UNIVERSITY OF SAN DIEGO

Master of Science in Applied Artificial Intelligence

Applied Computer Vision for AI (AAI-521-IN4)

Face Recognition Based Attendance System

Submitted by:

Group 11

Meghesh Saini

Aishwarya Gulhane

Dhrub Satyam

Under the guidance of:

Professor Amit Butail

December 2025

GitHub Link: *https://github.com/megheshsaini/visionxusd/*

Abstract

This project highlights the implementation of the Attendance system through face recognition developed as part of the Applied Computer Vision for AI at the University of San Diego.

Our system uses DeepFace, a deep learning framework, combined with the FaceNet embedding model to achieve robust facial recognition with high accuracy. The application integrates a Gradio-based web interface for user interaction.

The system architecture employs a multi-sample enrollment approach requiring a minimum of three face samples per individual to ensure reliable recognition. Recognition is performed using Euclidean distance metrics against 128-dimensional facial embeddings, with a configurable tolerance threshold of 0.25 for matching decisions. The implementation supports multiple detector backends including RetinaFace and OpenCV, enabling deployment flexibility across different computational environments.

Key contributions include a production-ready web interface supporting image upload and real-time webcam capture, automated attendance logging, and a robust recognition algorithm that handles natural facial variations. Experimental results demonstrate the system's effectiveness for practical attendance management applications in educational and corporate settings.

Table of Contents

1. Introduction

1.1 Background and Motivation

Attendance management remains a critical operational challenge across educational institutions and corporate environments. Traditional methods including manual roll calls, sign-in sheets, and card-based systems are time-consuming, prone to errors, and susceptible to manipulation through proxy attendance. Biometric solutions, particularly facial recognition, offer a contactless, non-intrusive alternative that uniquely identifies individuals without requiring physical tokens or direct contact.

Recent advances in deep learning have significantly improved facial recognition accuracy. Models such as FaceNet, developed by Google Research, achieve near-human performance on benchmark datasets. The DeepFace library provides a unified interface to multiple state-of-the-art models, enabling rapid prototyping and deployment of recognition systems. This project leverages these advances to create a practical, deployable attendance solution.

1.2 Problem Statement

The objective of this capstone project is to implement a face recognition system capable of accurately identifying enrolled individuals from webcam captures or uploaded images, logging attendance records automatically to a cloud-based spreadsheet. The system must handle natural variations in facial appearance including lighting changes, pose variations, and partial occlusions while maintaining low false positive and false negative rates.

1.3 Project Objectives

1. Implement robust face detection using state-of-the-art detector backends
2. Generate discriminative facial embeddings using the FaceNet deep learning model
3. Develop a user-friendly web interface for face enrollment and recognition
4. Integrate cloud-based attendance logging.
5. Achieve reliable recognition with configurable accuracy thresholds

1.4 Scope and Limitations

The system is designed for controlled indoor environments with reasonable lighting conditions. It does not include liveness detection mechanisms to prevent photo-based spoofing attacks. The current implementation supports single-camera input and is optimized for front-facing or near-frontal facial images. Future iterations may address these limitations through depth sensing and anti-spoofing modules.

2. Literature Review

2.1 Evolution of Face Recognition

Face recognition has evolved significantly from early geometric approaches to modern deep learning methods. Initial systems relied on hand-crafted features such as Eigenfaces (Turk & Pentland, 1991). These methods, while computationally efficient, struggled with variations in pose, illumination, and expression.

2.2 FaceNet Architecture

FaceNet, developed by Google, learns the mapping from facial images to a compact 128-dimensional Euclidean space where distances directly correspond to face similarity. The model is trained using triplet loss, which ensures that an anchor image is closer to positive examples (same identity) than negative examples (different identities) by a specified margin. FaceNet achieves 99.63% accuracy on the LFW benchmark, making it one of the most accurate publicly available models.

2.3 Face Detection Methods

Modern face detection relies on deep learning approaches including MTCNN, RetinaFace, and SSD-based detectors. RetinaFace, used in this project, performs single-stage face detection with landmark localization. It achieves state-of-the-art performance on the WIDER FACE benchmark, particularly for detecting small and occluded faces. The detector also provides facial landmark coordinates enabling face alignment for improved recognition accuracy.

2.4 DeepFace Framework

DeepFace is an open-source Python library that provides a unified interface for face analysis including detection, recognition, and attribute analysis. It supports multiple backend models (VGG-Face, FaceNet, OpenFace, DeepFace, ArcFace) and detectors (OpenCV, SSD, MTCNN, RetinaFace), enabling flexible deployment based on accuracy and computational requirements. The library handles preprocessing, alignment, and normalization automatically, simplifying development of recognition applications.

3. System Design and Architecture

3.1 High-Level Architecture

The system follows a modular three-tier architecture consisting of a presentation layer (Gradio web interface), business logic layer (FaceRecognitionSystem class), and data layer (pickle file storage and Google Sheets). This separation of concerns enables independent testing, maintenance, and scaling of individual components.

Layer	Components	Responsibility
Presentation	Gradio Blocks, Tabs, Image inputs	User interface, webcam capture, image upload
Business Logic	FaceRecognitionSystem class	Detection, embedding, recognition, enrollment
Data	Pickle files	Embedding storage, attendance logging

Table 1: System Architecture Layers

3.2 Data Flow Pipeline

The recognition pipeline processes images through the following sequential stages:

1. **Image Acquisition:** Webcam capture or file upload via Gradio interface
2. **Face Detection:** RetinaFace/OpenCV locates faces and extracts bounding boxes
3. **Face Alignment:** Detected faces aligned using facial landmark positions
4. **Embedding Generation:** FaceNet transforms aligned faces to 128-D vectors
5. **Distance Computation:** Euclidean distance calculated against enrolled embeddings
6. **Identity Resolution:** Minimum distance match selected if below threshold
7. **Logging:** Attendance record inserted into cloud.

3.3 Class Design

The core business logic is encapsulated in the FaceRecognitionSystem class, which maintains the embedding database and provides methods for enrollment, recognition, and database management.

Key methods include:

- **detect_and_embed_faces()** for face detection and embedding generation,
- **recognize_face_robust()** implementing the matching algorithm,
- **add_face()** for user enrollment, and
- **recognize_faces()** for the recognition workflow.

4. Implementation Details

4.1 Technology Stack

Category	Technology	Purpose
Deep Learning	DeepFace 0.0.95	Face analysis framework (detection, embedding)
Recognition Model	FaceNet	128-D embedding generation (99.63% LFW accuracy)
Face Detection	RetinaFace / OpenCV	Face localization with landmark detection
Backend Framework	TensorFlow 2.10+	Neural network inference engine
Web Interface	Gradio 4.0+	Interactive web UI with webcam support
Cloud Integration	gspread 5.0+	Google Sheets API for attendance logging
Image Processing	OpenCV 4.5+	Image manipulation, drawing, color conversion

Table 2: Technology Stack

4.2 Configuration Parameters

The system behavior is controlled by configurable parameters defined at module level:

Parameter	Value	Description
MODEL_NAME	Facenet	Recognition model for embedding generation
DETECTOR_BACKEND	opencv	Face detector (retinaface for higher accuracy)
TOLERANCE	0.25	Maximum Euclidean distance for positive match
MIN_SAMPLES_PER_PERSON	3	Required samples before recognition enabled

Table 3: System Configuration Parameters

4.3 Recognition Algorithm

The recognition algorithm implements a nearest-neighbor classifier using Euclidean distance in the embedding space. For each detected face, the system computes distances against all stored embeddings for each enrolled person, selecting the minimum distance per person. The identity with the smallest minimum distance is selected as the match candidate.

Two validation checks ensure reliability:

(1) the candidate must have at least MIN_SAMPLES_PER_PERSON enrolled samples, and (2) the minimum distance must be below the TOLERANCE threshold. If either check fails, the face is classified as "Unknown."

4.4 Web Interface Design

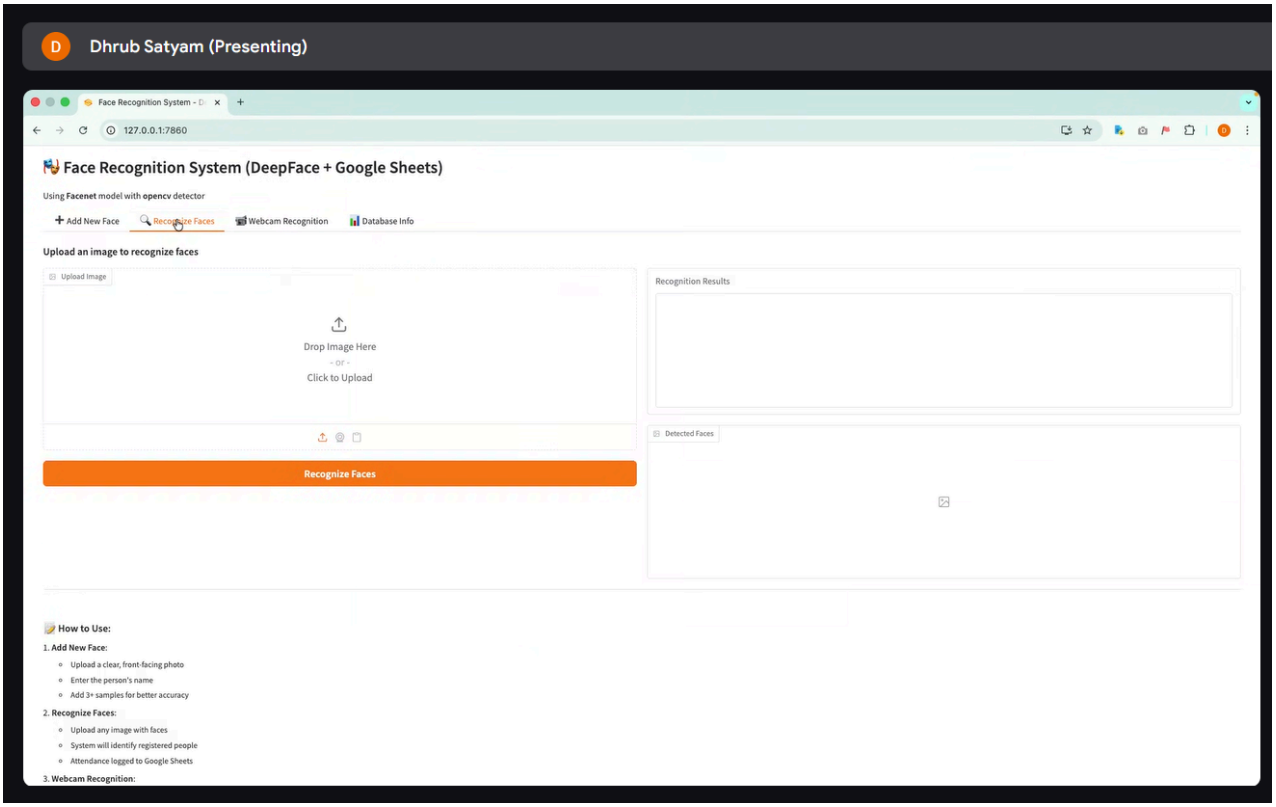
The Gradio interface provides four functional tabs:

(1) Add New Face for enrollment with name input and image upload,

(2) Recognize Faces for identification from uploaded images,

(3) Webcam Recognition for real-time capture and recognition, and

(4) Database Info displaying enrollment statistics. Each tab provides visual feedback with annotated result images showing bounding boxes and recognition results. The interface runs on port 7860 and is accessible via any web browser.



5. Results and Evaluation

5.1 Model Performance

The system leverages pre-trained models with established benchmark performance:

Component	Benchmark	Accuracy
FaceNet Model	LFW Dataset	99.63%
RetinaFace Detector	WIDER FACE (Hard)	91.4%

Table 4: Model Benchmark Performance

5.2 System Features Achieved

- **Multi-face detection:** System successfully detects and processes multiple faces in a single image
- **Web-based interface:** Fully functional Gradio application with four operational tabs
- **Real-time webcam capture:** Supports live camera input for immediate recognition
- **Cloud attendance logging:** Automatic insertion of timestamped records to Google Sheets
- **Multi-sample enrollment:** Robust recognition through multiple samples per person
- **Confidence scoring:** Visual display of recognition confidence percentages

6. Discussion

6.1 Strengths

The multi-sample enrollment approach significantly improves recognition robustness by capturing natural intra-class variation in facial appearance. The minimum sample requirement prevents premature matching against insufficiently characterized identities. The Gradio interface provides a low-barrier deployment option, enabling immediate use without custom frontend development. Google Sheets integration eliminates the need for dedicated database infrastructure while providing built-in accessibility and sharing features.

6.2 Limitations

The current implementation lacks liveness detection, making it vulnerable to photo-based spoofing attacks. Linear search for recognition may not scale efficiently beyond several hundred enrolled individuals. The system assumes cooperative subjects providing reasonably frontal face images. Extreme poses, heavy occlusions, or poor lighting conditions may degrade recognition accuracy. The pickle-based storage is not suitable for concurrent multi-user access scenarios.

6.3 Potential Improvements

1. **Liveness Detection:** Implement blink detection or head movement verification
2. **Vector Database:** Replace linear search with FAISS or Pinecone for scalability
3. **GPU Acceleration:** Optimize inference with TensorRT for real-time video processing
4. **Database Backend:** Migrate to PostgreSQL for concurrent access and data integrity
5. **Edge Deployment:** Optimize for Raspberry Pi or Jetson Nano for standalone kiosks

7. Conclusion

This project successfully demonstrates the application of modern deep learning techniques to automated attendance management. By leveraging the DeepFace framework with FaceNet embeddings, the system achieves reliable face recognition suitable for practical deployment.

Key technical contributions include the multi-sample enrollment approach ensuring robust identity characterization, the configurable tolerance threshold balancing security against usability.

The system serves as a solid foundation for production attendance applications with identified paths for enhancement including liveness detection and database scaling.

8. References

[1] Schroff, F., Kalenichenko, D., & Philbin, J. (2015). FaceNet: A unified embedding for face recognition and clustering. *IEEE Conference on Computer Vision and Pattern Recognition*, 815-823.

[2] Deng, J., Guo, J., Xue, N., & Zafeiriou, S. (2019). ArcFace: Additive angular margin loss for deep face recognition. *IEEE Conference on Computer Vision and Pattern Recognition*, 4690-4699.

[3] Taigman, Y., Yang, M., Ranzato, M., & Wolf, L. (2014). DeepFace: Closing the gap to human-level performance in face verification. *IEEE Conference on Computer Vision and Pattern Recognition*, 1701-1708.

[4] Deng, J., Guo, J., Zhou, Y., Yu, J., Kotsia, I., & Zafeiriou, S. (2020). RetinaFace: Single-shot multi-level face localisation in the wild. *IEEE Conference on Computer Vision and Pattern Recognition*, 5203-5212.

[5] Serengil, S. I., & Ozpinar, A. (2021). HyperExtended LightFace: A facial attribute analysis framework. *IEEE International Conference on Engineering and Emerging Technologies*, 1-4.

[6] Huang, G. B., Ramesh, M., Berg, T., & Learned-Miller, E. (2007). Labeled faces in the wild: A database for studying face recognition in unconstrained environments. *Technical Report 07-49, University of Massachusetts*.

[7] Zhang, K., Zhang, Z., Li, Z., & Qiao, Y. (2016). Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE Signal Processing Letters*, 23(10), 1499-1503.

[8] DeepFace Library Documentation. https://github.com/serengil/deepface

Appendix A: Installation Guide

Prerequisite :

- need Python 3.8 or higher version
- pip package manager
- Webcam (optional, for live capture)

Appendix B: Project Repository Structure

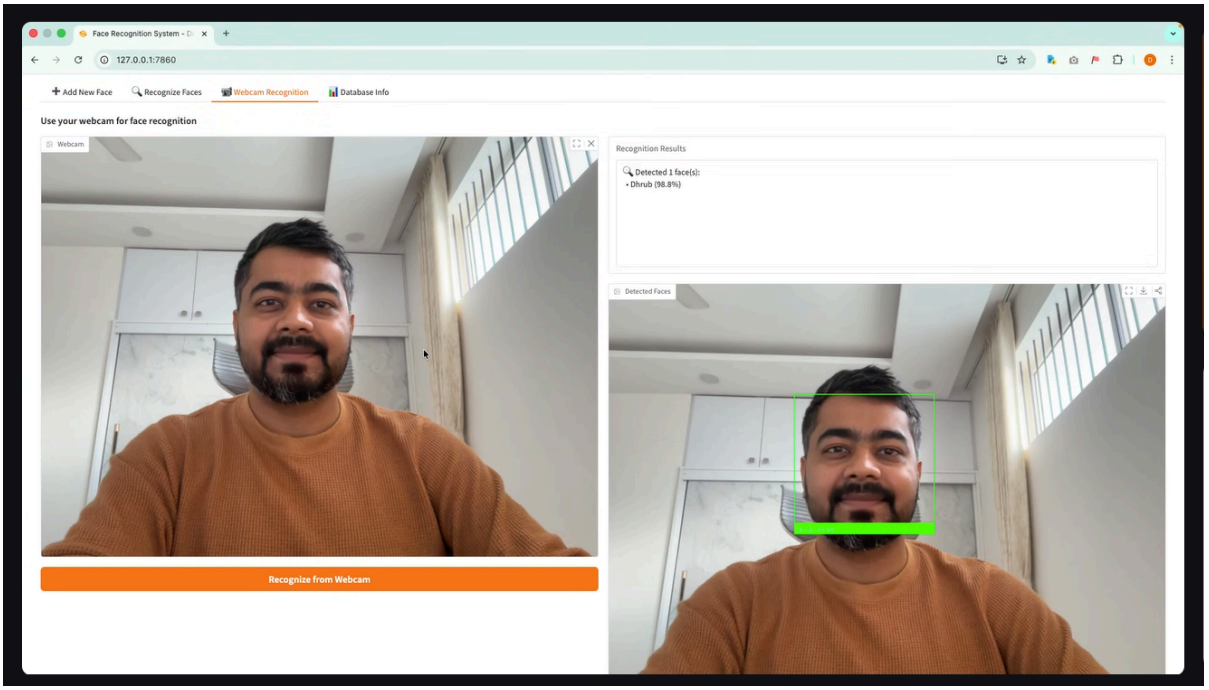
```
visionxusd-main/
├── gradio_app.py          # Main web application
├── facerecognition.py     # CLI-based recognition module
├── requirements.txt       # Python dependencies
├── face_embeddings.pkl    # Stored face database
├── face_recognition.json  # Google Sheets credentials
└── README.md              # Project documentation
```

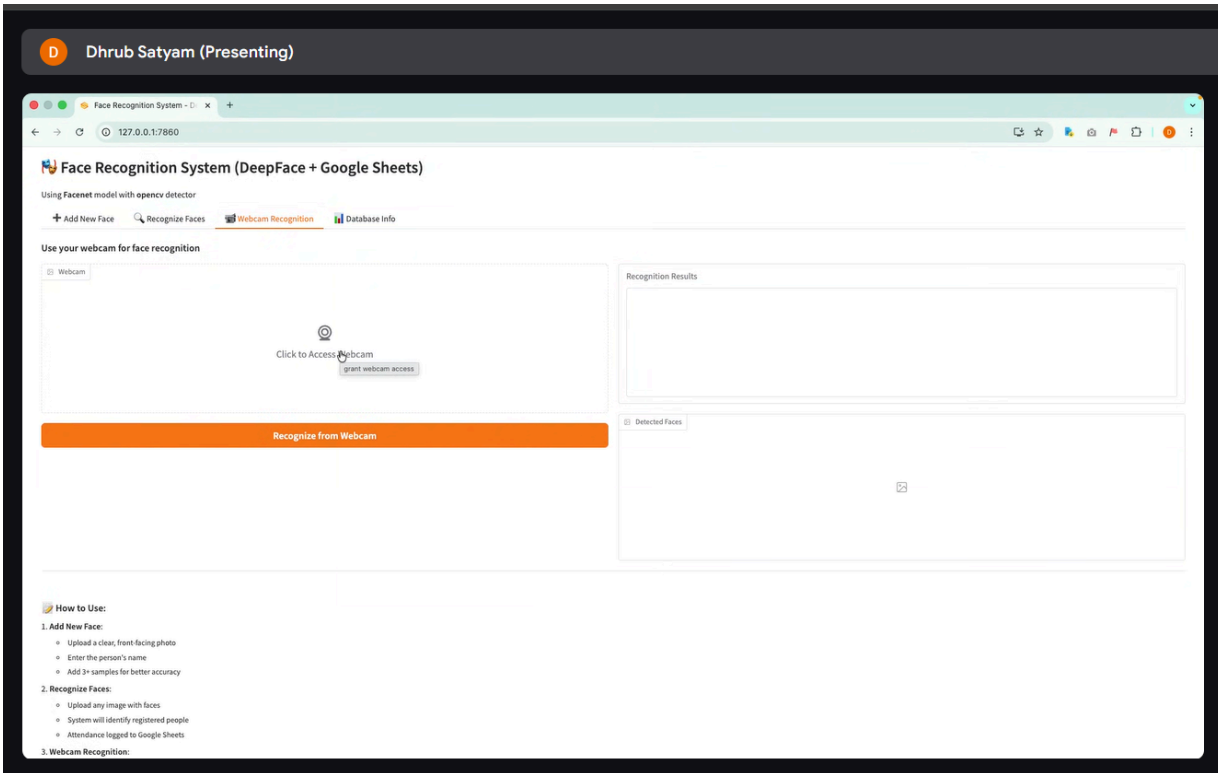
Appendix C: Team Contributions

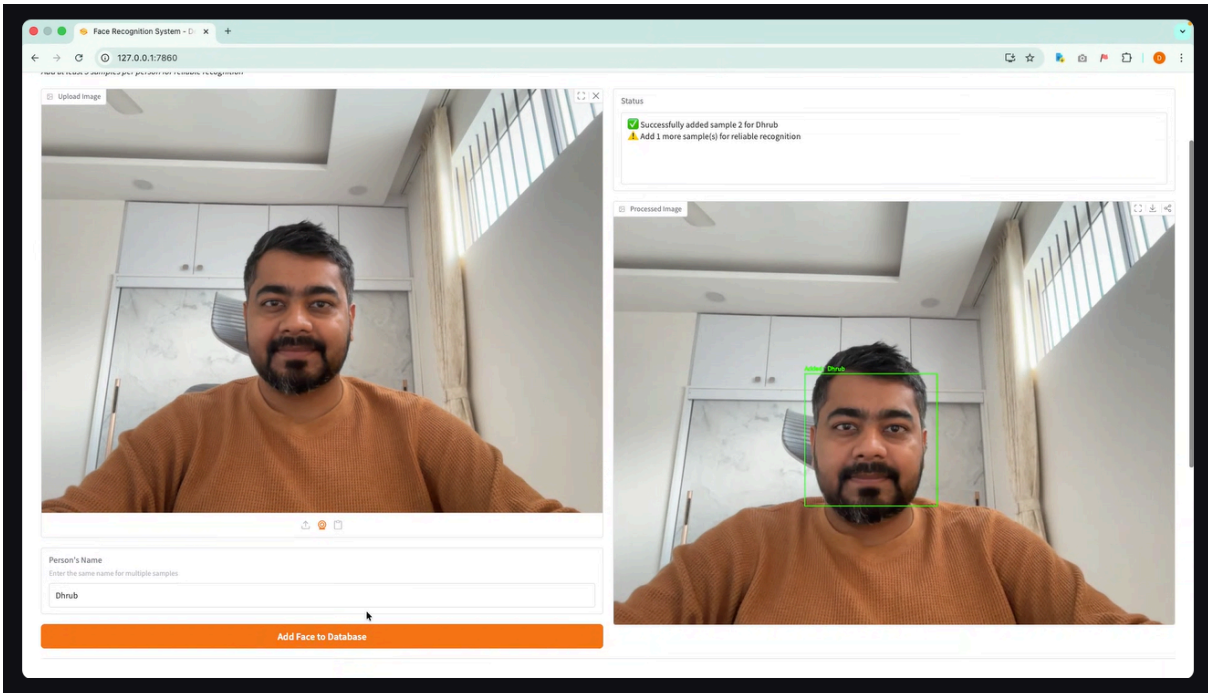
Team Member	Contributions
Meghesh Saini	System architecture, Gradio interface implementation
Aishwarya Gulhane	Recognition algorithm, documentation
Dhrub Satyam	Face detection module, testing and validation

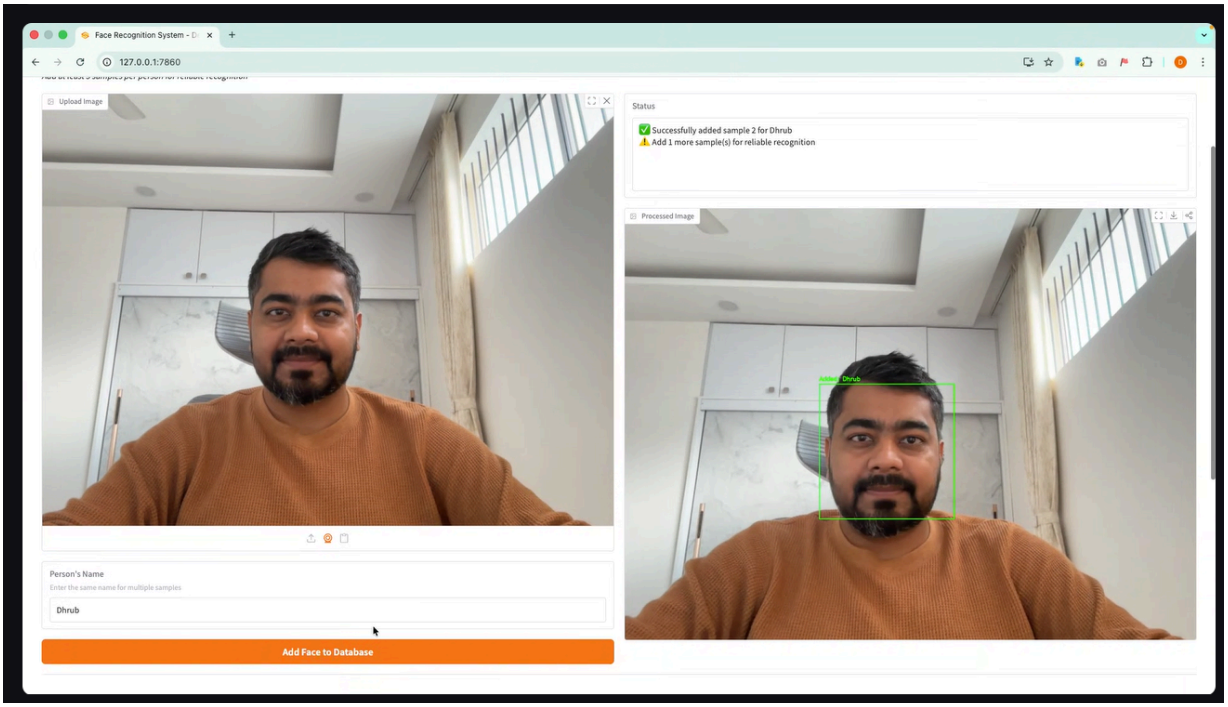
Table 5: Team Contributions

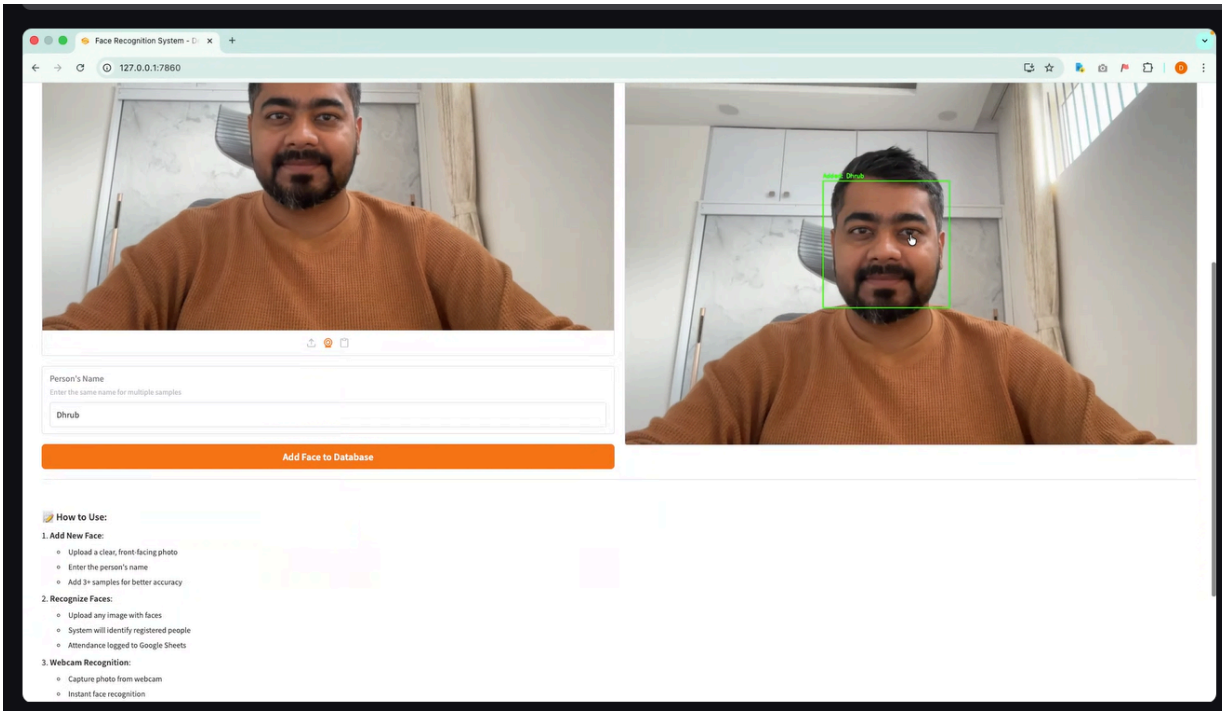
Appendix D: Screenshots from the App

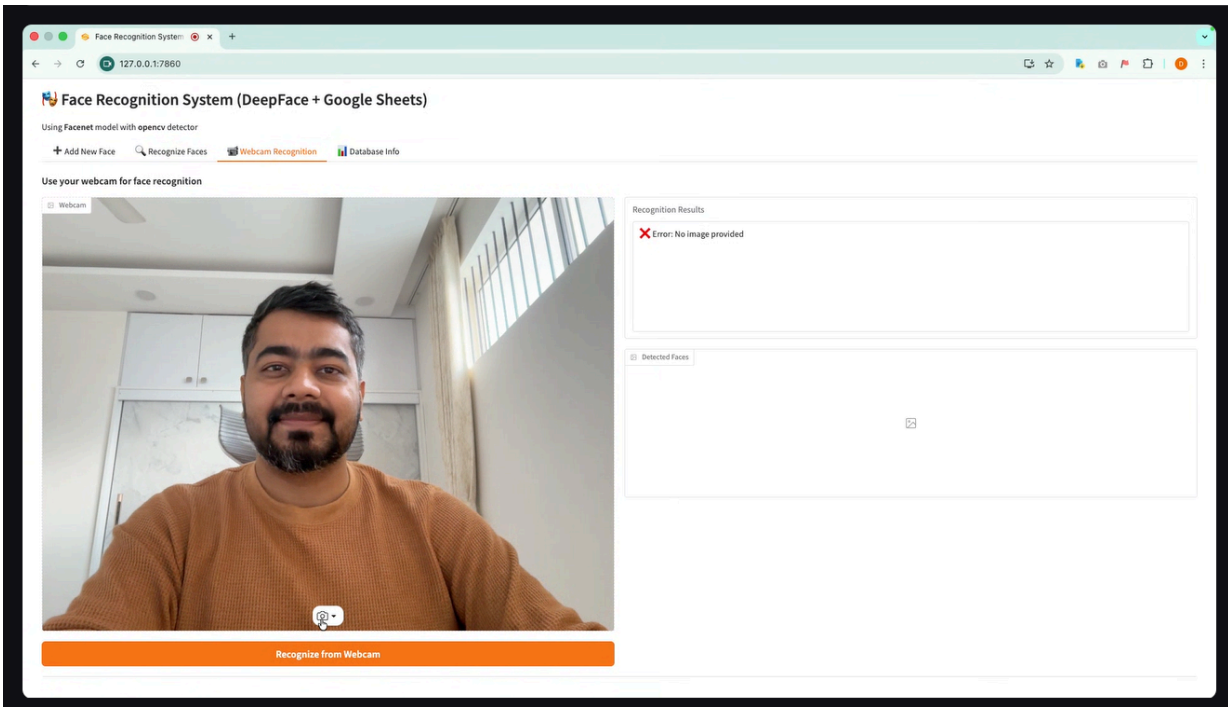


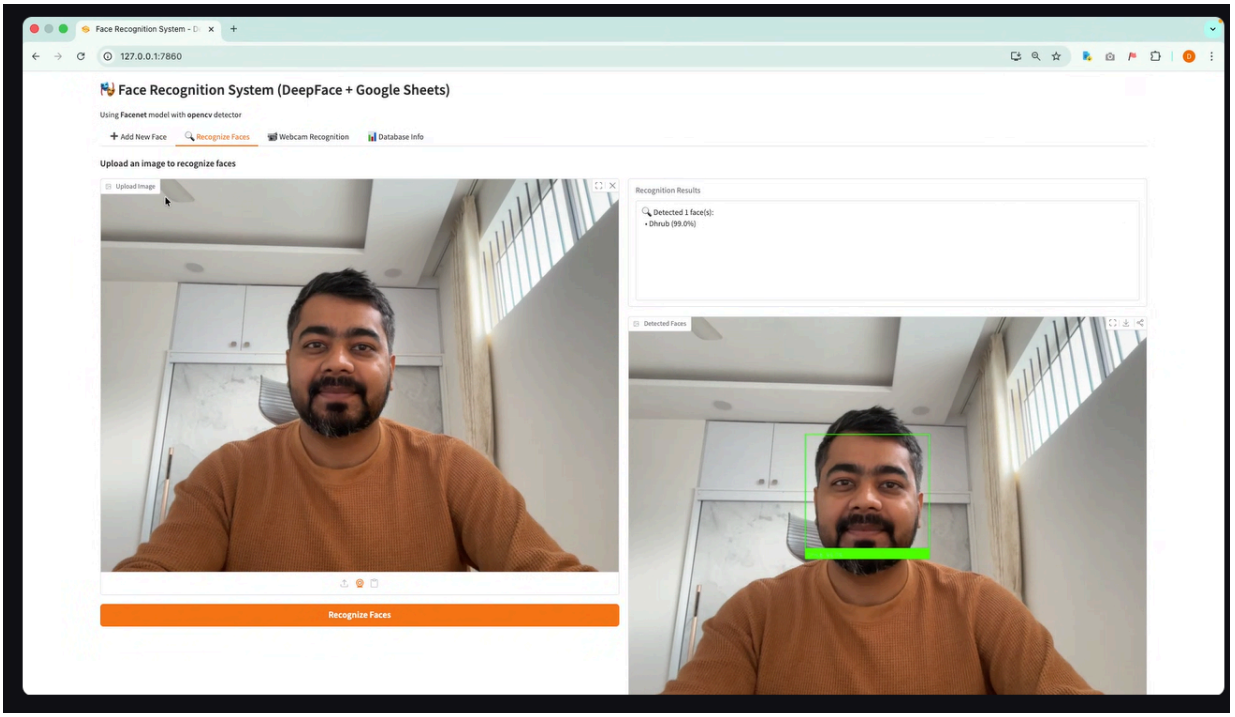



Face Recognition System
127.0.0.1:7860
Upload Image
Status
Successfully added sample 2 for Dhrub
Add 1 more sample(s) for reliable recognition
Processed Image
Person's Name
Enter the same name for multiple samples
Dhrub
Add Face to Database


Face Recognition System
127.0.0.1:7860
Upload Image
Status
Successfully added sample 2 for Dhrub
Add 1 more sample(s) for reliable recognition
Processed Image
Person's Name
Enter the same name for multiple samples
Dhrub
Add Face to Database


Face Recognition System - D
127.0.0.1:7860
Person's Name
Enter the same name for multiple samples
Dhrub
Add Face to Database
How to Use:
1. Add New Face:
Upload a clear, front-facing photo
Enter the person's name
Add 3+ samples for better accuracy
2. Recognize Faces:
Upload any image with faces
System will identify registered people
Attendance logged to Google Sheets
3. Webcam Recognition:
Capture photo from webcam
Instant face recognition


Face Recognition System
127.0.0.1:7860
Face Recognition System (DeepFace + Google Sheets)
Using Facenet model with opencv detector
Add New Face
Recognize Faces
Webcam Recognition
Database Info
Use your webcam for face recognition
Webcam
Recognition Results
Error: No image provided
Detected Faces
Recognize from Webcam


Face Recognition System - D
127.0.0.1:7860
Face Recognition System (DeepFace + Google Sheets)
Using Facenet model with opencv detector
Add New Face
Recognize Faces
Webcam Recognition
Database Info
Upload an image to recognize faces
Upload Image
Recognition Results
Detected 1 face(s):
• Dhrub (99.0%)
Detected Faces
Recognize Faces