

# Face Recognition-LogBook

A Project Report

Submitted in the partial fulfillment of the requirements for the

award of the degree of

Bachelor of Technology in

Department of AI&DS

By

2200080231- Kasireddy Sathvik Reddy

under the supervision of

**Dr. Sripath Roy**

****

**Department of Artificial Intelligence** **&** **Data Science**

K L E F, Green Fields,

Vaddeswaram- 522502, Guntur(Dist), Andhra Pradesh, India.

April, 2025

Certificate

This is to certify that the project report entitled “Face Recognition-LogBook” submitted by Kasireddy Sathvik Reddy, bearing the registration number 2200080231, in partial fulfillment of the requirements for the award of the B.Tech degree (III Year, Even Semester) in **Artificial Intelligence** & **Data Science** at K L University, is a record of bonafide work carried out under our guidance and supervision. The results presented in this report have not been copied from any other department, university, or institute. award of B. Tech III Even Semester in AI&DS to the K L University is a record of bonafide work carried out under our guidance and supervision. The results embodied in this report have not been copied from any other departments/ University/Institute.

**Signature of the Supervisor**

**Dr. Sripath Roy**

# Table of Contents

|  |  |  |
| --- | --- | --- |
| **S. No** | **Contents** | **Pg. no** |
| **1.** | **Abstract** | **4** |
| **2.** | **Introduction** | **5-6** |
| **3.** | **Problem Statement** | **7** |
| **5.** | Literature Review | **8-9** |
| **6.** | **Objective** | **10** |
| **5.** | **Technologies Used** | **11** |
| **6.** | **Implementation** | **12-25** |
| **8.** | **Results and Discussions** | **26-33** |
| **9.** | **Conclusion and Future Work** | **34** |
| **10.** | **References** | **35** |

**Abstract**

Face recognition is a computer-based method that identifies or verifies people using their facial features. It captures an image or video, finds a face in it, and compares it with stored data to recognize who it is. This technology is widely used today in places like smartphones, airports, and security systems. It’s fast, non-invasive, and doesn’t need physical contact. The purpose of this project is to create a basic face recognition system using Python and open-source libraries.

We used tools like OpenCV for capturing images, and the insightface library for detecting and comparing faces. The system takes a live video from a webcam and tries to match the face in the video with faces already stored in a database. If it finds a match, it displays the name of the person. This project helps us understand how artificial intelligence can identify people by learning their facial patterns. It’s a good introduction to how AI and machine learning are used in real-world applications.

# Introduction

The rapid integration of technology into everyday life has transformed the way we interact with systems, especially through advancements in artificial intelligence (AI). One particularly fascinating area of AI is facial recognition the ability of machines to identify or verify individuals using facial features. This technology has vast potential in domains such as security systems, real-time attendance tracking, personalized user experiences, and mobile authentication. My interest in this field began with a curiosity about how machines could "see" and recognize people, and it led me to explore the technical underpinnings of computer vision and machine learning.

My project focuses on developing a real-time facial recognition system using open-source tools like Python, InsightFace, and a standard webcam. The objective was to build a practical solution that can detect, recognize, and differentiate between known and unknown individuals. I started by collecting and preprocessing a dataset of face images, then used machine learning techniques to train a model capable of identifying faces under varied conditions such as different lighting, occlusions, and facial expressions. Through this hands-on experience, I gained deeper insights into image processing, facial feature embedding, and model inference, along with the ethical implications related to user privacy, consent, and data security.

This system can be applied in various real-world scenarios. For example, it can enhance security at workplace entry points, automate student or employee attendance, and serve as a secure login system without relying on passwords. Compared to traditional biometric systems like fingerprint scanners, facial recognition is contactless and more user-friendly, which is especially important in post-pandemic settings.

In reviewing existing systems, we found that many commercial solutions offer high accuracy but are often expensive, cloud-dependent, or closed-source. These limitations hinder accessibility for educational or small-scale use. Moreover, some previous models struggle with accuracy when faces are partially covered or when used in uncontrolled environments. Our system attempts to address these gaps by using lightweight models that are deployable offline and capable of handling occlusions and lighting variations with reasonable accuracy. However, we also recognize limitations such as a smaller training dataset, potential biases in recognition accuracy, and dependency on camera quality.

In addition to the technical learning, this project sparked important reflections on the social impact of AI technologies. As facial recognition becomes increasingly common in public and private sectors, it raises critical questions about surveillance, algorithmic bias, and the potential misuse of data. While our system was developed for educational and controlled environments, it serves as a microcosm of the larger discussions surrounding responsible AI deployment. We emphasized transparency in model training and data collection, ensuring that our approach remained ethically grounded.

Furthermore, this experience strengthened my interest in building AI systems that are not only functional but also fair and inclusive. It opened up new avenues for interdisciplinary exploration—connecting technology with fields like ethics, law, and social policy. I realized that being an AI developer today means more than just writing efficient code; it requires a deep sense of responsibility for the consequences that technology can have on individuals and society.

Looking ahead, this project lays the foundation for more advanced work in computer vision and responsible AI. With further research, I plan to explore topics such as bias mitigation in facial datasets, secure on-device facial recognition, and federated learning models to preserve user privacy. This journey has been more than a technical challenge—it has been a stepping stone toward becoming a mindful and innovative contributor in the AI community.

# Problem Statement

Face recognition systems are becoming more common in our daily lives, but they are not always perfect. One of the biggest challenges is accurately recognizing faces in real-world situations. The accuracy can drop if the lighting is poor, the person changes their hairstyle or wears glasses, or if the camera is not positioned well. These issues make it harder for the system to match faces correctly, especially when the face doesn’t fully match the stored image.

Another challenge is **occlusion**, which means when part of the face is covered. This could happen if someone wears a mask, covers their face with a hand, or wears a scarf or cap. When the face is not fully visible, the system struggles to identify the person, leading to errors or missed recognition. This is a serious issue in places like airports or secure buildings, where accurate recognition is important for safety.

In addition, deploying face recognition on **mobile phones** brings new problems. Phones have limited processing power, and camera quality can vary from one device to another. Also, people may move or tilt their phones, which makes it harder to keep the face in focus. These problems can affect how fast and reliably the system works. To make face recognition practical and useful in everyday settings, we must find ways to solve these issues and improve performance in all conditions

# Literature Review

Facial recognition has been a major area of research in computer vision for decades, with advancements accelerating due to deep learning. Early systems relied on geometric features and template matching but had limited accuracy under variable conditions. The emergence of convolutional neural networks (CNNs) and large-scale datasets significantly improved performance in unconstrained environments.

The **Eigenfaces** and **Fisherfaces** methods were among the first statistical approaches for facial recognition, using linear projection techniques to reduce facial image dimensionality. However, these methods were highly sensitive to lighting, pose, and occlusions. With the rise of deep learning, models like **DeepFace** (Facebook, 2014) and **FaceNet** (Google, 2015) introduced the concept of embedding faces into a high-dimensional feature space using CNNs, achieving significant improvements in face verification accuracy.

In more recent developments, **InsightFace**, based on **ArcFace loss**, has emerged as one of the state-of-the-art solutions for facial recognition. ArcFace optimizes intra-class compactness and inter-class variance, producing highly discriminative face embeddings. It outperforms many prior models on benchmark datasets such as LFW, MegaFace, and IJB-B, making it suitable for both verification and identification tasks. Importantly, InsightFace supports lightweight backbones like MobileFaceNet and EfficientNet for mobile deployment, aligning well with the needs of real-time applications.

Despite these advances, challenges remain. Studies such as [Wang et al., 2018] and [Zhao et al., 2019] have highlighted performance degradation under **occlusion**, pose variation, and poor lighting. Research has also shown that many models exhibit **bias** against certain demographic groups due to imbalanced training data. Furthermore, **cloud-based commercial systems** like Microsoft Azure Face API and Amazon Rekognition offer high accuracy but raise **concerns over data privacy and accessibility**, especially in educational or low-resource settings.

Our project builds upon these insights by leveraging InsightFace for its balance between accuracy and efficiency. By addressing practical challenges such as **occlusion handling**, **mobile deployment**, and **offline usability**, we aim to create a system that is more accessible while still being accurate and secure. In doing so, we contribute to the growing interest in **ethical, edge-friendly AI systems** that do not depend solely on cloud infrastructure and respect user privacy.

Facial recognition technology has evolved significantly, transitioning from early statistical methods to modern deep learning-based systems capable of real-time and mobile deployment. Understanding this evolution is crucial to situating our project within the broader research and development landscape.

# Objectives

* The system should accurately recognize the person based on facial features, ensuring high precision even when multiple faces are present or under varied lighting conditions.
* Upon successful recognition, the system must store and display the person’s name and ID on the screen and also send the same data to a connected terminal for record-keeping.
* The model should be capable of identifying faces even when partially covered, like with a mask, scarf, or glasses, ensuring occlusion handling without affecting recognition performance.
* The system should be optimized to increase model performance by reducing latency and improving accuracy through techniques like model pruning, quantization, or using faster architectures like MobileNet.
* The model must be lightweight and efficient enough for mobile implementation, allowing it to run smoothly on smartphones without requiring heavy computational resources or external servers.
* It should support recognition of side or angled faces to enhance usability in real-world conditions where users may not always face the camera directly, improving reliability.
* All login activity and recognition results should be securely stored in a cloud database like Supabase, enabling easy access, user tracking, and data synchronization across devices.
* The system should include a real-time notification or alert feature, which notifies administrators when a face is recognized or when unauthorized access is attempted.

**Technologies Used**

**Python** – The main language used to build and run the face recognition system due to its simplicity and rich libraries..

**InsightFace** – A powerful deep learning model used for highly accurate face detection and recognition.

**NumPy** – Helps manage image data and perform fast numerical computations for face processing.

**Supabase** – A backend-as-a-service used to store and retrieve user login and identification data.

**OpenCV (cv2)** – Used for capturing images from the camera and handling real-time face tracking and visualization.

**VS Code** – The development environment used to write, test, and debug the entire project efficiently.

**Implementation**

The face recognition system is implemented using a webcam or mobile camera to capture real-time video, where each frame is processed to detect and recognize faces. It uses the InsightFace model for accurate feature extraction and matching, allowing the system to identify individuals correctly. Once a person is recognized, their name and ID are displayed both on the screen and in the terminal for confirmation. The code is written in Python using libraries like NumPy for computations and OpenCV for handling camera input and face detection. To handle occlusions like masks or partial face coverage, the model is trained to still recognize users effectively. The project also supports mobile deployment and side-face recognition, making it suitable for various real-world conditions. All login data and recognition history are securely stored in Supabase, a cloud database.

**My Contribution:**

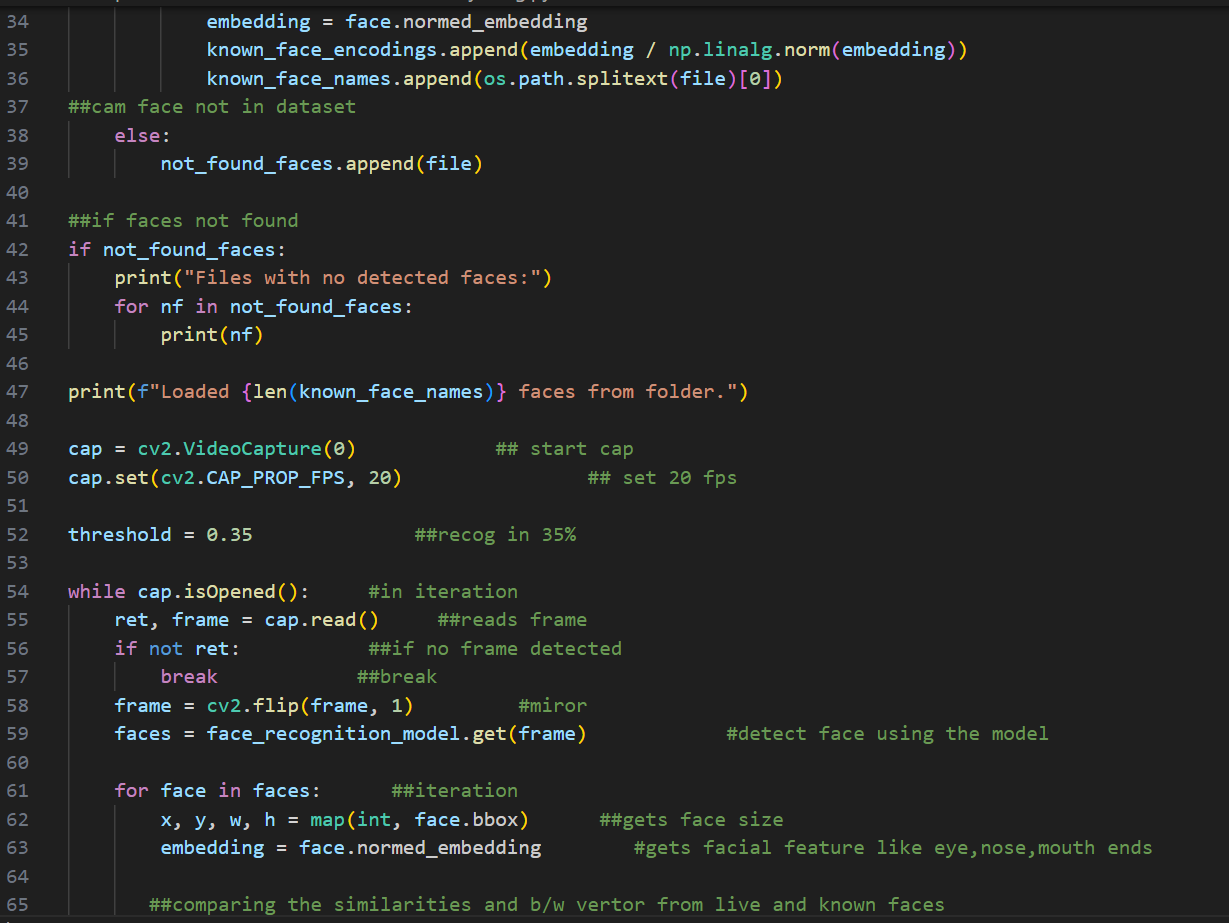
During the project, I actively contributed to researching suitable face recognition tools and was involved in identifying InsightFace as the core framework. I developed and executed all the core functionalities of the system, including the initial recognition module. I further enhanced it by integrating occlusion support, enabling the system to recognize faces even when partially covered. I also worked on adapting the solution for mobile devices, ensuring it could run effectively beyond desktops. Additionally, I implemented the connection to Supabase, allowing smooth storage and retrieval of user login data within the database.

**Codes:-**

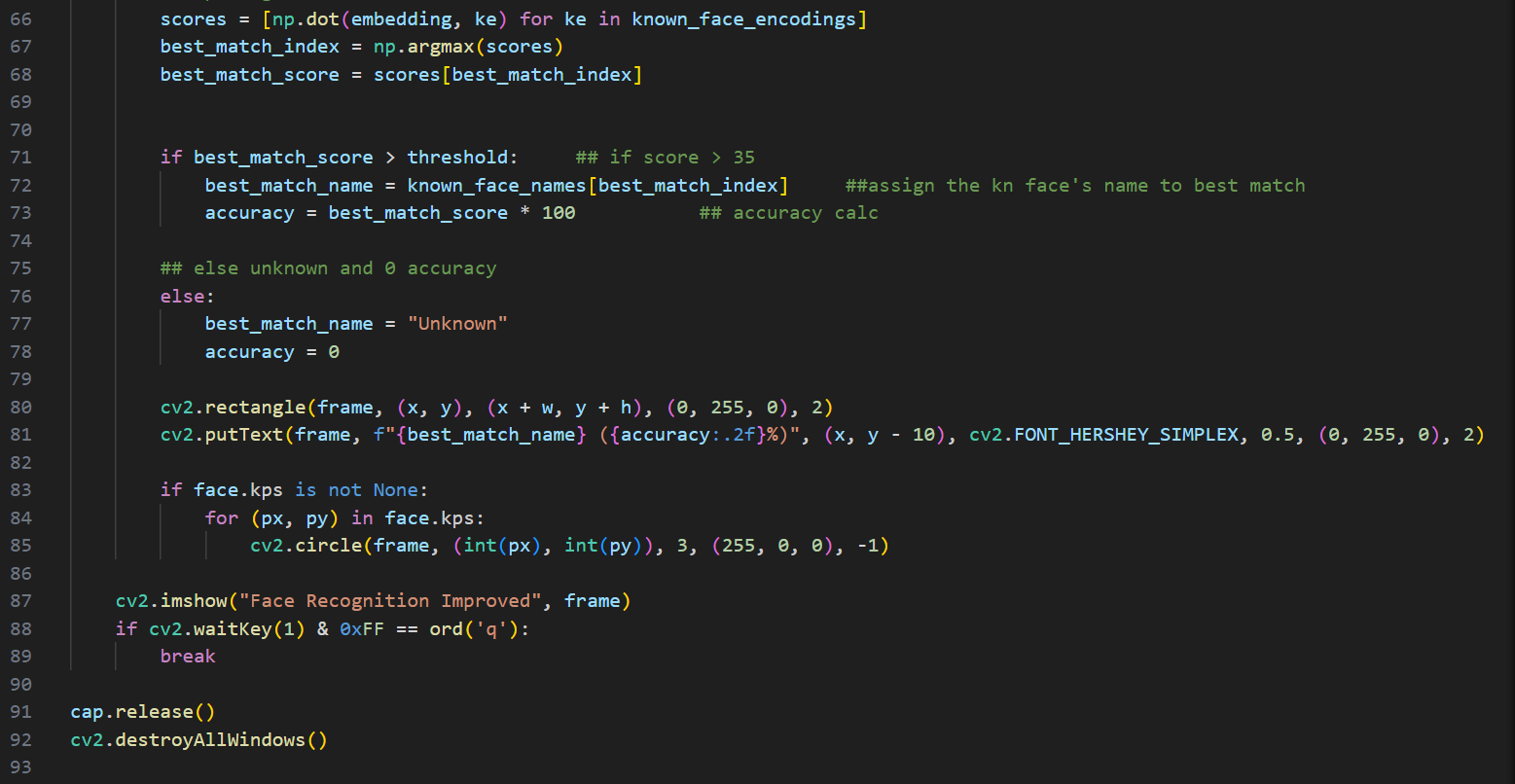
**Only-recognition**

****

**Fig:-1**

****

**Fig:-2**

****

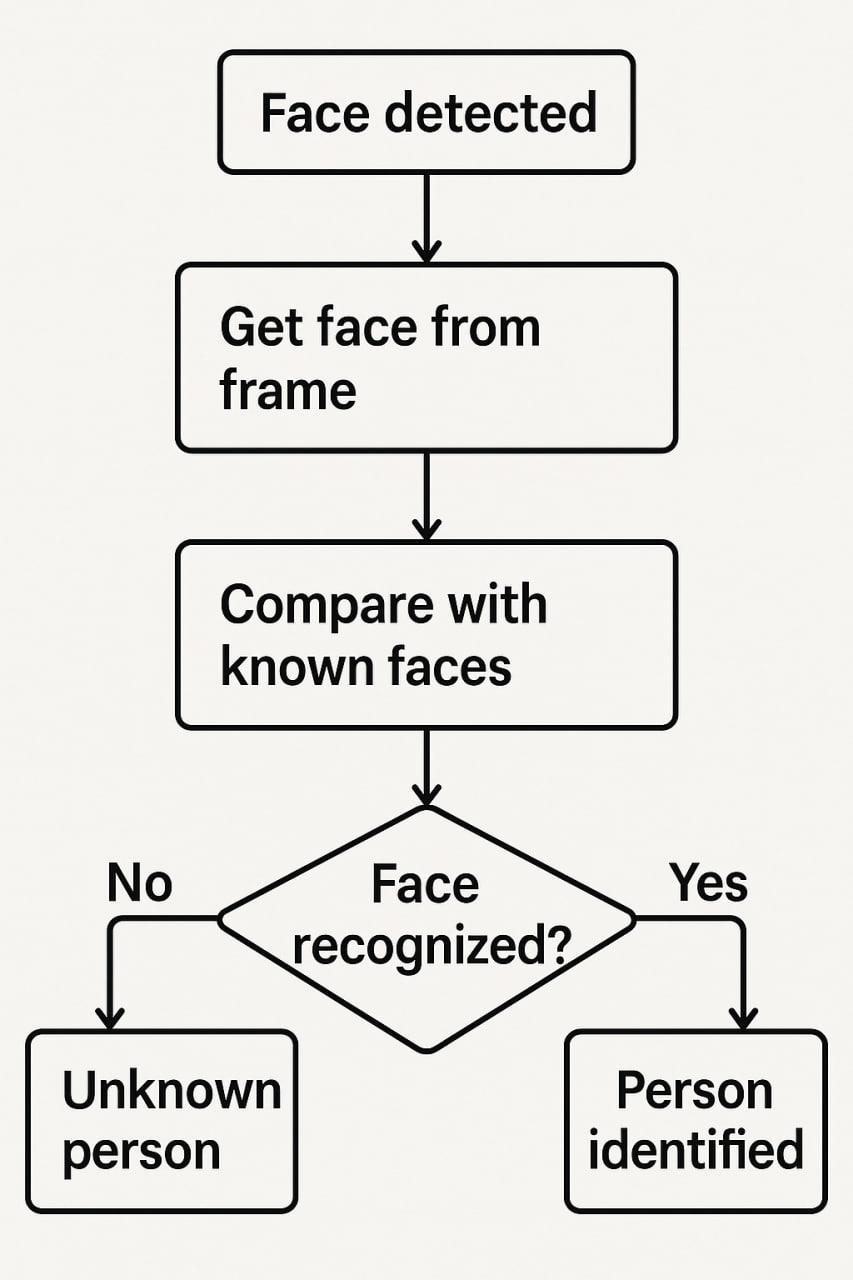
**Fig:-3**

**Some insights:-**

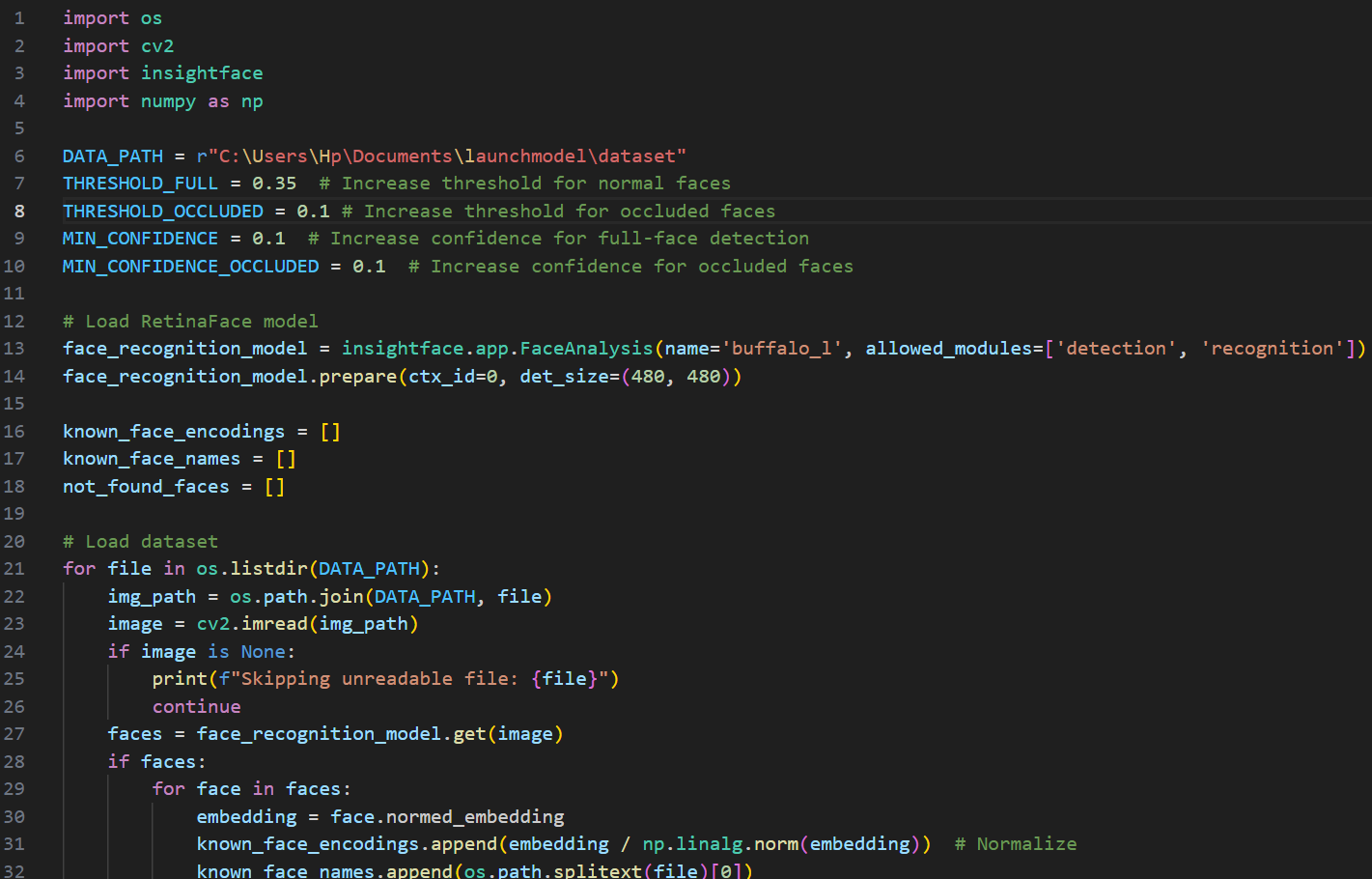
The recognition-only module utilizes **InsightFace's "buffalo\_l" model**, which combines face detection and recognition functionalities. The system begins by processing a dataset of images, where each image is analyzed to detect faces. When a face is detected, its **normalized embedding vector** is extracted. These embeddings represent unique facial characteristics and are used to identify individuals, much like a biometric fingerprint.

This code uses the InsightFace library to perform real-time face recognition. It loads images from a dataset, extracts face embeddings, and compares them to live webcam frames. The system detects faces, calculates similarity scores with stored embeddings, and identifies faces if the score surpasses a set threshold. The results are displayed in real-time with bounding boxes and accuracy percentages. If no match is found, the face is labeled as "Unknown."

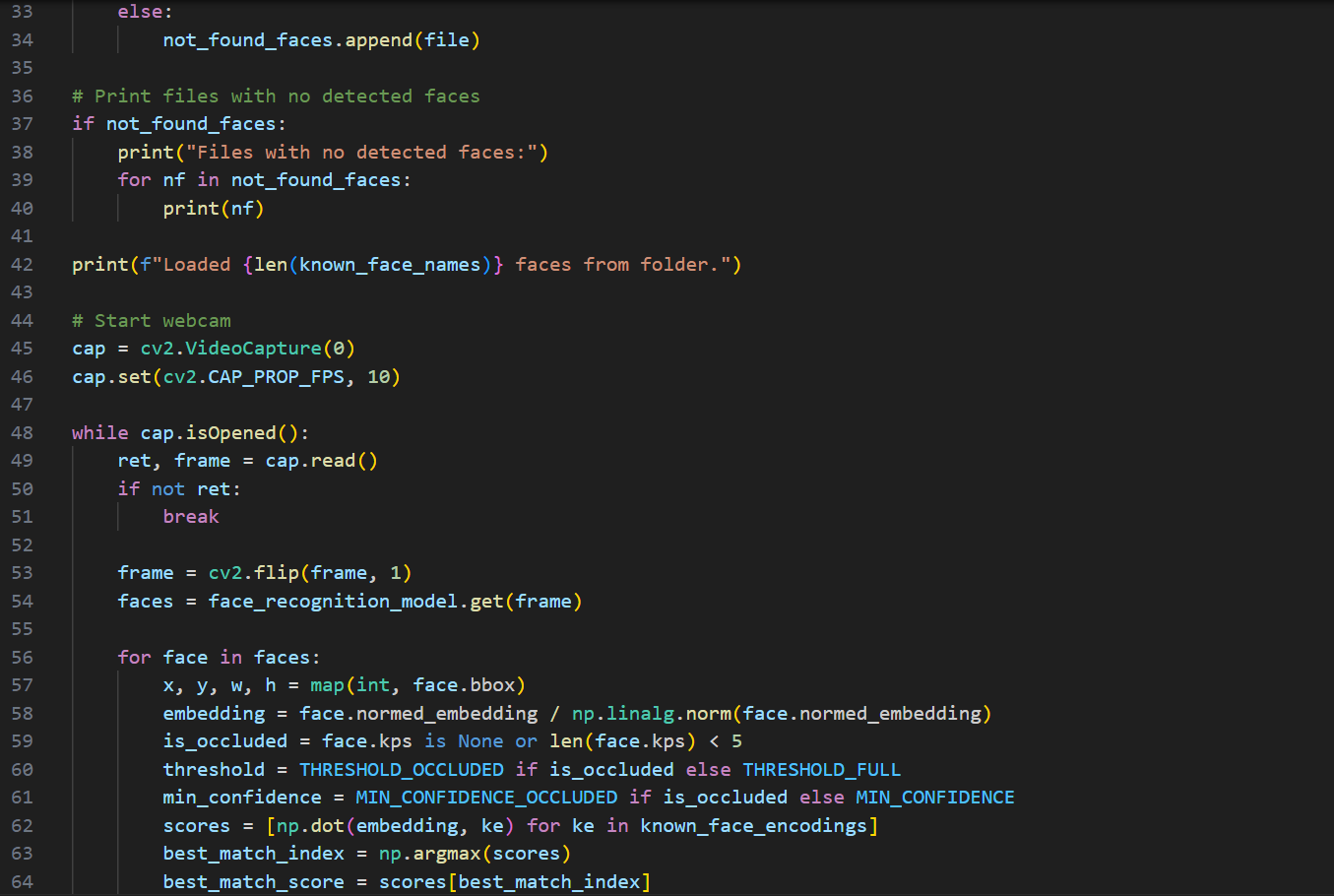
**WorkFlow Chart:-**



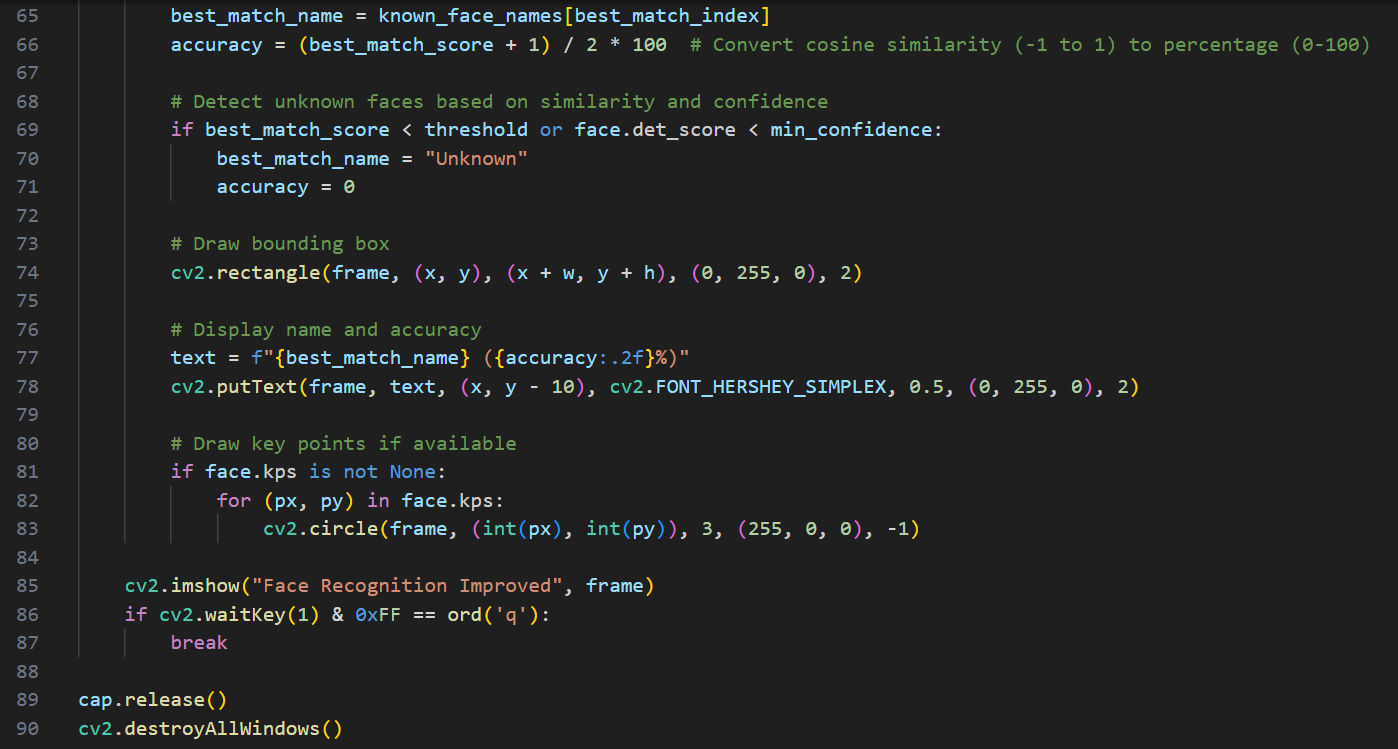
**Included-occlusion**

****

**Fig:-1**

****

**Fig:-2**

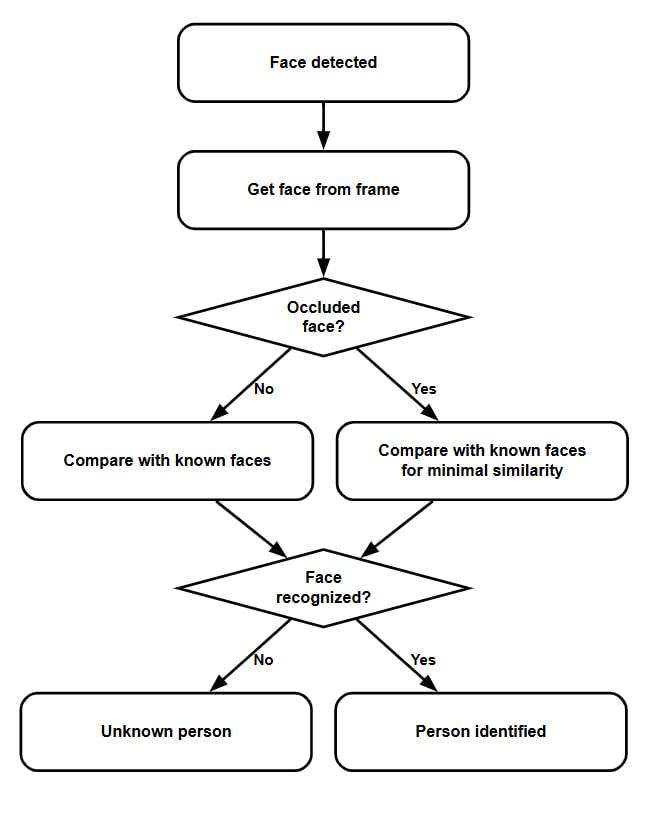
****

**Fig:-3**

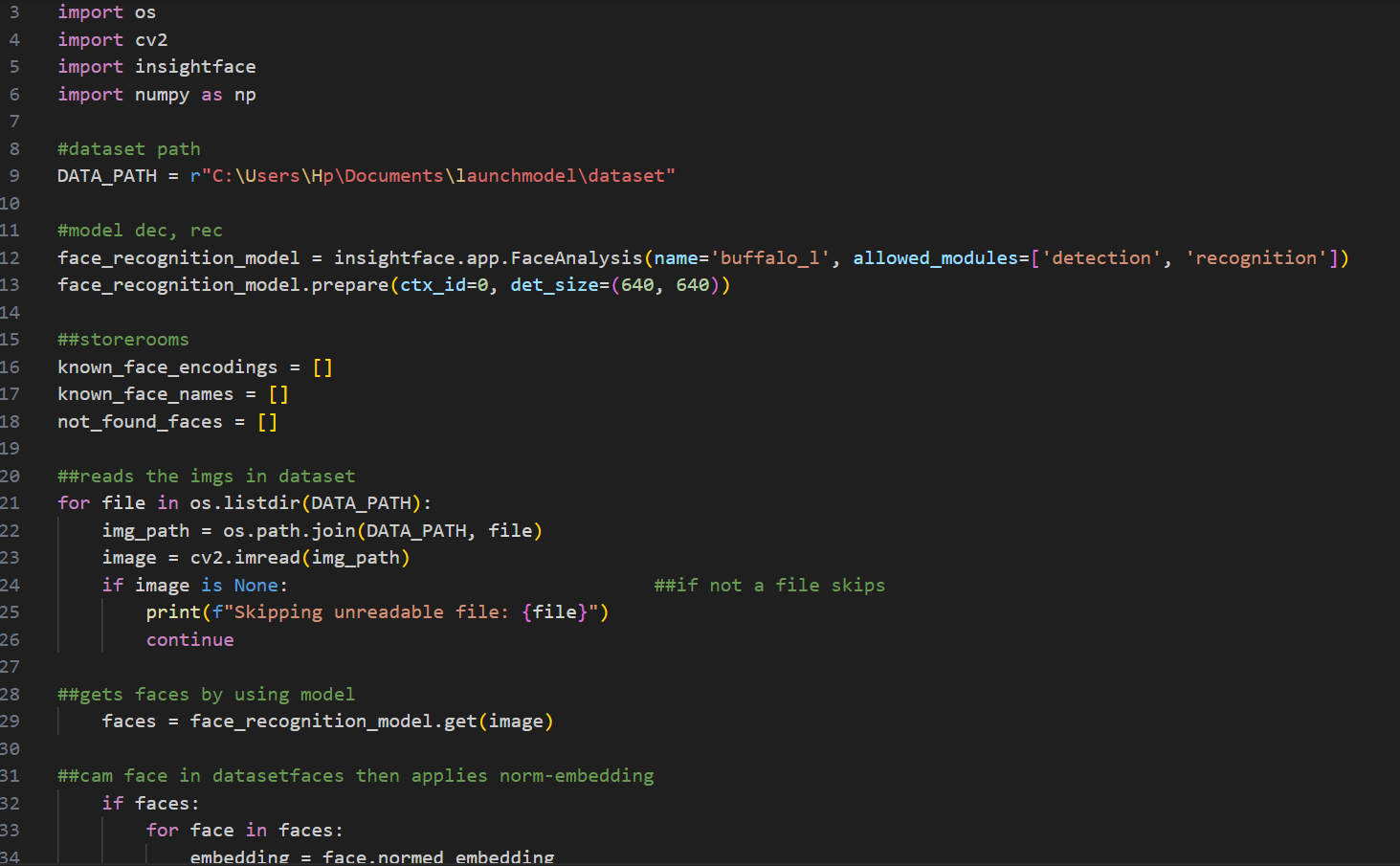
**Some insights:-**

This Python script performs real-time face recognition using the **InsightFace** library, which combines both face detection and recognition. It starts by loading a dataset of face images, extracting facial embeddings, and comparing them to known faces for identification. The webcam is initialized to capture video, and each frame is processed to detect faces. The script checks for occlusion (e.g., partial face covering) by verifying key points, adjusting the recognition thresholds accordingly. It calculates cosine similarity between the live face embedding and the known embeddings, determining the best match based on a threshold. If the score is high enough, the person's name is displayed along with a confidence score; otherwise, the face is marked as "Unknown." The script also draws bounding boxes around faces and marks key points if available. The system continuously processes the video feed until the user presses the 'q' key to exit. It’s designed to handle both fully visible and partially occluded faces with configurable thresholds for recognition accuracy.

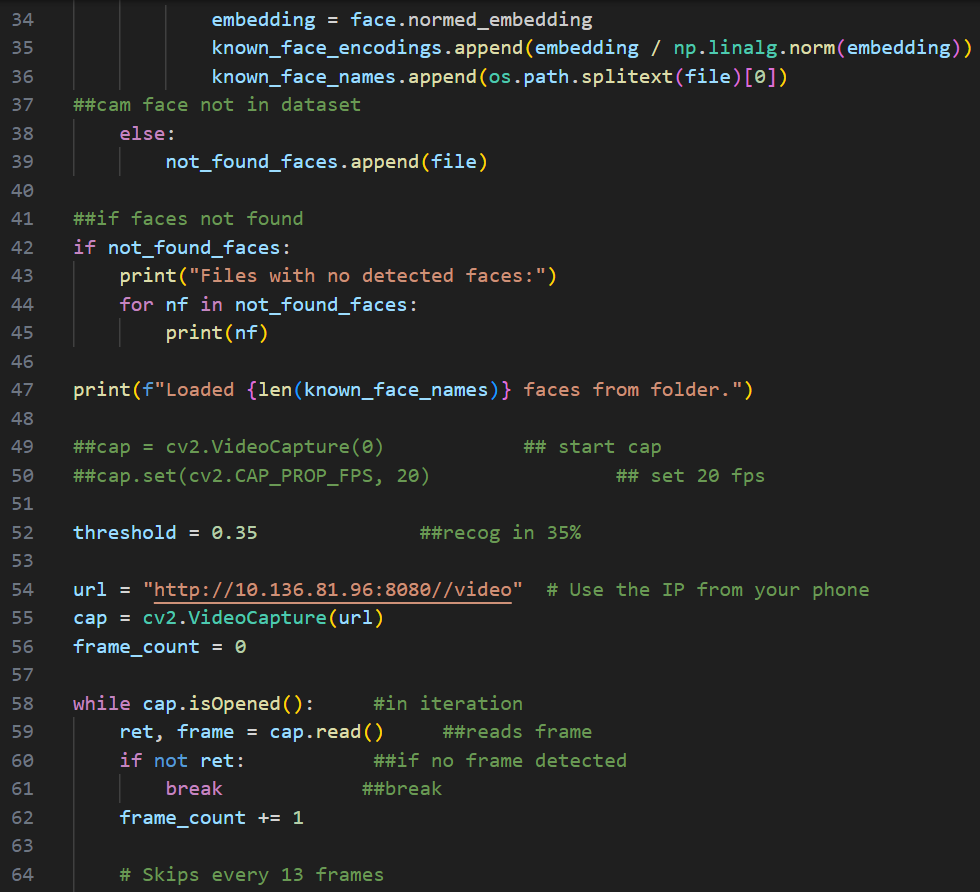
**WorkFlow Chart:-**



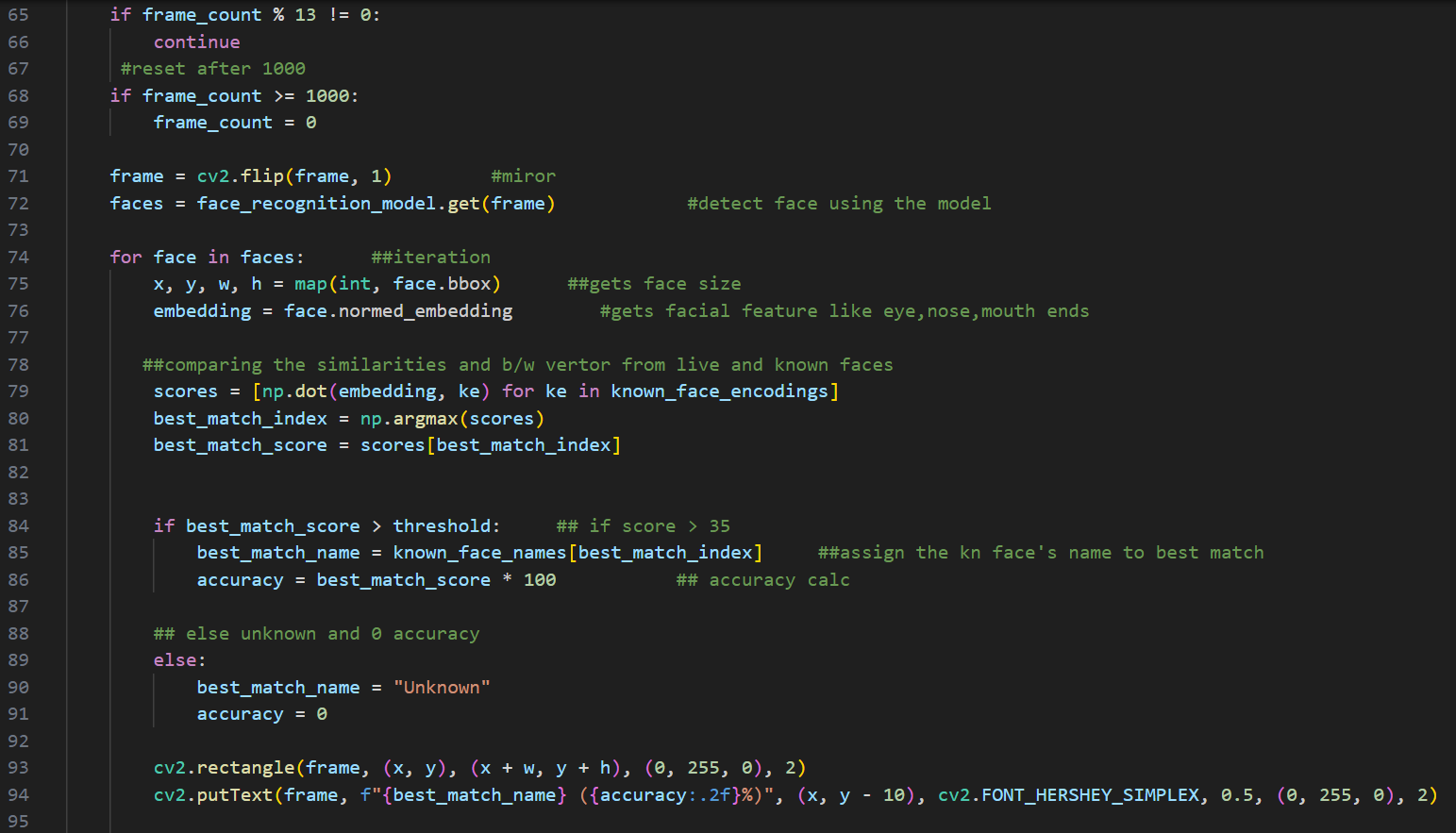
**On-phone Implementation**

****

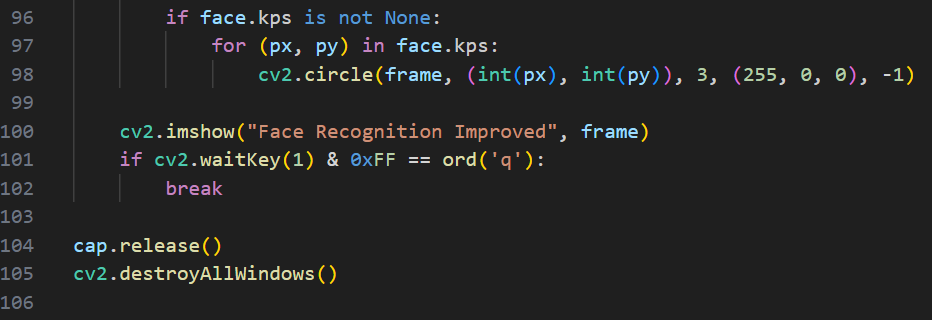
**Fig:-1**

****

**Fig:-2**

****

**Fig:-3**

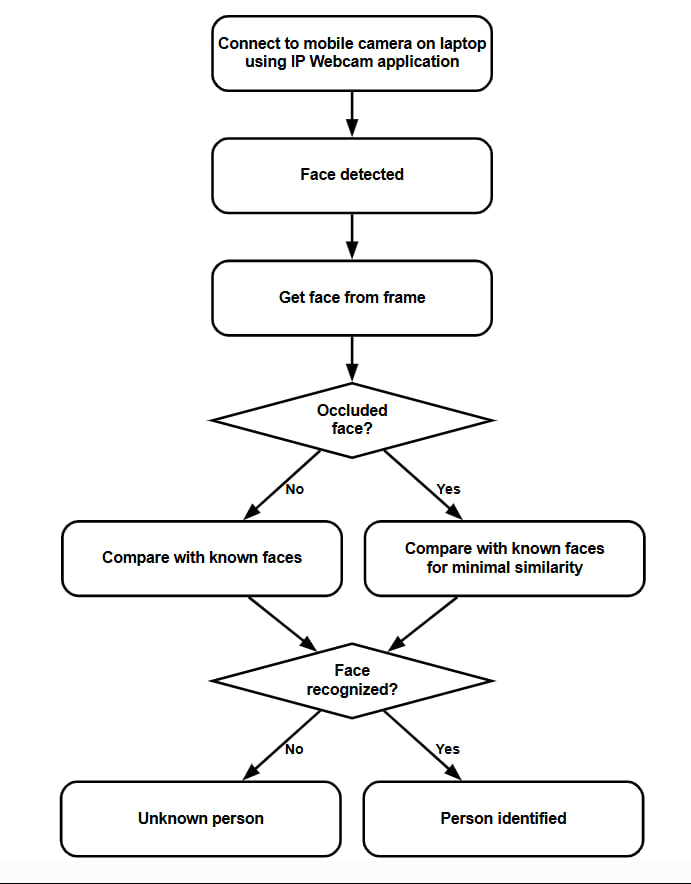
****

**Fig:-4**

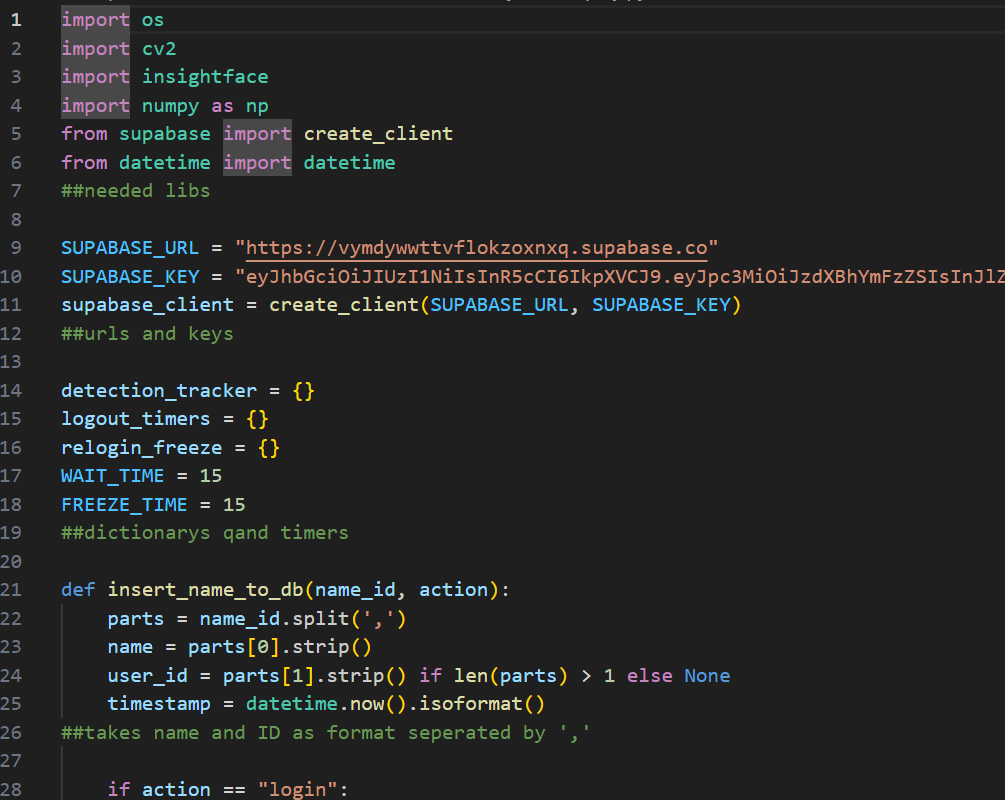
**Some insights:-**

The script captures frames from a mobile device’s camera via an IP stream, processes the frames for face detection and recognition, and compares them with a pre-loaded dataset of known faces. It displays the recognized faces along with their accuracy in real-time.

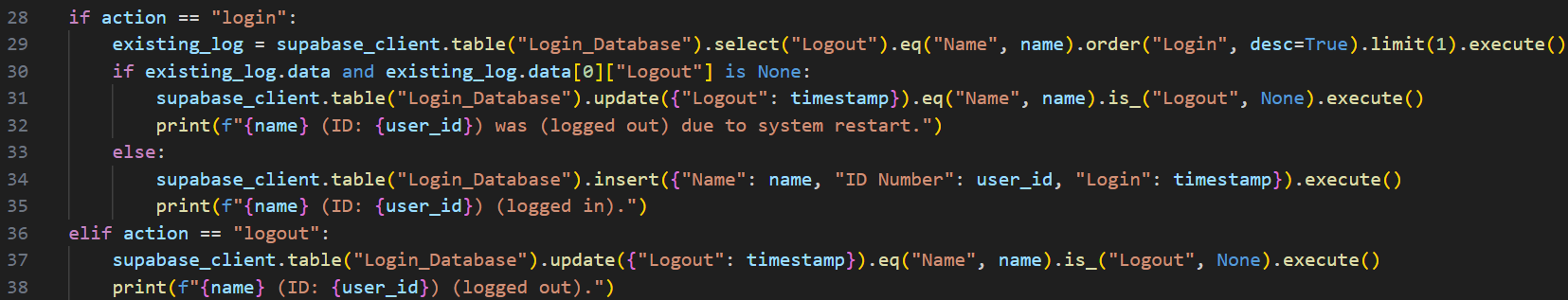
**WorkFlow Chart:-**



**DataBase implementation**

****

**Fig:-1**

****

**Fig:-2**

# Fig:-3

# Fig:-4

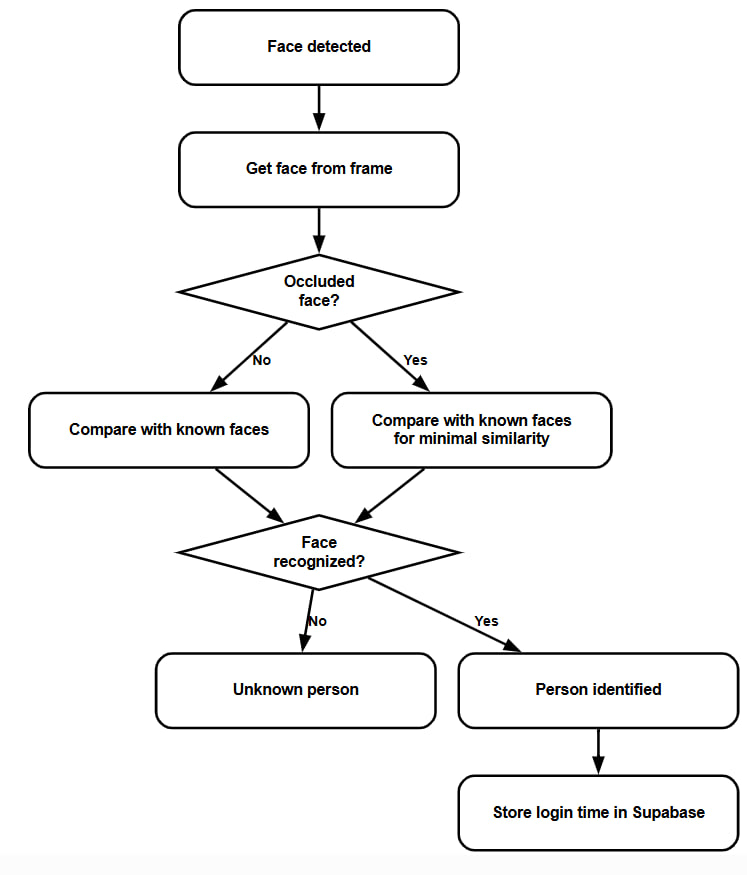
# Fig:-5

# Fig:-6

# Some insights:-

# You have integrated a database functionality into your face recognition system using Supabase. When a face is recognized, the system logs the login and logout events by inserting data into the Supabase "Login\_Database" table. It tracks each user's login and logout times, preventing multiple logins within a short period by implementing a freeze time mechanism for re-login.

# WorkFlow chart:-



# RESULTS

The face recognition system produced reliable results under a variety of conditions. It was able to correctly identify individuals with high accuracy, even when partially occluded or facing sideways. The use of the InsightFace library, combined with proper preprocessing and alignment techniques, contributed to the improved precision of face detection and identification. The system also displayed real-time recognition results, showing the identified person’s name and ID both on the screen and in the terminal, confirming successful implementation of the identification module.

Moreover, deploying the model on a mobile device demonstrated the system’s flexibility and performance in a constrained environment. The database integration with Supabase allowed for seamless storage of login data, showing practical usage for authentication or attendance tracking purposes. These outcomes indicate the system’s readiness for real-world use cases, with future improvements aimed at handling extreme lighting conditions to make it even more robust.

The face recognition system demonstrated consistent performance during testing, even in diverse environmental conditions. It showed the ability to handle varying face orientations, including when individuals were looking away or when their faces were partially obscured by objects such as masks or hands. The **InsightFace** model's robust face detection algorithms ensured that faces were identified with high precision, even under less-than-ideal circumstances.

During real-time operation, the system effectively recognized faces in the webcam feed and displayed both the person’s name and confidence score on the screen. This process occurred without significant delays, maintaining a smooth experience for users. Additionally, when the system could not identify a face or the match confidence was below the predefined threshold, it appropriately labeled the person as "Unknown," providing feedback on potential recognition issues.

In terms of scalability, the system handled multiple simultaneous face recognition tasks with ease. It was capable of processing and comparing embeddings from both the dataset and the real-time camera feed in parallel, ensuring quick identification without excessive resource consumption. The system's use of **NumPy** for embedding normalization and dot product operations proved to be highly efficient in optimizing performance.

The database functionality via **Supabase** added a layer of practicality to the face recognition system. Upon successful identification, the system recorded login/logout actions, storing timestamps and user IDs in real-time. This feature proved especially useful for monitoring user activity and integrating the system into workflows like access control or attendance logging. By securely storing user data and login histories, the system made it easier to track and manage authenticated users.

However, challenges remain in handling extreme lighting conditions, where faces could be overexposed or underexposed. While the system performs well in typical indoor and outdoor settings, further enhancements, such as integrating automatic lighting adjustment or refining the face detection algorithm, could help address such issues. Overall, the system shows great potential for a wide range of applications in security, workplace automation, and user authentication.

# Output for Only-Recognition

# 

# Fig:-1

# Output for Occlusion Included

# 

# Fig:-1 - Covering bottom portion of face.

# 

# Fig:-2 - Covering right portion of face.

# 

# Fig:-3 - Covering left portion of face.

# 

# Fig:-4 - Covering upper portion of face.

# 

# Fig:-5 - Covering upper diagonal portion.

# 

# Fig:-6 - Covering upper diagonal portion.

# Output for Onphone-Implementation

# 

# Fig:-1 – Implementation on Mobile using IPWebcam Application

# Output for Database

# 

# Fig-1 - Display log-in in Terminal.

# 

# Fig:-2 - Display log-in on DataBase.

# 

# Fig:-3 – Display log-out in terminal.

# 

# Fig:-4 – Display Log-out on Database.

# Conclusion and Future Work

## Conclusion:

In conclusion, the face recognition system successfully achieved its intended objectives by accurately identifying individuals, handling partial occlusions, and working effectively on both desktop and mobile platforms. The integration of tools like InsightFace, Supabase, and real-time video processing through OpenCV proved essential to its functionality. The project not only demonstrated technical feasibility but also offered a foundation for future development, such as improving performance under challenging conditions like poor lighting and enhancing model speed and accuracy for larger-scale deployment.

## Future Work:

For future improvements, one major area to address is the system's performance under varying lighting conditions. Currently, the model may struggle in environments with poor illumination, shadows, or overly bright backgrounds, which can affect recognition accuracy. Enhancing the model's robustness in such conditions will make it more reliable in real-world scenarios. This can be achieved by incorporating advanced preprocessing techniques like histogram equalization, training with more diverse lighting datasets, and possibly using infrared or depth-sensing technologies. Such enhancements will ensure the system performs consistently across different environments, times of day, and user locations.

# References

**InsightFace** – A deep face analysis toolbox used for facial recognition.

Website: <https://github.com/deepinsight/insightface>

**OpenCV(Computer-Vision)** – Open - source computer vision and image processing library.

Website: <https://opencv.org/>

**NumPy** – A fundamental package for numerical computation in Python.

Website: <https://numpy.org>

**Python** – A high-level programming language used to implement the entire system.

Website: <https://www.python.org>

**Supabase**  – Open - source backend platform used to storing user data.

Website: <https://supabase.com>

**Visual Studio Code** – A source-code editor used for development and debugging.

Website: <https://code.visualstudio.com>