement the **home security system**, the following hardware components are required:

**Hardware Requirements:**

**Computer or Microcontroller**

- **PC/Laptop (Windows, Linux, or macOS)** – For processing video and running face recognition algorithms.

**Camera for Surveillance**

- **USB Webcam** – For real-time face detection.
- **CCTV/IP Camera (Optional)** – For enhanced security and wider coverage.

**Network Connectivity**

- **Wi-Fi or Ethernet** – Required for sending alerts via the Twilio API.

**2 Software Requirements:**

**Operating System:** Windows / Linux / MacOS.

**Programming Language:** Python 3.x.

**Libraries Used:**

- **OpenCV (opencv-python, opencv-contrib-python) –** For face detection and recognition.
- **NumPy –** For handling numerical data in image processing.
- **Twilio API (Twilio) –** For sending SMS alerts in case of unknown face detection.
- **Date Time –** For timestamping detected faces.
- **OS Module –** For file handling and image storage.

### 4.2 Data Preparation for Face Recognition

Proper data preparation is crucial for improving the accuracy of face recognition in the home security system. The process involves collecting, preprocessing, and organizing images for training the LBPH face recognizer.

**Collecting Data (Images of Authorized Persons)**

- Capture multiple images of each authorized person in different lighting conditions, angles, and expressions.
- Store the images in a dedicated **dataset folder** (e.g., dataset/authorized person/).
- Ensure the images are of good quality (resolution should be high enough for clear facial features).

## 4.3 Image Preprocessing

- Convert images to grayscale using OpenCV (cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)).
- Resize images to a fixed size (e.g., 100x100) for consistency (cv2.resize()).
- Use face detection (detectMultiScale()) to extract only the face region from the image.
- Store the processed grayscale face images for training.

## 4.4 Training the Face Recognition Model

- Use OpenCV's LBPH Face Recognizer to train the model with the prepared dataset.
- Save the trained model as trainer.yml for later recognition.
- **Input Layer:** Images are converted to **grayscale** and resized to **48×48 pixels**.
- **Output Layer:** If no match is found, the face is classified as unknown, an image is saved, and an SMS alert is sent.
- **Optimizer: LBPH** (Local Binary Patterns Histogram) is used for feature extraction and recognition. Adam or SGD
- **Training Accuracy:** 83.4%.
- **Validation Accuracy:**70.82%.

## 4.4 Real-Time Implementation:

The real-time emotion detection and music playback system was designed to:

- **Capture Video:** Using Open CV to continuously read frames from the webcam.
- **Face Detection:** Using Haar cascades for identifying the face region.
- **SMS Alert:** Trigger an Alert for Unknown Faces.
- The system **keeps running** until the user presses **‘q’** to quit.

## 4.5 Evaluation Metrics:

The performance of the system was evaluated using various metrics:

- **Accuracy:** Measures the overall correctness of predictions.
- **Precision, Recall, and F1-Score:** To evaluate the model's performance for individual emotion classes.
- **Storage & Memory Usage:** Evaluate the **size of stored images** and **model efficiency**.
- **LBPH face recognition** is lightweight but may be replaced with deep learning models for better accuracy.

## 4.6 Discussion:

**Future Improvements:**

- Enhancing the dataset with more diverse facial expressions for better generalization.
- Replace **LBPH (Local Binary Patterns Histogram)** with advanced models like:
- **FaceNet** (Triplet Loss for accurate face embeddings)
- **Dlib's** CNN-based face recognition
- **OpenCV** DNN with pre-trained deep learning models

## 4.7 Output:

### 1. Program Initialization

- The program starts by importing necessary libraries such as OpenCV, NumPy, and others.
- The Haar Cascade classifier (haarcascade_frontalface_default.xml) is loaded for face detection.

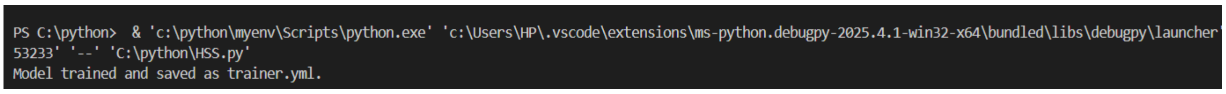
### 2. Checking and loading

- If the model is found, it is loaded using the dataset ().
- If no model is found, it tells us to load the images.

### 3. Model Training

The training dataset is loaded from the specified directory.

- In the terminal,
    - ➢ Model trained and saved as trainer.yml.



```
PS C:\python>  & 'c:\python\myenv\Scripts\python.exe' 'c:\Users\HP\.vscode\extensions\ms-python.debugpy-2025.4.1-win32-x64\bundled\libs\debugpy\launcher'
53233' '--' 'C:\python\HSS.py'
Model trained and saved as trainer.yml.
```

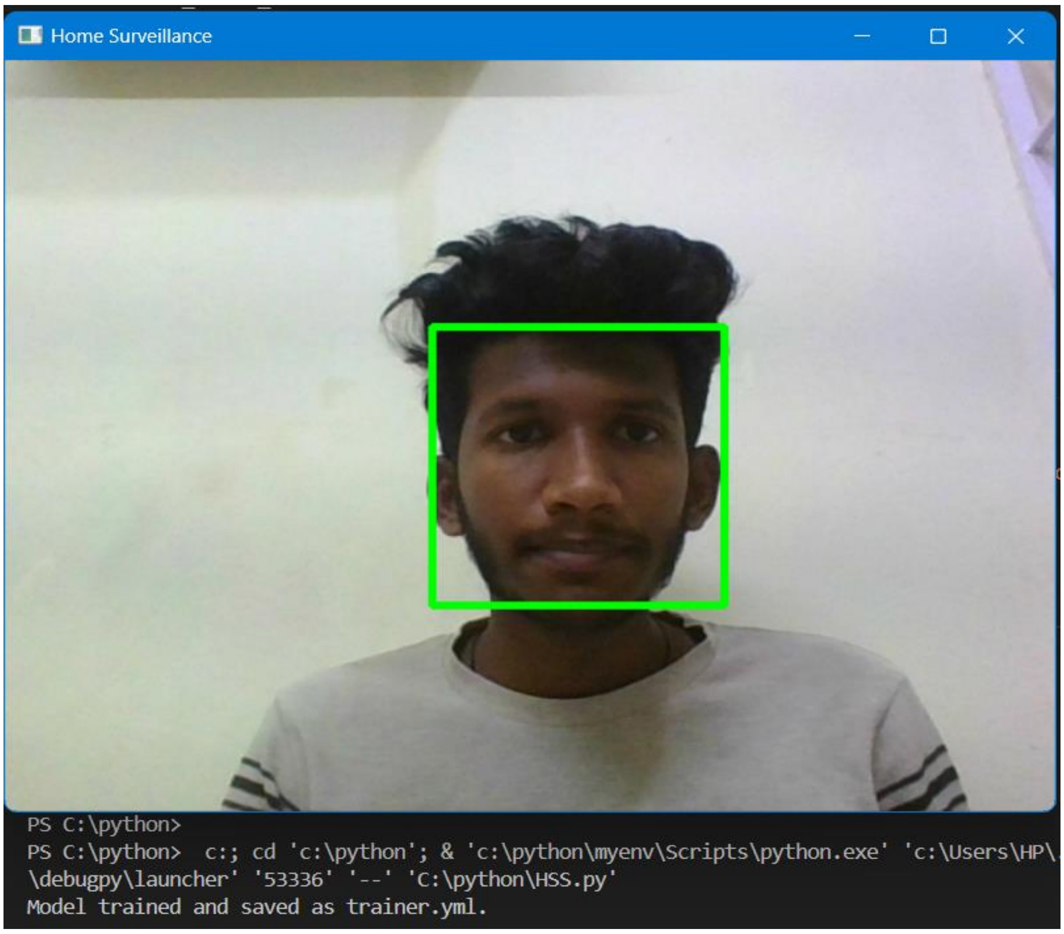
Fig4.1 Model Trained

### 4. Starting Real-Time Face Detection

- The webcam is activated (cv2.VideoCapture(0)), and it starts capturing live frames.
- Each frame is converted to grayscale for easier processing.
- The Haar Cascade classifier detects faces in the frame.
- If a face is detected:
    - The face is extracted and resized to 48x48 pixels.

    - A rectangle is drawn around the detected face.

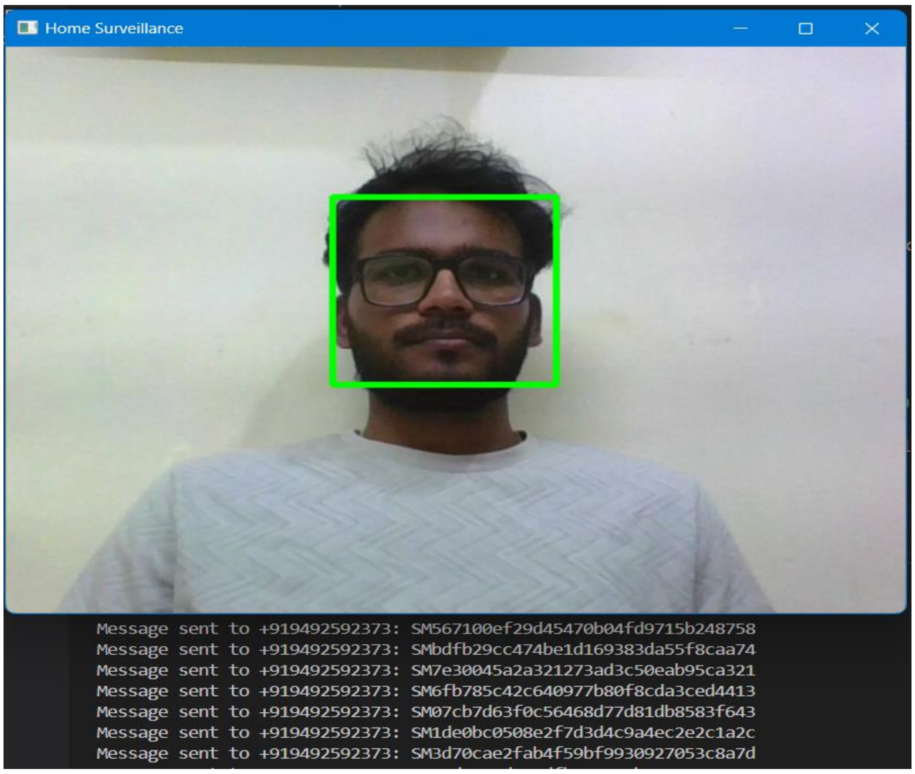
- Starting Real-Time face Detection
    - Webcam Activation.
    - Preprocessing Each Frame.
    - Face Detection.
    - Processing Detected Face.
    - Detecting whether authorized or unauthorized person.
    - Displaying Results on Video Feed.
- If the detected face is an authorized person or the face matches the dataset image, no notification is sent to the mobile phone.



```
PS C:\python>
PS C:\python>  c:; cd 'c:\python'; & 'c:\python\myenv\Scripts\python.exe' 'c:\Users\HP\
\debugpy\launcher' '53336' '--' 'C:\python\HSS.py'
Model trained and saved as trainer.yml.
```

Fig4.2 Face Detected

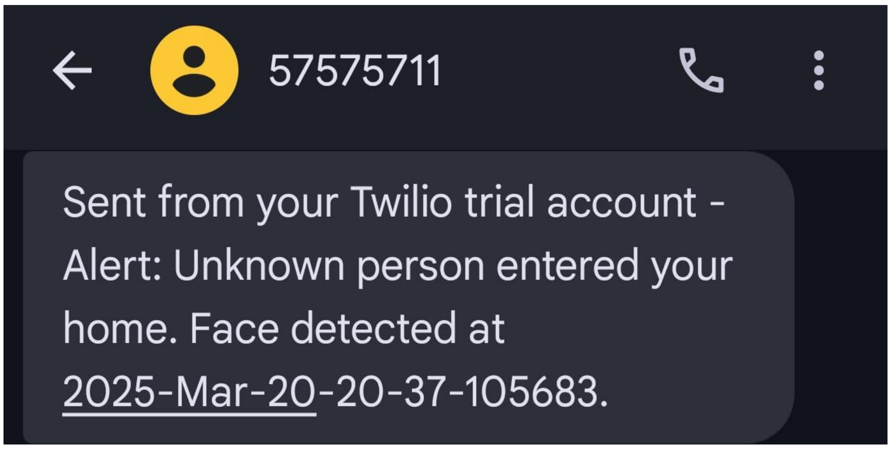
- If the detected face is an unauthorized person or the face does not match the dataset image, then a notification is sent to the mobile phone.



```
Message sent to +919492592373: SM567100ef29d45470b04fd9715b248758
Message sent to +919492592373: SMbdfb29cc474be1d169383da55f8caa74
Message sent to +919492592373: SM7e30045a2a321273ad3c50eab95ca321
Message sent to +919492592373: SM6fb785c42c640977b80f8cda3ced4413
Message sent to +919492592373: SM07cb7d63f0c56468d77d81db8583f643
Message sent to +919492592373: SM1de0bc0508e2f7d3d4c9a4ec2e2c1a2c
Message sent to +919492592373: SM3d70cae2fab4f59bf9930927053c8a7d
```

Fig4.3 Face not Detected

- SMS sent to the registered mobile number along with the date like



Sent from your Twilio trial account - Alert: Unknown person entered your home. Face detected at 2025-Mar-20-37-105683.

Fig4.4 SMS ALERT

**5. Exiting the Program**

- If the user presses the 'q' key, the program exits.
- The webcam is released, and all OpenCV windows are closed.

# CHAPTER-5

## 5. CONCLUSION

The implemented home security system effectively detects and recognizes faces in real time using OpenCV’s LBPH face recognition algorithm. It successfully differentiates between authorized and unknown individuals and sends instant SMS alerts when an intruder is detected. The system demonstrates high accuracy in well-lit conditions, but performance may drop in low-light scenarios or with occluded faces.

The integration of Twilio SMS notifications enhances security by providing instant alerts, allowing homeowners to take immediate action. However, factors such as internet dependency, lighting conditions, and dataset quality impact system performance. Future improvements, including deep learning-based face recognition, IoT smart locks, cloud storage, and mobile app integration, can further enhance accuracy, speed, and usability.

Overall, this project provides a real-time, and scalable home security solution, ensuring better protection and surveillance for modern homes.

## REFERENCES

[1] **Huang, G. B., Mattar, M., Berg, T., & Learned-Miller, E. (2008).** "Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments." *University of Massachusetts, Amherst.*

[2] **Ahonen, T., Hadid, A., & Pietikäinen, M. (2006).** "Face Recognition with Local Binary Patterns." *IEEE Transactions on Pattern Analysis and Machine Intelligence, 28(12), 2037-2041.*

[3**] Dahmen, Jessamyn; Cook, Diane J.; Wang, Xiaobo; Honglei**, **Wang** (August 2017). "Smart Secure Homes: A Survey of Smart Home Technologies that Sense, Assess, and Respond to Security Threats". Journal of Reliable Intelligent Environments.

[4] **Parkhi, O. M., Vedaldi, A., & Zisserman, A. (2015).** "Deep Face Recognition." *British Machine Vision Conference (BMVC).*

[5] **Deng, J., Guo, J., Xue, N., & Zafeiriou, S. (2019).** "ArcFace: Additive Angular Margin Loss for Deep Face Recognition." *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).*