FACETRACK: PERSONALIZED INFORMATION RETRIEVAL THROUGH FACE RECOGNITION

BY

OOI YUNN SUEN

A REPORT
SUBMITTED TO
Universiti Tunku Abdul Rahman
in partial fulfillment of the requirements
for the degree of
BACHELOR OF COMPUTER SCIENCE (HONOURS)
Faculty of Information and Communication Technology
(Kampar Campus)

FEBRUARY 2025

COPYRIGHT STATEMENT

This Final Year Project report is submitted in partial fulfillment of the requirements for the degree of Bachelor of Computer Science (Honours) at Universiti Tunku Abdul Rahman (UTAR). This Final Year Project report represents the work of the author, except where due acknowledgment has been made in the text. No part of this Final Year Project report 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 express my sincere thanks and appreciation to my ex-supervisor, Ts Dr. Ooi Boon Yaik, and my supervisor, Ts. Tan Teik Boon, for their invaluable guidance and support throughout my FYP project. When I was facing problems in this project, the advice from them always assists me in overcoming the problems. Once again, a million thanks to both my former and current supervisors.

In addition, I would like to extend my heartfelt gratitude to my parents and family for their unwavering love, encouragement, and support throughout this journey.

Finally, I would also like to thank my friends for their patience, unconditional support, and assistance in testing the performance of the system.

ABSTRACT

This project addressed the challenge of managing and recalling personal information in scenarios involving high-volume interactions, infrequent encounters, and age-related memory decline using face recognition. Existing face recognition solutions were primarily limited to government applications, such as national identification systems, suspect tracking, and border control [1], [2], highlighting the need for accessible, personalized tools for everyday users. The system employed OpenCV and YuNet for face detection and movement analysis with a sharpness threshold above 1.8, while DeepFace (FaceNet512) enabled face matching with a similarity threshold of 0.7. WebSocket facilitated low-latency communication between the frontend and backend, supported by user interaction modules with form validation and notifications, and PostgreSQL handled database operations. The system was tested across multiple stages, including initialization, detection, recognition, user interaction, and database management. It achieved an end-to-end latency of 2–5 seconds and a recognition accuracy of 98.54% for known faces under controlled lighting. The novelty of this system lay in its movement-adaptive processing, which cleared the recognition queue during rapid motion. This allowed the system to quickly process new faces entering the camera view, as high movement often indicated a change in person. Additionally, its sharpness-based image capture mechanism ensured that only high-quality images were processed. Combined with WebSocket-driven pause/resume functionality, these features enabled a seamless and uninterrupted user experience. Overall, the system demonstrated the feasibility and scalability of a personalized face recognition solution for everyday environments such as kindergartens, hotels, and events.

Area of Study (Minimum 1 and Maximum 2): Computer Vision

Keywords (Minimum 5 and Maximum 10): Face Recognition, Personalized Information Retrieval, Movement-Adaptive Processing, Sharpness-Based Image Capture, User Interaction Module, WebSocket Communication, YuNet, DeepFace (FaceNet512), PostgreSQL Database

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Introduction

This chapter provides an overview of the background and motivation behind the project, outlines the problem statement, defines the project objectives and scope, highlights the contributions to the field, and presents the structure of the thesis.

1.1 Background Information

1.1.1 Computer Vision

Computer Vision is a part of artificial intelligence (AI) and computer science that focuses on enabling machines to interpret, understand, and respond to visual information in a way like humans [3]. The goal of computer vision is to automate tasks that the human visual system can perform, such as detecting objects, recognizing faces, reading text, and interpreting scenes in images or videos [3].

Computer vision started in the 1950s and 1960s when researchers began studying how machines could handle and understand images and visual information. Early experiments used some of the first neural networks to perform basic image-processing tasks, such as detecting edges and sorting simple objects into categories like circles and squares [4]. By the 1970s, the field had its first practical use with optical character recognition (OCR), a technology that could read typed or handwritten text [4]. This technology was used to assist the visually impaired by converting written text into a readable format. As computational power increased in the 1970s and 1980s, researchers developed more advanced algorithms, which improved object recognition and motion tracking [3]. The field advanced further in the 1990s with the rise of machine learning methods, especially neural networks [3]. The use of machine learning allowed computer vision systems to improve over time by learning from data, which was a breakthrough in pattern recognition and classification tasks.

In recent years, the integration of deep learning techniques, especially convolutional neural networks (CNNs), has transformed computer vision. CNNs have greatly improved accuracy in tasks like classifying images, detecting objects, and recognizing faces [5]. Face recognition is one of the most popular applications within computer vision [5]. It is used in various domains such as security for identity verification, smartphones for user authentication, social media for tagging individuals in photos, and law enforcement for suspect identification.

1.2 Project Motivations

The motivation for this project stems from the need to provide personalized face recognition solutions for non-technical users such as kindergarten teachers, hotel staff, and event organizers. Existing systems often rely on centralized infrastructures and are not designed for personalized, context-specific use. This system addresses that gap by enabling user-owned data storage, allowing faces and associated personal details (e.g., “Parent of Alice,” “Room 201,” or “VIP Identification”) to be stored locally. It also supports custom workflows tailored to individual use cases, such as verifying parents during kindergarten pickups or streamlining hotel check-ins. Additionally, the project is motivated by the limitations of human memory, particularly in high-volume or age-related contexts, where recalling names and faces becomes increasingly difficult. The system aims to reduce social friction by minimizing misidentifications, enhance security by restricting access to recognized individuals, and eliminate operational inefficiencies by replacing manual name or photo cross-referencing with automated recognition.

1.3 Problem Statements

The first problem addressed by this project is the limited accessibility of existing face recognition systems, which are primarily developed for government applications such as border control and national ID systems [1], [2]. These solutions typically lack user-friendly interfaces and do not support user-owned data architectures, making them unsuitable for personal or small-scale deployment in everyday environments such as kindergartens, hotels, or event guest management. In addition, the project responds to the inadequate support for human memory demands in high-volume social settings and among individuals experiencing age-related memory decline. Current practices fail to prevent errors in sensitive scenarios, such as misidentifications during child pickups or VIP identification, and continue to rely on manual verification methods like photo directories or sign-in sheets, which are inefficient and vulnerable to human error.

1.4 Project Objectives

The first objective of this project is to enable personalized face recognition by delivering customizable identification that includes user-defined additional details such as “Parent of Alice,” “Room 201,” or “VIP identification.” This allows users, such as kindergarten teachers, to instantly recognize parent-child relationships during pickups. The second objective is to address human memory limitations by automating contextual recall, thereby replacing error-prone manual verification methods such as photo directories. This includes the integration of safety checks for sensitive scenarios, for instance, by generating alerts when unrecognized individuals attempt to perform actions like picking up a child.

1.5 Project Scope

This project focuses on developing a personalized face recognition system designed for everyday environments such as kindergartens, hotels, and event venues. The system addresses common challenges, including human memory limitations, operational inefficiencies, and the lack of accessible, user-owned face recognition tools. The project scope encompasses key functionalities in face detection and recognition, image quality control, user interaction, and database management, as well as defined hardware, software, and performance constraints.

The core functionality of the system includes real-time face detection using YuNet (OpenCV) and face recognition powered by DeepFace (FaceNet512), with a similarity threshold set at 0.7. To ensure that only high-quality images are processed, the system integrates sharpness filtering based on Laplacian variance, discarding frames with a value below 1.8. Additionally, motion detection clears recognition queues when rapid movement is detected, preventing the system from processing outdated frames and ensuring responsiveness.

User interaction is supported through guided registration and search workflows for unrecognized individuals, with built-in form validation to maintain input consistency (e.g., "Surname Firstname" format). Upon successful recognition, the system displays personalized details, such as "Parent of Alice" or "Room 201." Recognized names are also announced using text-to-speech (TTS) via gTTS and Pygame, enhancing the user experience.

The backend infrastructure relies on a local PostgreSQL database for storing user profiles, metadata, and image paths. The system automatically handles folder creation and database updates for new or existing users, enabling streamlined data management without manual intervention.

From a technical standpoint, the system operates entirely on local hardware, specifically using a built-in laptop camera with a resolution of 640×480 at 1 FPS. There is no reliance on cloud-based services, ensuring full control over data privacy and availability. The software stack includes Python 3.12 with libraries such as OpenCV, DeepFace, and Flask-SocketIO, and it is deployed with a web-based interface.

In terms of performance, the system achieves an end-to-end recognition latency of under 3 to 5 seconds, with an accuracy of 98.54% under controlled lighting conditions. These results demonstrate the feasibility and responsiveness of the system for personalized face recognition in dynamic, real-world environments.

1.6 Contributions

The first contribution of this project is the development of personalized face recognition. This system enables users, such as teachers, hotel staff, and event staff, to utilize face recognition without relying on centralized databases in everyday scenarios. The system uses local PostgreSQL and file-based image storage with customizable additional details, such as "Parent of Alice," "Room 201," or "VIP identification." For example, hotel staff can record special requests, like a preference to avoid rooms near the elevator, in additional detail. This innovation addresses the accessibility limitations of face recognition systems, which have traditionally been used only by governments, making face recognition more accessible for everyday use.

The second contribution is the implementation of memory-augmented interfaces, which compensate for human memory limits. The system displays names and additional details, such as “Cousin last seen at Chinese New Year,” helping individuals recall context in social settings. This feature enhances personal connectivity and reduces the social awkwardness that can arise from forgotten names or details. It eliminates the errors associated with manual memory recall, providing accurate recognition and related information, thus improving user interaction and reducing potential misidentifications.

The third contribution is the creation of a modular workflow that adapts to various use cases, including kindergartens, hotels, and event venues. This system allows users to configure additional detail fields, such as “Parent of Alice,” “Room 201,” or “VIP identification.” For instance, event staff can log guest information, including seat numbers and section, in the additional details field. This modular approach ensures that one system can support diverse sectors without requiring code changes, offering flexibility for different industries.

1.7 Report Organization

The report is organized into seven chapters. Chapter 1, the Introduction, provides background information, motivations, problem statements, project objectives, project scope, project contributions, and the overall report organization. Chapter 2, the Literature Review, evaluates existing systems in the market, identifying the strengths and weaknesses of each. Chapter 3, System Methodology/Approach, discusses the overall system flow of the project. Chapter 4, System Design, details the hardware and software specifications, along with the module functionalities of the system. Chapter 5, System Implementation, outlines the implementation process and the challenges encountered. Chapter 6, System Evaluation and Discussion, covers the testing setup and results. The final chapter, Chapter 7, provides the conclusion and recommendations of the system.

Chapter 2

Literature Review

2.1 Previous Works on Face Recognition Systems for Storing and Retrieving Information

2.1.1 Clearview AI, Malaysia Immigration Department's Biometric System, and Malaysia National Registration Department (NRD) Biometric Identification System

Existing Systems	Features				
	Face Detection & Recognition	Store Information Based on Face	Retrieve Information Based on Face		Personalized
			1-to-1	1-to-many	
Clearview AI	✓	✓		✓	
Malaysia Immigration Department's Biometric System	✓	✓	✓	✓	
Malaysia National Registration Department (NRD) Biometric Identification System	✓	✓	✓		
Developed System	✓	✓		✓	✓

Table 2.1 Feature Analysis Table of Existing Systems and Developed System

Clearview AI is a powerful face recognition technology widely used by law enforcement agencies in the United States to quickly identify suspects, witnesses, and victims, thereby helping to solve cases more efficiently and enhance community safety [6]. It boasts the largest known facial image database, with over 50 billion images sourced from publicly accessible online platforms such as news media, mugshot websites, and public social media [6]. Clearview AI's system operates on a 1-to-many basis, where a submitted face is compared against its vast database to find potential matches with high accuracy. The platform allows users to group searches, save them, set alerts for new matches, archive inactive investigations, and easily access active cases [6].

The Malaysia Immigration Department’s Biometric System is a key component of Malaysia’s border control and immigration management, using advanced technologies such as face recognition and fingerprint scanning [1]. As part of the Malaysian Immigration System (MyIMMs), it is deployed at immigration checkpoints to verify the identities of travelers, ensuring that only authorized individuals enter or exit the country [1]. During passport applications and renewals, high-resolution facial images, fingerprints, digital signatures, and personal information are collected and stored in a centralized database [1]. This data is later used at border checkpoints to confirm identities, provide relevant information to immigration officers, and strengthen national security by preventing unauthorized access and identity fraud [1]. The system performs both 1-to-many identification, comparing travelers against databases of individuals on watchlists or with prior immigration violations, and 1-to-1 verification, matching a traveler’s biometric data with that stored in their passport [1].

The Malaysia National Registration Department (NRD) Biometric Identification System is a key tool for managing and verifying the identities of Malaysian citizens using biometric data, such as face recognition and fingerprints [2]. Primarily employed during the issuance and renewal of the national identification card, MyKad, the system ensures that each citizen’s identity is unique, authentic, and secure. This biometric system operates on a 1-to-1 verification basis, comparing the biometric data captured at the time of application or renewal with the information stored in a centralized database to prevent identity fraud and duplication [2]. The NRD’s centralized database securely stores personal details, including digital facial images, fingerprints, and demographic information like name, date of birth, and address. This data is used for identity verification and supports various government services, enhancing the accuracy and security of civil registration in Malaysia [2].

Clearview AI, Malaysia's Immigration Department Biometric System, and the National Registration Department (NRD) Biometric Identification System offer advanced face detection, recognition, and metadata management features. These systems accurately identify faces from images or video feeds and match them against stored databases, linking each face with relevant details such as name, age, and contact information.

2.1.2 Limitations of Previous Works

Despite the sophistication of existing face recognition systems, there is a lack of effective solutions designed to assist individuals in recognizing and recalling faces in personalized settings. These systems, which are typically developed for security purposes such as customs control, are not widely accessible and do not cater to everyday use scenarios where remembering people can be challenging, such as in schools, hotels, or events.

This project aims to bridge this gap by creating a more accessible system, focusing on personalization and practicality for daily use. The system offers a customized approach to face detection, recognition, and metadata management, making these technologies usable beyond governmental applications. It delivers personalized recognition by storing user-defined metadata (e.g., "Parent of Alice," "Room 201," "VIP Guest") and solving human memory limitations by automating contextual recall, thereby replacing error-prone manual checks such as photo directories. Additionally, the system integrates safety checks for sensitive scenarios, such as generating alerts for unrecognized pickups, enhancing the overall safety and efficiency in environments requiring face recognition.

2.2 Method Review

Face recognition in computer vision began in the 1960s with early methods focusing on facial feature measurements [7]. Over time, the field evolved through two major approaches: Good Old-Fashioned Artificial Intelligence (GOFAI) and deep learning [7], [8].

GOFAI, prevalent in the 1990s, used rule-based systems and algorithms like Eigenfaces to recognize faces based on geometric and statistical features [7], [8]. These methods relied on manually crafted rules and feature extraction techniques [8]. In contrast, deep learning emerged in the 2000s, utilizing convolutional neural networks (CNNs) to automatically learn and extract features from large datasets, significantly enhancing recognition accuracy [7].

Today, advanced deep learning models such as VGG-Face, OpenFace, DeepID, Dlib, and FaceNet 512 use large datasets and powerful neural network architectures to achieve high accuracy in face recognition [3]. Among these, FaceNet 512 stands out for its key features, including 512-dimensional embeddings, triplet loss optimization, and a ResNet-like architecture [9].

FaceNet 512 is a deep learning-based face recognition model designed to embed facial images into a 512-dimensional Euclidean space [9]. This representation is highly effective for tasks like face verification, recognition, and clustering. It was developed by Google Research, and the original FaceNet framework introduced the concept of 'triplet loss' [9]. This concept ensures that embeddings of the same identity are closer together in the space, while embeddings of different identities are farther apart [9], optimizing the model's performance. Additionally, FaceNet 512 employs a ResNet-like architecture. Its final layer consists of 512 embedding nodes, enabling compact and highly descriptive facial representations. These embeddings facilitate precise recognition and efficient storage [10], making FaceNet 512 ideal for real-time and large-scale applications [9].

The key features of FaceNet 512 include its high accuracy, compact embeddings, scalability, and seamless compatibility. Its accuracy results from training on extensive datasets, achieving state-of-the-art results on benchmarks such as Labelled Faces in the Wild (LFW) [9]. The 512-dimensional embeddings not only support efficient storage and comparisons but also ensure stable performance across varying demographics, lighting conditions, and poses [9].

Furthermore, its seamless integration with frameworks like DeepFace provides developers with an adaptable and practical solution for implementing face recognition in diverse applications, including security systems, personalized user experiences, and attendance tracking [10].

For these reasons, FaceNet 512 has been chosen for the solution, as it offers the perfect balance of accuracy, speed, and adaptability to meet the system's requirements effectively. The face recognition technology is flexible; while the current implementation uses FaceNet 512, other recognition methods can also be employed.

Chapter 3

System Methodology/Approach

3.1 System Flow

The system begins with an Initialization Phase, during which essential hardware and software components are prepared. The camera is initialized using OpenCV, with the resolution configured to 640×480 at 1 FPS—an implementation choice made to reduce CPU load during development. The YuNet face detection model is loaded, and the DeepFace (FaceNet512) recognition model is initialized. A connection to the PostgreSQL database is also established to support profile retrieval and data storage. Concurrently, a WebSocket handshake is performed to establish communication between the frontend (browser) and backend (Flask-SocketIO), enabling real-time interaction.

During the Continuous Capture Loop, the system continuously reads frames from the camera for processing. Movement detection is applied by computing the inter-frame difference using the Euclidean norm. If the movement exceeds a threshold of 50, based on the implemented design to indicate that the previous person has walked away, the system clears the recognition queue to facilitate rapid identification of newly arriving individuals. Frame processing is skipped until motion stabilizes. To ensure image quality, the system calculates the Laplacian variance of each frame and discards frames with a sharpness score of 1.8 or lower, based on thresholds selected during implementation testing.

In the Face Processing Pipeline, YuNet detects faces and returns face coordinates, with the system prioritizing the largest face when multiple faces are present. If no face is detected, the user interface is cleared, and the recognition queue is reset. Once a face is isolated, the system extracts the region of interest and passes it to DeepFace for recognition. If a match is found in the database with a similarity score of 0.7 or higher, the system fetches the associated metadata, such as name and additional details, from PostgreSQL. This information is then displayed on the user interface, and a greeting is played using text-to-speech. The new face image is saved to the local image folder, and the database is updated with the corresponding image path. If the face is unrecognized, the system pauses the capture loop and presents options to either register a new user or search the database by name.

The User Interaction Flows consist of two main pathways. In the Registration Flow, selecting "Register New User" prompts a form requiring input of the individual's name and relevant details. The backend performs input validation, including name formatting (e.g., "Surname Firstname"). Upon submission, the system saves the image, adds a new record to PostgreSQL with the provided metadata and image path, and resumes the capture process. In the Search Flow, the user can choose "Search by Name" and use type-ahead suggestions to locate an existing profile. Once selected, the system saves the current image to the individual’s folder and updates the database before resuming the capture loop.

The Shutdown Sequence ensures proper resource management by releasing the camera handle, closing the PostgreSQL connection, and terminating any active WebSocket threads. This systematic approach ensures data integrity and a clean application exit.

3.2 System Architecture Diagram

Figure 3.1 illustrates the architecture of the system. The system is divided into three major components: Frontend, Backend, and Data Storage. These components interact to deliver a face recognition-based user identification and interaction system.

The Frontend (User Interface Layer) serves as the primary interaction point between users and the system, accessed through a web browser. The web interface includes a live video stream presented in the Video Feed Display area, allowing real-time monitoring. Recognized individuals’ names and associated metadata are dynamically updated and shown in the Name and Additional Detail Panel. Users can get feedback through a Notification Area, which displays system alerts and status messages. Additional user interface components include a “Register New User” form for adding new individuals to the system and a “Search by Name” feature for retrieving existing users. All user actions, such as form submissions and control commands, are transmitted from the browser to the backend for processing.

The Backend (Processing and Logic Layer) is powered by a Flask server that manages communication between the frontend and the processing components. Incoming video frames are first processed by the Image Processor, which performs movement and sharpness calculations to ensure frame quality. Once validated, the frames are passed to the face detection module, where the YuNet model identifies and extracts the Face Region of Interest (ROI) from each frame. The extracted face is then passed to the DeepFace Recognizer, which attempts to identify the individual. When a match is found, the recognized name is returned to the User Interaction module for display and is also forwarded to a Text-to-Speech (TTS) engine to generate an audible greeting. The User Interaction logic further manages the handling of registration/search forms and relays system notifications back to the frontend, ensuring seamless communication between the interface and logic layers.

The Data Storage Layer comprises two key repositories. The PostgreSQL database stores structured information such as users' names, additional metadata, and image file paths. Complementing this, an image folder maintains the actual face images used in the recognition process. When a new user is registered, their profile data is added to the PostgreSQL database while the corresponding face image is saved in the designated image folder. This dual-storage approach ensures efficient retrieval and reliable identification during system operation.

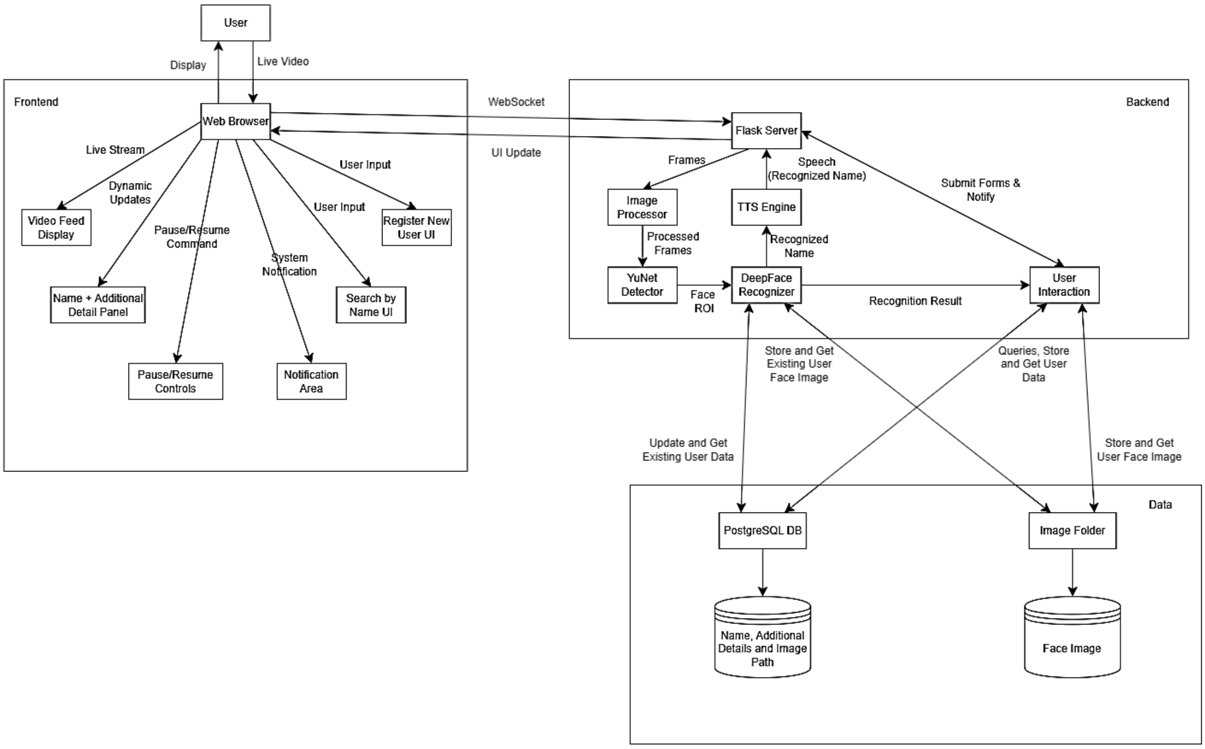


Figure 3.1 Architecture Diagram of Personalized Face Recognition System

3.3 Use Case Diagram

Figure 3.2 illustrates the use case diagram of the personalized face recognition system. The system involves interactions between the user and the backend processes. Users can perform several actions through the interface, including viewing the live video feed, searching for individuals by name, registering a new user, and cancelling ongoing operations.

On the system side, a series of automated processes are triggered in response to user actions. The system processes incoming images, detects faces using the YuNet detection model, and performs face recognition using DeepFace (FaceNet512). It manages the database and image folder by updating records and storing image paths. Upon successful recognition, the system displays the recognized details on the interface and activates the text-to-speech engine to greet the individual. In cases where a face is unrecognized, the system presents options for further user interaction, such as registration or search.

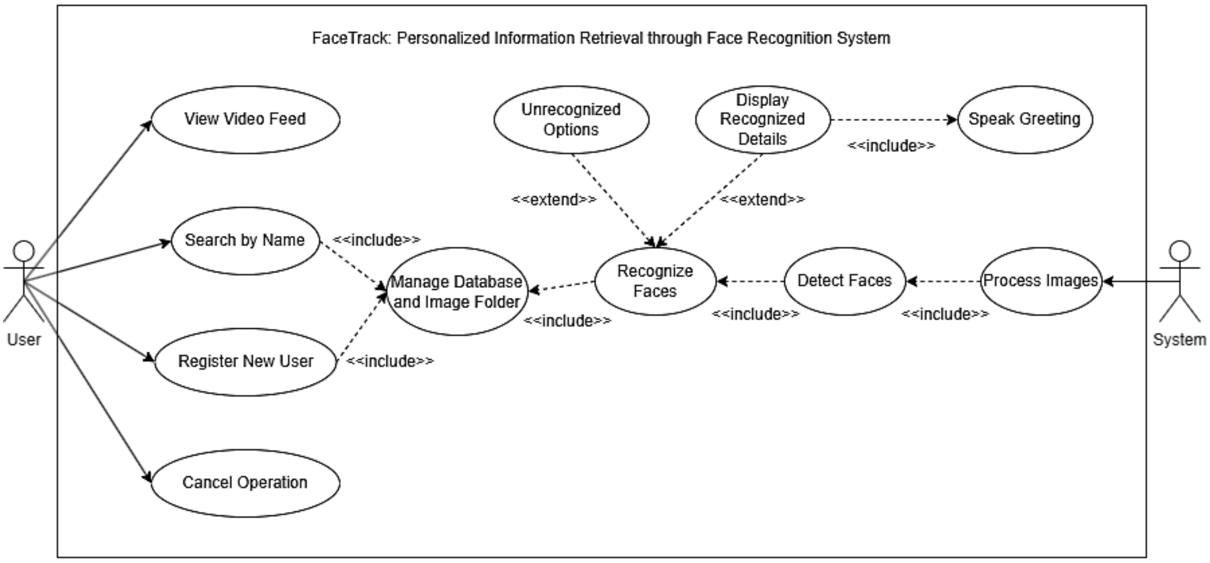


Figure 3.2 Use Case Diagram of Personalized Face Recognition System

3.4 Use Case Description

Use Case ID	UC001	**Version**	1.0
Use Case	View Video Feed		
Purpose	To allow the user to view real-time camera feed		
Actor	User		
Trigger	User accesses the system interface		
Precondition	Camera is connected and system is running		
Scenario Name	**Step**	**Action**	
Main Flow	1	User opens the system interface	
	2	System initializes camera feed	
	3	System displays live video stream	
	4	User observes the video feed	
	5	User closes the system interface (exit condition)	
Rules		-	
Author		Ooi Yunn Suen	

Table 3.1 Use Case Description - View Video Feed

Use Case ID	UC002	**Version**	1.0
Use Case	Search by Name		
Purpose	To manually identify existing users when face recognition fails (false-negative cases)		
Actor	User		
Trigger	User selects "Search by Name" for unrecognized face		
Precondition	Face detected but not recognized, database and image folder contain user's records		
Scenario Name	**Step**	**Action**	
Main Flow	1	User clicks "Search by Name" option	
	2	System shows search interface with face preview	
	3	User starts typing name	
	4	System provides real-time name suggestions	
	5	User selects valid name from suggestions	
	6	System retrieves user details from database	
	7	System displays user information	

	8	System saves new face image to selected user's image folder and update image path to database
	9	System shows success notification " Name found. Face successfully saved!"
	10	System resumes video capture
	11	System hides pause overlay
Alternate Flow- No Select Name	5.1	System shows "Please select a name"
Rules		-
Author		Ooi Yunn Suen

Table 3.2 Use Case Description - Search by Name

Use Case ID	UC003	**Version**	1.0
Use Case	Register New User		
Purpose	To register new users in the system		
Actor	User		
Trigger	User selects "Register New User" for unrecognized face		
Precondition	Face detection active, unrecognized face present		
Scenario Name	**Step**	**Action**	
Main Flow	1	User clicks "Register New User"	
	2	System shows registration form with face preview	
	3	User enters valid full name (Surname Firstname)	
	4	User enters additional details	
	5	User submits form	
	6	System validates name format	
	7	System creates new user image folder	
	8	System saves face image to image folder	
	9	System adds record to database	
	10	System shows success notification "Successfully registered [Name of the New User]	
	11	System resumes video capture	
	12	System hides pause overlay	

Alternate Flow- Invalid Name Format	6.1	System shows " Please use format: Surname Firstname (e.g., Ng Ei Wei)"
Rules		-
Author		Ooi Yunn Suen

Table 3.3 Use Case Description - Register New User

Use Case ID	UC004	**Version**	1.0
Use Case	Cancel Operation		
Purpose	To allow user to abort ongoing processes (registration/search)		
Actor	User		
Trigger	User clicks "Cancel" during registration/search		
Precondition	System is in paused state (after unrecognized face)		
Scenario Name	**Step**	**Action**	
Main Flow	1	User clicks "Cancel" button	
	2	System clears all input forms	
	3	System resumes video capture	
	4	System hides pause overlay	
	5	System resets recognition state	
Rules		-	
Author		Ooi Yunn Suen	

Table 3.4 Use Case Description - Cancel Operation

Use Case ID	UC005	**Version**	1.0
Use Case	Process Images		
Purpose	To preprocess frames for face detection		
Actor	System		
Trigger	New video frame captured		
Precondition	Camera feed active, three frames buffered (prev/current/next)		
Scenario Name	**Step**	**Action**	
Main Flow	1	System calculates movement value between frames	
	2	System converts current frame to grayscale (Movement <=50)	
	3	System applies Gaussian blur	

	4	System calculates sharpness score
	5	System passes frame to Detect Faces (Sharpness >1.8)
Alternate Flow- If Movement >50	1.1	System discard frames
	1.2	System clears recognition queue
Alternate Flow- If Sharpness <=1.8	4.1	System discard frames
Rules		-
Author		Ooi Yunn Suen

Table 3.5 Use Case Description - Process Images

Use Case ID	UC006	**Version**	1.0
Use Case	Detect Faces		
Purpose	To detect human faces in video frames		
Actor	System		
Trigger	Receives sharpness validated frame form Process Images (UC005)		
Precondition	Face detection model (face_detection_yunet_2023mar.onnx) loaded, frame sharpness validated		
Scenario Name	**Step**	**Action**	
Main Flow	1	System sets input size based on frame dimensions	
	2	System runs face detection model	
	3	System sorts faces by size (largest first)	
	4	System returns sorted face coordinates	
	5	System triggers Recognize Faces (UC007)	
Alternate Flow- If No Face Detected	2.1	System discard frame	
	2.2	System clears recognition queue	
Rules		-	
Author		Ooi Yunn Suen	

Table 3.6 Use Case Description - Detect Faces

Use Case ID	UC007	**Version**	1.0
Use Case	Recognize Faces		
Purpose	To identify detected faces against known users		
Actor	System		

Trigger	Receives face coordinates from Detect Faces (UC006)	
Precondition	Face recognition model (DeepFace/Facenet512) loaded, face coordinates available, database accessible	
Scenario Name	**Step**	**Action**
Main Flow	1	System extracts face region from frame
	2	System runs DeepFace recognition
	3	System finds match with similarity ≥70%
	4	System retrieves user details from database
	5	System triggers Display Recognized Details (UC009)
	6	System triggers Speak Greeting (UC010)
Alternate Flow- If Similarity <=70%	3.1	System marks face as “Unknown”
	3.2	System triggers Unrecognized Options (UC008)
Rules		-
Author		Ooi Yunn Suen

Table 3.7 Use Case Description - Recognize Faces

Use Case ID	UC008	**Version**	1.0
Use Case	Unrecognized Options		
Purpose	To handle unrecognized faces by showing user choices		
Actor	System		
Trigger	Face recognition similarity <=70%		
Precondition	Face detected but unknown		
Scenario Name	**Step**	**Action**	
Main Flow	1	System pauses video capture	
	2	System display pause overlay	
	3	System displays face preview	
	4	System shows options: “Search by Name”/ “Register New User”	
	5	System waits for user selection	
Alternate Flow- If No Selection within 30s	4.1	System auto resumes capture	
	4.2	System hide pause overlay	
Rules		-	

Author	Ooi Yunn Suen

Table 3.8 Use Case Description - Unrecognized Options

Use Case ID	UC009	**Version**	1.0
Use Case	Display Recognized Details		
Purpose	To show recognized user information		
Actor	System		
Trigger	Face recognition similarity >70%		
Precondition	User matched in database		
Scenario Name	**Step**	**Action**	
Main Flow	1	System retrieves user details from database	
	2	System updates UI with user’s name and additional details	
	3	Triggers Speak Greeting (UC010)	
Rules		-	
Author		Ooi Yunn Suen	

Table 3.9 Use Case Description - Display Recognized Details

Use Case ID	UC010	**Version**	1.0
Use Case	Speak Greeting		
Purpose	To audibly greet recognized users via TTS		
Actor	System		
Trigger	Face recognition similarity >70%		
Precondition	TTS initialized, user details available		
Scenario Name	**Step**	**Action**	
Main Flow	1	System generates text ("[Name]")	
	2	System converts text to speech via gTTS	
	3	System plays audio through speakers	
Alternate Flow- If Audio Busy	3.1	System queues greeting and plays when available	
Rules		-	
Author		Ooi Yunn Suen	

Table 3.10 Use Case Description - Speak Greeting

Use Case ID	UC011	**Version**	1.0
Use Case	Manage Database and Image Folder		
Purpose	To handle all database operations and face image storage in the filesystem		
Actor	System		
Trigger	Called by Search by Name (UC002), Register New User (UC003) and Recognize Faces (UC007)		
Precondition	DB connection established, image folder accessible		
Scenario Name	**Step**	**Action**	
Main Flow	1	Receives query request (search/save/update)	
	2	Validates SQL parameters	
	3	Executes transaction	
	4	Returns results to caller	
Alternate Flow- If Query is Save/Update	1.1	Receives face image + user's name for storage	
	1.2	Creates user-specific subfolder if missing (Database/[Name]/)	
	1.3	Saves face image with timestamp ([Name]_YYYYMMDD_HHMMSS.jpg)	
	1.4	Updates database with new image path	
Alternate Flow- If Connection Lost	3.1	System attempts reconnection	
Rules		- Image folder path: C:/Users/ys/Desktop/fypproject/FYP2/Database/ - Timestamp format: YYYYMMDD_HHMMSS	
Author		Ooi Yunn Suen	

Table 3.11 Use Case Description - Manage Database and Image Folder

3.5 Activity Diagram

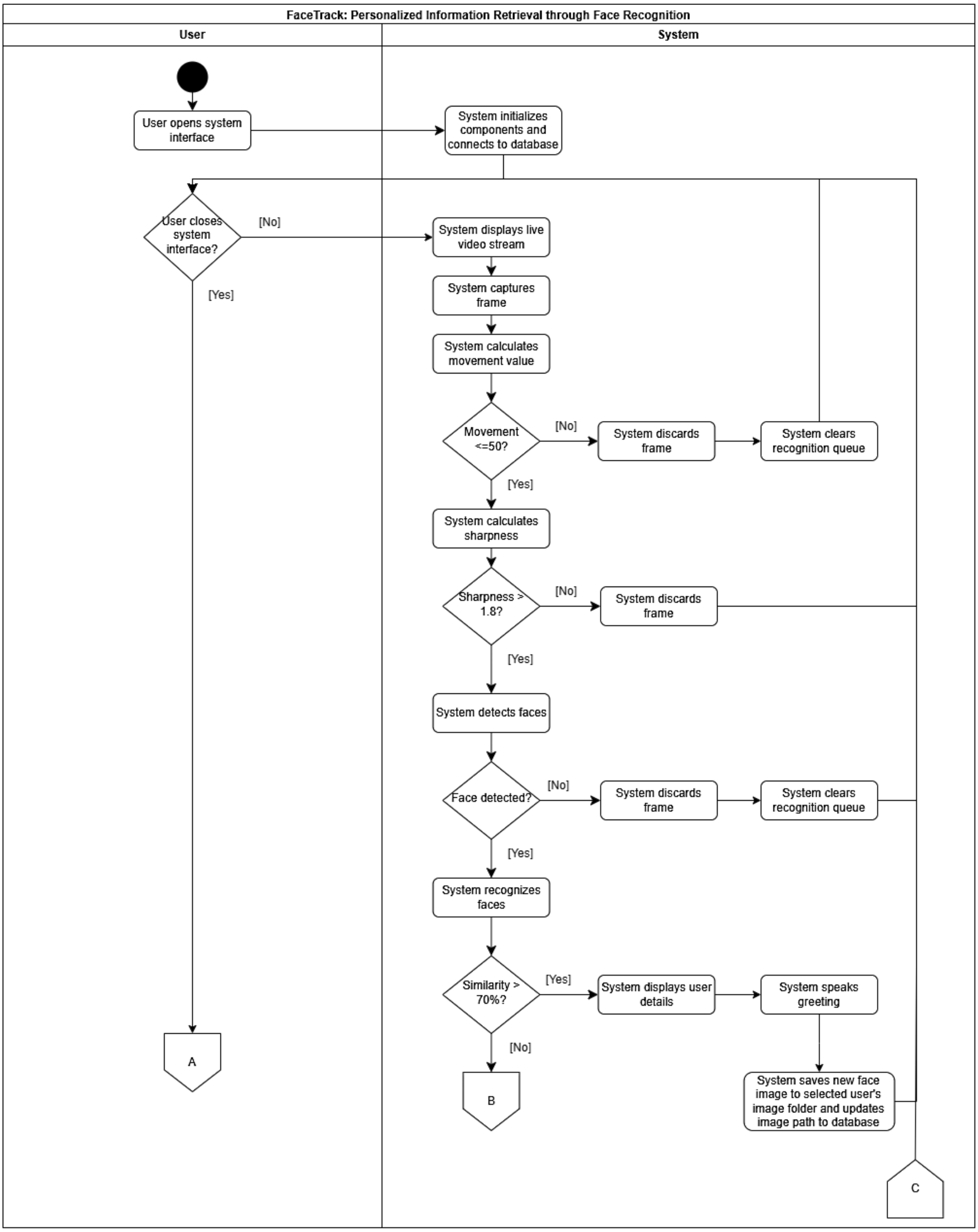

FaceTrack: Personalized Information Retrieval through Face Recognition
User
System
User opens system interface
System initializes components and connects to database
User closes system interface?
[No]
[Yes]
System displays live video stream
System captures frame
System calculates movement value
Movement <=50?
[No]
System discards frame
System clears recognition queue
[Yes]
System calculates sharpness
Sharpness > 1.8?
[No]
System discards frame
[Yes]
System detects faces
Face detected?
[No]
System discards frame
System clears recognition queue
[Yes]
System recognizes faces
Similarity > 70%?
[Yes]
System displays user details
System speaks greeting
System saves new face image to selected user's image folder and updates image path to database
[No]
A
B
C

Figure 3.3 Activity Diagram (Part A) of Personalized Face Recognition System

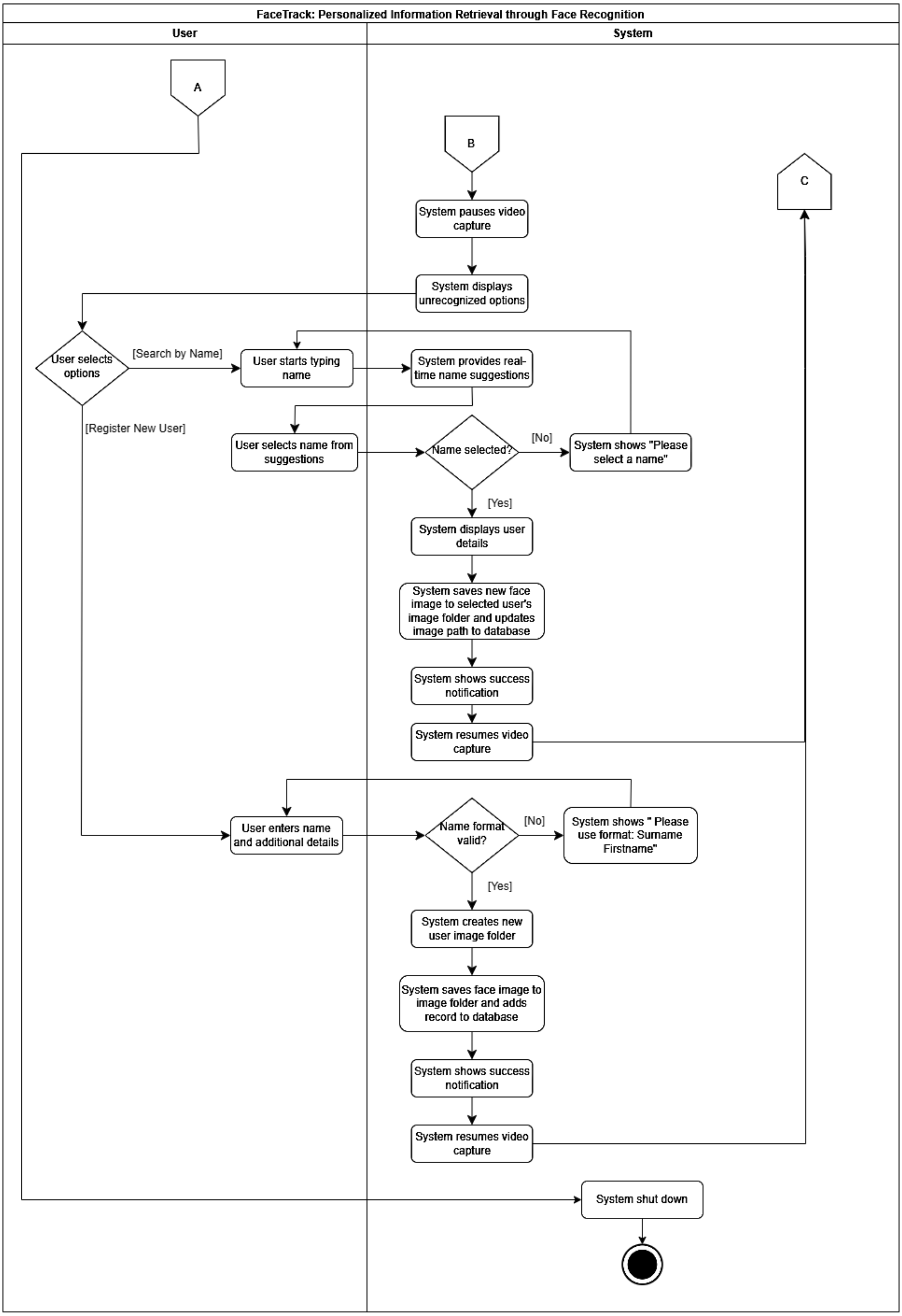

FaceTrack: Personalized Information Retrieval through Face Recognition
User
System
A
B
C
System pauses video capture
System displays unrecognized options
User selects options
[Search by Name]
User starts typing name
System provides real-time name suggestions
[Register New User]
User selects name from suggestions
Name selected?
[No]
System shows "Please select a name"
[Yes]
System displays user details
System saves new face image to selected user's image folder and updates image path to database
System shows success notification
System resumes video capture
User enters name and additional details
Name format valid?
[No]
System shows " Please use format: Surname Firstname"
[Yes]
System creates new user image folder
System saves face image to image folder and adds record to database
System shows success notification
System resumes video capture
System shut down

Figure 3.4 Activity Diagram (Part B) of Personalized Face Recognition System

Chapter 4

System Design

4.1 Flowchart Diagram

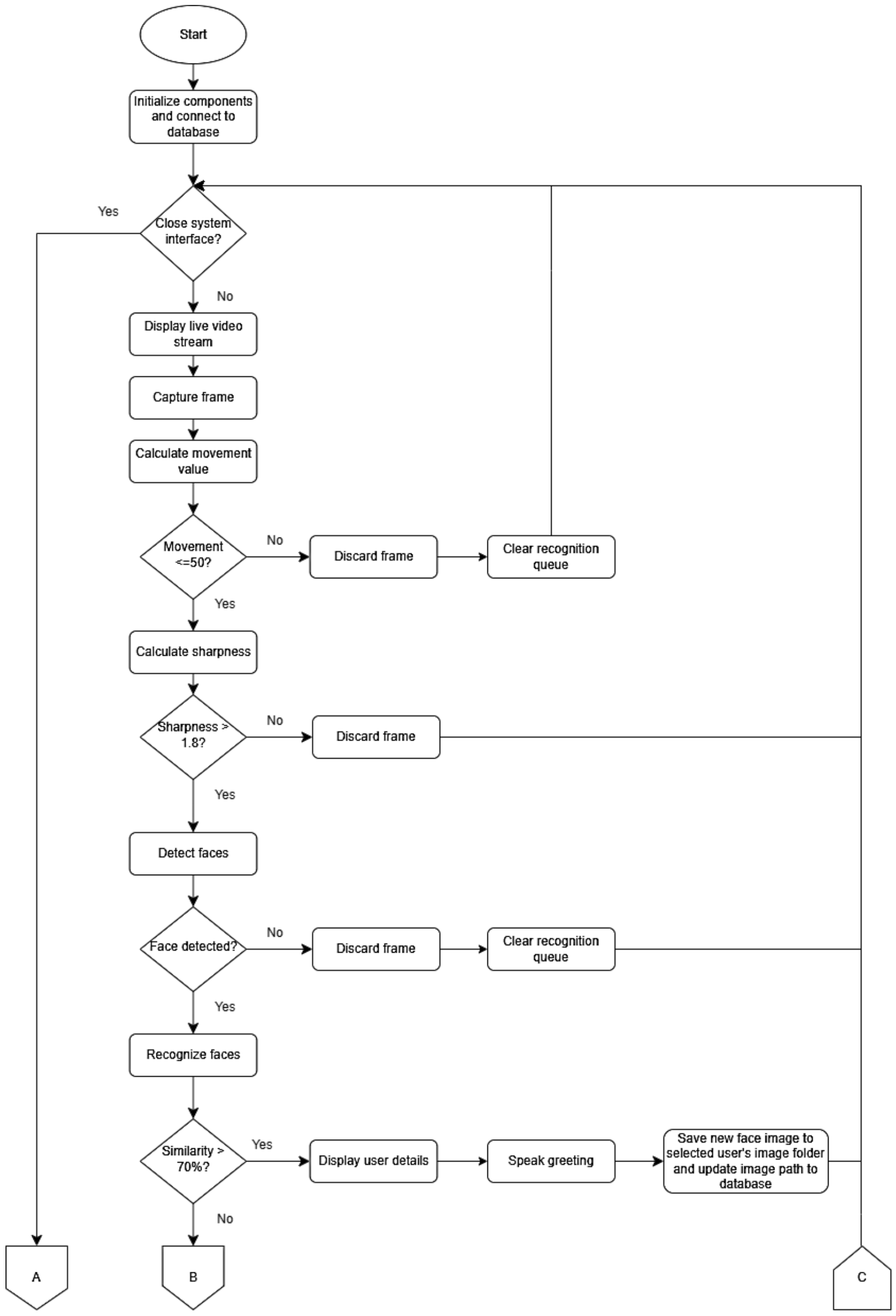


Figure 4.1 Flowchart Diagram (Part A) of Personalized Face Recognition System

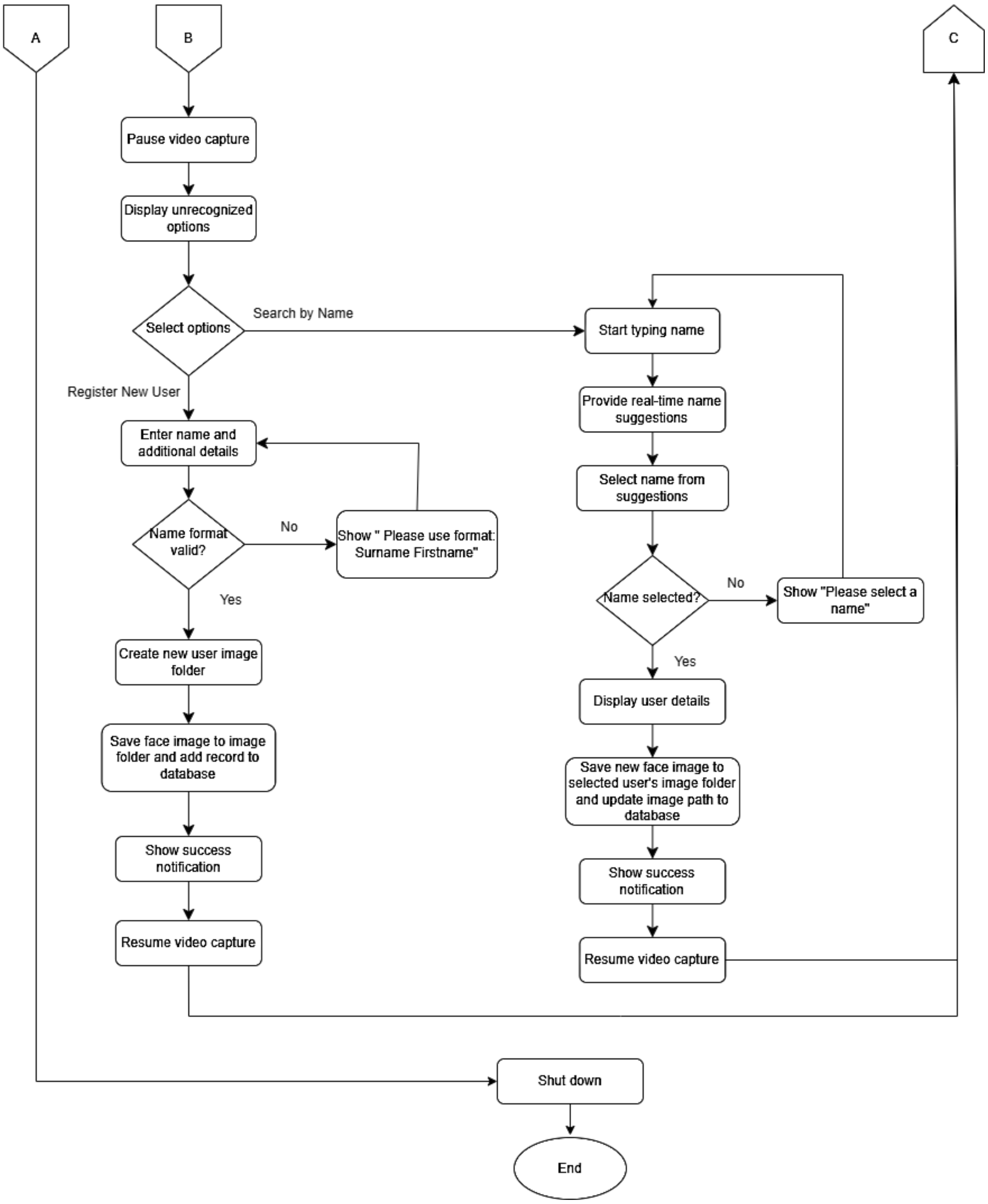

A
B
C
Pause video capture
Display unrecognized options
Select options
Search by Name
Register New User
Enter name and additional details
Name format valid?
No
Show " Please use format: Surname Firstname"
Yes
Create new user image folder
Save face image to image folder and add record to database
Show success notification
Resume video capture
Start typing name
Provide real-time name suggestions
Select name from suggestions
Name selected?
No
Show "Please select a name"
Yes
Display user details
Save new face image to selected user's image folder and update image path to database
Show success notification
Resume video capture
Shut down
End

Figure 4.2 Flowchart Diagram (Part B) of Personalized Face Recognition System

4.2 System Block Diagram

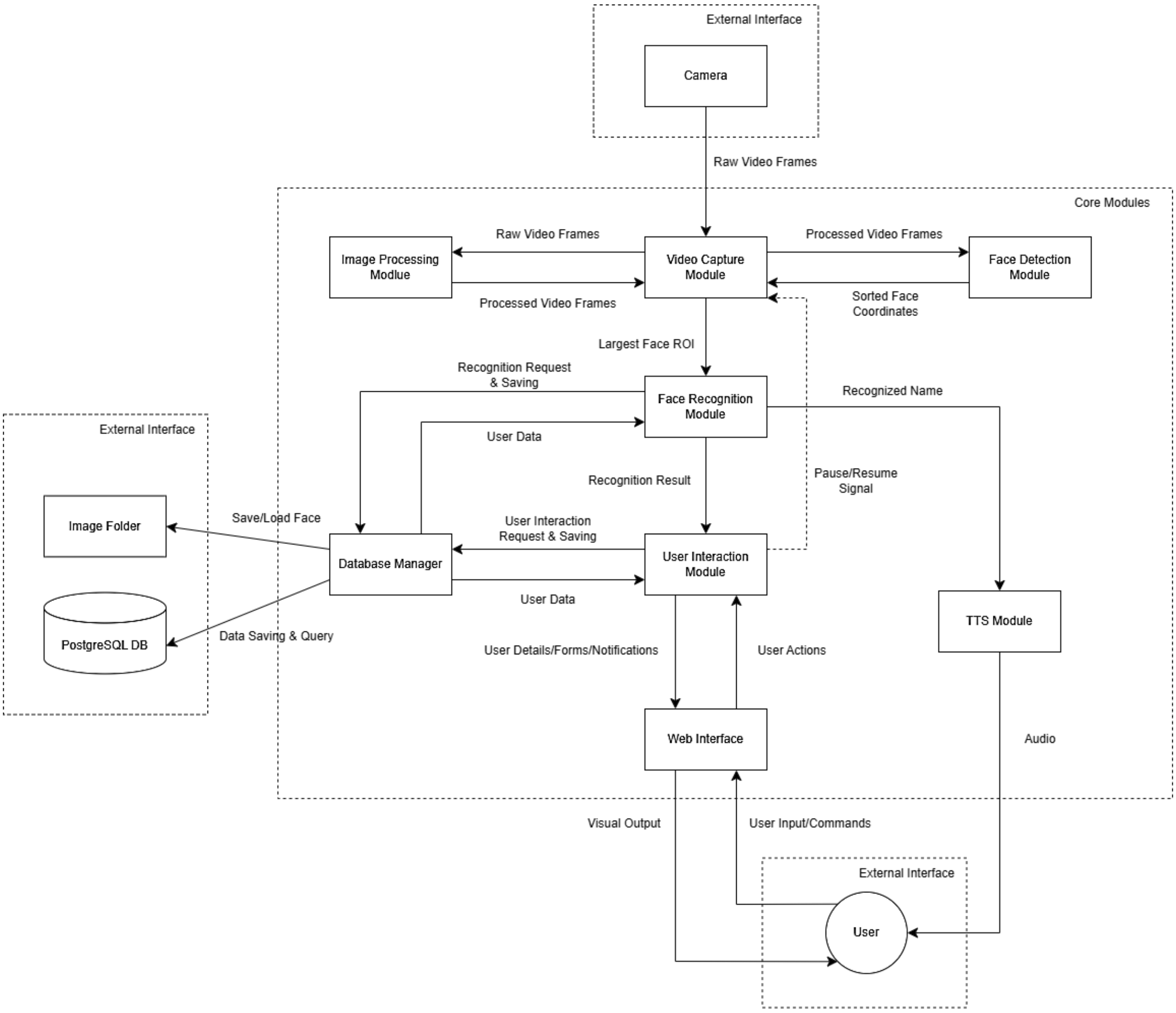


Figure 4.3 System Block Diagram of Personalized Face Recognition System

4.3 System Components Specifications

The personalized face recognition system was developed and tested using a laptop with a built-in camera. The camera supported a minimum resolution of 640x480 and was configured to capture frames at 1 FPS to reduce CPU load. Media Foundation was used as the interface for Windows compatibility. The processing unit consisted of an AMD Ryzen 7 4800H CPU and 16GB of RAM, which provided smooth system operation. Storage was handled by a 477GB SSD, offering ample space for storing database entries and facial images. Audio output for the text-to-speech feature was successfully tested using both laptop speakers and headphones.

The system ran on Windows 11 and was developed using Python 3.12. Several essential libraries and frameworks were employed: OpenCV (cv2) for image handling and face detection, DeepFace for facial recognition, Flask and Flask-SocketIO for handling the web interface and real-time communication, psycopg2 for PostgreSQL database interaction, gTTS and Pygame for speech synthesis and playback, and NumPy for numerical operations. PostgreSQL version 16.8 was used as the database, with a dedicated users table designed to store names, metadata, image paths, and timestamps. An index was created on the name field to improve query speed.

The Video Capture Module managed frame acquisition from the camera, with integrated movement and sharpness detection to skip low-quality or redundant frames. It supported pausing and resuming based on system state and was designed to reconnect automatically in case of camera failure.

The Face Detection Module utilized the YuNet model from OpenCV for real-time face detection. It was capable of handling multiple faces per frame and prioritized the largest face. Additional logic was implemented to manage cases where no face or partial faces were detected.

The Face Recognition Module employed the Facenet512 model via DeepFace to perform identity matching. A similarity threshold of 0.7 was used to validate known faces. The module processed detections asynchronously using a queue and logged recognition events, including duration, database size, and match results.

The Database Module maintained a persistent connection to PostgreSQL with retry logic in case of failure. It supported full CRUD operations for user metadata and coordinated with the image folder for storing face images. Cleanup routines were added to safely close connections and release resources during shutdown.

The User Interaction Module handled registration and search functionalities for unrecognized faces. It validated input formats (e.g., "Surname Firstname"), managed timeouts for user actions, and sent dynamic forms and notifications to the frontend through WebSocket communication.

The Text-to-Speech (TTS) Module used Google Text-to-Speech (gTTS) for name synthesis and Pygame for audio playback. The module included logic to prevent duplicate greetings and supported graceful termination on exit.

The Web Interface was divided into frontend and backend components. The frontend displayed a real-time video feed, dynamically updated user information, and provided interfaces for registration and search. The backend, implemented using Flask and Socket.IO, handled bidirectional communication between the client and server.

During system launch, it was necessary to ensure the PostgreSQL server was running and that the Python virtual environment was activated with all required dependencies installed. The main application was started through main.py, and the web interface was made accessible via http://localhost:5000.

4.4 System Components Interaction Operations

The startup sequence of the system began with the execution of main.py, which launched the Flask and Socket.IO server via app.py. The WebApp class was initialized to set up key components, including VideoCapture (which activated the camera and started the capture_thread), FaceDetector (YuNet), and FaceRecognizer (DeepFace). These models were preloaded to reduce response latency. Additionally, the DatabaseManager established a connection to the PostgreSQL server, while the frontend (index.html) initiated a WebSocket connection and requested the video feed. During the camera warmup phase, the system skipped five frames to stabilize exposure and focus.

The main operation loop functioned continuously while the system was active. Each frame captured by VideoCapture was processed by ImageProcessor, which calculated movement. If the movement level was below or equal to 50 and sharpness exceeded 1.8, the frame was deemed suitable for further processing. Otherwise, the system resets the recognition queue and internal states to avoid redundant operations. Valid frames were passed to FaceDetector, which used the YuNet model to identify facial regions. If no faces were detected, the recognition queue was cleared again. When one or more faces were detected, the system extracted the largest face, converted it to RGB format, and passed it to the recognition queue for FaceRecognizer. This face was also stored as last_detected_face to facilitate later registration or search interactions.

The FaceRecognizer operated on a separate worker thread, processing the queued faces asynchronously. If a face was matched with a similarity score of 0.7 or higher, user details were retrieved from the database using DatabaseManager, and the information was emitted to the frontend via Socket.IO. The recognized image was saved to the image folder, and its path was appended to the corresponding database entry. A greeting was then generated using the TTS module, provided it had not already been triggered recently. In cases where the face was not recognized, the system labeled it as “Unknown,” paused the frame capture, and emitted the show_unrecognized_options event to present registration or search options to the user.

User interaction flows included registration and search functionalities for unrecognized faces. In the registration process, the frontend displayed a form through the show_registration_form event. Once the user submitted their name and details, the backend validated the input format (e.g., "Ng Ei Wei") and saved the facial image using DatabaseManager.save_face_to_image_folder(). The new user data was recorded in the database via save_to_database(), and a success notification was emitted to resume video capture. In the search flow, the frontend presented a search form with name suggestions. Upon name selection, the backend verified the input, searched the database, saved the image to the user's folder, and updated the database with the new image path. Success notifications were also emitted to confirm the action and resume scanning. Cancellation events, such as cancel_search or cancel_registration, cleared all temporary states and resumed frame capture.

The notification system was used to deliver system feedback to the frontend. It captured both backend errors (e.g., FaceRecognitionError, DatabaseConnectionError) and user-related events (e.g., registration success or input failure). These messages were formatted as notifications and sent to the frontend via the notification event.

Pause and resume mechanisms were employed to manage system behavior based on real-time conditions. The system paused automatically when an unrecognized face was detected or when the user engaged with registration or search forms. Capture resumed after successful interaction or when timeouts and cancellations occurred. This was controlled using VideoCapture.set_pause(True/False) to lock or unlock frame processing accordingly.

All database operations followed a CRUD model. New users were created through save_to_database(), which inserted names, details, and image paths. Search and read functionalities were handled by search_in_database() and get_all_names(). Existing entries were updated by appending new image paths. To ensure consistency and prevent data corruption during concurrent access, all database operations were enclosed in a threading.Lock() to maintain thread safety.

The system also included a cleanup and shutdown procedure. When clients disconnected from the frontend, the WebApp monitored the active connection count and triggered cleanup if no clients remained. This process involved releasing the camera, closing the database connection, and terminating both the face recognition and TTS threads, ensuring that no resources were left open or idle.

Chapter 5
System Implementation

5.1 Hardware Setup

The system was deployed using a built-in laptop camera, which eliminated the need for external mounting. The camera supported a minimum resolution of 640x480 and accommodated higher resolutions when required. To optimize processing efficiency, the frame rate was configured to 1 frame per second. Media Foundation was utilized as the video capture interface due to its compatibility with Windows systems. The laptop was positioned strategically to face the area of interest, such as an entrance gate or user interaction zone, ensuring effective coverage for recognition tasks.

Processing was handled by an AMD Ryzen 7 4800H CPU, a high-performance 8-core processor capable of managing simultaneous tasks, including real-time face detection, recognition, and database operations. The system was equipped with 16GB DDR4 RAM, which facilitated the concurrent operation of multiple modules such as DeepFace recognition, the Flask web server, and image processing tasks without significant performance degradation. A 477GB SSD provided fast read/write speeds, enabling rapid access to image data and smooth interaction with the PostgreSQL database.

Audio output was managed through either built-in speakers or external headphones, depending on availability. This component was essential for delivering name-based greetings using the gTTS engine, played through Pygame.

Lastly, networking requirements were minimal, as the web interface, powered by Flask and WebSocket, ran entirely on the local machine. No internet connection was necessary for the system's core functionalities. However, appropriate firewall configurations were ensured, particularly keeping port 5000 open to allow smooth internal HTTP and WebSocket communication.

5.2 Software Setup

The system was developed and tested on Windows 11, which provided compatibility with the Media Foundation framework, essential for camera interfacing. This operating system also supported the development and execution of Python applications and Flask-based web servers, making it suitable for a locally hosted real-time recognition system.

The Python environment was configured using Python 3.12, selected to ensure compatibility with updated libraries and enhanced language features. Installation was performed via the official Python installer from python.org. A dedicated virtual environment was created and activated to isolate dependencies and prevent conflicts with system-level packages, allowing consistent and reproducible behavior during development and deployment.

Several essential Python libraries and frameworks were employed to implement core functionalities. OpenCV (cv2) was used for image capture and preprocessing, while DeepFace handled facial recognition by generating and comparing embeddings. Flask and Flask-SocketIO powered the web interface and managed real-time data communication between the frontend and backend. psycopg2 facilitated database interaction with PostgreSQL. For audio output, gTTS generated speech from recognized names, and Pygame was used to play the resulting audio files. NumPy supported numerical operations involved in image processing.

The system incorporated two primary models for face detection and recognition. YuNet, an OpenCV-based detector, was chosen for its lightweight structure and efficient performance in live video scenarios. For recognition, Facenet512, accessed through DeepFace, was employed to generate facial embeddings and perform similarity comparisons. A similarity threshold of 0.7 was defined to determine recognition matches based on empirical testing.

The web interface consisted of a real-time video feed and dynamic user interactions. The frontend employed WebSocket technology to stream video and update interface elements like registration/search forms and notifications. The backend used Flask routes to handle page rendering and data submission, while Flask-SocketIO maintained real-time communication for recognition updates and user feedback.

The PostgreSQL database, version 16.8, was used to manage user information. The schema included a users table with fields for name, details, image_path, and timestamp. The name field was indexed to allow faster lookup operations, particularly during face search and match verification. It was essential to start the PostgreSQL service before running the application.

Text-to-Speech (TTS) functionality was integrated to provide audio feedback. Recognized names were converted into .mp3 files using gTTS, and Pygame was used to handle playback. A duplicate prevention mechanism was implemented to avoid repeating greetings for the same individual during a session, improving user experience and reducing redundancy.

To launch the system, the PostgreSQL server had to be running and the Python virtual environment activated. The application was started by executing main.py, after which the user interface could be accessed locally through the address http://localhost:5000.

5.3 Setting and Configuration

The system utilized a built-in webcam, which was accessed through OpenCV's cv2.VideoCapture interface. The resolution was configured to 640x480 by default, and the frame rate was set to 1 frame per second to reduce processing load while maintaining sufficient temporal accuracy. To ensure compatibility with Windows 11, the Media Foundation API (cv2.CAP_MSMF) was selected as the preferred backend.

For movement detection, a frame differencing method was employed. This approach calculated pixel-level differences between consecutive frames to identify significant motion. To filter out unusable frames, sharpness was calculated using the Laplacian variance method, which effectively prevented the capture of blurry images, ensuring only high-quality images proceeded to the recognition phase.

Face detection was carried out using the YuNet model, which is based on the OpenCV DNN module. This model was chosen due to its real-time performance and low computational overhead, making it suitable for integration with lightweight systems.

For face recognition, the system leveraged the DeepFace library and used the Facenet512 model. This model generated facial embeddings, which were then compared using cosine similarity to evaluate identity matches. A similarity threshold of 0.7 was established through experimentation, striking a balance between false positives and recognition accuracy.

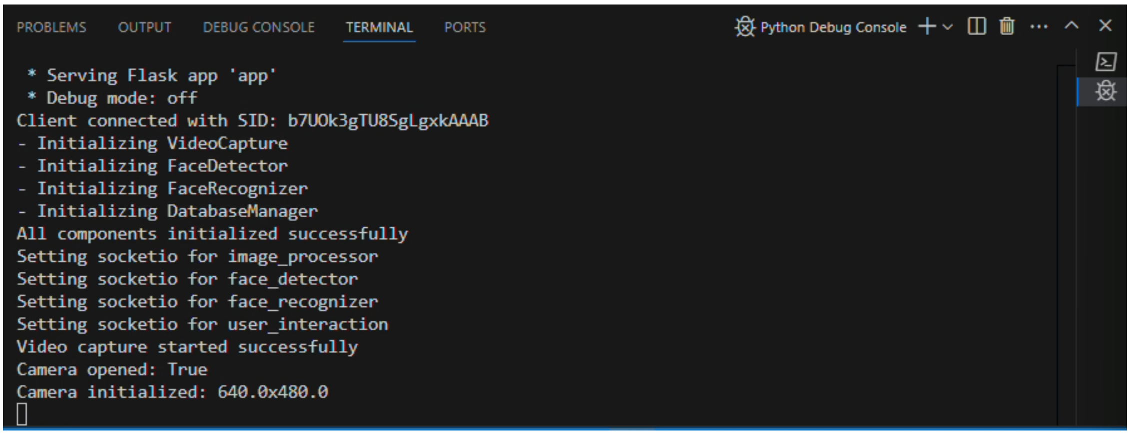
The PostgreSQL database was used for storing user data, with connection parameters including localhost as the host, port 5432, and credentials configured for local access. The target database, face_recognition_db, included a users table with fields such as name, details, image_path, and timestamp. An index was created on the name field to facilitate faster search and retrieval operations during the recognition process.

Text-to-Speech (TTS) functionality was implemented using the gTTS library, which generated audio in MP3 format. The playback of audio greetings was managed through the Pygame mixer. The system was designed to greet each recognized user only once per session to prevent redundancy. Additionally, audio files were deleted immediately after playback to conserve storage space.

The web interface was hosted on a Flask server integrated with Flask-SocketIO. The server ran on port 5000, with debug mode enabled during development to assist in troubleshooting. WebSocket communication allowed the backend to push real-time updates—including live video frames, user information, and status alerts—directly to the frontend without the need for page reloads, improving responsiveness and interactivity.

5.4 System Operation (with Screenshot)

The system began by initializing the camera, loading the YuNet face detection model via OpenCV DNN, loading the DeepFace (Facenet512) face recognition model, connecting to the PostgreSQL database, and launching the Flask web server with Socket.IO for real-time communication. Once initialization was complete, the user interface became accessible through a web browser, displaying a real-time video feed and a section to show user details retrieved from the database after successful face recognition.



```
 * Serving Flask app 'app'
 * Debug mode: off
Client connected with SID: b7UOk3gTU8SgLgxkAAAB
- Initializing VideoCapture
- Initializing FaceDetector
- Initializing FaceRecognizer
- Initializing DatabaseManager
All components initialized successfully
Setting socketio for image_processor
Setting socketio for face_detector
Setting socketio for face_recognizer
Setting socketio for user_interaction
Video capture started successfully
Camera opened: True
Camera initialized: 640.0x480.0
```

Figure 5.1 Initialization of Personalized Face Recognition System

If the face was not recognized, the system paused the video capture and displayed a pause overlay along with a notification. It then showed the unrecognized face image and presented options to either search for an existing name or register the new user.

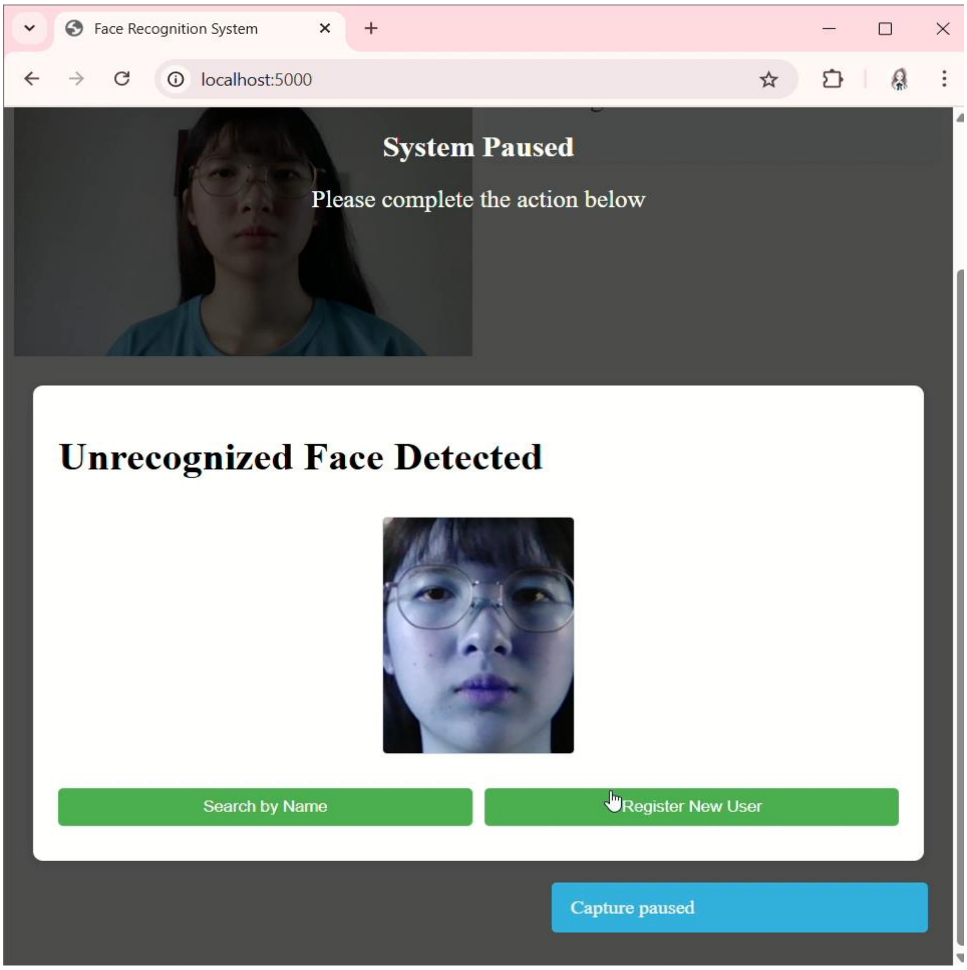
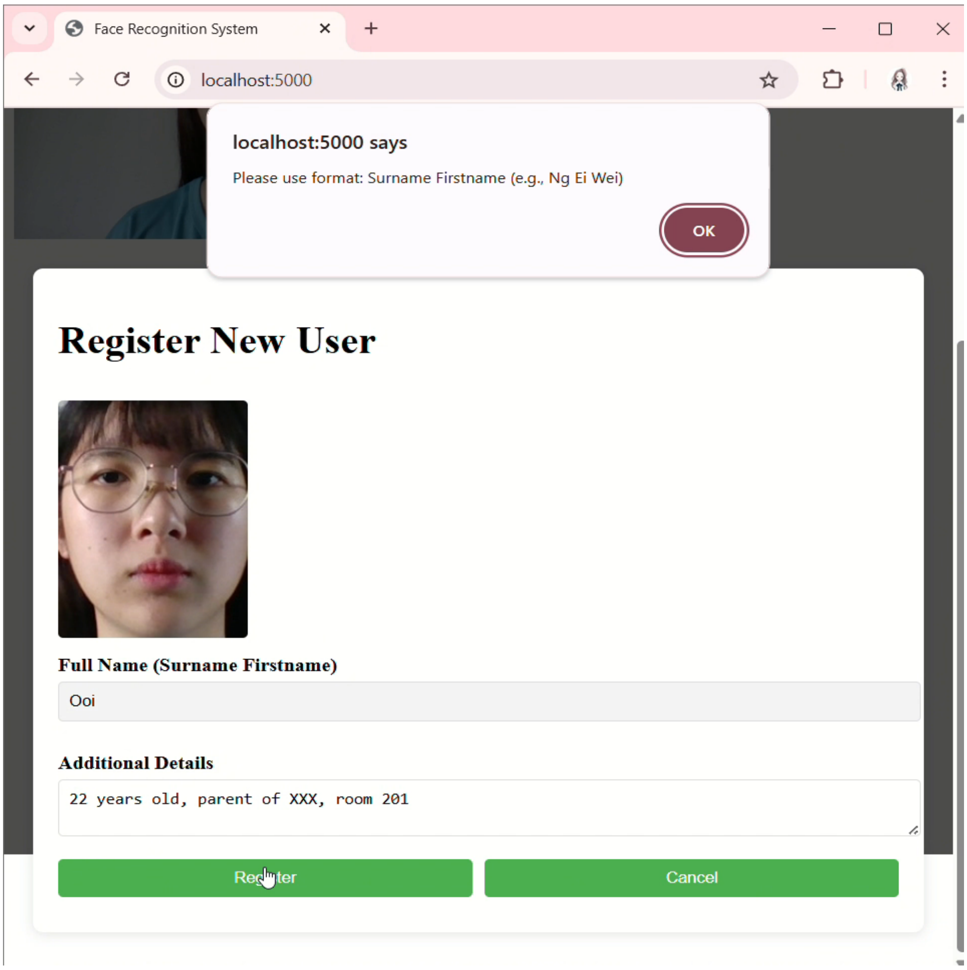


Figure 5.2 Unrecognized Options of Personalized Face Recognition System

If the user chose to register a new user, the system displayed the unrecognized face image along with sections to input the name and additional details. If the name was not entered in the correct format (Surname Firstname), the system alerted the user.



localhost:5000 says

Please use format: Surname Firstname (e.g., Ng Ei Wei)

Register New User

Full Name (Surname Firstname)

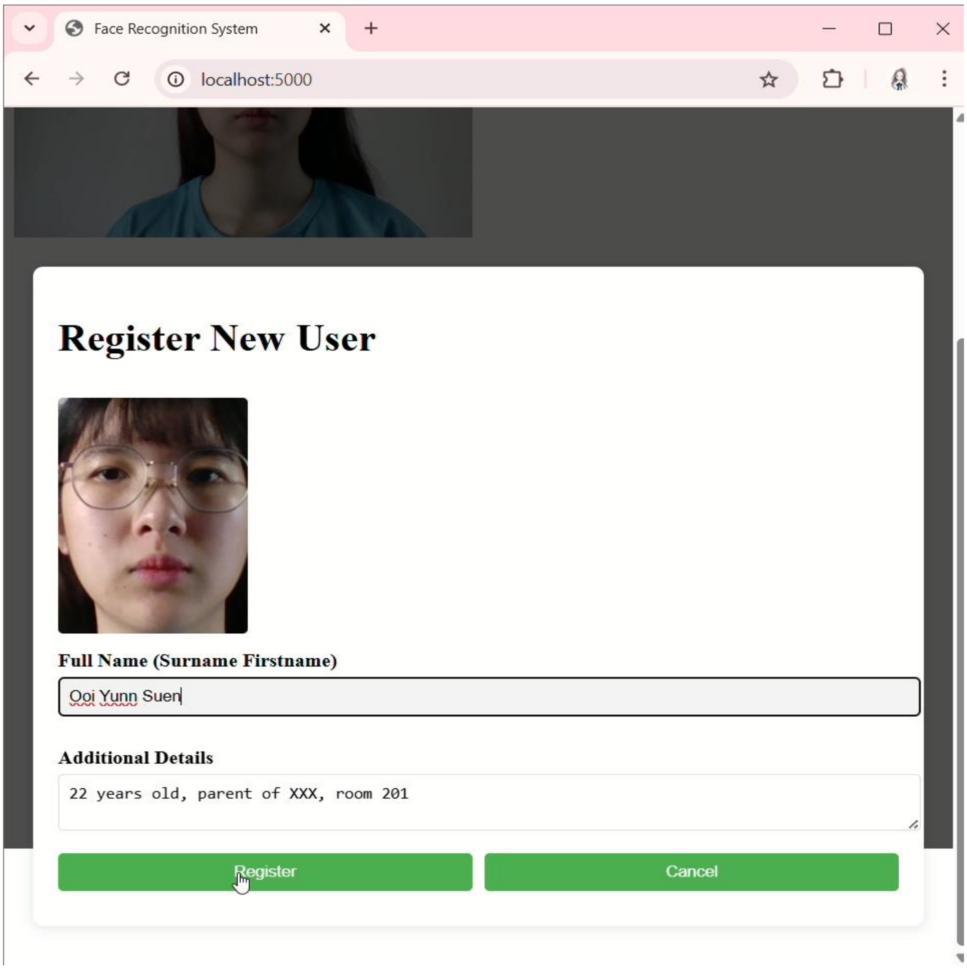
Ooi

Additional Details

22 years old, parent of XXX, room 201

Figure 5.3 Register New User (Wrong Name Format) of Personalized Face Recognition System

The user typed the name in the correct format along with additional details, then clicked Register.



Register New User

Full Name (Surname Firstname)

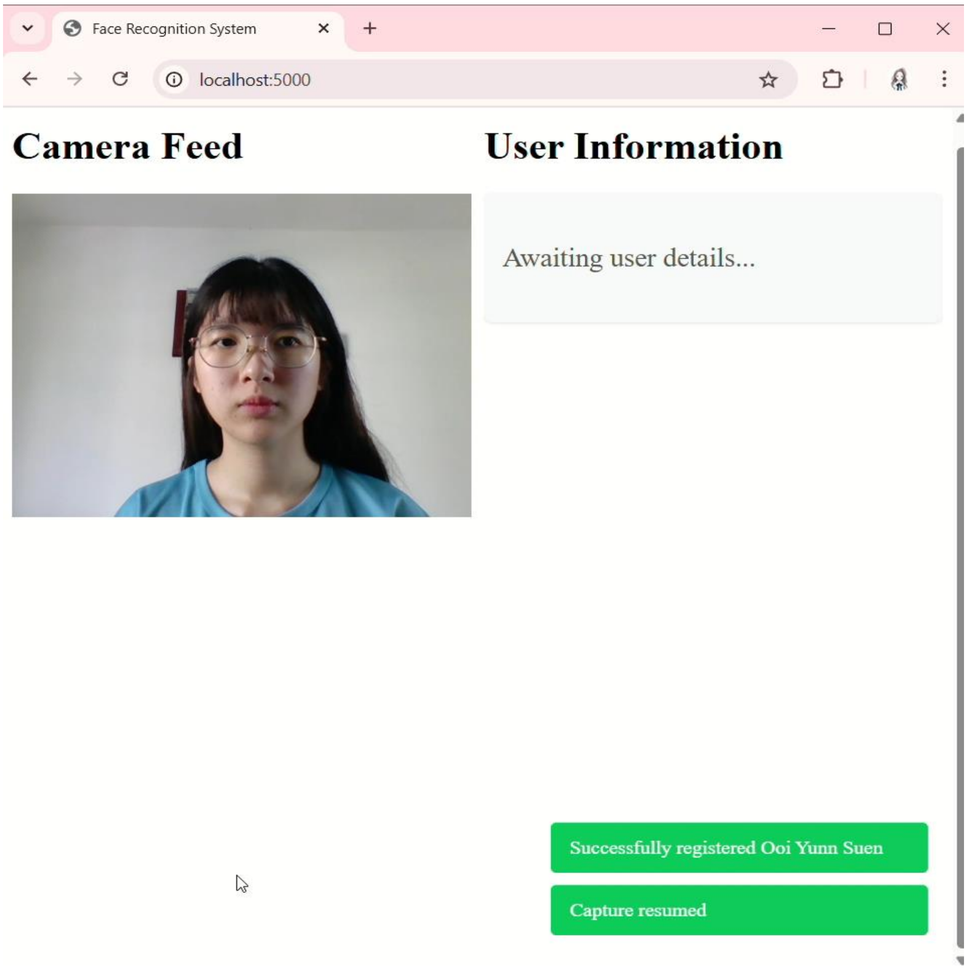
Ooi Yunn Suen

Additional Details

22 years old, parent of XXX, room 201

Figure 5.4 Register New User of Personalized Face Recognition System

After clicking 'Register,' the system created a new image folder and saved the captured face image to it. It then added a record with the name, additional details, and image path to the database. Once the process was complete, a notification appeared: "Successfully registered [name]." The system then resumed video capture.



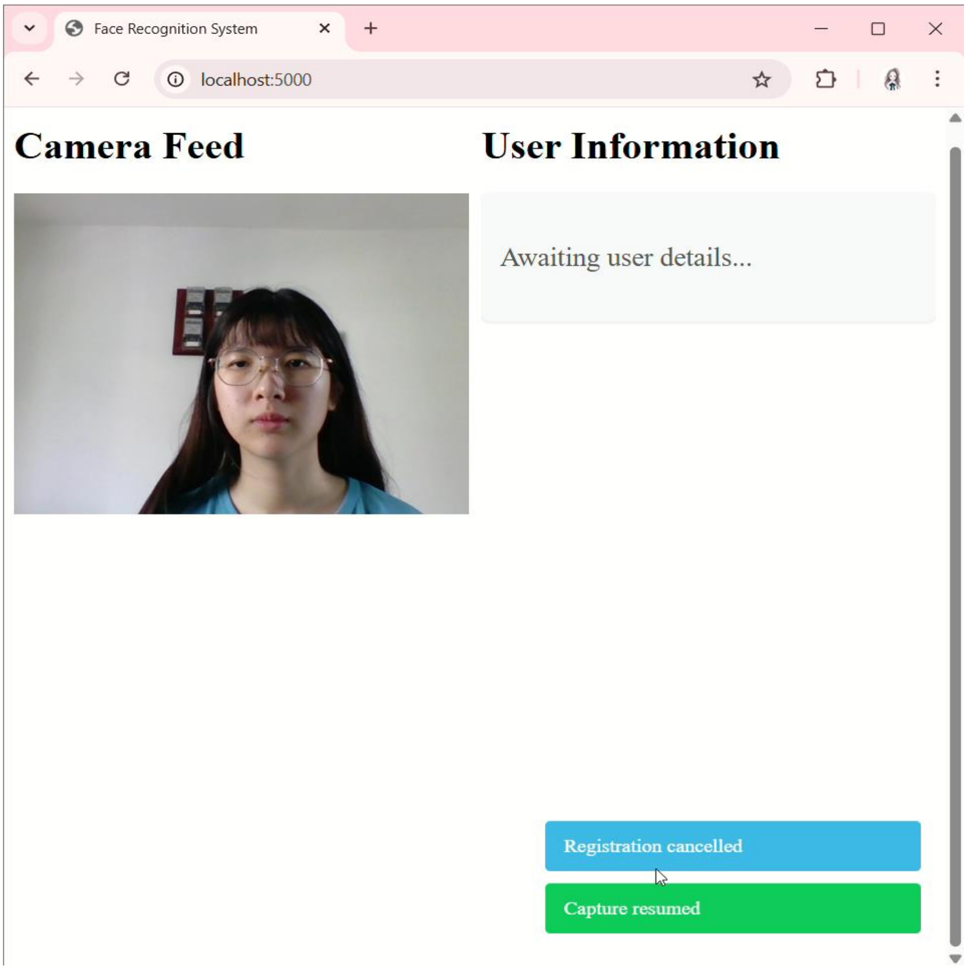
Awaiting user details...

Successfully registered Ooi Yunn Suen

Capture resumed

Figure 5.5 Register New User (Success) of Personalized Face Recognition System

If the user pressed 'Cancel' during registration, the system resumed video capture and displayed a notification: "Registration cancelled."



Camera Feed

User Information

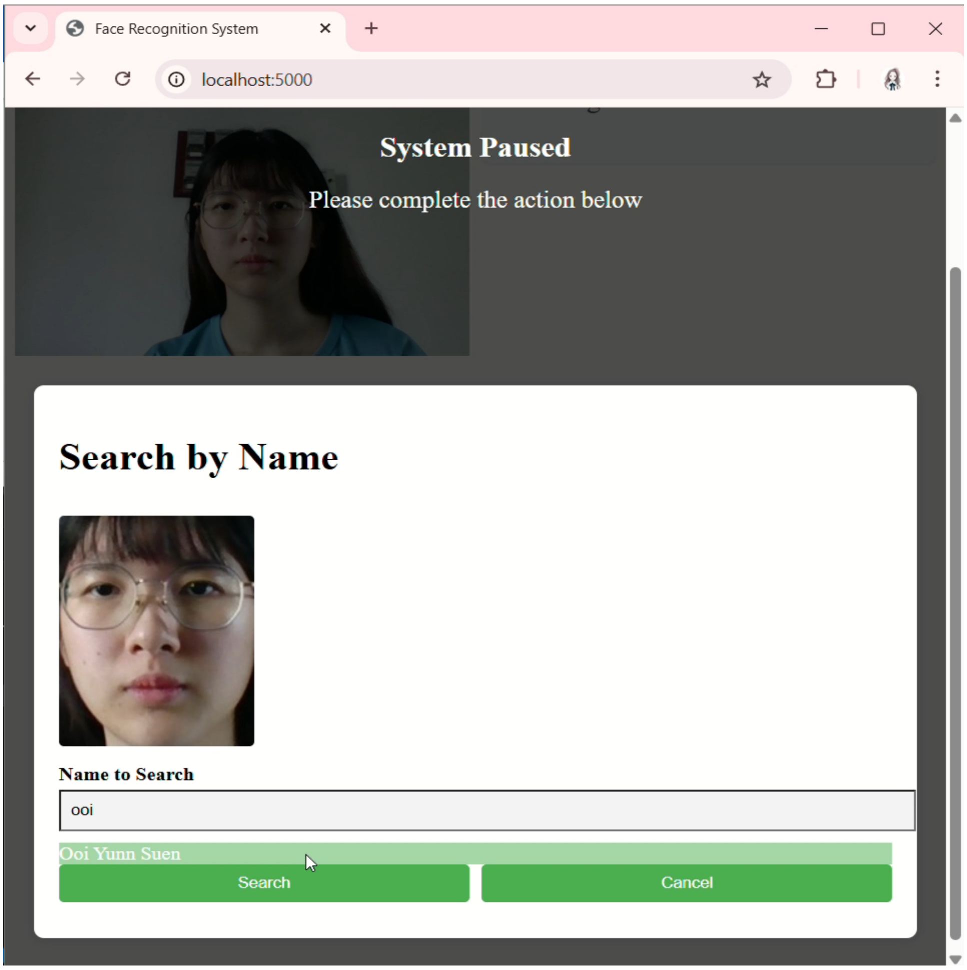
Awaiting user details...

Registration cancelled

Capture resumed

Figure 5.6 Register New User (Cancel) of Personalized Face Recognition System

If the user chose to search by name, the system displayed the unrecognized face image along with a search box containing name suggestions for selection. Users typed part of the name, and the system automatically suggested matching names from the database. They selected a suggested name and clicked 'Search'.



System Paused

Please complete the action below

Search by Name

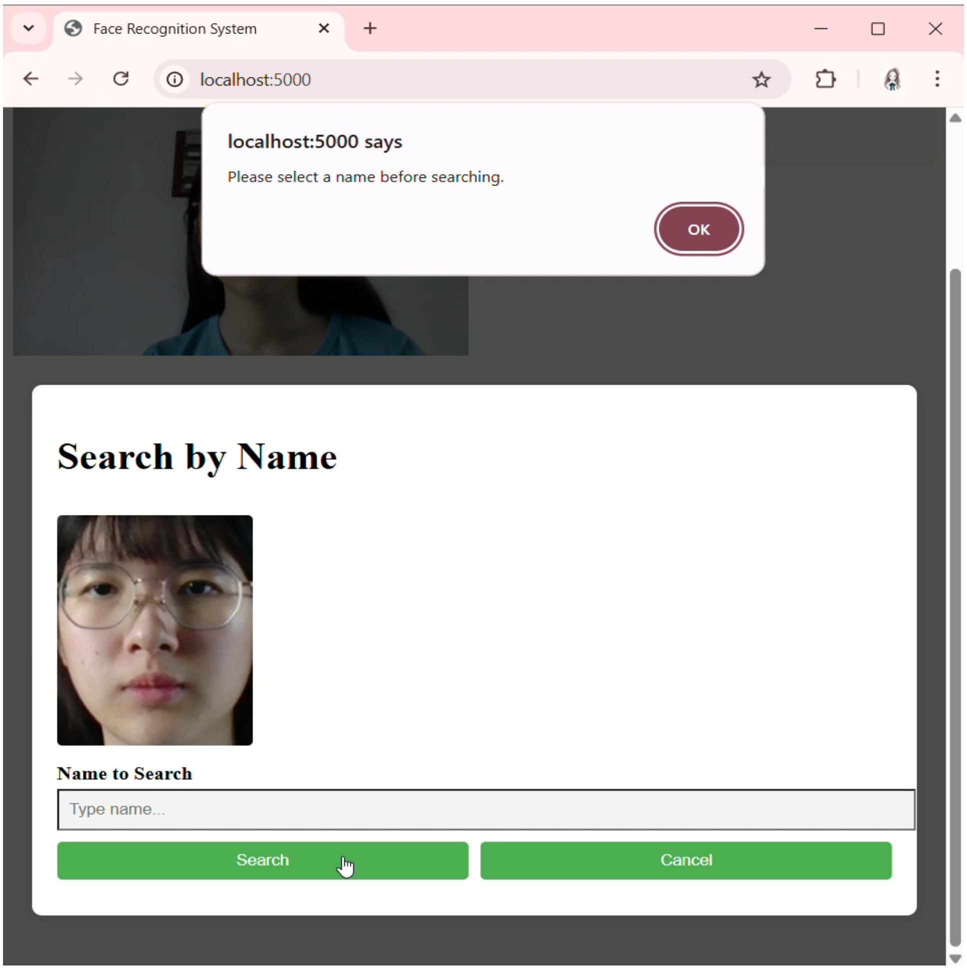
Name to Search

ooi

Ooi Yunn Suen

Figure 5.7 Search by Name of Personalized Face Recognition System

The system alerted the user if no name was selected, and the search button was pressed.



localhost:5000 says

Please select a name before searching.

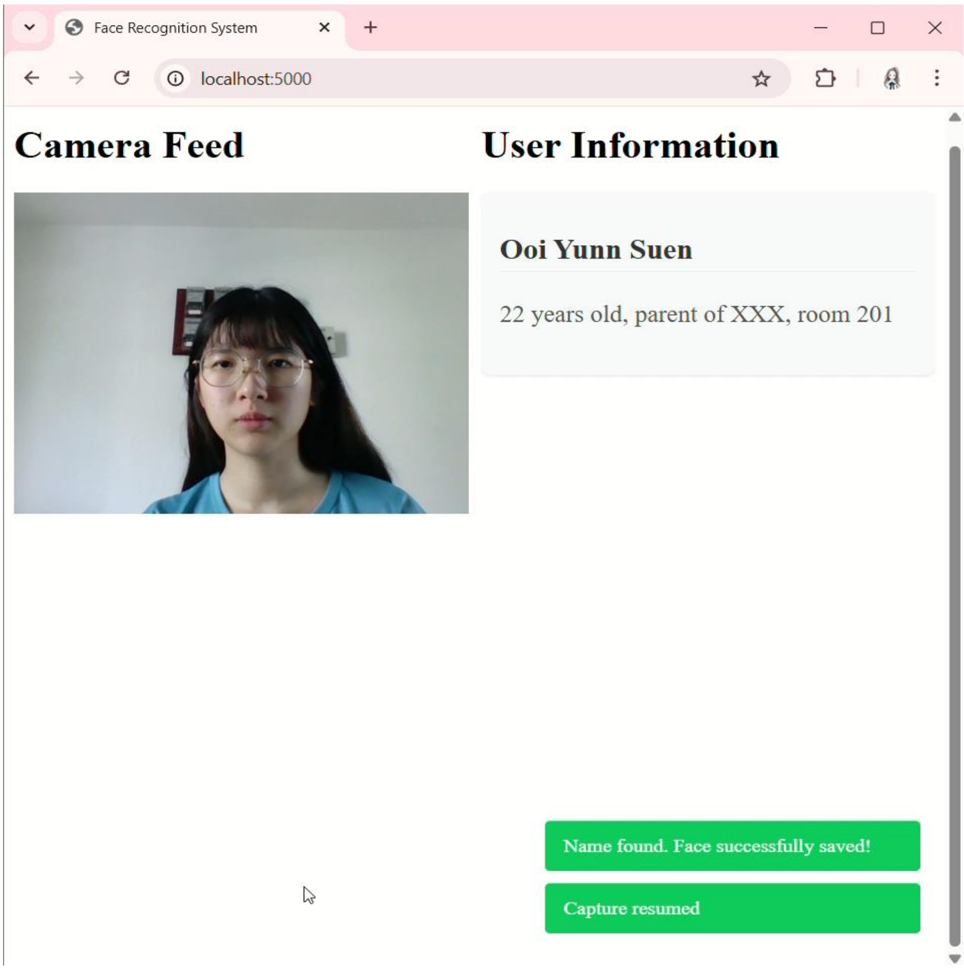
Search by Name

Name to Search

Type name...

Figure 5.8 Search by Name (Empty Search) of Personalized Face Recognition System

After clicking ‘Search’, the system saved the face image to the selected user's existing image folder and updated the image path in the database. Once saving was complete, it displayed a notification: "Name found. Face successfully saved!" Then, it resumed video capture and displayed the selected user's details.



Camera Feed

User Information

Ooi Yunn Suen

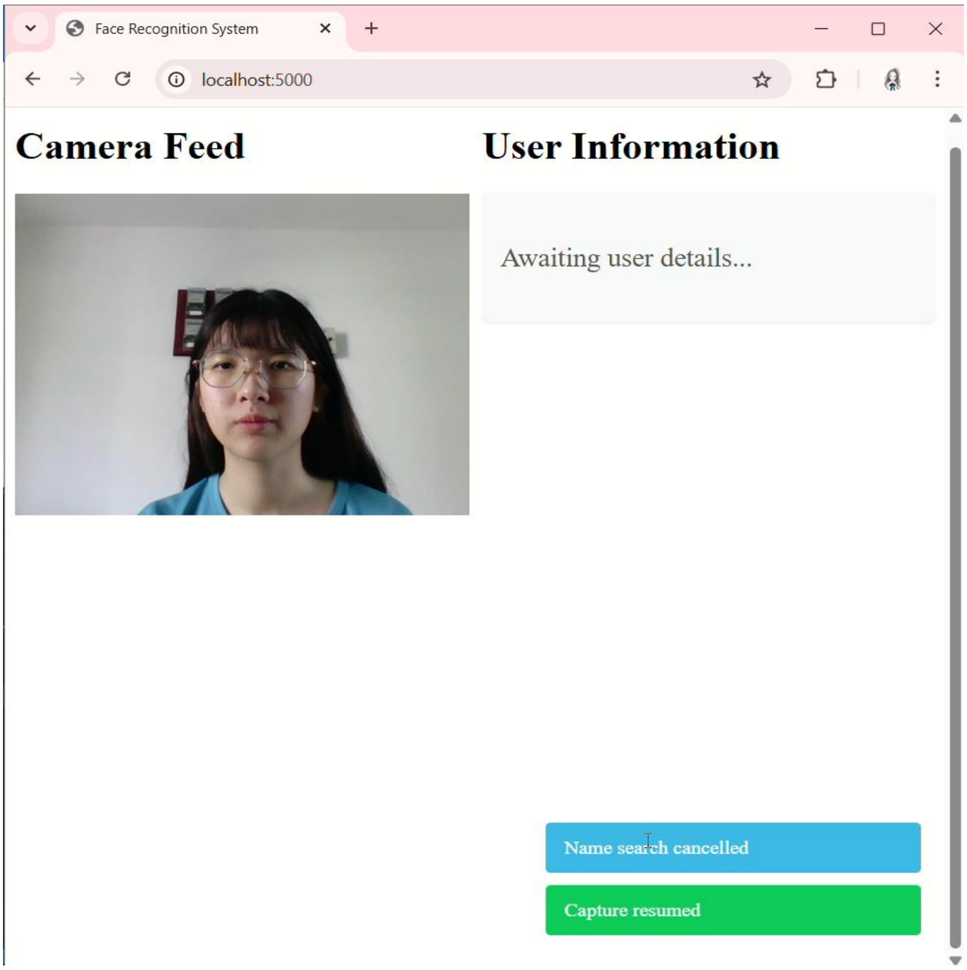
22 years old, parent of XXX, room 201

Name found. Face successfully saved!

Capture resumed

Figure 5.9 Search by Name (Success) of Personalized Face Recognition System

If the user pressed cancel during the search, the system resumed video capture and displayed the notification: "Name search cancelled."



Camera Feed

User Information

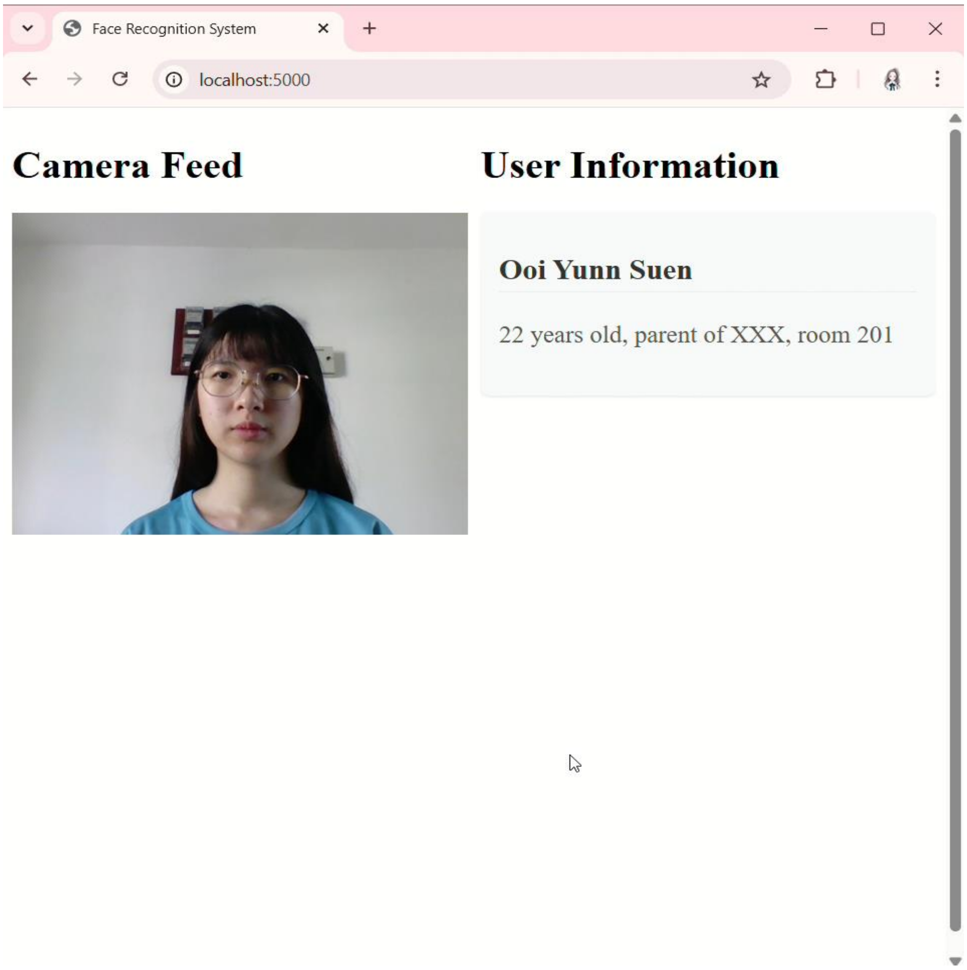
Awaiting user details...

Name search cancelled

Capture resumed

Figure 5.10 Search by Name (Cancel) of Personalized Face Recognition System

If the detected face matched a known user in the database (with a similarity score > 0.7), the system displayed their name and details. It then spoke the recognized name via TTS. The system saved the face image to the image folder and updated the image path in the database.



Camera Feed

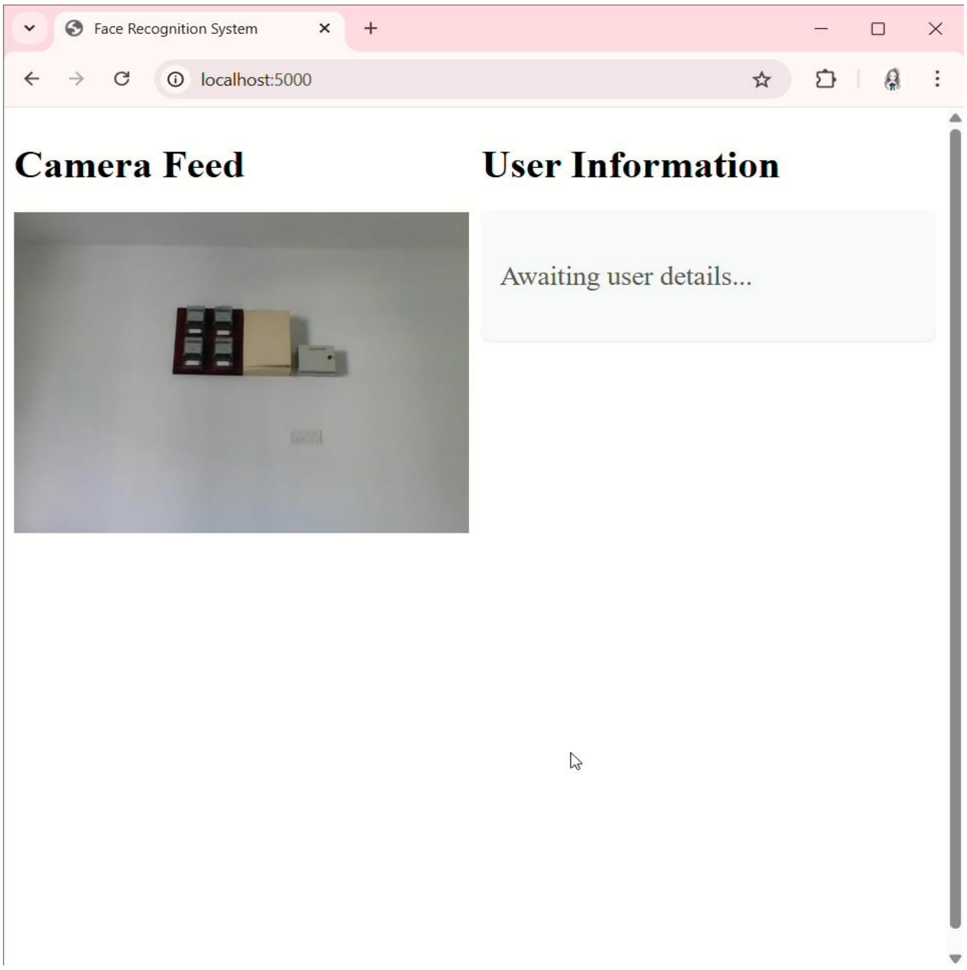
User Information

Ooi Yunn Suen

22 years old, parent of XXX, room 201

Figure 5.11 Recognized Face of Personalized Face Recognition System

If no face was detected, the system cleared the displayed details and reset the recognition queue. This reset allowed the system to immediately process new faces that entered the camera's view.



Camera Feed

User Information

Awaiting user details...

Figure 5.12 Empty Frame of Personalized Face Recognition System

If multiple faces were detected, the system prioritized and processed only the largest face. This ensured that in scenarios with many people, the closest face was processed.

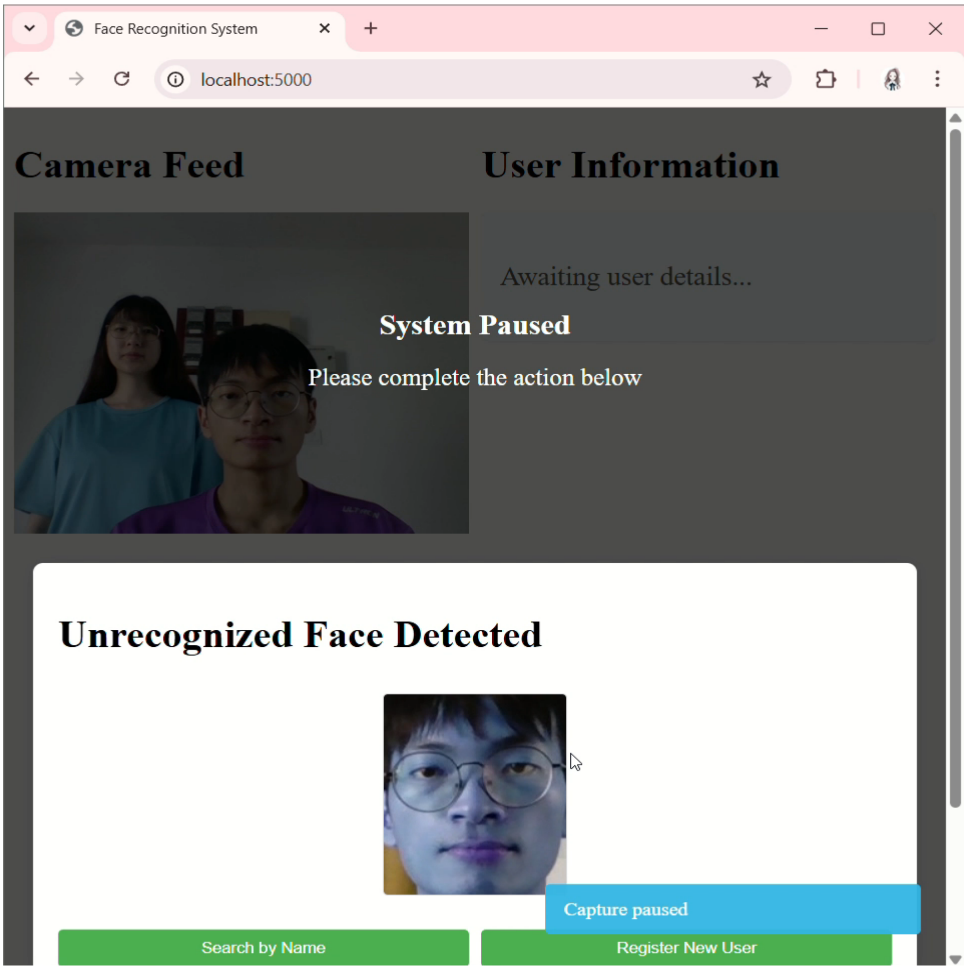


Figure 5.13 Multiple Faces Detected of Personalized Face Recognition System

5.5 Implementation Issues and Challenges

One of the main challenges encountered during the implementation of this system was ensuring accurate face detection and recognition under varying real-world conditions. The system needed to handle different image qualities, such as sharp, blurred, and low resolution. To address this, the system employed sharpness thresholds to ensure that only high-quality, relevant images were processed, which helped reduce false positives and computational waste.

The most significant challenge faced during the implementation of this system was the lagging issue during face recognition. As the system processed multiple faces simultaneously or when there was a heavy computational load, performance degraded, causing noticeable delays in recognizing faces. This issue was particularly critical in real-time systems, where fast and responsive recognition was required. To address this, a queue for face recognition was implemented. This ensured that only one face was processed at a time, preventing the system from becoming overwhelmed by multiple simultaneous recognition tasks. Once the current face recognition process was completed, the system checked if there were faces added to the queue. If the queue was not empty, the next face was processed. This approach helped manage the processing load efficiently and prevented performance bottlenecks. In addition to the queue, multithreading was used to further improve system responsiveness. A dedicated thread was allocated to the face recognition process, allowing it to run independently from other tasks. This ensured that even if other processes were running in the system, face recognition could continue in parallel, reducing the likelihood of lag. By combining the queue with multithreading, the system efficiently managed the recognition process, even with a larger volume of faces or under heavy system loads, resulting in a smoother user experience.

The next challenge was that the TTS system could trigger multiple greetings for the same user, resulting in redundant announcements. To address this, the system was designed to track previous interactions, preventing the TTS from repeating the same greeting and ensuring a more natural user experience.

Another challenge was that the system continued running during user interactions, such as when a form was being filled out. This caused the user interface to constantly update—either displaying recognized details or showing a new form for unrecognized options—while still capturing images. To address this, the system was designed to pause image capturing during user interactions and resume once the form submission was complete, ensuring seamless functionality without disruption.

5.6 Concluding Remark

The system's hardware setup and software configuration worked seamlessly together to provide an efficient face recognition solution. The built-in laptop camera and optimized network requirements ensured that real-time face detection and recognition were handled effectively, even under varying conditions. The system's portability, achieved by leveraging a laptop, enhanced its adaptability in different environments.

The software environment, built on Windows 11 and Python 3.12, supported the integration of key technologies such as YuNet for face detection, DeepFace for recognition, PostgreSQL for efficient database management, and gTTS for text-to-speech functionality. The Flask-based web interface allowed for smooth user interactions with real-time video streaming, status alerts, and easy access to dynamic forms.

Despite these robust setups, the system faced several challenges during implementation, including issues with processing blurry images, lagging during face recognition, TTS system duplication, and interference during user interactions. However, the system implemented effective solutions to address these problems.

Ultimately, the system's hardware and software configurations, along with its thoughtful consideration of implementation challenges, combined to create an efficient and responsive solution for real-time face recognition and user interaction.

Chapter 6

System Evaluation and Discussion

6.1 System Testing and Performance Metrics

To assess the system's effectiveness, both accuracy and performance speed were evaluated. Accuracy testing focused on the correctness of face recognition across both known and unknown individuals, while speed testing measured the end-to-end duration for three recognition flows: recognized face, unrecognized face (register), and unrecognized face (search). Metrics such as True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN) were used to construct confusion matrices, from which recognition accuracy was calculated. Meanwhile, execution time per flow was measured with increasing database sizes to evaluate system responsiveness and determine time complexity.

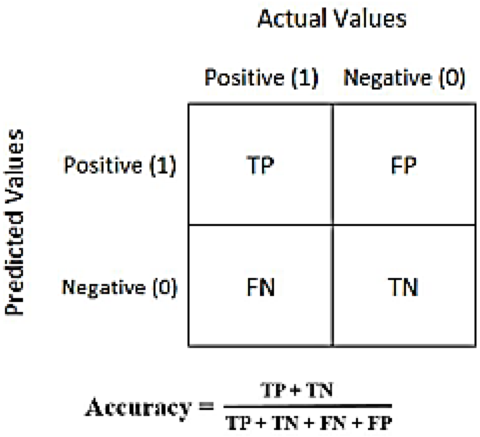
6.2 Testing Setup and Result

The system was tested with a dataset consisting of five known individuals (Persons A–E), each with five images stored in the database. For accuracy testing, 80 test images per known individual and 80 test images of unknown individuals were used. For performance testing, database sizes were increased from 100 to 1000 images, and the execution time for each flow was recorded.

6.2.1 Accuracy Testing

Figure 6.1 below shows the confusion matrix and the formula for accuracy calculation. In this test, a True Positive (TP) occurred when a face belonged to one of the five known individuals (Person A, B, C, D, or E) and was correctly recognized by the system. A True Negative (TN) refers to an unknown person who was correctly not recognized and was classified as ‘unknown'. A False Positive (FP) occurred when an unknown person was incorrectly recognized as one of the known individuals. A False Negative (FN) happened when a face belonging to one of the known individuals was not recognized or was incorrectly classified as unknown.

To evaluate the accuracy of face recognition, a confusion matrix was generated based on the results from database creation and testing. The values in the confusion matrix represented the number of images correctly or incorrectly placed into each person's folder.

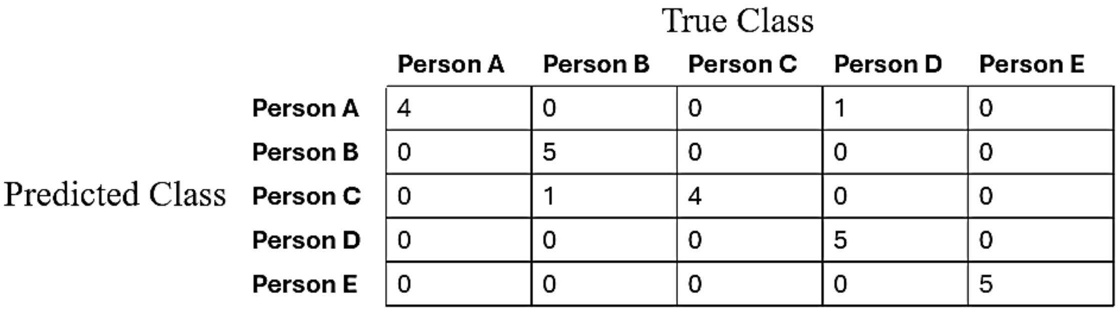


		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FN + FP}$$

Figure 6.1 Confusion Matrix and Accuracy Formula

Figure 6.2 below shows the confusion matrix for database creation.

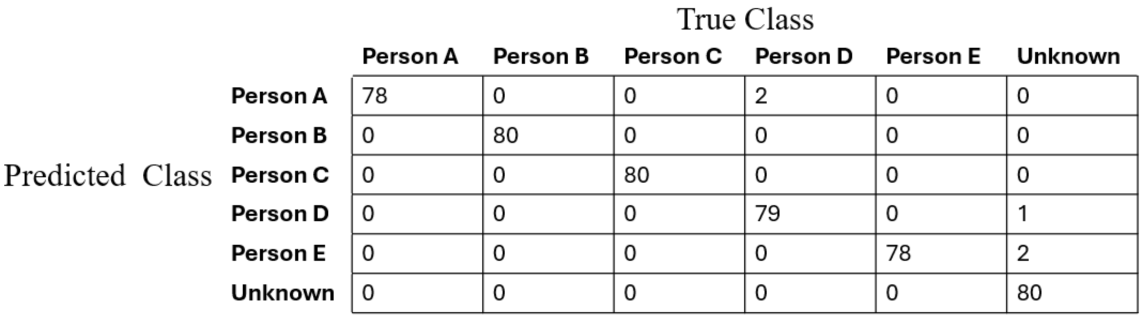


		True Class				
		Person A	Person B	Person C	Person D	Person E
Predicted Class	Person A	4	0	0	1	0
	Person B	0	5	0	0	0
	Person C	0	1	4	0	0
	Person D	0	0	0	5	0
	Person E	0	0	0	0	5

Figure 6.2 Confusion Matrix for Database Creation

During the creation of the database, 4 images of Person A were correctly recognized, but 1 image was incorrectly recognized as Person D. Similarly, 1 image of Person C was incorrectly recognized as Person B. All other images (5 each for Persons B, D, and E) were correctly recognized. Based on the confusion matrix shown in Figure 6.2 above, the calculated accuracy for this process was 92%.

Figure 6.3 below shows the confusion matrix for the testing phase.



		True Class					
		Person A	Person B	Person C	Person D	Person E	Unknown
Predicted Class	Person A	78	0	0	2	0	0
	Person B	0	80	0	0	0	0
	Person C	0	0	80	0	0	0
	Person D	0	0	0	79	0	1
	Person E	0	0	0	0	78	2
	Unknown	0	0	0	0	0	80

Figure 6.3 Confusion Matrix for Testing

Out of the 80 images of Person A, 78 were correctly recognized, while 2 were incorrectly recognized as Person D. One image of Person D was mistakenly recognized as unknown, and two images of Person E were also misrecognized as unknown. All 80 images of Person B and Person C were correctly recognized. Finally, all 80 images of unknown individuals were correctly identified as unknown. Based on the confusion matrix shown in Figure 6.3 above, the calculated accuracy was 98.54%.

6.2.2 Speed Testing

Figure 6.4 below shows how the end-to-end duration of the recognized face flow changed with database size. As the number of images in the database increased, the end-to-end duration of the recognized face flow also increased. While the actual data points did not form a perfectly straight line, the trendline suggested a linear growth pattern. Based on this observation, the time complexity of the end-to-end process could be approximated as O(n), where n is the size of the database.

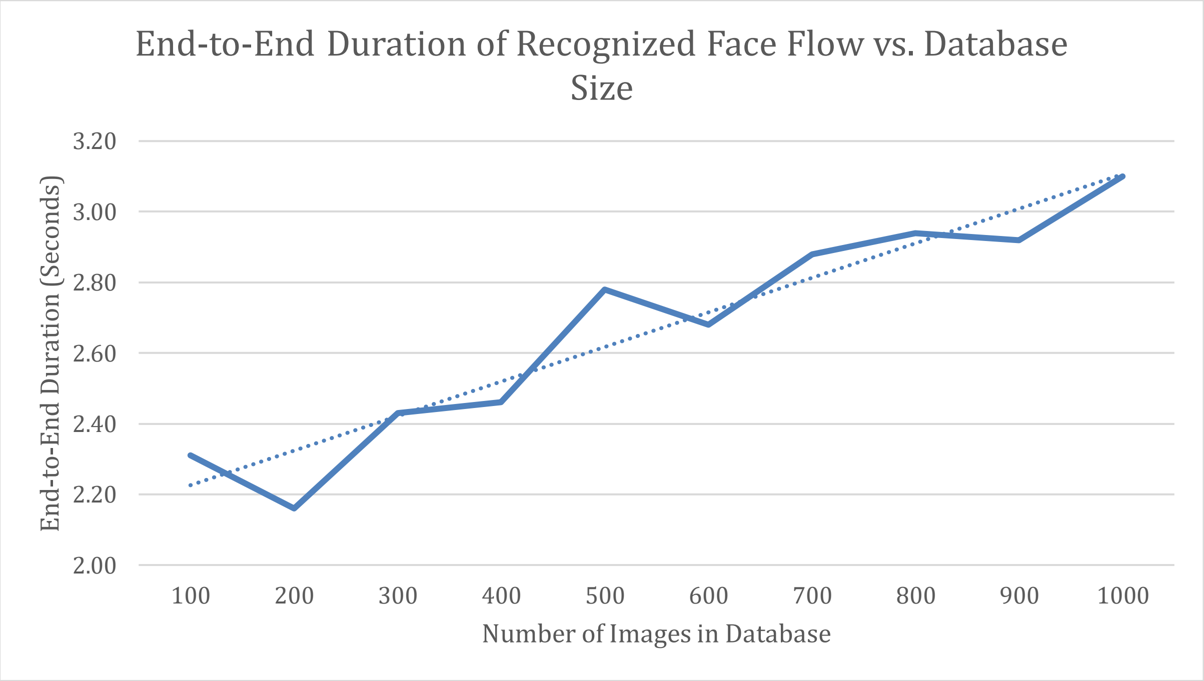


Figure 6.4 End-to-End Duration of Recognized Face Flow vs. Database Size

Figure 6.5 below shows how the end-to-end duration of the unrecognized face (register) flow changed with database size. As the number of images in the database increased, the end-to-end duration of the unrecognized face (register) flow also increased. While the actual data points did not form a perfectly straight line, the trendline suggested a linear growth pattern. Based on this observation, the time complexity of the end-to-end process could be approximated as O(n), where n is the size of the database.

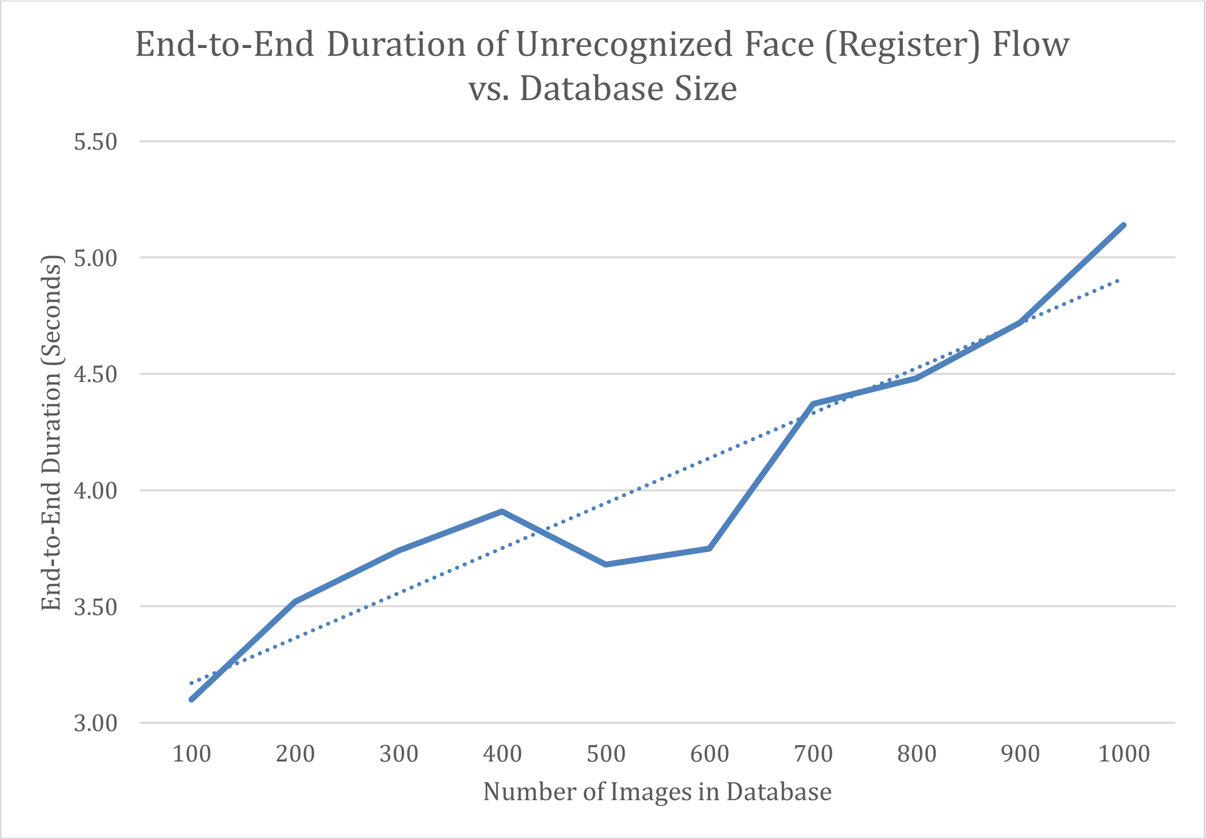


Figure 6.5 End-to-End Duration of Unrecognized Face (Register) Flow vs. Database Size

Figure 6.6 below shows how the end-to-end duration of the unrecognized face (search) flow changed with the database size. As the number of images in the database increased, the end-to-end duration of the unrecognized face (search) flow also increased. While the actual data points did not form a perfectly straight line, the trendline suggested a linear growth pattern. Based on this observation, the time complexity of the end-to-end process could be approximated as O(n), where n is the size of the database.

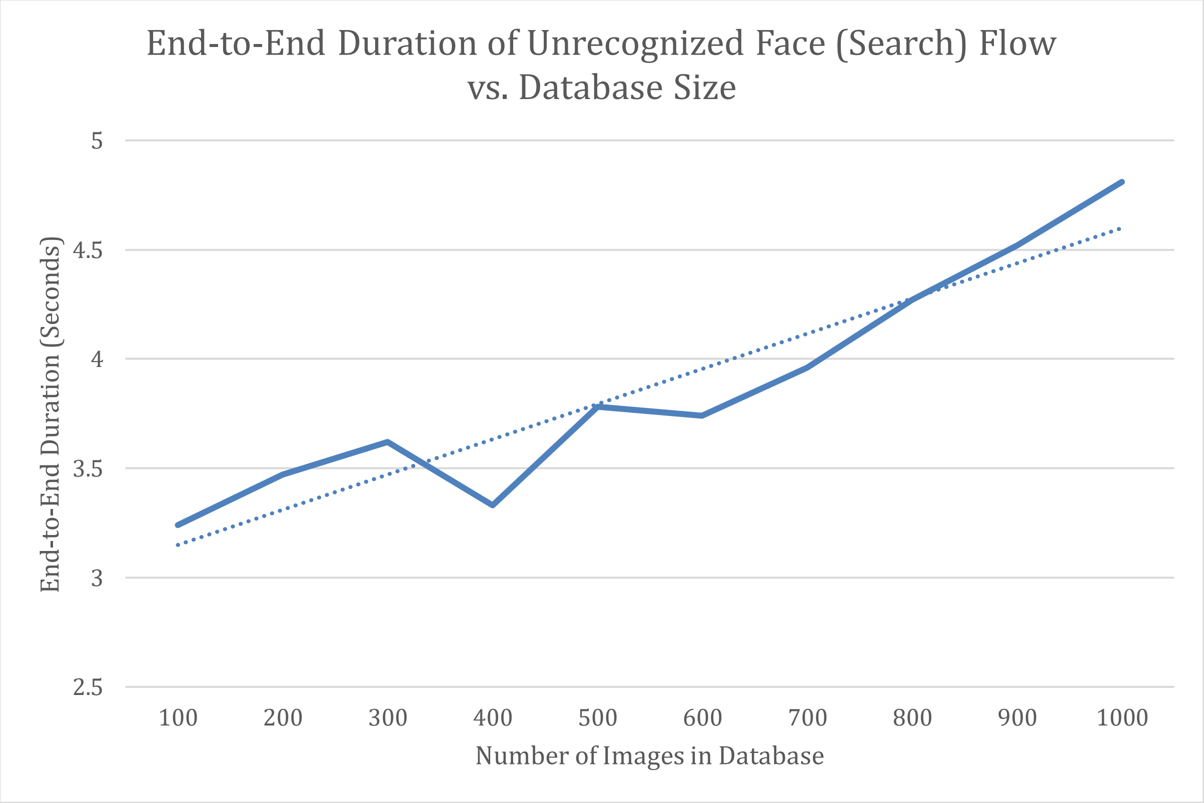


Figure 6.6 End-to-End Duration of Unrecognized Face (Search) Flow vs. Database Size

Figure 6.7 below shows the comparison of end-to-end durations for three face recognition flows, recognized face, unrecognized face (register), and unrecognized face (search), as the database size increased from 100 to 1000 images. Across all flows, the end-to-end duration increased with the database size. While the data points for each flow did not form perfectly straight lines, the trendlines indicated a generally linear growth pattern. This suggested that the time complexity of the end-to-end process in all three flows could be approximated as O(n), where n was the size of the database. Among the three, the recognized face flow consistently performed faster, followed by the unrecognized (search) and unrecognized (register) flows.

The unrecognized face (search) flow took more time than the recognized flow because it involved matching the input face against all faces in the database. In contrast, the recognized flow stopped as soon as a match was found, making it faster. Additionally, the search flow included querying the database for names and providing auto-suggestions. This process primarily involved read operations (name suggestions) and minimal write operations (saving the image and updating the path). On the other hand, the unrecognized face (register) flow took even more time than the search flow because it required creating a new image folder and inserting a new record into the database. These additional write operations were generally slower.

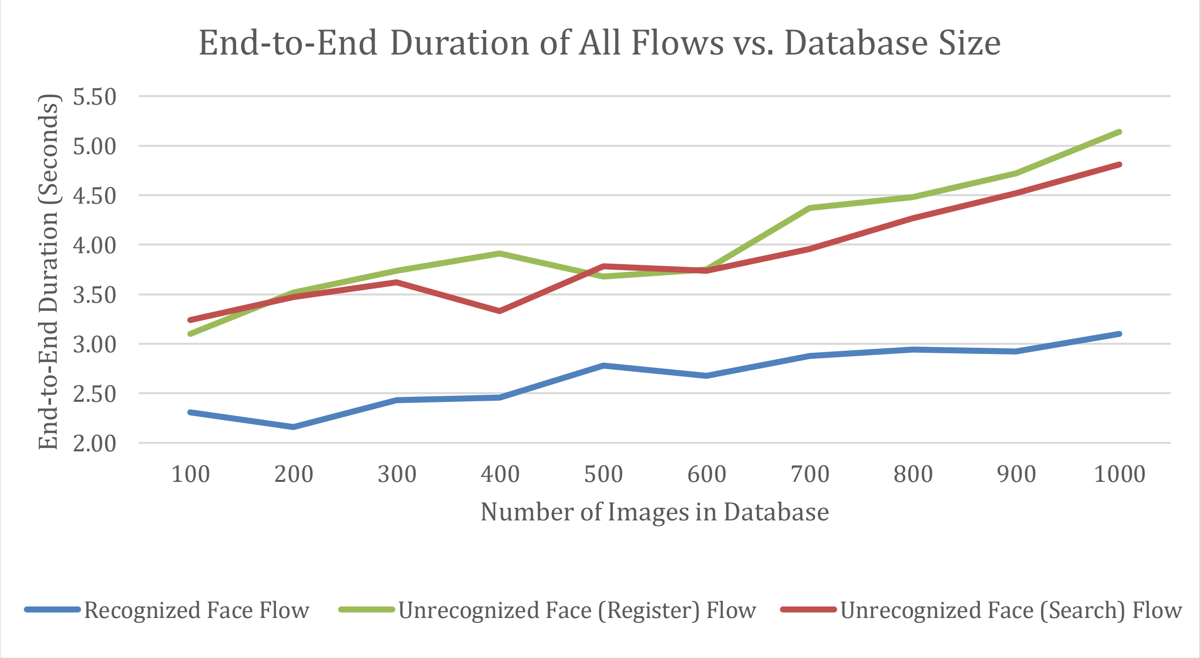


Figure 6.7 End-to-End Duration of All Flows vs. Database Size

6.3 Project Challenges

A key project challenge was ensuring only clear images were processed, as blurry or out-of-focus images could lead to inaccurate face recognition. To resolve this, the system enforces a sharpness threshold (>1.8). This proactive quality control filters out blurry inputs caused by motion or focus issues.

Another challenge was the lagging issue during face recognition. When the system processed multiple faces simultaneously, performance degraded, leading to noticeable delays in recognition. To address this, a queue was implemented to ensure that only one face was processed at a time. Additionally, a dedicated thread was allocated to the face recognition process, allowing it to run independently of other tasks. This approach ensured that face recognition could continue smoothly, even while other processes were running in parallel, significantly reducing the likelihood of lag.

A challenge arose when the Text-to-Speech (TTS) module would repeat greetings for the same individual, creating unnecessary audio redundancy. To prevent this, the system tracks the last spoken name in the TextToSpeech module. Before playing a greeting, the current name is compared to the last spoken name to avoid repeating the greeting.

During user interactions (e.g., form registration/search), the system risked race conditions by continuing frame processing while performing database operations. To solve this, the system halts frame capturing during user interactions. This pause state is synchronized with the frontend through Socket.IO events. The implementation ensures smooth state management.

6.4 Objectives Evaluation

The system successfully met its first objective, Enable Personalized Face Recognition, by achieving a high final testing accuracy of 98.54%. This ensured that individuals were correctly identified and associated with user-defined details, such as "Parent of Alice" or "Room 201." Such a level of accuracy facilitated reliable personalization, which proved to be especially useful in environments like schools, where quick and accurate recognition of parents during pickups enhances both safety and convenience. The ability to swiftly identify individuals contributed to a smoother and more efficient process, improving the overall user experience in high-traffic scenarios.

In terms of meeting the second objective, Solve Human Memory Limitations, the system effectively automated recognition and reduced the need for memory-based or manual checks, such as printed photos. Given that human recall can be slow and error-prone, particularly in fast-paced environments, this feature streamlined the process. The system handled both known and unknown faces, clearly flagging any unrecognized individuals. With consistent speed performance (average end-to-end time between 2–5 seconds) and the ability to scale linearly with large databases, the system remained responsive and provided timely alerts, ensuring safety in situations where unrecognized individuals attempted pickups. This addressed both the safety and automation aspects of the objective effectively.

6.5 Concluding Remark

The face recognition system demonstrated strong accuracy performance, achieving a final testing accuracy of 98.54%, indicating robust recognition capabilities. In terms of speed, the system maintained consistent performance across varying database sizes (100–1000 images). All recognition flows, recognized, unrecognized (register), and unrecognized (search), scaled linearly, which aligns well with real-world expectations. The average processing time for these three flows ranged from 2 to 5 seconds, which is considered fast. The system's design effectively handles both recognized and unrecognized individuals, offering practical usability for real-time face recognition applications.

Chapter 7

Conclusion and Recommendation

7.1 Conclusion

The personalized face recognition system developed in this project successfully addressed the challenges of human memory limitations and the lack of accessible, user-friendly face recognition tools for everyday scenarios. By integrating technologies such as OpenCV (YuNet) for face detection, DeepFace (Facenet512) for recognition, and PostgreSQL for local data management, the system achieved high accuracy (98.54% in testing) and efficient performance. Key features like sharpness-based image filtering (>1.8 threshold), movement-adaptive processing, and WebSocket-driven pause/resume mechanics ensured robustness in real-world conditions.

The system demonstrated practical applicability in diverse settings such as kindergartens, hotels, and event management, where it automated personalized identification, reduced operational inefficiencies, and enhanced security (e.g., preventing unauthorized child pickups). Its modular design allowed customization for different workflows without code changes, while the local database architecture ensured user-owned data privacy.

Despite challenges like processing lag and blurry images, the implemented solutions, such as a recognition queue, multithreading, and sharpness thresholds, optimized system responsiveness. The linear scalability (O(n) time complexity) ensured consistent performance even as the database grew. Overall, the project bridged the gap between government-centric face recognition systems and personalized, everyday applications, offering a scalable, user-centric solution.

7.2 Recommendations

The system can significantly enhance user experience by incorporating multi-language support for text-to-speech (TTS) greetings, ensuring inclusivity for diverse user bases in settings like multicultural schools or international hotels. This feature would allow the system to greet users in their preferred language, improving accessibility and personalization.

To bolster security and privacy, the system should implement encryption for stored face images and database entries, safeguarding sensitive data against unauthorized access. This measure is critical for maintaining user trust, especially in scenarios involving personal or confidential information.

These improvements would make the system more robust and adaptable to a wider range of applications.

REFERENCES

[1] “Home – Malaysian Immigration Department,” Imi.gov.my, 2023. https://www.imi.gov.my/index.php/en/home/ (accessed Apr. 2, 2025).

[2] Super User, “Portal JPN - Home,” Jpn.gov.my, 2023. https://www.jpn.gov.my/en/ (accessed Apr. 2, 2025).

[3] D. Bhatt et al., “CNN Variants for Computer Vision: History, Architecture, Application, Challenges and Future Scope,” Electronics, vol. 10, no. 20, p. 2470, Oct. 2021, doi: https://doi.org/10.3390/electronics10202470.

[4] “Computer Vision: What it is and why it matters,” www.sas.com. https://www.sas.com/en_my/insights/analytics/computer-vision.html

[5] A. Voulodimos, N. Doulamis, A. Doulamis, and E. Protopapadakis, “Deep Learning for Computer Vision: a Brief Review,” Computational Intelligence and Neuroscience, vol. 2018, pp. 1–13, Feb. 2018, doi: https://doi.org/10.1155/2018/7068349.

[6] Clearview AI, “Clearview AI | The World’s Largest Facial Network,” Clearview AI, 2023. https://www.clearview.ai/

[7] I. Adjabi, A. Ouahabi, A. Benzaoui, and A. Taleb-Ahmed, “Past, Present, and Future of Face Recognition: A Review,” Electronics, vol. 9, no. 8, p. 1188, Jul. 2020, doi: https://doi.org/10.3390/electronics9081188.

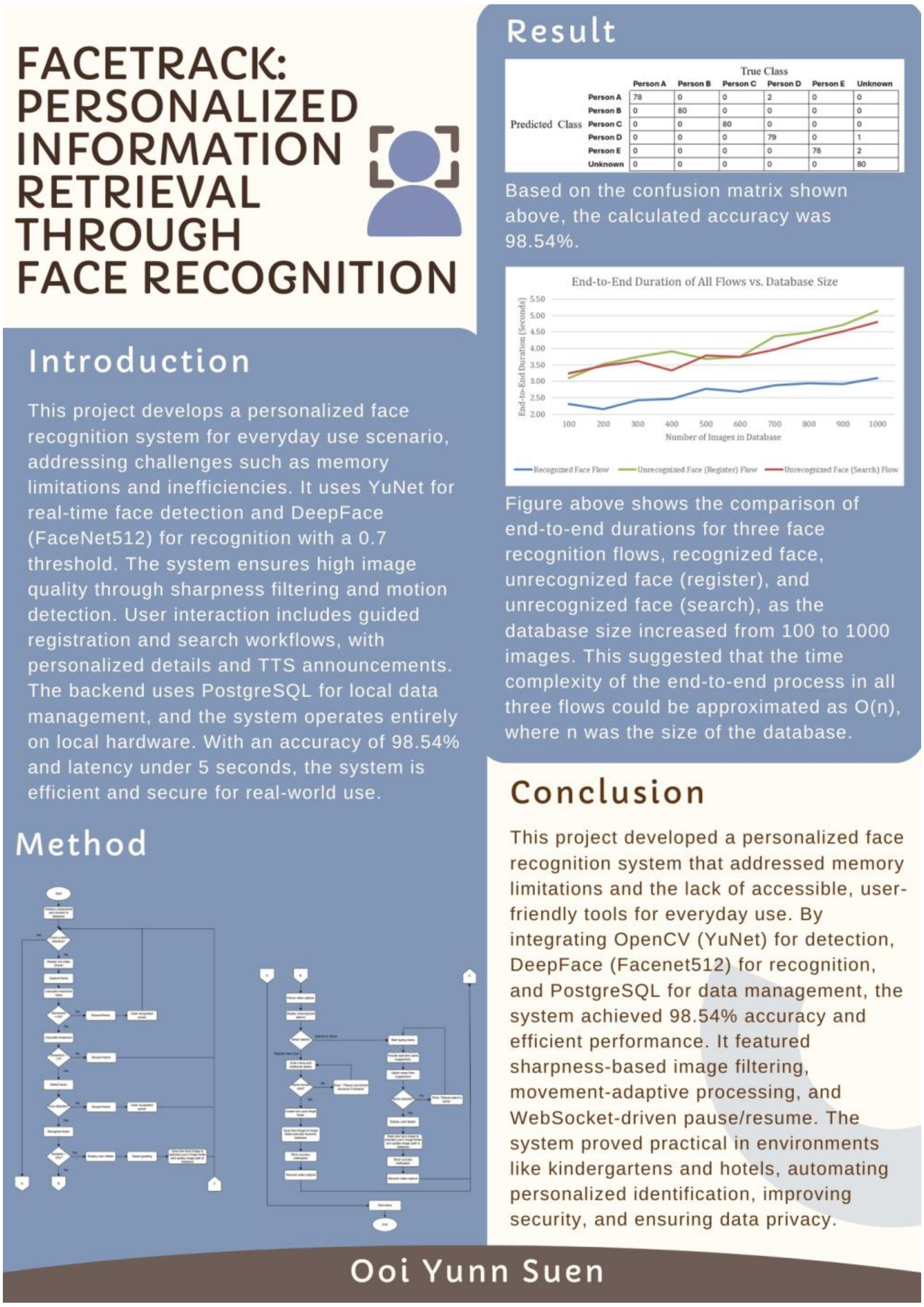
[8] P. Boucher, “How artificial intelligence works,” EPRS–European Parliamentary Research Service, In: europarl. europa, 2, https://www.sipotra.it/wpcontent/ uploads/2019/03/How-artificial-intelligence-works.pdf.

[9] Z. Deng, H. Chiang, L. Kang, and H. Li, “A lightweight deep learning model for

real-time face recognition," Iet Image Processing, vol. 17, no. 13, pp. 3869–3883, Aug. 2023, doi: https://doi.org/10.1049/ipr2.12903.

[10] H.-C. Li, Z.-Y. Deng, and H.-H. Chiang, "Lightweight and Resource-Constrained Learning Network for Face Recognition with Performance Optimization," Sensors, vol. 20, no. 21, p. 6114, Oct. 2020, doi: https://doi.org/10.3390/s20216114.

POSTER



FACETRACK: PERSONALIZED INFORMATION RETRIEVAL THROUGH FACE RECOGNITION

Introduction

This project develops a personalized face recognition system for everyday use scenario, addressing challenges such as memory limitations and inefficiencies. It uses YuNet for real-time face detection and DeepFace (FaceNet512) for recognition with a 0.7 threshold. The system ensures high image quality through sharpness filtering and motion detection. User interaction includes guided registration and search workflows, with personalized details and TTS announcements. The backend uses PostgreSQL for local data management, and the system operates entirely on local hardware. With an accuracy of 98.54% and latency under 5 seconds, the system is efficient and secure for real-world use.

Method

Result

		True Class					
		Person A	Person B	Person C	Person D	Person E	Unknown
Predicted Class	Person A	78	0	0	2	0	0
	Person B	0	80	0	0	0	0
	Person C	0	0	80	0	0	0
	Person D	0	0	0	79	0	1
	Person E	0	0	0	0	78	2
	Unknown	0	0	0	0	0	80

Based on the confusion matrix shown above, the calculated accuracy was 98.54%.

End-to-End Duration of All Flows vs. Database Size

End-to-End Duration (Seconds)
5.50
5.00
4.50
4.00
3.50
3.00
2.50
2.00
100
200
300
400
500
600
700
800
900
1000
Number of Images in Database
Recognized Face Flow
Unrecognized Face (Register) Flow
Unrecognized Face (Search) Flow

Figure above shows the comparison of end-to-end durations for three face recognition flows, recognized face, unrecognized face (register), and unrecognized face (search), as the database size increased from 100 to 1000 images. This suggested that the time complexity of the end-to-end process in all three flows could be approximated as O(n), where n was the size of the database.

Conclusion

This project developed a personalized face recognition system that addressed memory limitations and the lack of accessible, user-friendly tools for everyday use. By integrating OpenCV (YuNet) for detection, DeepFace (Facenet512) for recognition, and PostgreSQL for data management, the system achieved 98.54% accuracy and efficient performance. It featured sharpness-based image filtering, movement-adaptive processing, and WebSocket-driven pause/resume. The system proved practical in environments like kindergartens and hotels, automating personalized identification, improving security, and ensuring data privacy.

Ooi Yunn Suen