"IMAGE SEGMENTATION AND CLASSIFICATION "

A

Project Report

submitted

in partial fulfillment

for the award of the Degree of

Bachelor of Technology

in Department of Information Technology

Project Mentor:
Mrs. Shalini Singhal
Assistant Prof.-2

Submitted By :
Samyak Jain (21ESKIT102)

Department of Computer Science and Engineering
Swami Keshvanand Institute of Technology, M & G, Jaipur
Rajasthan Technical University, Kota
Session 2024-2025

Swami Keshvanand Institute of Technology, Management & Gramothan, Jaipur

Department of Computer Science and Engineering

CERTIFICATE

This is to certify that Mr. Samyak Jain, a student of B.Tech(Information Technology) 8th semester has submitted his/her Project Report entitled ”Image Segmentation and Classification” under my guidance

Mentor	**Coordinator**
Mrs. Shalini Singhal	Mrs. Richa Rawal
Assistant Prof.-2	Assistant Prof.-1
Signature.....	Signature.....

DECLARATION

We hereby declare that the report of the project entitled ”Image Segmentation and Classification” is a record of an original work done by us at Swami Keshvanand Institute of Technology, Management and Gramothan, Jaipur under the mentorship of ”Mrs. Shalini Singhal”(Dept. of Information Technology) and coordination of ”Mrs. Richa Rawal” (Dept.of Information Technology). This project report has been submitted as the proof of original work for the partial fulfillment of the requirement for the award of the degree of Bachelor of Technology (B.Tech) in the Department of Computer Science and Technology.It has not been submitted anywhere else, under any other program to the best of our knowledge and belief.

Team Members **Signature**

Samyak Jain 21ESKIT102

Acknowledgement

A project of such a vast coverage cannot be realized without help from numerous sources and people in the organization.We take this opportunity to express our gratitude to all those who have been helping us in making this project successful.

We are highly indebted to our faculty mentor Mrs. Shalini Singhal. She has been a guide, motivator source of inspiration for us to carry out the necessary proceedings for the project to be completed successfully. We also thank our project coordinator Mrs. Richa Rawal for her co-operation, encouragement, valuable suggestions and critical remarks that galvanized our efforts in the right direction.

We would also like to convey our sincere thanks to Dr. Anil Chaudhary, HOD, Department of Information Technology, for facilitating, motivating and supporting us during each phase of development of the project.Also, we pay our sincere gratitude to all the Faculty Members of Swami Keshvanand Institute of Technology, Management and Gramothan, Jaipur and all our Colleagues for their co-operation and support.

Last but not least we would like to thank all those who have directly or indirectly helped and cooperated in accomplishing this project.

Team Members:

Samyak Jain 21ESKIT102

Abstract

The Image Segmentation and Recognition System is developed to enhance personal identification using advanced facial recognition techniques. This project addresses key challenges such as variations in age, race, gender, and image quality, which hinder existing recognition systems. Built with Python, OpenCV, and machine learning, the system segments facial images and recognizes individuals through feature extraction and classification. Users can be registered by capturing multiple facial images which are processed and stored for accurate future identification. Real-time detection captures images and matches them with the database to confirm identity or alert unauthorized access. Image preprocessing, including grayscale conversion and noise reduction, is applied for improved accuracy. The system is thoroughly tested for database integration, face matching accuracy, and real-time functionality. The solution aims to offer a robust and efficient identification method suitable for use in security systems, access control, and personal authentication platforms.

Table of Content

List of Figures

List of Tables

Chapter 1
Introduction

1.1 Problem Statement and Objective

A problem statement is a useful communication tool, as it keeps the whole team on track and tells them why the project is important. A problem statement helps someone to define and understand the problem, identify the goals of the project, and outline the scope of work. A problem statement is a useful communication tool, as it keeps the whole team on track and tells them why the project is important. A problem statement helps someone to define and understand the problem, identify the goals of the project, and outline the scope of work.

1.2 Literature Survey /Market Survey/Investigation and Analysis

This study reviews existing research on facial expression recognition, focusing on traditional handcrafted methods like Haarcascade and modern computer vision techniques using OpenCV. While older approaches had limitations in accuracy and real–time performance, advancements in OpenCV's optimization and deep learning integrations have significantly improved recognition capabilities. The growing demand for automated expression analysis in psychology, security, and customer sentiment analysis highlights the relevance of this project.

1.3 Introduction to Project

The project focuses on developing a facial expression recognition system utilizing computer vision techniques. The primary technologies involved include OpenCV for real-time face detection, Haarcascade for feature extraction, and Tkinter for creating an interactive user interface. The system will capture facial expressions from real–

time video feeds and classify emotions such as happiness, sadness, anger, surprise, and neutral expressions. The project aims to provide a user-friendly and efficient solution applicable in various domains such as surveillance, human-computer interaction, and healthcare. This report will detail the methodology, implementation, system design, and testing of the developed model.

1.4 Proposed Logic / Algorithm / Business Plan / Solution / Device

User Registration:

- The user enters their **User ID** and clicks **Register**.
- The camera captures **100 images** of the user.
- These images undergo **digital image processing** (converted to **grayscale**, noise reduction).
- Processed images are stored in a folder named after the **User ID**.

Attendance Marking:

- The user clicks **Take Attendance**, which opens the **camera**.
- The system **matches** the captured image with the stored dataset using **Haar-cascade and OpenCV-based recognition techniques**.
- If a match is found, attendance is marked with **time and date**.

This system ensures accurate, real-time, and automated attendance tracking using **AI-based facial recognition**.

1.5 Scope of the Project

The **Face Expression Recognition System** aims to automate attendance tracking using **AI-powered facial recognition**. The scope of this project includes:

- **User Registration:** Users can register by capturing **100 images**, which are pre-processed (grayscale conversion, noise reduction) and stored for recognition.
- **Attendance Marking:** The system matches live images with the stored dataset using **Haarcascade and OpenCV-based recognition techniques**, ensuring accurate identification.
- **Real-time Processing:** The system captures attendance with **date and time** without manual intervention.
- **User-Friendly Interface:** Developed using **Tkinter**, providing an intuitive and interactive UI.
- **Scalability:** Can be expanded for **multiple users** and integrated into **educational institutions, offices, and security systems**.

This project enhances accuracy, reduces manual effort, and ensures efficient attendance management.

Chapter 2
Software Requirement Specification

2.1 Overall Description

The Face Expression Recognition System is designed to provide an automated, real-time attendance tracking solution using AI-powered facial recognition. The system consists of two main functionalities: User Registration and Attendance Marking. During registration, the user provides a User ID, and the system captures 100 images, which are preprocessed using grayscale conversion and noise reduction before being stored in a dedicated folder. For attendance marking, the system opens the camera, captures an image, and matches it against the stored dataset using Haarcascade and OpenCV-based recognition techniques. If a match is found, the system records attendance along with the date and time. The project is implemented using Python, Tkinter, and Machine Learning techniques, ensuring an efficient, scalable, and user-friendly experience. This system can be integrated into schools, offices, and security applications, enhancing accuracy and eliminating manual errors in attendance tracking.

2.1.1 Product Perspective

The **Face Attendance System** is designed as a standalone application that automates attendance tracking using **AI-powered facial recognition**. It integrates with **hardware** (webcam), **computer vision techniques** (Haarcascade, OpenCV), and a **GUI framework** (Tkinter) to provide an intuitive and efficient user experience.The system follows a **client-server architecture**, where the front end (Tkinter-based UI) interacts with the backend (Python and OpenCV-based image processing) to process images and store attendance records. The solution is **scalable** and can be deployed in **schools, offices, and security systems** for seamless attendance monitoring.

2.1.1.1 System Interfaces

The System Interface section in an SRS document defines how the system interacts with external components, such as **hardware, software, databases, and networks**. It specifies **communication protocols, data exchange formats, and APIs** used for integration. This ensures seamless interaction between the system and other entities for efficient operation.

2.1.1.2 User Interfaces

This section describes how users will interact with the system, including the **design, layout, and usability aspects**. It outlines visual elements such as **menus, buttons, input fields, and navigation flow** to ensure a seamless user experience. This section helps define **accessibility, responsiveness, and the overall look and feel** of the system.

2.1.1.3 Hardware Interfaces

It defines the **physical devices** that interact with the system, such as **webcams, storage devices, and servers**. The system primarily requires a **camera** to capture images for both user registration and attendance marking. It also utilizes a **local storage system** to maintain user image datasets. The interface ensures seamless **hardware-software integration** for accurate recognition and real-time processing.

2.1.1.4 Software Interfaces

This section defines how the system interacts with other **software applications, operating systems, databases, or APIs**. The system is developed using **Python**, with **Tkinter** for GUI, and **computer vision techniques** (Haarcascade, OpenCV) for face detection and recognition. The attendance records are stored in a **local database or file system**, with an option for future integration with cloud-based platforms.

2.1.1.5 Communications Interfaces

The communication interface defines the protocols and methods used for data exchange within the system. The Face Attendance System is designed to work **offline**,

meaning it does not rely on an internet connection for its primary functions. However, if integrated with cloud storage or an external database, it can use **HTTP APIs or database queries** for data transmission. The system supports **real-time image processing** to match faces against stored datasets.

2.1.1.6 Memory Constraints

The system processes and stores multiple user images, requiring **adequate storage and memory management**. Each registered user has **100 grayscale images** stored in their respective folders, which helps improve recognition accuracy. Memory constraints depend on the number of users stored and the available disk space. Efficient storage techniques such as **image compression and optimized data structures** ensure that memory usage remains minimal while maintaining accuracy.

2.1.1.7 Operations

The system performs two primary operations:

- **User Registration:** Captures and processes user images for recognition.
- **Attendance Marking:** Matches real-time captured images with stored datasets and marks attendance.

Additional operations include **data management, user authentication, and system maintenance**.

2.1.1.8 Project Functions

The core functions of the system include:

- Capturing and storing user images.
- Processing images using **digital image processing techniques** (grayscale conversion, noise removal).
- Detecting and recognizing faces using **Haarcascade and OpenCV-based recognition techniques**.
- Maintaining **attendance records** with **date and time stamps**.

- Providing a **user-friendly interface** for registration and attendance.

2.1.1.9 User Characteristics

The system is designed for a **wide range of users**, including students, employees, and administrators. Users are expected to have **basic computer skills** to interact with the system. The UI is kept **simple and intuitive** to ensure easy operation.

2.1.1.10 Constraints

Some constraints that affect the system's performance include:

- **Lighting conditions:** Poor lighting can affect face recognition accuracy.
- **Camera quality:** Low-resolution cameras may lead to false recognition.
- **Processing time:** Real-time face recognition requires an efficient algorithm for fast processing.
- **Storage limitations:** Large datasets may require additional storage space.

2.1.1.11 Assumptions and Dependencies

- The system assumes users will **register with a clear face image** for accurate recognition.
- It depends on **Haarcascade and OpenCV** for face detection.
- The system requires **a working camera** for capturing images.
- The accuracy of face recognition depends on the **quality and number of images stored**.

Chapter 3
System Design Specification

3.1 System Architecture

The **Face Expression Recognition System** consists of three primary modules: **User Registration, Face Recognition, and Attendance Management**. The system architecture defines the interaction between these modules and their data flow. The system follows a **client-server model** with a local storage system to manage user records and attendance logs.

3.1.1 Data Flow Overview

1. **User Registration:** Captures and processes user images, storing them for future recognition.
2. **Face Recognition:** Identifies users by matching real-time images with stored datasets.
3. **Attendance Management:** Marks attendance if a match is found and logs the date and time.

3.1.2 Technical Components

- **Frontend:** Tkinter-based GUI for user interaction.
- **Backend:** Python-based logic for face detection and recognition.
- **Algorithms Used:** Haarcascade for face detection, OpenCV for image processing and recognition.
- **Storage:** User images and attendance records stored locally in structured directories.

3.2 Module Decomposition

The system is structured into three core modules:

3.2.1 Module 1: User Registration

- The user enters their **User ID** and clicks ”Register”.
- The system captures **100 images** of the user via a webcam.
- Images undergo preprocessing (**grayscale conversion, noise reduction**).
- Processed images are saved in a folder named after the **User ID**.

3.2.2 Module 2: Face Recognition

- The user clicks ”Take Attendance,” and the camera captures an image.
- The captured image is analyzed using **Haarcascade** for face detection.
- The detected face is compared with stored images using **OpenCV-based face recognition techniques**.
- If a match is found, the system proceeds to mark attendance.

3.2.3 Module 3: Attendance Management

- If a face match is confirmed, the system marks attendance.
- Attendance records are logged with **User ID, date, and time**.
- The system updates attendance logs for future reference.

This modular approach ensures an efficient, scalable, and easy-to-maintain system for face-based attendance tracking.

3.3 High Level Design Diagrams

3.3.1 Use Case Diagram

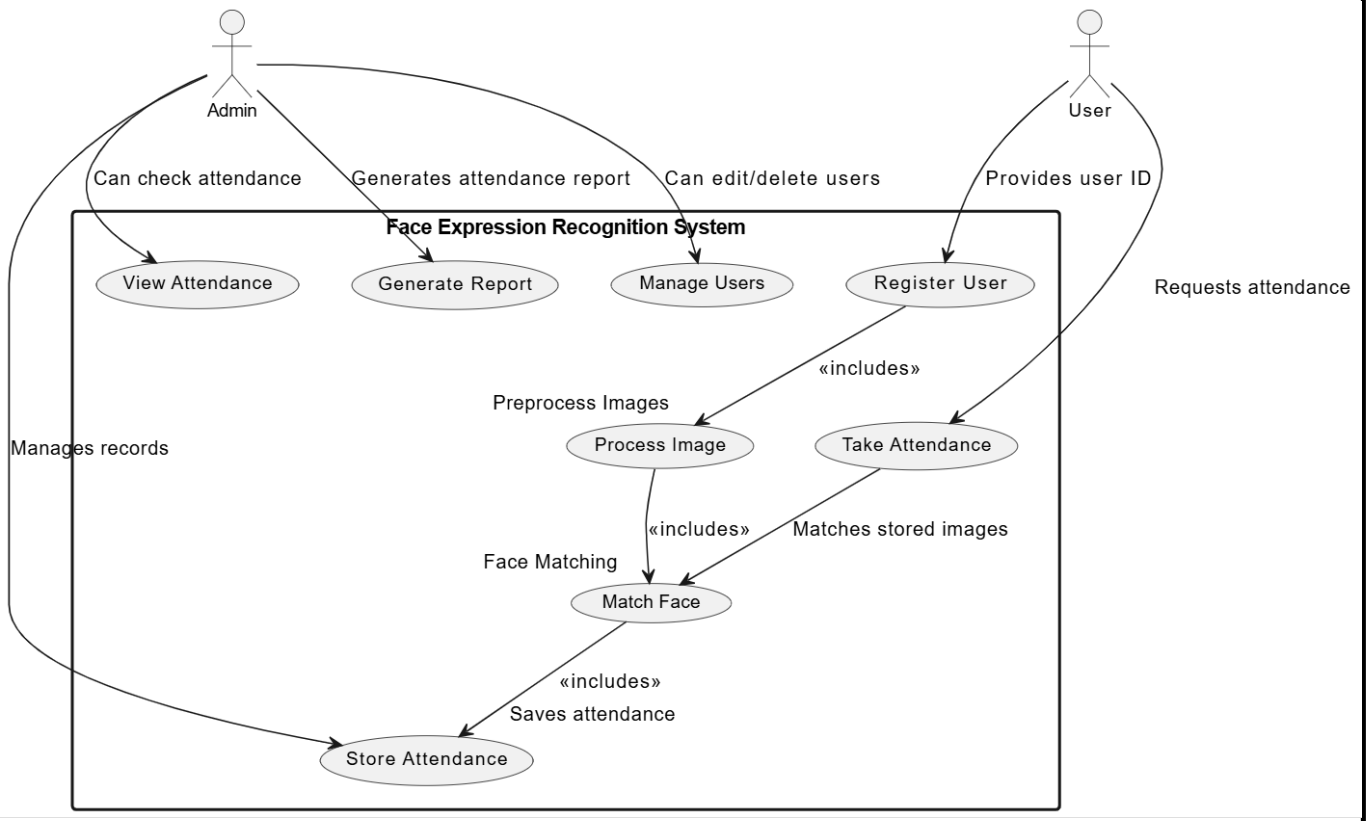

Figure 3.1: Use Case diagram

3.3.2 Activity Diagram

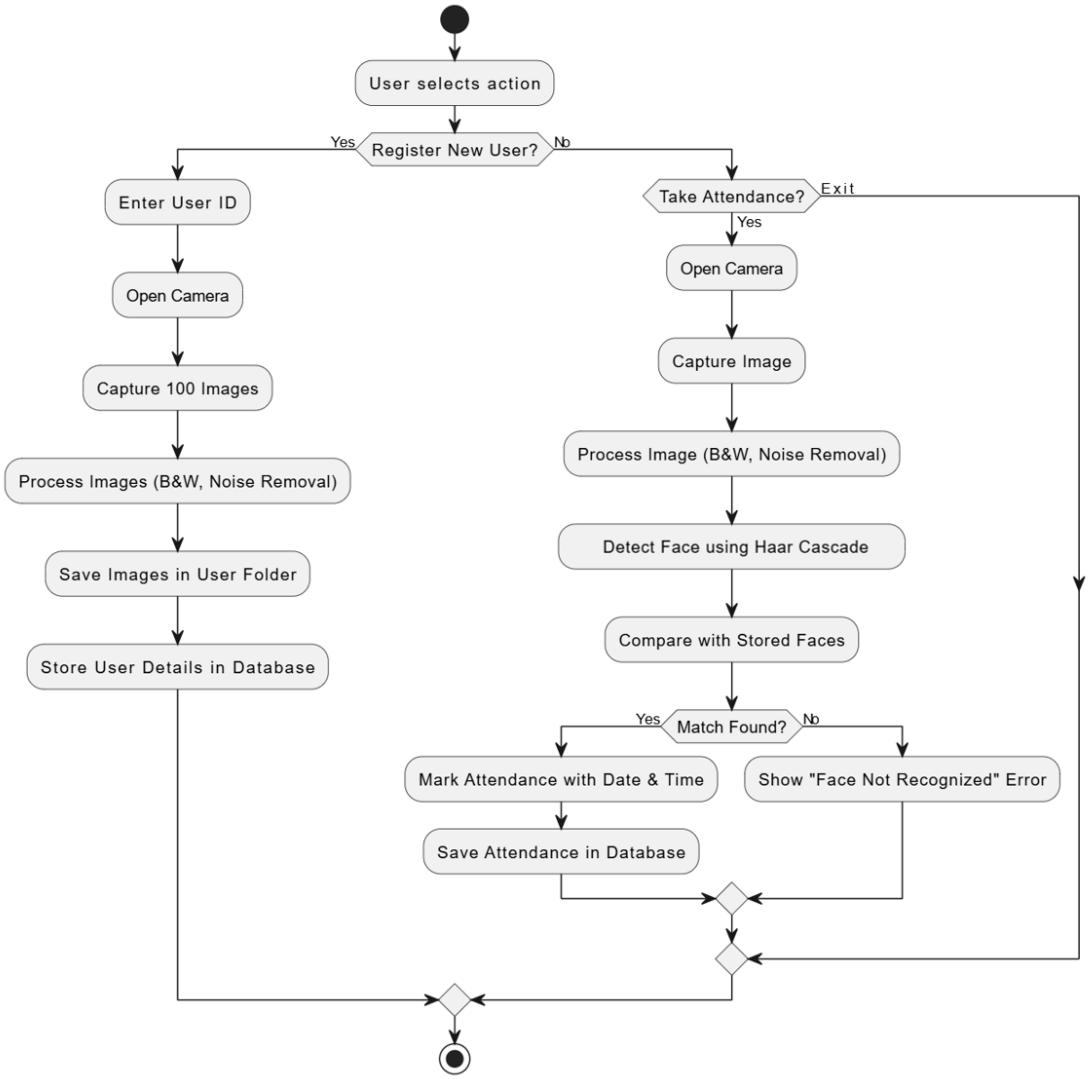

Figure 3.2: Activity Diagram

3.3.3 Data-Flow Diagram

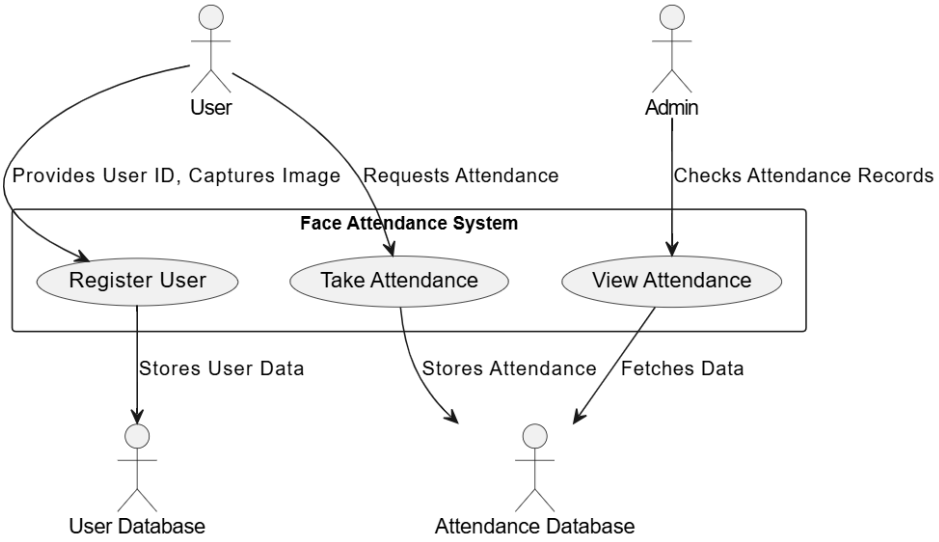

Figure 3.3: DFD Level 0: High-level Overview of the Face Attendance System

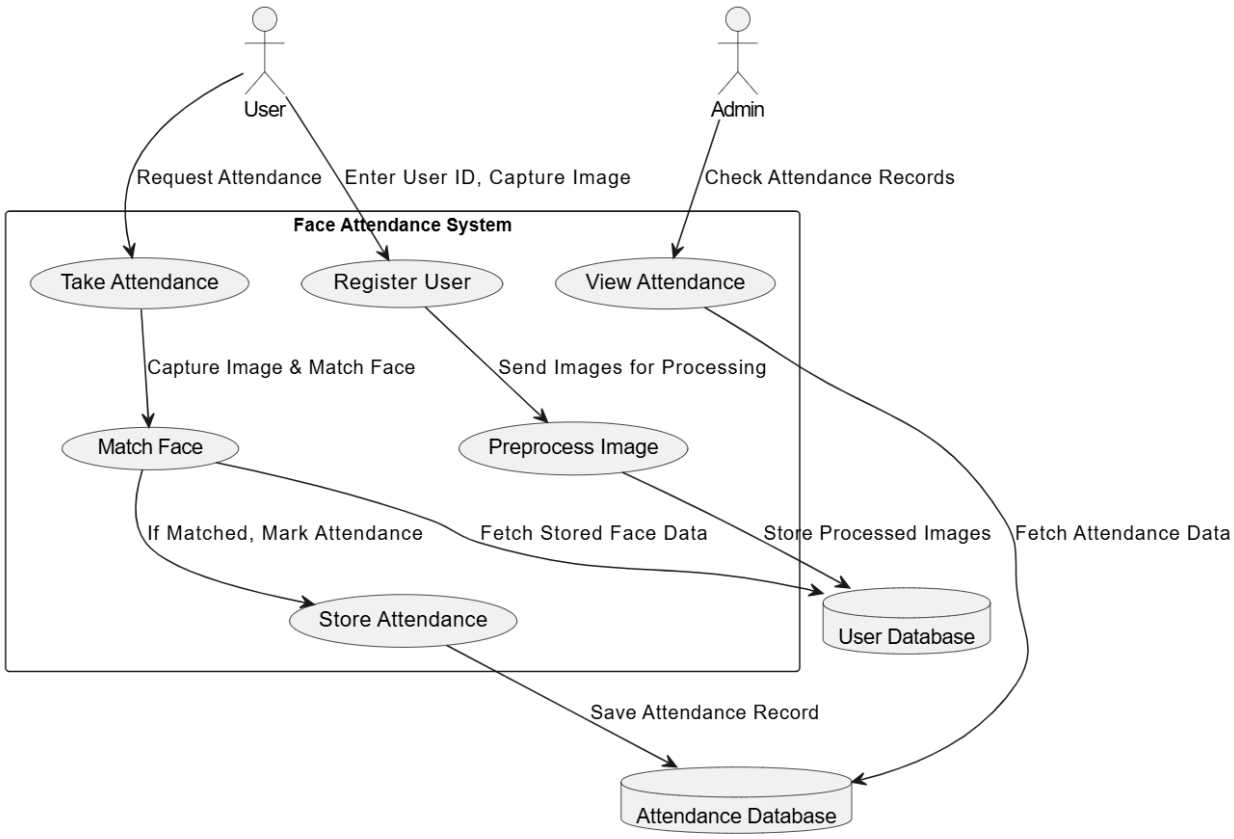

Figure 3.4: DFD Level 1: Detailed View of User Registration and Face Recognition

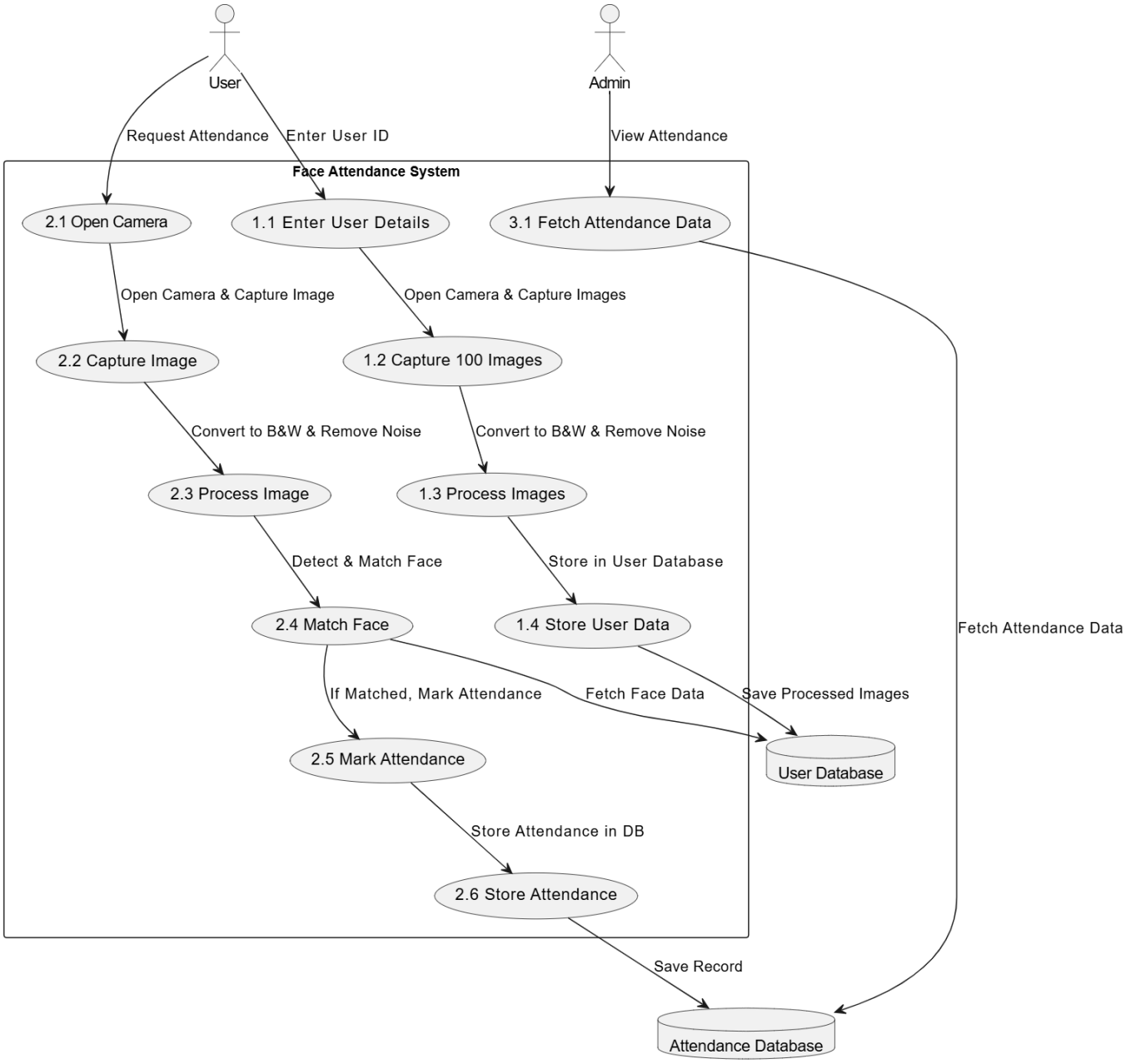

Figure 3.5: DFD Level 2: Detailed View of Attendance Management and Data Handling

3.3.4 Class Diagram

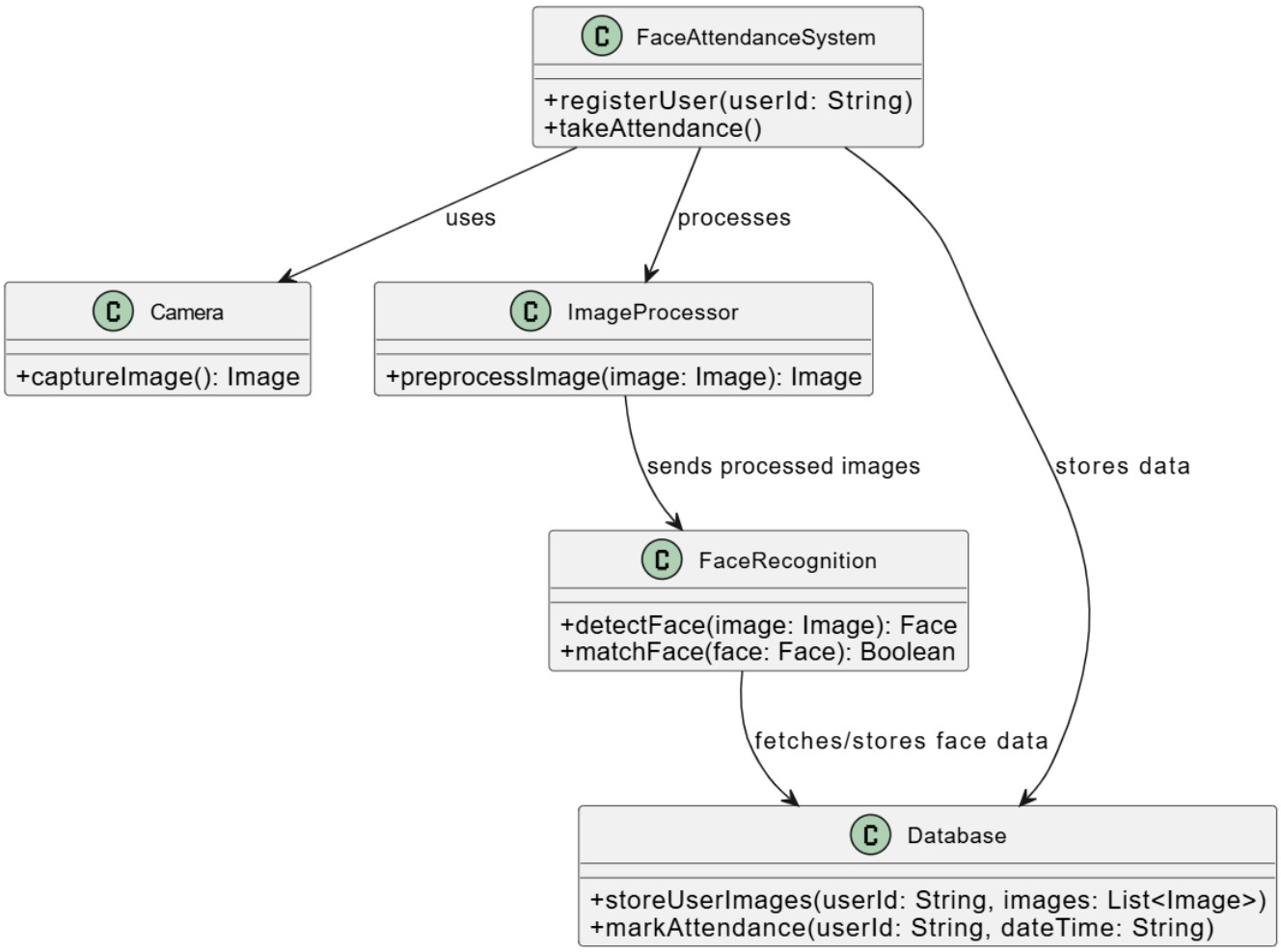

Figure 3.6: Class Diagram

Chapter 4

Methodology and Team

4.1 Introduction to Waterfall Framework

The Waterfall Model was first Process Model to be introduced. It is also referred to as a linear-sequential life cycle model. It is very simple to understand and use. In a waterfall model, each phase must be completed before the next phase can begin and there is no overlapping in the phases. The waterfall Model illustrates the software development process in a linear sequential flow; hence it is also referred to as a linear-sequential life cycle model. This means that any phase in the development process begins only if the previous phase is complete. In waterfall model phases do not overlap. In "The Waterfall" approach, the whole process of software development is divided into separate phases. In Waterfall model, typically, the outcome of one phase acts as an input for the next phase sequentially. Following is a diagrammatic representation of different phases of waterfall model.

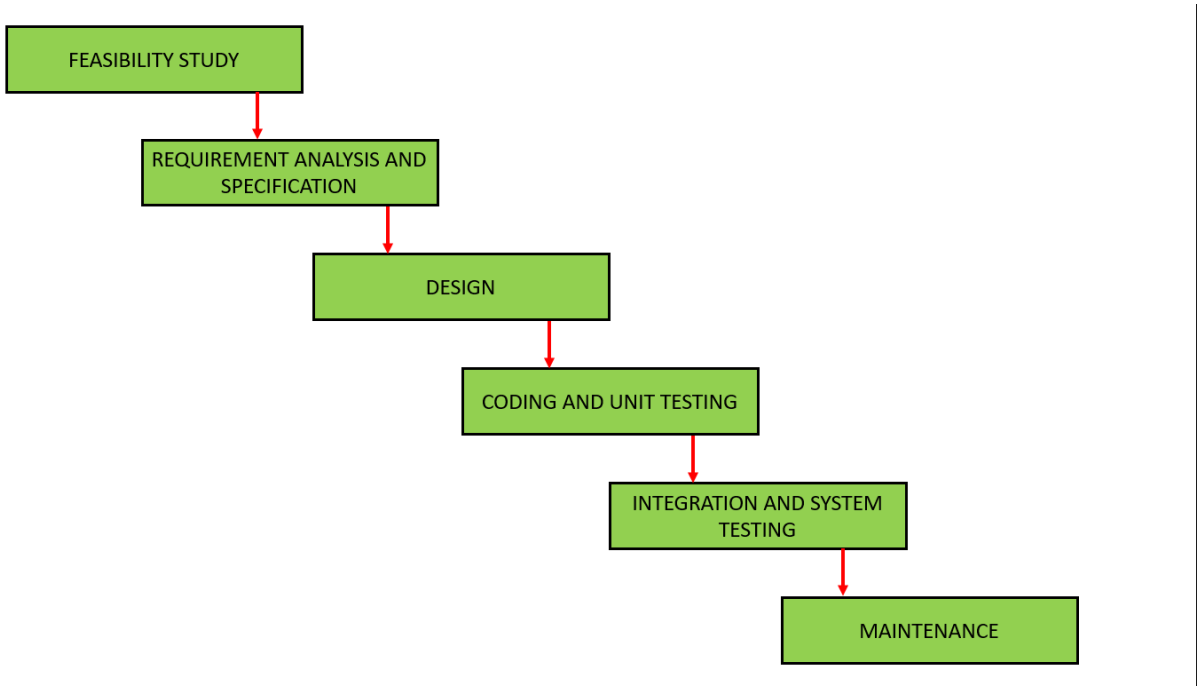

Figure 4.1: WaterFall model

The sequential phases in Waterfall model are-

1. **Requirement Gathering and analysis:** All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification doc.

2. **System Design:** The requirement specifications from first phase are studied in this phase and system design is prepared. System Design helps in specifying hardware and system requirements and also helps in defining overall system architecture.

3. **Implementation:** With inputs from system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality which is referred to as Unit Testing.

4. **Integration and Testing:** All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.

5. **Deployment of system:** All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.

6. **Maintenance:** All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.

All these phases are cascaded to each other in which progress is seen as flowing steadily downwards (like a waterfall) through the phases. The next phase is started only after the defined set of goals are achieved for previous phase and it is signed off, so the name ”Waterfall Model”. In this model phases do not overlap.

Waterfall Model Pros & Cons

Advantage The advantage of waterfall development is that it allows for departmen-

talization and control. A schedule can be set with deadlines for each stage of development and a product can proceed through the development process model phases one by one. Development moves from concept, through design, implementation, testing, installation, troubleshooting, and ends up at operation and maintenance. Each phase of development proceeds in strict order.

Disadvantage The disadvantage of waterfall development is that it does not allow for much reflection or revision. Once an application is in the testing stage, it is very difficult to go back and change something that was not well-documented or thought upon in the concept stage.

4.2 Team Members, Roles & Responsibilities

Naman Jain - Image Processing

- **Image Detection:** Detection of face using algorithms.
- **Image Enhancement:** Enhance image quality for better feature extraction.
- **Optimize Image:** Ensure post-processing times are optimized for real-time application.
- **Feature Extraction:** Extraction of facial features to be used in recognition.
- **Testing & Deployment:** Testing of image processing techniques and deployment to ensure robustness.

Nitin Verma - AI/ML

- **Multi-Modal Training:** Designing machine learning algorithms for face expression recognition.
- **Feature Extraction:** Extraction of relevant facial features from images.
- **Prediction:** Developing prediction models to recognize facial expressions.
- **Testing & Deployment:** Testing and deploying the machine learning model to ensure high accuracy and performance.

Naman Jain - Frontend Developer

- **Design UI/UX for Data:** Creating a user-friendly interface where users can upload images.
- **Implement FER Interface:** Developing visual components that display the results of facial expression recognition.
- **Develop Interactive Feedback System:** Using sigmoids to define and train expressions for feedback.
- **Fully Responsive:** Ensuring that the system is suitable for every device (desktop, mobile, etc.).
- **Integration with Backend:** Handling data and fixing errors during the integration of the frontend and backend.

Chapter 5
Centering System Testing

The designed system has been testing through following test parameters.

5.1 Functionality Testing

In testing the functionality of the web sites the following features were tested:

1. Links

   (a) Internal Links: All internal links of the website were checked by clicking each link individually and providing the appropriate input to reach the other links within.

   (b) External Links: Till now no external links are provided on our website but for future enhancement we will provide the links to the candidate's actual profile available online and link up with the elections updates online etc.

   (c) Broken Links : Broken links are those links which so not divert the page to specific page or any page at all. By testing the links on our website, there was no link found on clicking which we did not find any page.

2. Forms

   (a) Error message for wrong input : Error messages have been displayed as and when we enter the wrong details (eg. Dates), and when we do not enter any details in the mandatory fields. For example: when we enter wrong password we get error message for acknowledging us that we have entered it wrong and when we do not enter the username and/or password we get the messages displaying the respective errors.

   (b) Optional and Mandatory fields : All the mandatory fields have been marked with a red asterisk (*) and apart from that there is a display of error messages when we do not enter the mandatory fields. For example: As the first

name is a compulsory field in all our forms so when we do not enter that in our form and submit the form we get an error message asking for us to enter details in that particular field.

3. Database Testing is done on the database connectivity.
4. Note : In testing - Unit Testing,Integration Testing, System Testing,Performance Testing,User Acceptance Testing (UAT) should be discussed.

5.2 Performance Testing

Performance testing was conducted to evaluate the system's responsiveness, stability, and overall efficiency. The key aspects assessed include:

- Load Testing: Measuring system behavior under expected user loads.
- Stress Testing: Evaluating system performance under extreme conditions.
- Scalability Testing: Checking if the system can handle increased traffic and data volume.
- Response Time Analysis: Ensuring the website loads efficiently within acceptable time frames.

5.3 Usability Testing

Usability testing was carried out to ensure the system provides a smooth and user-friendly experience. The main areas analyzed include:

- Ease of Navigation: Assessing how easily users can navigate through the system.
- User Interface Design: Evaluating the layout, colors, fonts, and responsiveness of UI components.
- Accessibility: Ensuring the system is accessible to users with disabilities.
- Verifying that users receive clear and helpful messages when incorrect inputs are provided.

Chapter 6

Test Execution Summary

S.No	Test Case Id	Test Case Description	Test Case Status	Resources Consumed
GUI Testing				
1	GUI-001	Verify the Tkinter interface loads without errors	Passed	Low
2	GUI-002	Validate the ”Register User” button click opens the camera	Passed	Medium
3	GUI-003	Ensure the ”Take Attendance” button opens the camera	Passed	Medium
4	GUI-004	Check if the user ID input field accepts valid alphanumeric input	Passed	Low
5	GUI-005	Verify that invalid characters in the user ID field show an error	Passed	Low
6	GUI-006	Ensure the captured images are displayed properly before saving	Passed	Medium
7	GUI-007	Test if the UI buttons are responsive across different resolutions	Passed	Low
8	GUI-008	Verify the error message when the camera is not available	Passed	Medium
9	GUI-009	Check if the UI displays a success message after a user registers	Passed	Low
10	GUI-010	Ensure the UI updates attendance status after a face match	Passed	Medium

Table 6.1: GUI Testing Summary

S.No	Test Case Id	Test Case Description	Test Case Status	Resources Consumed
Business Logic Testing				
11	BL-001	Validate that 100 images are correctly saved for a new user	Passed	High
12	BL-002	Check if images are converted to grayscale correctly	Passed	Medium
13	BL-003	Test noise reduction algorithm applied to saved images	Passed	Medium
14	BL-004	Verify that face detection with Haar Cascade works properly	Passed	High
15	BL-005	Ensure the system detects and matches a registered user's face	Passed	High
16	BL-006	Test if an unregistered user is correctly rejected	Passed	High
17	BL-007	Validate that the system logs attendance with a timestamp	Passed	Medium
18	BL-008	Check if duplicate attendance for the same user is prevented	Passed	Medium
19	BL-009	Ensure face recognition works in different lighting conditions	Passed	High
20	BL-010	Test if attendance records are saved in the correct format (CSV/DB)	Passed	Medium

Table 6.2: Business Logic Testing Summary

GUI Testing

GUI-001: Verify the Tkinter interface loads without errors

This test case ensures that the Tkinter-based graphical user interface (GUI) loads properly without any errors. The system should open with the necessary UI elements such as buttons, input fields, and labels. The GUI should be functional and free from any runtime crashes or graphical glitches. This is manually verified by launching the application and ensuring all elements are visible and interactive.

GUI-002: Validate the "Register User" button click opens the camera

When the "Register User" button is clicked, the system should open the camera for capturing the user's image for registration. This test checks whether clicking this button triggers the camera feed to open, allowing the user to position themselves for the image capture. This behavior is verified manually by interacting with the button and confirming the camera functionality using OpenCV.

GUI-003: Ensure the "Take Attendance" button opens the camera

Similar to the "Register User" button, the "Take Attendance" button should also

open the camera for real-time face recognition. Upon clicking, the camera should start detecting the user's face and mark attendance. The test is manually conducted by verifying that the button correctly opens the camera for face recognition.

GUI-004: Check if the user ID input field accepts valid alphanumeric input

The "User ID" field in the registration form must accept valid alphanumeric characters (letters and numbers). This test case involves manually entering a mix of letters and numbers to ensure that the input field accepts these characters without triggering errors. Invalid characters like special symbols should be rejected by the system.

GUI-005: Verify that invalid characters in the user ID field show an error

When invalid characters such as special symbols or spaces are entered into the "User ID" field, the system should display an error message. This test is meant to confirm that the input validation mechanism is working as expected and that only valid alphanumeric IDs are accepted.

GUI-006: Ensure the captured images are displayed properly before saving

After capturing an image during the user registration process, the system should display a preview of the image to the user. This test checks if the system shows the captured image correctly on the GUI, allowing the user to confirm the image before saving it. The image preview must be clear and properly displayed.

GUI-007: Test if the UI buttons are responsive across different resolutions

The UI should be responsive across different screen resolutions, meaning that buttons and input fields should resize or align correctly. This test case ensures that all UI components remain functional and do not overlap or get distorted when the application window is resized or displayed on different screen sizes.

GUI-008: Verify the error message when the camera is not available

In the event that the camera is unavailable, such as when it is not connected or malfunctioning, the system should display an appropriate error message. This test is manually performed by disabling the camera and ensuring that the system shows a clear error indicating that the camera could not be accessed.

GUI-009: Check if the UI displays a success message after a user registers

After a successful user registration, the system should show a success message on the GUI. This test case involves manually verifying that, once a user's image is cap-

tured and their details saved, the system displays a confirmation message that the user has been successfully registered.

GUI-010: Ensure the UI updates attendance status after a face match

When a user's face is successfully matched with a registered user, the system should update the attendance status. This test checks whether the attendance status on the UI is correctly updated with the user's details and the current timestamp once the face match is successful.

Business Logic Testing

BL-001: Validate that 100 images are correctly saved for a new user

When a new user is registered, the system should capture and store 100 images to build a robust facial recognition model. This test verifies that exactly 100 images are saved in the appropriate directory for each user. It also ensures that the images are properly named and stored in the user's unique folder.

BL-002: Check if images are converted to grayscale correctly

The system should convert all captured images to grayscale before processing them for face recognition. This test confirms that the conversion to grayscale happens without errors and that the saved images are indeed in grayscale format. It is verified manually by inspecting the saved images and ensuring they have a monochrome appearance.

BL-003: Test noise reduction algorithm applied to saved images

To improve the quality of captured images, a noise reduction algorithm should be applied during preprocessing. This test case ensures that the noise reduction (e.g., smoothing or filtering) is applied correctly to the captured images, improving their clarity and suitability for face recognition.

BL-004: Verify that face detection with Haar Cascade works properly

The system uses the Haar Cascade classifier to detect faces in captured images. This test verifies that the Haar Cascade algorithm is working effectively, correctly identifying and locating faces in different conditions (lighting, angle, etc.). It is manually tested by feeding various images into the system and ensuring faces are detected.

BL-005: Ensure the system detects and matches a registered user's face

This test ensures that when a user tries to mark attendance, the system successfully matches their face with the registered images. If the user's face is recognized, attendance should be recorded. The face matching process is verified manually by checking if the system identifies registered faces correctly during attendance.

BL-006: Test if an unregistered user is correctly rejected

An unregistered user's face should not be matched to any registered users, and the system should reject their attendance attempt. This test case ensures that the system correctly handles face recognition failures and ensures that unregistered users are not mistakenly marked as present.

BL-007: Validate that the system logs attendance with a timestamp

When a user's face is recognized, the system should log their attendance with the exact time and date. This test ensures that the system accurately records the time of attendance, and that the timestamp is correctly displayed and stored along with the user's ID.

BL-008: Check if duplicate attendance for the same user is prevented

The system should prevent duplicate attendance marks for the same user within a single session. This test case ensures that once a user's face is matched and attendance is recorded, they cannot be marked present again until a new session.

BL-009: Ensure face recognition works in different lighting conditions

The system should be robust enough to recognize faces under various lighting conditions, such as bright or dim environments. This test verifies that the face recognition algorithm works effectively under different lighting scenarios, ensuring accurate attendance marking regardless of the lighting.

BL-010: Test if attendance records are saved in the correct format (CSV/DB)

The system must save attendance records in a format that is easily accessible and readable, such as CSV or a database. This test case ensures that attendance data is correctly stored in the specified format, and that the saved records contain accurate information about the user and the timestamp.

Chapter 7

Project Screen Shots

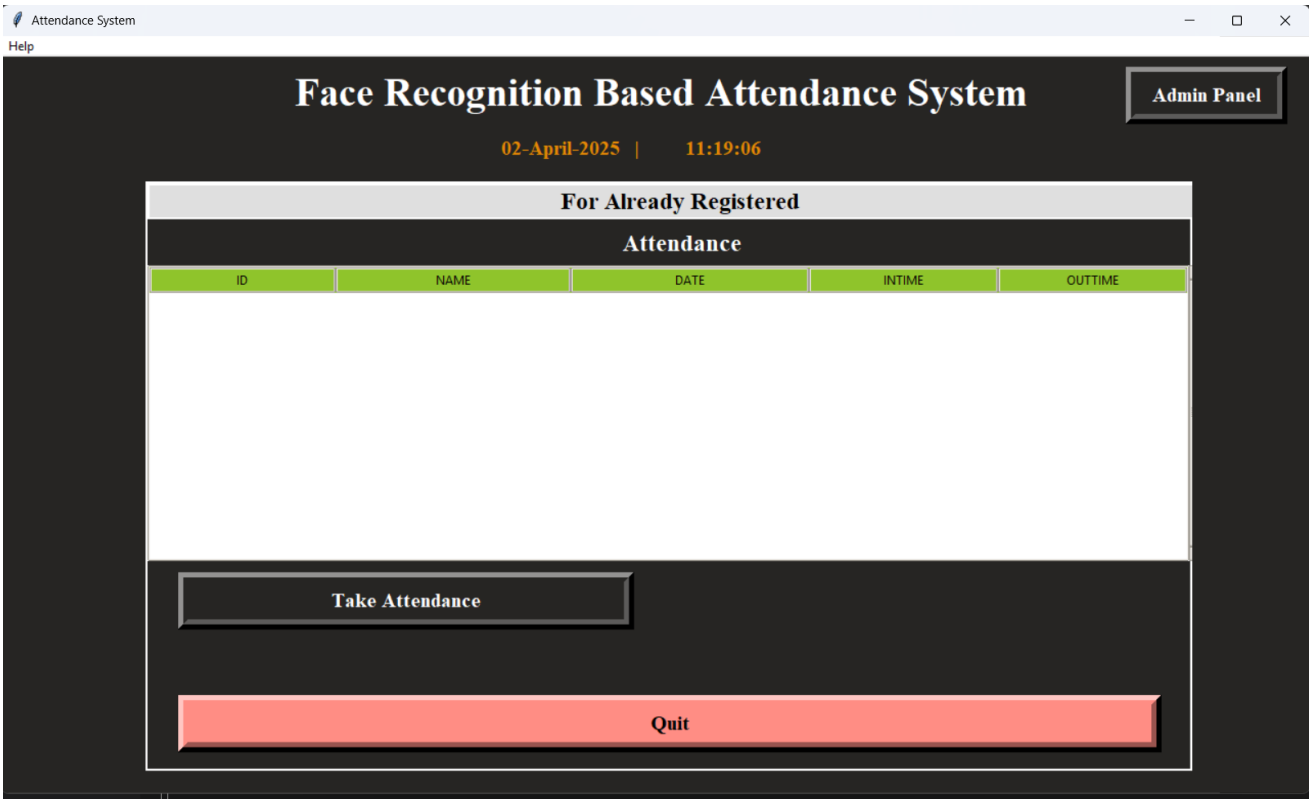

Figure 7.1: Dashboard

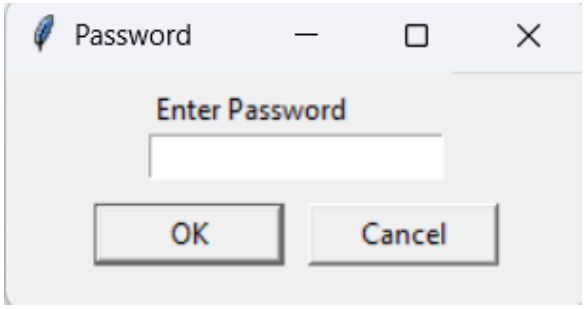

Figure 7.2: Admin Password

Figure 7.3: Admin Panel

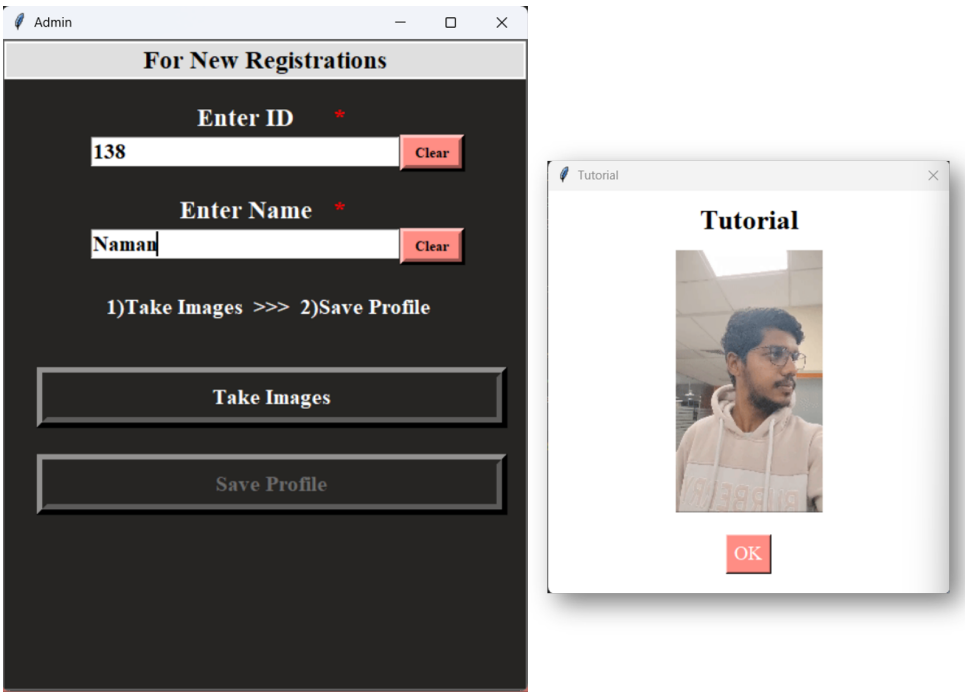

Figure 7.4: New Registration

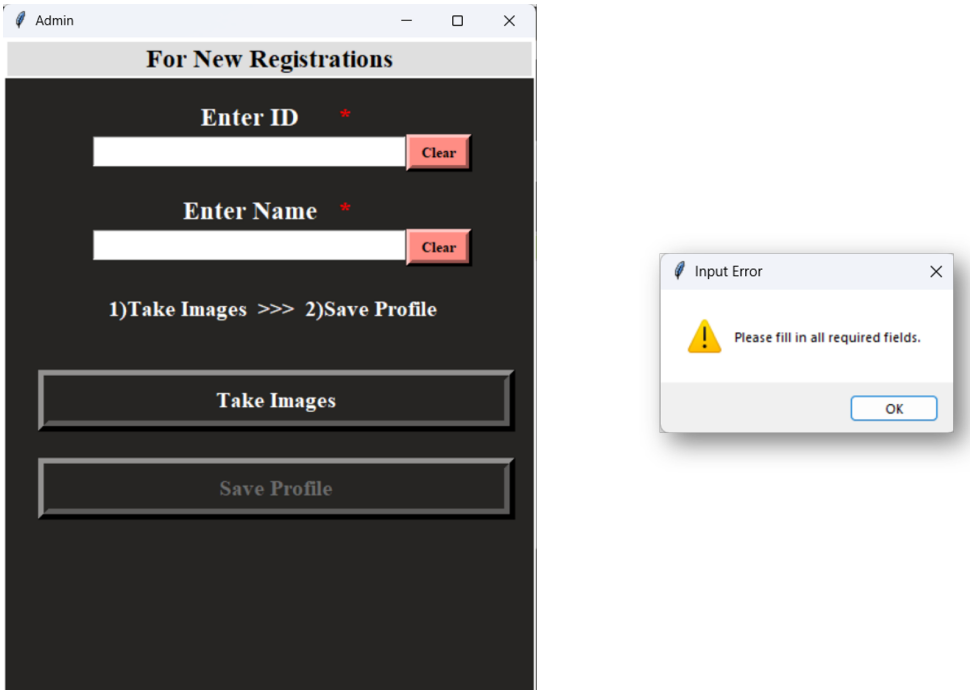

Figure 7.5: New Registration - Wrong Input

Figure 7.6: Taking Attendance

Chapter 8
Conclusion and Future Scope

8.1 Conclusion

The Face Attendance System, implemented using OpenCV for facial recognition, provides a cutting-edge solution for automating attendance management. By leveraging Python and computer vision techniques, the system efficiently identifies and authenticates users in real-time. OpenCV, a widely used computer vision library, ensures reliable facial recognition even under various environmental conditions.

This system significantly reduces human error in attendance marking and enhances security and convenience in educational institutions or workplaces. Through the integration of image processing and machine learning techniques, the project has successfully addressed key challenges such as scalability, speed, and reliability, offering a seamless and efficient solution for modern attendance systems.

8.2 Future Scope

While the Face Attendance System using OpenCV is effective, there are multiple directions for future improvements and enhancements. The following potential developments could further elevate the system's performance and usability:

- **Integration with Cloud-Based Systems:** The system could be extended to integrate with cloud platforms, allowing remote management of attendance data and real-time monitoring. Cloud storage would enable better data accessibility, scalability, and backup, ensuring a more flexible solution.
- **Real-Time Analytics and Reporting:** The inclusion of real-time analytics could provide valuable insights, such as attendance trends, patterns, and forecasting absenteeism. This would help in decision-making and resource management, particularly in large organizations or institutions.

- **Enhanced Facial Recognition with Deep Learning:** Further improvements in facial recognition accuracy can be achieved by integrating more advanced deep learning models. This could include multi-modal recognition, such as integrating mask detection or emotion recognition, to handle real-world scenarios like users wearing face masks.
- **Mobile Application Integration:** To enhance accessibility, a mobile application could be developed to allow users to register, mark attendance, and view reports directly from their smartphones. This would increase user engagement and make the system more versatile.
- **Multi-Location and Multi-Device Support:** Expanding the system to allow for attendance tracking across multiple locations and devices would be beneficial for organizations with branches in different areas. The centralized monitoring would allow real-time updates and data consolidation.
- **Incorporation of Other Biometric Data:** To further enhance security and reduce fraud, the system could incorporate additional biometric authentication methods such as fingerprint or iris recognition alongside facial recognition.

Chapter 9
UN Sustainable Development Goals

9.1 Mapping with UN sustainable development goals

The Facial Recognition Attendance System supports several United Nations Sustainable Development Goals (SDGs), contributing to the enhancement of educational efficiency and digital transformation. By automating attendance tracking, the system promotes Quality Education (SDG 4) and fosters Industry, Innovation, and Infrastructure (SDG 9), ensuring a more secure and technology-driven learning environment.

SDG Title : Goal 4 - Quality Education		
S.No	**Objective**	**Outcome**
1	Ensuring accurate and automated attendance tracking in educational institutions	Reduces proxy attendance and ensures fair assessment of student participation
2	Reducing administrative workload for teachers and institutions	Saves time spent on manual attendance, allowing teachers to focus on quality education

SDG Title : Goal 9 - Industry, Innovation, and Infrastructure		
S.No	**Objective**	**Outcome**
1	Enhancing security and efficiency using AI-based automation	Prevents unauthorized access and improves campus security
2	Promoting digital innovation in educational and corporate sectors	Encourages the adoption of smart technology, aligning with global digital transformation trends

GitHub Link

Link:

https://github.com/Samyak-it-is/AttendanceMS

Project Poster

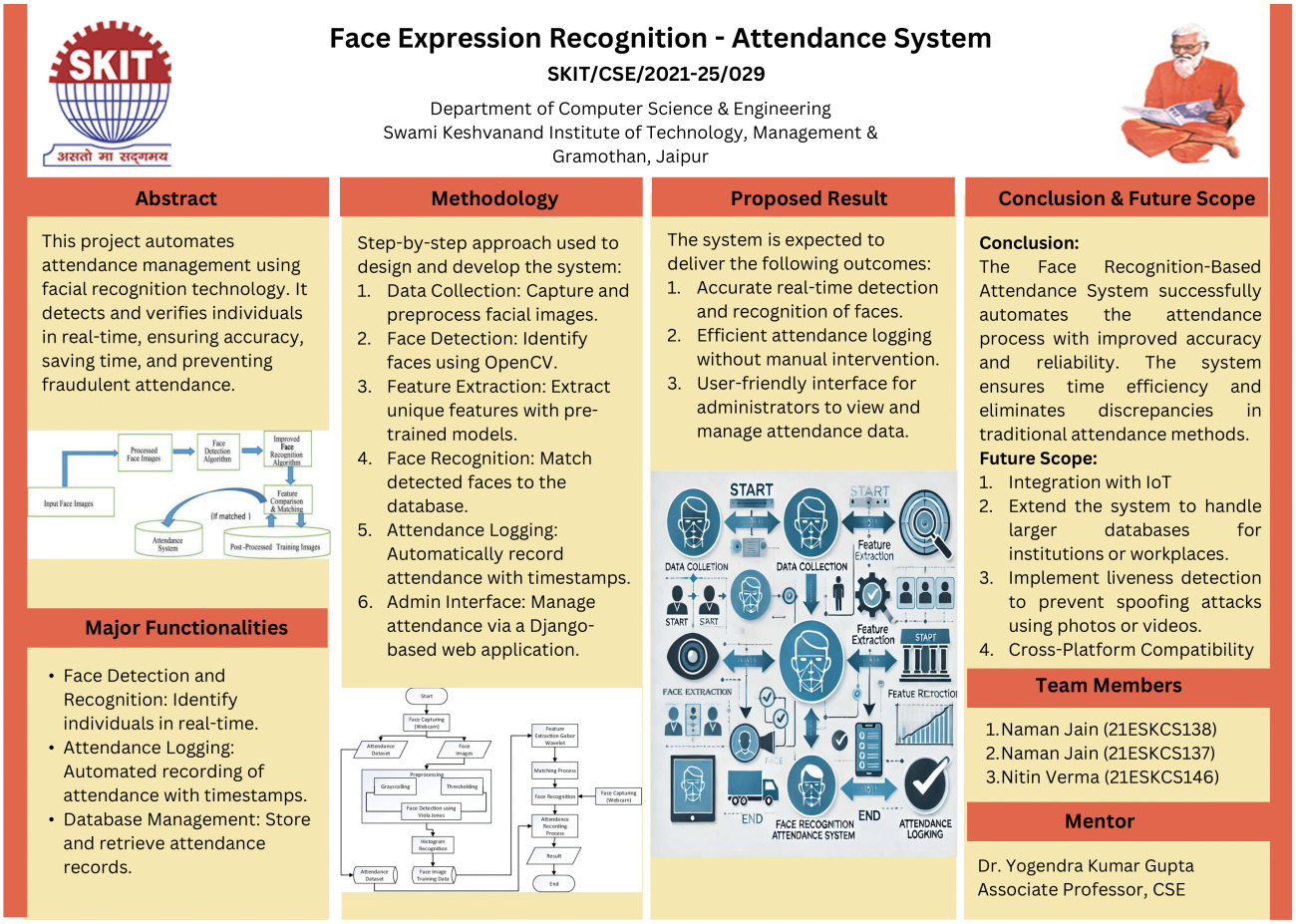

Figure 9.1: Project Poster

References

[1] *Python Official Docs:* https://docs.python.org/3/

[2] *Tkinter GUI Programming:* https://docs.python.org/3/library/tkinter.html

[3] *OpenCV Documentation:* https://docs.opencv.org/4.x/

[4] *Haar Cascade Classifier in OpenCV:* https://docs.opencv.org/4.x/db/d28/tutorial_cascade_classifier.html

[5] *Face Recognition with OpenCV:* https://docs.opencv.org/4.x/dc/dc3/tutorial_py_face_detection.html

[6] *NumPy:* https://numpy.org/doc/stable/

[7] *Pandas (if used for data processing):* https://pandas.pydata.org/docs/

[8] *Pillow (for Image Processing):* https://pillow.readthedocs.io/en/stable/