|  |  |  |
| --- | --- | --- |
| CHAPTER | TITLE | PAGE  NO |
| 1 | 1. SYNOPSIS    1. Title of the project    2. Introduction and Background    3. Objectives    4. Scope    5. Methodology and Approach    6. Software Used    7. Components    8. Advantages    9. Disadvantages   1.10 Conclusion | 04-07 |
| 2 | 1. SOFTWARE REQUIREMENT SPECIFICATION    1. INTRODUCTION       1. Purpose       2. Intended Audience and Readinf Suggestiona       3. Project Scope       4. Reference’s    2. Overall Description       1. Product Perspective       2. Product Features       3. User Classes and Characteristics       4. Operating Environment       5. Design and Implementation Constraints       6. User Documentation       7. Assumption and Dependencies    3. System Features       1. Face Detection and Recognition       2. Database Management       3. Attendance Tracking       4. Reporting and Analytics       5. Assumption and Dependency    4. Externam Interface Requirements   2.4.1 User Interface  2.4.2 Harware Interface  2.4.3 Software Interface  2.4.4 Cpmmunication Interface   * 1. Other Non-functional Requirements      1. Performance Requirement      2. Safety Requirement      3. Security Requirement      4. Software Quality Attributes   2. Other Requirements   2.6.1 Legal Requirements  2.6.2 Maintainance and Support  2.6.3 Training  2.6.4 Documentation  2.6.5 Scalability  2.6.6 Backup and Recovery  2.7 Appendix A:Glossary |  |

|  |  |  |
| --- | --- | --- |
| 3 | 1. SYSTEM DESIGN    1. Introduction    2. Context Flow Diagram       1. Notifications used in Context Flow Diagram    3. Data Flow Diagram       1. Notifications used in Data Flow Diagram    4. Entity -Relationship Diagram    5. ER-Diagram Symbols | 16-25 |
| 4 | 1. DETAILED DESIGN    1. Introduction    2. Applicable documents    3. Structure of the software package    4. Modular decomposition of components | 26-30 |
| 5 | 1. IMPLEMENTATION    1. Implementation of software    2. Introduction    3. Normalised tables | 31-34 |

|  |  |  |
| --- | --- | --- |
| 6 | 1. CODING    1. Introduction | 35-62 |
| 7 | 1. TESTING    1. Introduction 7.2Testing Objectives 7.3Testing Methods 7.4Testing Steps       1. Unit Testing       2. Integration Testing       3. Validation       4. Output Testing       5. User Acceptance Testing 7.5Test Cases | 63-69 |
| 8 | 8.USER INTERFACE | 70-80 |
| 9 | CONCLUSION | 81-82 |
| 10 | LIMITATIONS | 83-84 |
| 11 | BIBLIOGRAPHY | 85-86 |

SYNOPSIS

## Chapter 1 SYNOPSIS

**1.1 Title of the Project:**

**“Face Recognition Attendance Management System”**

**1.2 Introduction and Background:**

we are making a system which can recognize face and match with its own database. It will make the attendance system more authentic. Our primary goal is to help the lecturers, improve and organize the process of track and manage student attendance and absenteeism.

Additionally, we seek to provide a valuable attendance service for both teachers and students. Reduce manual process errors by provide automated and a reliable attendance system uses face recognition technology.

Increase privacy and security which student cannot presenting himself or his friend while they are not. Flexibility, Lectures capability of editing attendance records.

**1.3 Objectives:**

Our aim is to build up a face recognition system where a human will stand in front of the system and a camera will match the face along with its database.

There will no extra RFID card people need to carry any more and this system will be the most authentic system of taking attendance. We will try to build this system as efficient as possible.

**1.4 Scope:**

We have divided our work into two parts.

1. Sensing Face and Capture.

2. Match with database.

**1.5 Methodology and Approach:**

Face detection involves separating image windows into two classes; one containing faces (turning the background (clutter).

The first step is a classification task that takes some arbitrary image as input and outputs a binary value of yes or no, indicating whether there are any faces present in the image. The second step is the face localization task that aims to take an image as input and output the location of any face or faces within that image as some bounding box with (x, y, width, height).

After taking the picture the system will compare the equality of the pictures in its database and give the most related result.

There is still no work combining image processing by Raspbian operating system with open CV platform. So, we want to introduce this model.

**1.6 Software Used:**

Python, Open CV , Raspbian OS

**1.7 Components:**

The total estimated components to complete the project is provided in Table 2.

|  |
| --- |
| Name of Item |
| HD Webcam (Camera Module) |
| Raspberry Pi with LED Display |
| SD card |
| Battery |
| Wires and Other |

**1.8 Advantages:**

* The system stores the faces that are detected and automatically marks attendance.
* Ease of use is manipulate and recognize the faces in real time using. Multiple face detection. Multipurpose software Can be used in different places.

**1.9 Disadvantages:**

* The accuracy of the system is not 100%. It can only detect face from a limited distance.

**1.10 Conclusion:**

In conclusion, the Face Recognition Attendance Management System project offers an innovative solution for efficient attendance tracking. By utilizing facial recognition technology, employees,students can easily record their presence, saving time and resources.

The system enhances security by securely storing unique facial features, preventing fraudulent attendance practices and unauthorized access.

Overall, this project simplifies attendance management, improves productivity, and fosters a transparent and accountable work environment.

# SOFTWARE REQUIREMENT SPECIFICATION (SRS)

## Chapter 2

**SOFTWARE REQUIREMENT SPECIFICATION (SRS)**

## : Introduction

The introduction section provides an overview of the Software Requirements Specification (SRS) document. It includes the purpose, intended audience and reading suggestions, project scope, and references

## : Purpose

The purpose of this SRS document is to provide a clear and complete description of the requirements for the Attendance Management System based on face recognition technology. The document serves as a communication tool between the development team, stakeholders, and clients, ensuring that all parties have a common understanding of what the software is supposed to do and how it is supposed to behave.

## : Intended Audience and Reading

The intended audience for this document includes the development team, project managers, stakeholders, and clients. This document is intended to be read by those involved in the development, testing, and maintenance of the Attendance Management System based on face recognition technology.

## : Project Scope

The project scope of this document includes the development of an attendance management system that uses face recognition technology to capture and match student faces with a database. The system will be able to accurately track student attendance, provide automated and reliable attendance records, and increase privacy and security in the classroom.

## : References

* Mark Andrejevic & Neil Selwyn (2020) “Facial recognition technology in schools: critical questions and concerns, Learning, Media and Technology,” URL: [https://www.tandfonline.com/doi/full/10.1080/17439884.2020.1686 014](https://www.tandfonline.com/doi/full/10.1080/17439884.2020.1686%20014)
* Farah Deeba, Aftab Ahmed, Hira Memon “Real Time Face Recognition

of Human Faces by using LBPH and Viola Jones Algorithm.” 2019.

URL: https://pdfs.semanticscholar.org/3255/0898eec9c4424932e70e5e32c98

b0220a747.pdf

* X. Zhao and C. Wei, "A real-time face recognition system based on the improved LBPH algorithm," 2017 URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8124508&is number=8124489

## 2.2 OVERALL DESCRIPTION

* + 1. **: Product perspective**

The attendance management system based on face recognition technology is a standalone application that is designed to be integrated with existing attendance management systems. The system will provide an additional layer of security and accuracy to the attendance management process.

## : Product Features

The attendance management system based on face recognition technology will include features such as face detection and recognition, database management, attendance tracking, and reporting.

**2.2.3 : User Classes and Characteristics**

The users of the attendance management system based on face recognition technology will include lecturers, students, and administrators. Lecturers will be able to take attendance easily, while students will not need to carry any additional cards. Administrators will be able to manage the attendance records and generate reports.

**2.2.4 : Operating Environment**

The attendance management system based on face recognition technology will be developed to work on a Windows or Linux operating system. The system will require a camera for face detection and recognition and a reliable internet connection to access the database

**2.2.5 : Design and Implementation Constraints**

The attendance management system based on face recognition technology will be designed and implemented using the latest technologies and best practices. The system will need to be scalable, secure, and efficient.

**2.2.6 : User Documentation**

The attendance management system based on face recognition technology will include user manuals, installation guides, and tutorials for users and administrators.

**2.2.7 : Assumption and Dependencies**

The development of the attendance management system based on face recognition technology assumes that the hardware and software required for the system are readily available. The system is dependent on a reliable internet connection to access the database.

## 2.3 SYSTEM FEATURES

* + 1. **: Face Recognition and Detection:**

The system will be able to detect and recognize human faces accurately and quickly using face recognition technology.

## : Database Management:

The system will have a database management system to store and manage the data related to the students and faculty members.

**2.3.3 Attendance Tracking:**

The system will track and record attendance for each class and store it in the database. The attendance tracking will be done automatically through face recognition technology.

**2.3.4 Reporting and Analytics**

The system will provide various reports and analytics related to attendance records for teachers and administrators to access.

**2.4 EXTERNAL INTERFACE REQUIREMENTS**

The external interface requirements section includes details of the user interfaces, hardware interfaces, software interfaces, and communication interfaces required for the attendance management system based on face recognition technology.

**2.4.1 User Interfaces:**

The user interface for the attendance system will consist of a web-based interface that can be accessed through any device with an internet connection. The interface will allow teachers to view and manage attendance records, add and remove students from the system, and edit attendance records. Students will be able to view their own attendance records through a separate interface.

The user interface will be designed to be intuitive and user-friendly, with clear navigation and easy-to-use controls. The interface will be responsive, meaning it will adjust to the size of the device screen and be optimized for both desktop and mobile use.

**2.4.2 Hardware Interfaces**

The attendance system will require a camera to capture the image of the student's face. The camera should have a resolution of at least 720p to ensure accurate face recognition. The system will also require a computer or server to process the face recognition algorithms and store attendance records.

**2.4.3 Software Interfaces**

The attendance system will be built using open-source software libraries for face detection and recognition. The system will communicate with the camera to capture images and with the database to store attendance records. The system will be built using a web-based framework and will utilize standard web protocols such as HTTP and WebSocket.

**2.4.4 Communication Interfaces**

The attendance system will communicate with the camera using standard communication protocols such as USB or Ethernet. The system will also communicate with the database using standard database protocols such as SQL. The system will communicate with the user interface using standard web protocols such as HTTP and WebSocket.

**2.5 OTHER NON FUNCTIONAL REQUIREMENTS**

**2.5.1 Peformance Requirements**

The attendance system should be able to recognize a face and record attendance within 5 seconds of the student standing in front of the camera. The system should be able to handle up to 1000 students and 10 teachers simultaneously**.**

**2.5.2 Safety Requirements**

The attendance system should not collect any personal information other than the student's face image. The system should ensure the privacy and security of the student's personal data.

**2.5.3 Security Requirements**

The attendance system should be secure and protect against unauthorized access to attendance records. The system should utilize standard security protocols such as SSL encryption and password protection.

**2.5.4 Software Quality Attributes**

The attendance system should be reliable, maintainable, and scalable. The system should be designed using best practices for software development and should be thoroughly tested to ensure proper functionality.

**2.6 OTHER REQUIREMENTS**

**2.6.1 Legal Requirements**

The system must comply with all legal requirements and regulations related to data privacy and protection.

**2.6.2 Maintenance and Support**

The system must be easy to maintain and support. The vendor must provide regular updates and upgrades to the system.

**2.6.3 Training**

The vendor must provide comprehensive training to users on how to use the system. The training must cover all aspects of the system, including data input, system management, and troubleshooting.

**2.6.4 Documentation**

The documentation must cover all aspects of the system, including system requirements, installation, configuration, operation, maintenance, and troubleshooting.

**2.6.5 Scalability**

The system must be scalable to accommodate additional users and functionality as needed. The vendor must provide guidelines for adding new users and functionality to the system.

**2.6.6 Backup and Recovery**

The system must have a backup and recovery mechanism in place. The vendor must provide guidelines for backing up data and recovering the system in case of a failure.

**2.7 Appendix A:Glossary**

|  |  |
| --- | --- |
| **TERM** | **DEFINITION** |
| Attendance | The act of being present in a particular location or event. |
| CFD | Context Flow Diagram is a visual representation of the flow of data through a system or process. |
| Database | A structured collection of data that is stored and accessed electronically. |
| DFD | Data Flow Diagram is a graphical representation of the flow of data through a system or process. |
| HFR | Human Face Recognition is a technology that uses biometrics to map facial features |
| RFID | Radio-frequency identification is a technology that uses electromagnetic fields to automatically identify and track tags attached to objects. |
| SRS | Software Requirements Specification is a document that describes the system's functional and non-functional requirements. |
| User Interface | The part of the system or software that allows the user to interact with it. |

# SYSTEM DESIGN

## Chapter 3

## SYSTEM DESIGN

* 1. **INTRODUCTION**

System design is solution to the creation of a new system. This important phase is composed of several steps. It provides the understanding and procedural details necessary for implementing the system recommended in feasibility study.

Emphasis is on translating the performance requirements into design specifications.

Design goes through logical and physical development. Logical designs preview the present physical system input and output specifications and details the implementations plan and internal logic of each of the modules. Here it should be ensured that the design is technically sound with minimum of redundancy and maintains. The physical design maps out the details of the physical system, plans the system and specifies any hardware and software.The goal of the system design phase is to produce a model or representation of the system which can be used to build the system. Here, the emphasis is on translating the requirements of the system into design specification.

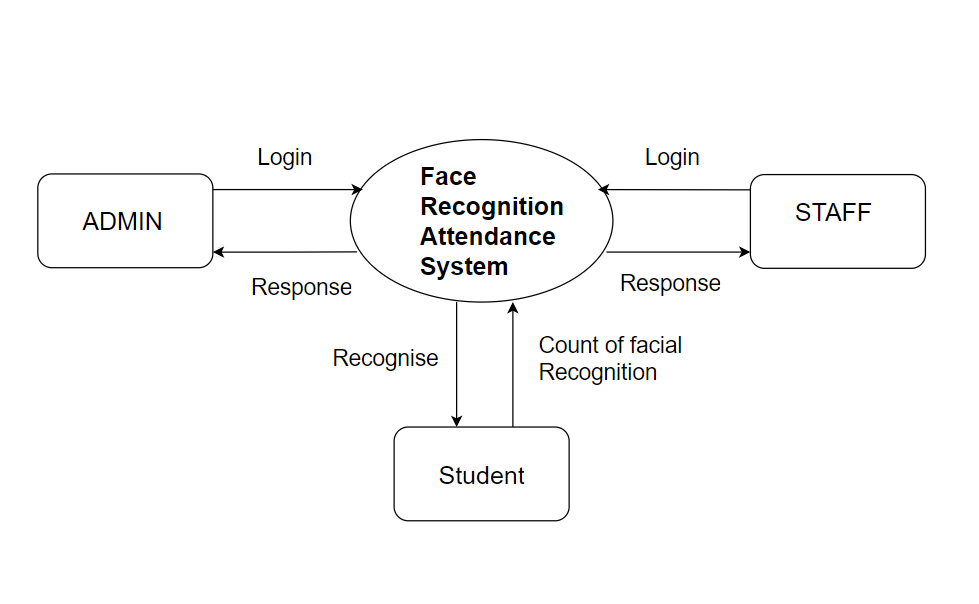
## : CONTEXT FLOW DIAGRAM

A context flow diagram is a top-level data flow diagram. It only contains one process node that generalises the function of the entire system in relationship to external entities. In the context flow diagram, the entire system is treated as a single process and all its inputs, outputs, sinks and sources are identified and shown.

### Notations used in Context Flow diagram

|  |  |  |
| --- | --- | --- |
| Name | Notation | Description |
| Process |  | A process transforms incoming data flow into outgoing data flow. The process are shown by named  circles. |
| Data flows |  | Data flows are pipelines through which packets of information flow. Label the arrows with the names of the data that moves through it. |
| External Entity |  | External entities are the objects outside the system, with which the system communicates. External Entities are the sources and destinations of the systems  inputs and outputs. |

**CONTEXT FLOW DIAGRAM (Level 0):**



## : DATA FLOW DIAGRAM

A Data Flow Diagram is the graphical representation of the data through an information system, modelling its process aspects. A DFD also known as bubble chart or dataflow graph are commonly used during problem analysis. DFD’S are very useful in understanding a system and can be efficiently used during analysis. A DFD will show what kind of information will be input to and output from the system, how the data will advance through the system, and where the data will be stored it does not show information about process timing or whether process will operate in sequence or in parallel format, a DFD is often used as the preliminary step to create an overview of the system without going into great detail, which can

later be elaborated, it can also be used for the visualization of data processing DFD shows the movement of data through different transformations or process in the system. A Data Flow Diagram (DFD) illustrates how data is processed by a system in terms of inputs and outputs. As its name indicates its focus is on the flow of information, where it goes and how it gets stored. DFD provide critical insights into the systems and way the information passes through it. DFD helps structure every element of the system, keep them logically intact and interconnected. On the other hand, you have the customers who need to know what is going on in a digestible easy to follow manner. All data flow diagrams include four main elements: entity, process, data store and data flow.

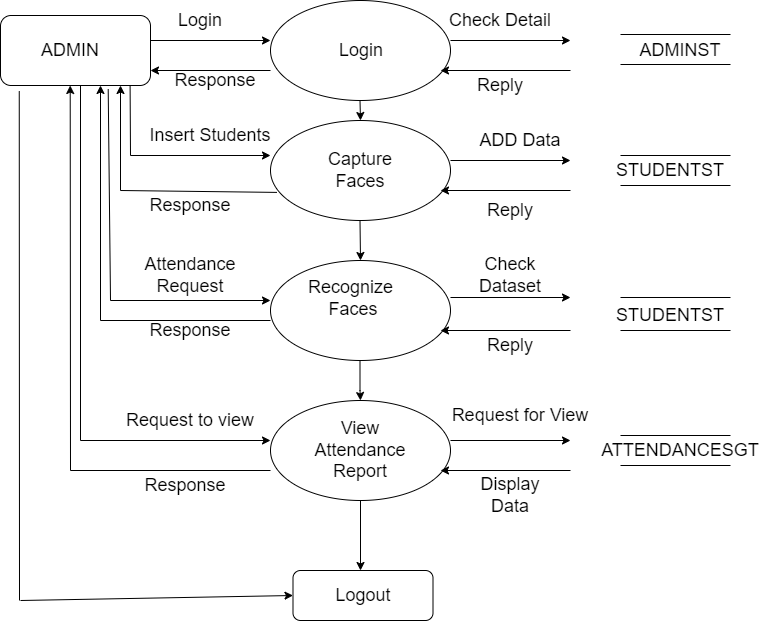
Rules Regarding DFD Construction:

* A process cannot have only outputs.
* A process cannot have only inputs.
* The inputs to a process must be sufficient to produce output from the processes.
* All data stores must be connected to at least one process.
* All data stores must be connected to a sink or source
* A data flow can have only one direction of flow. Multiple data to and or from the same processes and data store must be shown by separate arrows.
* If the same exact data flows to two separate arrows, it should be represented by forked arrow.
* Data cannot flow directly back into the process it just left. All data flow must be named using a noun phrase.

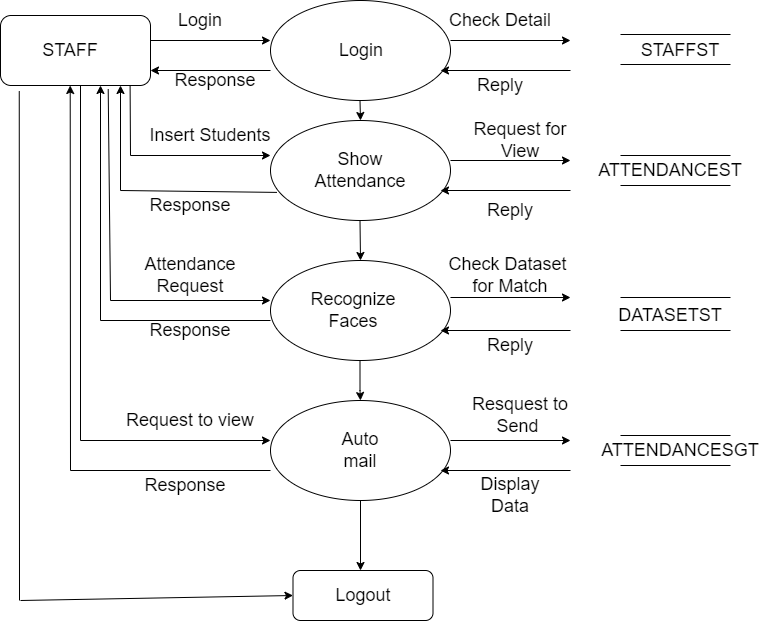
### 3.1.1: Notations used to create DFD

|  |  |
| --- | --- |
| **Symbol** | **Description** |
|  | The circle represents a process. A process is named and each process is  represented by a named circle. |
|  | The source or the sink is represented as a rectangular box. It represents the net originator or the consumer of the  data that flows in the system. |
|  | The arrow represents the flow of data through the system. The labelled arrows enter or leave the bubbles. |
|  | The database is represented with the open box symbol. |

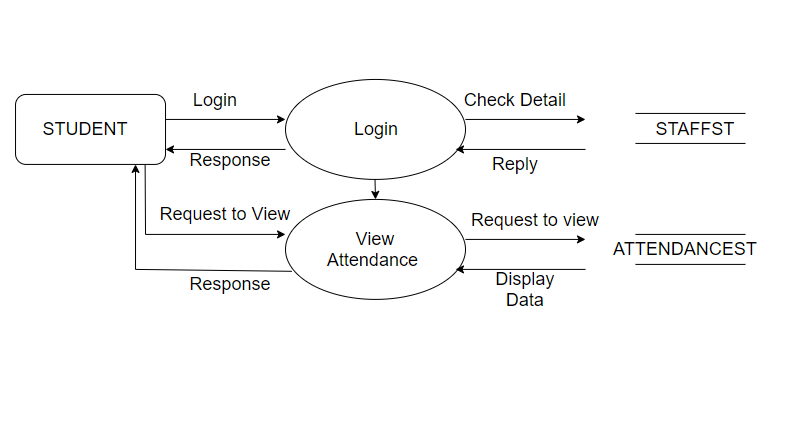
**ADMIN – Data Flow Diagram for Attendance management system**

****

**Staff Side Data Flow Diagram – Attendance Management System**



**Student Side Data Flow Diagram Attendance Management System**



## : ENTITY-RELATIONSHIP (ER) DIAGRAM

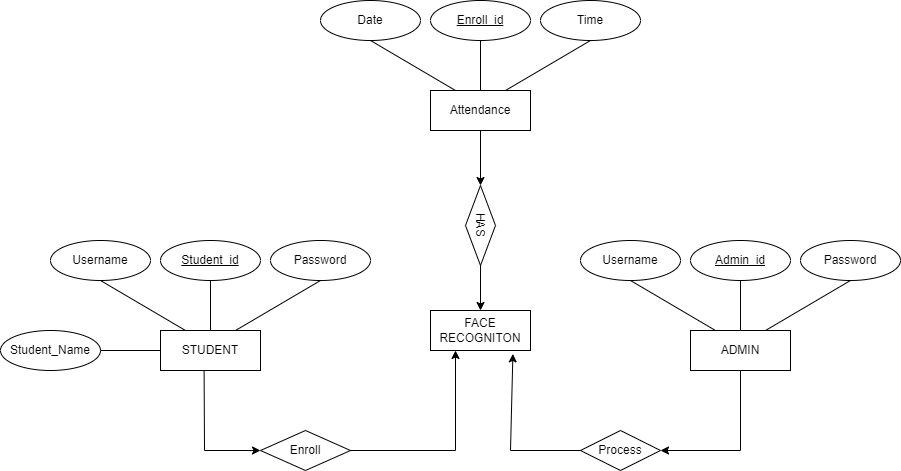
ERD is a diagram that displays the relationship of entity sets stored in a database. In other words, ER diagrams help to explain the logical structure of databases.

ER diagrams are created based on three basic concepts: entities, attributes and relationships. Following are the main components and its symbols in ER Diagrams:

|  |  |  |
| --- | --- | --- |
| **Name** | **Notation** | **Description** |
| **Entity** |  | It may be an object with physical existence or conceptual existence. It is represented by a rectangle |
| **Attribute** |  | The properties of the entity can be a attribute. It is represented by Ellipse. |
| **Relationship** |  | Whenever an attribute of one entity refer to the another entity, some relationship exists.  It is represented by a Diamond. |
| **Link** |  | Lines link attribute to entity sets and entity  sets to relation. |
| **Derived Attribute** |  | Dashed ellipse denotes derived attributes. |
| **Key Attribute** |  | An entity type usual has an attribute whose values are distinct for each individual entry in the entity set. It is represented by a Underlined word in ellipse. |

|  |  |  |
| --- | --- | --- |
| **Multivalued Attribute** |  | Attribute that have different number of values for particular attribute.  It is represents multi-valued attributes. |
| **Cardinality Ratio** | 1) 1:1  2) 1:M  3) M:1  4) M:M | It specifies the maximum number of relationship instance that an entity can participate in. There are are four cardinality ratios. |

**ER Diagram :**



# DETAILED DESIGN

## Chapter 4

## DETAILED DESIGN

* 1. **: INTRODUCTION**

This document's preparation aims to explain full design specifications. The general definition and features of the project, design constraints, the overall system architecture, and the data architecture are all included in this detailed design report. Additionally, there will be a brief explanation of our current status and the project schedule in the relevant sections. In order to aid the programmer in correctly and easily understanding all the information stated in this document, the design of the system and its subsystems/modules will be described verbally and visually using diagrams.

### : Applicable documents

### During the design process of the face recognition project, the following documents were referred to and used as a basis for the design decisions:

### Technical specifications: These documents outline the requirements and expectations for the face recognition system, including performance targets, accuracy metrics, and any constraints that need to be considered.

### Design guidelines: These guidelines provide best practices and recommendations for designing robust and efficient face recognition systems. They cover topics such as feature extraction methods, algorithm selection, and system architecture.

### 4.3: Structure of the software package

### 

Fig : Structure Chart

* Face detection module: Responsible for detecting and localizing faces within an input image or video stream. It utilizes the Haar cascade classifier to identify potential face regions.
* Feature extraction module: Extracts representative features from the detected face regions. Currently, the code uses a simple method of resizing the face images to a fixed size of 50x50 pixels to create feature vectors.
* Face recognition module: Uses the K-Nearest Neighbors (KNN) algorithm for face recognition. It trains the KNN classifier using the extracted face features and associated labels.
* Attendance recording module: Manages the recording of attendance based on the recognized faces. It captures the current date and time and stores the attendance data in a CSV file.

These components interact with each other in a sequential manner, where face detection is performed first, followed by feature extraction and face recognition. The attendance recording module is invoked upon user input to mark attendance.

### 4.4 :Modular decomposition of components

### The face recognition project consists of several functional modules that work together to enable accurate face recognition and attendance recording. These modules are designed to encapsulate specific functionality and promote code modularity, reusability, and maintainability. The following modules are part of the face recognition system:

### Face Detection Module:

### Purpose: This module is responsible for detecting and localizing faces within an input image or video stream.

### Functionality: It utilizes the Haar cascade classifier, an object detection algorithm, to identify potential face regions.

### Input: Input image or video frames.

### Output: Detected face regions (bounding boxes) within the input data.

### Feature Extraction Module:

### Purpose: This module extracts representative features from the detected face regions.

### Functionality: Currently, it resizes the face images to a fixed size of 50x50 pixels and flattens them to create feature vectors.

### Input: Detected face regions (bounding boxes).

### Output: Feature vectors representing the face images.

### Face Recognition Module:

### Purpose: This module performs face recognition using the K-Nearest Neighbors (KNN) algorithm.

### Functionality: It trains the KNN classifier using the extracted face features and corresponding labels.

### Input: Feature vectors representing the face images, along with their associated labels.

### Output: Predicted labels for new face images

### Attendance Recording Module:

### Purpose: This module manages the recording of attendance based on the recognized faces.

### Functionality: It captures the current date and time and stores the attendance data in a CSV file.

### Input: Recognized face labels and timestamps.

### Output: Attendance records stored in a CSV file.

### By decomposing the face recognition system into these functional modules, the system becomes more modular, making it easier to develop, test, and maintain each module independently. The interaction between these modules ensures the accurate recognition of faces and the recording of attendance based on the recognized faces.

# CODING

## Chapter 5

## CODING

**5.1 Introduction**

The goal of the coding or programming phase is to translate the design of the system produced during the design phase into code in a given programming language. It is then executed by the which can be executed by a computer and that performs the consumptions specified by the design. The coding during this phase affects the both testing and maintainers profoundly. The goal during this phase is not simplify the job of the programmer. Rather the goal during simplify the job of the tester and maintainer.

The coding is done with following characteristics in mind:

* Ease of design to code translation
* Code efficiency
* Memory efficiency
* Response time
* Maintenance
* Security
* Simple easy to understand
* Efficient and consistent logic

### Source Code

### Main.py

### import tkinter as tk

### import os

### def open\_admin():

### os.system("python admin.py")

### def open\_staff():

### os.system("python staff.py")

### def open\_student():

### os.system("python student.py")

### window = tk.Tk()

### window.title("Home")

### screen\_width = window.winfo\_screenwidth()

### screen\_height = window.winfo\_screenheight()

### window\_width = 500

### window\_height = 300

### window.geometry(f"{window\_width}x{window\_height}+{int((screen\_width - window\_width) / 2)}+{int((screen\_height - window\_height) / 2)}")

### primary\_color = "#3498db"

### secondary\_color = "#ffffff"

### font\_color = "#333333"

### heading\_frame = tk.Frame(window, bg="white")

### heading\_frame.pack(pady=20)

### heading\_label = tk.Label(heading\_frame, text="ATTENDANCE SYSTEM USING FACE RECOGNITION", font=("Arial", 14, "bold"), fg=font\_color, bg="white")

### heading\_label.pack()

### button\_frame = tk.Frame(window, bg="white")

### button\_frame.pack(pady=20)

### buttons = []

### labels = ["Admin", "Staff", "Student"]

### button\_width = 15

### button\_height = 3

### for i, label in enumerate(labels):

### button = tk.Button(button\_frame, text=label, command=lambda l=label: open\_file(l), bg=primary\_color, fg=secondary\_color, relief=tk.RAISED, width=button\_width, height=button\_height)

### button.grid(row=0, column=i, padx=10)

### def open\_file(label):

### if label == "Admin":

### open\_admin()

### elif label == "Staff":

### open\_staff()

### elif label == "Student":

### open\_student()

### button\_frame.grid\_columnconfigure(0, weight=1)

### button\_frame.grid\_columnconfigure(1, weight=1)

### button\_frame.grid\_columnconfigure(2, weight=1)

### subheading\_frame = tk.Frame(window, bg="white")

### subheading\_frame.pack(pady=10)

### project\_label = tk.Label(subheading\_frame, text="Project by:", font=("Arial", 12))

### project\_label.pack(side=tk.LEFT)

### names\_label = tk.Label(subheading\_frame, text="Abhishek & Allan", font=("Arial", 12, "bold"))

### names\_label.pack(side=tk.LEFT)

### window.mainloop()

### Admin.py

### import tkinter as tk

### from tkinter import messagebox

### import subprocess

### def validate\_login():

### admin\_username = "admin"

### admin\_password = "pass"

### entered\_username = username\_entry.get()

### entered\_password = password\_entry.get()

### if entered\_username == admin\_username and entered\_password == admin\_password:

### messagebox.showinfo("Login Successful", "Welcome, Admin!")

### window.withdraw()

### subprocess.call(["python", "admin\_home.py"])

### window.destroy()

### 

### else:

### messagebox.showerror("Login Failed", "Invalid username or password. Please try again.")

### window = tk.Tk()

### window.title("Attendance Management System")

### primary\_color = "#3498db"

### secondary\_color = "#ffffff"

### font\_color = "#333333"

### bg\_color = "#f2f2f2"

### entry\_style = {"foreground": font\_color, "background": secondary\_color, "highlightbackground": primary\_color, "highlightcolor": primary\_color, "highlightthickness": 2, "relief": tk.FLAT}

### label\_style = {"foreground": font\_color, "background": bg\_color, "font": ("Arial", 12), "padx": 5, "pady": 5}

### button\_style = {"foreground": secondary\_color, "background": primary\_color, "font": ("Arial", 12), "padx": 10, "pady": 5, "relief": tk.RAISED}

### window.configure(bg=bg\_color)

### window\_width = 400

### window\_height = 250

### screen\_width = window.winfo\_screenwidth()

### screen\_height = window.winfo\_screenheight()

### x\_coordinate = int((screen\_width / 2) - (window\_width / 2))

### y\_coordinate = int((screen\_height / 2) - (window\_height / 2))

### window.geometry(f"{window\_width}x{window\_height}+{x\_coordinate}+{y\_coordinate}")

### window.resizable(False, False)

### heading\_label = tk.Label(window, text="ADMIN LOGIN", font=("Arial", 16, "bold"))

### heading\_label.configure(\*\*label\_style)

### heading\_label.pack(pady=10)

### username\_label = tk.Label(window, text="Username:", \*\*label\_style)

### username\_label.pack()

### username\_entry = tk.Entry(window, \*\*entry\_style)

### username\_entry.pack()

### password\_label = tk.Label(window, text="Password:", \*\*label\_style)

### password\_label.pack()

### password\_entry = tk.Entry(window, show="\*", \*\*entry\_style)

### password\_entry.pack()

### login\_button = tk.Button(window, text="Login", command=validate\_login, \*\*button\_style)

### login\_button.pack(pady=10)

### window.mainloop()

### staff.py

### import tkinter as tk

### from tkinter import messagebox

### import subprocess

### def validate\_login():

### admin\_username = "staff"

### admin\_password = "pass"

### entered\_username = username\_entry.get()

### entered\_password = password\_entry.get()

### if entered\_username == admin\_username and entered\_password == admin\_password:

### messagebox.showinfo("Login Successful", "Welcome, Staff!")

### window.withdraw()

### subprocess.call(["python", "staff\_home.py"])

### window.destroy()

### 

### else:

### messagebox.showerror("Login Failed", "Invalid username or password. Please try again.")

### window = tk.Tk()

### window.title("Attendance Management System")

### primary\_color = "#3498db"

### secondary\_color = "#ffffff"

### font\_color = "#333333"

### bg\_color = "#f2f2f2"

### entry\_style = {"foreground": font\_color, "background": secondary\_color, "highlightbackground": primary\_color, "highlightcolor": primary\_color, "highlightthickness": 2, "relief": tk.FLAT}

### label\_style = {"foreground": font\_color, "background": bg\_color, "font": ("Arial", 12), "padx": 5, "pady": 5}

### button\_style = {"foreground": secondary\_color, "background": primary\_color, "font": ("Arial", 12), "padx": 10, "pady": 5, "relief": tk.RAISED}

### window.configure(bg=bg\_color)

### window\_width = 400

### window\_height = 250

### screen\_width = window.winfo\_screenwidth()

### screen\_height = window.winfo\_screenheight()

### x\_coordinate = int((screen\_width / 2) - (window\_width / 2))

### y\_coordinate = int((screen\_height / 2) - (window\_height / 2))

### window.geometry(f"{window\_width}x{window\_height}+{x\_coordinate}+{y\_coordinate}")

### window.resizable(False, False)

### heading\_label = tk.Label(window, text="STAFF LOGIN", font=("Arial", 16, "bold"))

### heading\_label.configure(\*\*label\_style)

### heading\_label.pack(pady=10)

### username\_label = tk.Label(window, text="Username:", \*\*label\_style)

### username\_label.pack()

### username\_entry = tk.Entry(window, \*\*entry\_style)

### username\_entry.pack()

### password\_label = tk.Label(window, text="Password:", \*\*label\_style)

### password\_label.pack()

### password\_entry = tk.Entry(window, show="\*", \*\*entry\_style)

### password\_entry.pack()

### login\_button = tk.Button(window, text="Login", command=validate\_login, \*\*button\_style)

### login\_button.pack(pady=10)

### window.mainloop()

### Student.py

### import tkinter as tk

### from tkinter import messagebox

### import subprocess

### def validate\_login():

### admin\_username = "test"

### admin\_password = "pass"

### entered\_username = username\_entry.get()

### entered\_password = password\_entry.get()

### if entered\_username == admin\_username and entered\_password == admin\_password:

### messagebox.showinfo("Login Successful", "Welcome!")

### window.withdraw()

### subprocess.call(["python", "stu\_home.py"])

### window.destroy()

### 

### else:

### messagebox.showerror("Login Failed", "Invalid username or password. Please try again.")

### window = tk.Tk()

### window.title("Attendance Management System")

### primary\_color = "#3498db"

### secondary\_color = "#ffffff"

### font\_color = "#333333"

### bg\_color = "#f2f2f2"

### entry\_style = {"foreground": font\_color, "background": secondary\_color, "highlightbackground": primary\_color, "highlightcolor": primary\_color, "highlightthickness": 2, "relief": tk.FLAT}

### label\_style = {"foreground": font\_color, "background": bg\_color, "font": ("Arial", 12), "padx": 5, "pady": 5}

### button\_style = {"foreground": secondary\_color, "background": primary\_color, "font": ("Arial", 12), "padx": 10, "pady": 5, "relief": tk.RAISED}

### window.configure(bg=bg\_color)

### window\_width = 400

### window\_height = 250

### screen\_width = window.winfo\_screenwidth()

### screen\_height = window.winfo\_screenheight()

### x\_coordinate = int((screen\_width / 2) - (window\_width / 2))

### y\_coordinate = int((screen\_height / 2) - (window\_height / 2))

### window.geometry(f"{window\_width}x{window\_height}+{x\_coordinate}+{y\_coordinate}")

### window.resizable(False, False)

### heading\_label = tk.Label(window, text="STUDENT LOGIN", font=("Arial", 16, "bold"))

### heading\_label.configure(\*\*label\_style)

### heading\_label.pack(pady=10)

### username\_label = tk.Label(window, text="Username:", \*\*label\_style)

### username\_label.pack()

### username\_entry = tk.Entry(window, \*\*entry\_style)

### username\_entry.pack()

### password\_label = tk.Label(window, text="Password:", \*\*label\_style)

### password\_label.pack()

### password\_entry = tk.Entry(window, show="\*", \*\*entry\_style)

### password\_entry.pack()

### login\_button = tk.Button(window, text="Login", command=validate\_login, \*\*button\_style)

### login\_button.pack(pady=10)

### window.mainloop()

### Admin\_Home.py

### import os # accessing the os functions

### import check\_camera

### import Capture\_Image

### import Recognize

### from tkinter import \*

### import tkinter as tk

### import threading

### # creating the title bar function

### # ---------------------------------------------------------

### # calling the camera test function from check camera.py file

### def cc\_call():

### tkStatus.set("Accessing Camera...")

### status\_label.update()

### check\_camera.camer()

### tkStatus.set("")

### status\_label.update()

### def checkCamera():

### t1=threading.Thread(target=cc\_call,daemon=True)

### t1.start()

### 

### def autom\_call():

### tkStatus.set("Sending Mail...")

### status\_label.update()

### os.system("py automail.py")

### tkStatus.set("Mail Sent")

### status\_label.update()

### def autom():

### t5=threading.Thread(target=autom\_call,daemon=True)

### t5.start()

### # --------------------------------------------------------------

### # calling the take image function form capture image.py file

### def cfaces\_call():

### tkStatus.set("Capturing Faces...")

### status\_label.update()

### try:

### Capture\_Image.takeImages(str(tkName.get()))

### tkStatus.set("Faces Captured")

### id\_label.update()

### tkName.set("")

### name\_label.update()

### except Exception as e:

### if (tkID.get().isnumeric()==False):

### tkStatus.set("Enter valid user id")

### status\_label.update()

### else:

### tkStatus.set("Enter valid user email")

### status\_label.update()

### def CaptureFaces():

### t2=threading.Thread(target=cfaces\_call,daemon=True)

### t2.start()

### # -----------------------------------------------------------------

### # calling the train images from train\_images.py file

### def timages\_call():

### tkStatus.set("Showing Attendance...")

### status\_label.update()

### os.system("streamlit run app.py")

### tkStatus.set("Attendance Displayed")

### status\_label.update()

### 

### def Trainimages():

### t3=threading.Thread(target=timages\_call,daemon=True)

### t3.start()

### 

### # --------------------------------------------------------------------

### # calling the recognize\_attendance from recognize.py file

### def rfaces\_call():

### tkStatus.set("Recognizing Faces...")

### status\_label.update()

### Recognize.recognize\_attendence()

### tkStatus.set("Faces Recognized.")

### status\_label.update()

### def RecognizeFaces():

### t4=threading.Thread(target=rfaces\_call,daemon=True)

### t4.start()

### # ---------------main driver ------------------

### # create a tkinter window

### root = Tk()

### root.title("Contactless Attendance System")

### tkID = tk.StringVar()

### tkName = tk.StringVar()

### tkEmail = tk.StringVar()

### tkStatus = tk.StringVar()

### 

### # Open window having dimension 100x100

### #root.geometry('100x100')

### screen\_width = root.winfo\_screenwidth()

### screen\_height = root.winfo\_screenheight()

### x\_coordinate = int((screen\_width / 2) - (400 / 2)) # Adjust the window width as needed

### y\_coordinate = int((screen\_height / 2) - (300 / 2)) # Adjust the window height as needed

### # Set the window position to the center of the screen

### root.geometry(f"400x400+{x\_coordinate}+{y\_coordinate}")

### # Create a Button

### btn1 = tk.Button(

### root,

### text='CHECK CAMERA',

### command=checkCamera,

### width=42,

### bg='#3498db',

### fg='#ffffff',

### bd=2,

### relief=tk.FLAT,

### activebackground = "Green",

### activeforeground = "White",

### )

### btn1.grid(

### padx=15,

### pady=8,

### ipadx=24,

### ipady=6,

### row=0,

### column=0,

### columnspan=4,

### sticky=tk.W + tk.E + tk.N + tk.S,

### )

### name\_label = tk.Label(root,

### text='Enter Name:', bg='#eeeeee',

### anchor=tk.W)

### name\_label.grid(

### padx=12,

### pady=(8, 0),

### ipadx=0,

### ipady=1,

### row=2,

### column=0,

### columnspan=1,

### sticky=tk.W + tk.E + tk.N + tk.S,

### )

### name\_entry = tk.Entry(root, textvariable=tkName,

### bg='#fff', exportselection=0,

### relief=tk.FLAT)

### name\_entry.grid(

### padx=15,

### pady=6,

### ipadx=8,

### ipady=8,

### row=2,

### column=1,

### columnspan=3,

### sticky=tk.W + tk.E + tk.N + tk.S,

### )

### btn2 = tk.Button(

### root,

### text='CAPTURE FACES',

### command=CaptureFaces,

### width=42,

### bg='#3498db',

### fg='#ffffff',

### bd=2,

### relief=tk.FLAT,

### activebackground = "Green",

### activeforeground = "White",

### )

### btn2.grid(

### padx=15,

### pady=8,

### ipadx=24,

### ipady=6,

### row=4,

### column=0,

### columnspan=4,

### sticky=tk.W + tk.E + tk.N + tk.S,

### )

### btn3 = tk.Button(

### root,

### text='SHOW ATTENDANCE',

### command=Trainimages,

### width=42,

### bg='#3498db',

### fg='#ffffff',

### bd=2,

### relief=tk.FLAT,

### activebackground = "Green",

### activeforeground = "White",

### )

### btn3.grid(

### padx=15,

### pady=8,

### ipadx=24,

### ipady=6,

### row=5,

### column=0,

### columnspan=4,

### sticky=tk.W + tk.E + tk.N + tk.S,

### )

### btn4 = tk.Button(

### root,

### text='RECOGNIZE FACES',

### command=RecognizeFaces,

### width=42,

### bg='#3498db',

### fg='#ffffff',

### bd=2,

### relief=tk.FLAT,

### activebackground = "Green",

### activeforeground = "White",

### )

### btn4.grid(

### padx=15,

### pady=8,

### ipadx=24,

### ipady=6,

### row=6,

### column=0,

### columnspan=4,

### sticky=tk.W + tk.E + tk.N + tk.S,

### )

### btn5 = tk.Button(

### root,

### text='AUTO MAIL',

### command=autom,

### width=42,

### bg='#3498db',

### fg='#ffffff',

### bd=2,

### relief=tk.FLAT,

### activebackground = "Green",

### activeforeground = "White",

### )

### btn5.grid(

### padx=15,

### pady=8,

### ipadx=24,

### ipady=6,

### row=7,

### column=0,

### columnspan=4,

### sticky=tk.W + tk.E + tk.N + tk.S,

### )

### btn6 = tk.Button(

### root,

### text='EXIT',

### command=root.destroy,

### width=42,

### bg='#3498db',

### fg='#ffffff',

### bd=2,

### relief=tk.FLAT,

### activebackground = "Green",

### activeforeground = "White",

### )

### btn6.grid(

### padx=15,

### pady=8,

### ipadx=24,

### ipady=6,

### row=8,

### column=0,

### columnspan=4,

### sticky=tk.W + tk.E + tk.N + tk.S,

### )

### status\_label = tk.Label(

### root,

### textvariable=tkStatus,

### bg='#eeeeee',

### anchor=tk.W,

### justify=tk.LEFT,

### relief=tk.FLAT,

### wraplength=350,

### )

### status\_label.grid(

### padx=12,

### pady=(0, 12),

### ipadx=0,

### ipady=1,

### row=9,

### column=0,

### columnspan=4,

### sticky=tk.W + tk.E + tk.N + tk.S,

### )

### 

### # Set the position of button on the top of window.

### 

### root.mainloop()

### #mainMenu()

### Staff\_home.py

### 

### import os # accessing the os functions

### import check\_camera

### import Recognize

### from tkinter import \*

### import tkinter as tk

### import threading

### def cc\_call():

### tkStatus.set("Accessing Camera...")

### status\_label.update()

### check\_camera.camer()

### tkStatus.set("")

### status\_label.update()

### def checkCamera():

### t1=threading.Thread(target=cc\_call,daemon=True)

### t1.start()

### 

### def autom\_call():

### tkStatus.set("Sending Mail...")

### status\_label.update()

### os.system("py automail.py")

### tkStatus.set("Mail Sent")

### status\_label.update()

### def autom():

### t5=threading.Thread(target=autom\_call,daemon=True)

### t5.start()

### def timages\_call():

### tkStatus.set("Showing Attendance...")

### status\_label.update()

### os.system("streamlit run app.py")

### tkStatus.set("Attendance Displayed")

### status\_label.update()

### 

### def Trainimages():

### t3=threading.Thread(target=timages\_call,daemon=True)

### t3.start()

### 

### def rfaces\_call():

### tkStatus.set("Recognizing Faces...")

### status\_label.update()

### Recognize.recognize\_attendence()

### tkStatus.set("Faces Recognized.")

### status\_label.update()

### def RecognizeFaces():

### t4=threading.Thread(target=rfaces\_call,daemon=True)

### t4.start()

### # ---------------main driver ------------------

### # create a tkinter window

### root = Tk()

### root.title("Contactless Attendance System")

### tkID = tk.StringVar()

### tkName = tk.StringVar()

### tkEmail = tk.StringVar()

### tkStatus = tk.StringVar()

### 

### # Open window having dimension 100x100

### #root.geometry('100x100')

### screen\_width = root.winfo\_screenwidth()

### screen\_height = root.winfo\_screenheight()

### x\_coordinate = int((screen\_width / 2) - (400 / 2)) # Adjust the window width as needed

### y\_coordinate = int((screen\_height / 2) - (300 / 2)) # Adjust the window height as needed

### # Set the window position to the center of the screen

### root.geometry(f"400x400+{x\_coordinate}+{y\_coordinate}")

### # Create a Button

### btn1 = tk.Button(

### root,

### text='CHECK CAMERA',

### command=checkCamera,

### width=42,

### bg='#3498db',

### fg='#ffffff',

### bd=2,

### relief=tk.FLAT,

### activebackground = "Green",

### activeforeground = "White",

### )

### btn1.grid(

### padx=15,

### pady=8,

### ipadx=24,

### ipady=6,

### row=0,

### column=0,

### columnspan=4,

### sticky=tk.W + tk.E + tk.N + tk.S,

### )

### btn3 = tk.Button(

### root,

### text='SHOW ATTENDANCE',

### command=Trainimages,

### width=42,

### bg='#3498db',

### fg='#ffffff',

### bd=2,

### relief=tk.FLAT,

### activebackground = "Green",

### activeforeground = "White",

### )

### btn3.grid(

### padx=15,

### pady=8,

### ipadx=24,

### ipady=6,

### row=5,

### column=0,

### columnspan=4,

### sticky=tk.W + tk.E + tk.N + tk.S,

### )

### btn4 = tk.Button(

### root,

### text='RECOGNIZE FACES',

### command=RecognizeFaces,

### width=42,

### bg='#3498db',

### fg='#ffffff',

### bd=2,

### relief=tk.FLAT,

### activebackground = "Green",

### activeforeground = "White",

### )

### btn4.grid(

### padx=15,

### pady=8,

### ipadx=24,

### ipady=6,

### row=6,

### column=0,

### columnspan=4,

### sticky=tk.W + tk.E + tk.N + tk.S,

### )

### btn5 = tk.Button(

### root,

### text='AUTO MAIL',

### command=autom,

### width=42,

### bg='#3498db',

### fg='#ffffff',

### bd=2,

### relief=tk.FLAT,

### activebackground = "Green",

### activeforeground = "White",

### )

### btn5.grid(

### padx=15,

### pady=8,

### ipadx=24,

### ipady=6,

### row=7,

### column=0,

### columnspan=4,

### sticky=tk.W + tk.E + tk.N + tk.S,

### )

### btn6 = tk.Button(

### root,

### text='EXIT',

### command=root.destroy,

### width=42,

### bg='#3498db',

### fg='#ffffff',

### bd=2,

### relief=tk.FLAT,

### activebackground = "Green",

### activeforeground = "White",

### )

### btn6.grid(

### padx=15,

### pady=8,

### ipadx=24,

### ipady=6,

### row=8,

### column=0,

### columnspan=4,

### sticky=tk.W + tk.E + tk.N + tk.S,

### )

### status\_label = tk.Label(

### root,

### textvariable=tkStatus,

### bg='#eeeeee',

### anchor=tk.W,

### justify=tk.LEFT,

### relief=tk.FLAT,

### wraplength=350,

### )

### status\_label.grid(

### padx=12,

### pady=(0, 12),

### ipadx=0,

### ipady=1,

### row=9,

### column=0,

### columnspan=4,

### sticky=tk.W + tk.E + tk.N + tk.S,

### )

### 

### # Set the position of button on the top of window.

### 

### root.mainloop()

### #mainMenu()

### Stu\_home.py

### import os # accessing the os functions

### import check\_camera

### import Capture\_Image

### import Recognize

### from tkinter import \*

### import tkinter as tk

### import threading

### def rfaces\_call():

### tkStatus.set("Recognizing Faces...")

### status\_label.update()

### Recognize.recognize\_attendence()

### tkStatus.set("Faces Recognized.")

### status\_label.update()

### def RecognizeFaces():

### t4=threading.Thread(target=rfaces\_call,daemon=True)

### t4.start()

### root = Tk()

### root.title("Contactless Attendance System")

### tkID = tk.StringVar()

### tkName = tk.StringVar()

### tkEmail = tk.StringVar()

### tkStatus = tk.StringVar()

### 

### screen\_width = root.winfo\_screenwidth()

### screen\_height = root.winfo\_screenheight()

### x\_coordinate = int((screen\_width / 2) - (400 / 2)) # Adjust the window width as needed

### y\_coordinate = int((screen\_height / 2) - (300 / 2)) # Adjust the window height as needed

### # Set the window position to the center of the screen

### root.geometry(f"400x400+{x\_coordinate}+{y\_coordinate}")

### btn4 = tk.Button(

### root,

### text='RECOGNIZE FACES',

### command=RecognizeFaces,

### width=42,

### bg='#3498db',

### fg='#ffffff',

### bd=2,

### relief=tk.FLAT,

### activebackground = "Green",

### activeforeground = "White",

### )

### btn4.grid(

### padx=15,

### pady=8,

### ipadx=24,

### ipady=6,

### row=6,

### column=0,

### columnspan=4,

### sticky=tk.W + tk.E + tk.N + tk.S,

### )

### btn6 = tk.Button(

### root,

### text='EXIT',

### command=root.destroy,

### width=42,

### bg='#3498db',

### fg='#ffffff',

### bd=2,

### relief=tk.FLAT,

### activebackground = "Green",

### activeforeground = "White",

### )

### btn6.grid(

### padx=15,

### pady=8,

### ipadx=24,

### ipady=6,

### row=8,

### column=0,

### columnspan=4,

### sticky=tk.W + tk.E + tk.N + tk.S,

### )

### status\_label = tk.Label(

### root,

### textvariable=tkStatus,

### bg='#eeeeee',

### anchor=tk.W,

### justify=tk.LEFT,

### relief=tk.FLAT,

### wraplength=350,

### )

### status\_label.grid(

### padx=12,

### pady=(0, 12),

### ipadx=0,

### ipady=1,

### row=9,

### column=0,

### columnspan=4,

### sticky=tk.W + tk.E + tk.N + tk.S,

### )

### 

### 

### 

### root.mainloop()

### Capture\_image.py

### import cv2

### import pickle

### import numpy as np

### import os

### def takeImages(name):

### video=cv2.VideoCapture(1)

### facedetect=cv2.CascadeClassifier('data/haarcascade\_frontalface\_default.xml')

### faces\_data=[]

### i=0

### name=name

### while True:

### ret,frame=video.read()

### gray=cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)

### faces=facedetect.detectMultiScale(gray, 1.3 ,5)

### for (x,y,w,h) in faces:

### crop\_img=frame[y:y+h, x:x+w, :]

### resized\_img=cv2.resize(crop\_img, (50,50))

### if len(faces\_data)<=100 and i%10==0:

### faces\_data.append(resized\_img)

### i=i+1

### cv2.putText(frame, str(len(faces\_data)), (50,50), cv2.FONT\_HERSHEY\_COMPLEX, 1, (50,50,255), 1)

### cv2.rectangle(frame, (x,y), (x+w, y+h), (50,50,255), 1)

### cv2.imshow("Frame",frame)

### k=cv2.waitKey(1)

### if k==ord('q') or len(faces\_data)==100:

### break

### video.release()

### cv2.destroyAllWindows()

### faces\_data=np.asarray(faces\_data)

### faces\_data=faces\_data.reshape(100, -1)

### if 'names.pkl' not in os.listdir('data/'):

### names=[name]\*100

### with open('data/names.pkl', 'wb') as f:

### pickle.dump(names, f)

### else:

### with open('data/names.pkl', 'rb') as f:

### names=pickle.load(f)

### names=names+[name]\*100

### with open('data/names.pkl', 'wb') as f:

### pickle.dump(names, f)

### if 'faces\_data.pkl' not in os.listdir('data/'):

### with open('data/faces\_data.pkl', 'wb') as f:

### pickle.dump(faces\_data, f)

### else:

### with open('data/faces\_data.pkl', 'rb') as f:

### faces=pickle.load(f)

### faces=np.append(faces, faces\_data, axis=0)

### with open('data/faces\_data.pkl', 'wb') as f:

### pickle.dump(faces, f)

### Check\_camera.py

### def camer():

### import cv2

### # Load the cascade

### face\_cascade = cv2.CascadeClassifier('data/haarcascade\_frontalface\_default.xml')

### # To capture video from webcam.

### cap = cv2.VideoCapture(1)

### while True:

### # Read the frame

### \_, img = cap.read()

### # Convert to grayscale

### gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

### # Detect the faces

### faces = face\_cascade.detectMultiScale(gray, 1.3, 5, minSize=(30, 30),flags = cv2.CASCADE\_SCALE\_IMAGE)

### # Draw the rectangle around each face

### for (x, y, w, h) in faces:

### cv2.rectangle(img, (x, y), (x + w, y + h), (10,159,255), 2)

### # Display

### cv2.imshow('Webcam Check', img)

### # Stop if escape key is pressed

### if cv2.waitKey(1) & 0xFF == ord('q'):

### break

### # Release the VideoCapture object

### cap.release()

### cv2.destroyAllWindows()

### Recognize.py

### from sklearn.neighbors import KNeighborsClassifier

### import cv2

### import pickle

### import numpy as np

### import os

### import csv

### import time

### from datetime import datetime

### from win32com.client import Dispatch

### def speak(str1):

### speak=Dispatch(("SAPI.SpVoice"))

### speak.Speak(str1)

### def recognize\_attendence():

### video=cv2.VideoCapture(1)

### facedetect=cv2.CascadeClassifier('data/haarcascade\_frontalface\_default.xml')

### with open('data/names.pkl', 'rb') as w:

### LABELS=pickle.load(w)

### with open('data/faces\_data.pkl', 'rb') as f:

### FACES=pickle.load(f)

### print('Shape of Faces matrix --> ', FACES.shape)

### knn=KNeighborsClassifier(n\_neighbors=5)

### knn.fit(FACES, LABELS)

### imgBackground=cv2.imread("background.png")

### COL\_NAMES = ['NAME', 'TIME']

### while True:

### ret,frame=video.read()

### gray=cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)

### faces=facedetect.detectMultiScale(gray, 1.3 ,5)

### for (x,y,w,h) in faces:

### crop\_img=frame[y:y+h, x:x+w, :]

### resized\_img=cv2.resize(crop\_img, (50,50)).flatten().reshape(1,-1)

### output=knn.predict(resized\_img)

### ts=time.time()

### date=datetime.fromtimestamp(ts).strftime("%d-%m-%Y")

### timestamp=datetime.fromtimestamp(ts).strftime("%H:%M-%S")

### exist=os.path.isfile("Attendance/Attendance\_" + date + ".csv")

### cv2.rectangle(frame, (x,y), (x+w, y+h), (0,0,255), 1)

### cv2.rectangle(frame,(x,y),(x+w,y+h),(50,50,255),2)

### cv2.rectangle(frame,(x,y-40),(x+w,y),(50,50,255),-1)

### cv2.putText(frame, str(output[0]), (x,y-15), cv2.FONT\_HERSHEY\_COMPLEX, 1, (255,255,255), 1)

### cv2.rectangle(frame, (x,y), (x+w, y+h), (50,50,255), 1)

### attendance=[str(output[0]), str(timestamp)]

### imgBackground[162:162 + 480, 55:55 + 640] = frame

### cv2.imshow("Frame",imgBackground)

### k=cv2.waitKey(1)

### if k==ord('o'):

### speak("Attendance Taken..")

### time.sleep(5)

### if exist:

### with open("Attendance/Attendance\_" + date + ".csv", "+a") as csvfile:

### writer=csv.writer(csvfile)

### writer.writerow(attendance)

### csvfile.close()

### else:

### with open("Attendance/Attendance\_" + date + ".csv", "+a") as csvfile:

### writer=csv.writer(csvfile)

### writer.writerow(COL\_NAMES)

### writer.writerow(attendance)

### csvfile.close()

### if k==ord('q'):

### break

### video.release()

### cv2.destroyAllWindows()

### Automail.py

### import yagmail

### import os

### import datetime

### import Info

### import pandas as pd

### import numpy as np

### date = datetime.date.today().strftime("%B %d, %Y")

### path = 'Attendance'

### os.chdir(path)

### files = sorted(os.listdir(os.getcwd()), key=os.path.getmtime)

### df = pd.read\_csv(r'EmployeeDetails\EmployeeDetails.csv')

### 

### receivers = df["el3ktrz06@gmail.com"]

### newest = files[-1]

### filename = newest

### sub = "Attendance Report for " + str(date)

### body = " Attendance Submitted."

### for receiver in receivers:

### # mail information

### if pd.isnull(reciver):

### continue

### else:

### yag = yagmail.SMTP(Info.EMAIL\_ID, Info.PASSWORD)

### # sent the mail

### yag.send(

### to=receiver,

### subject=sub, # email subject

### contents=body, # email body

### attachments= filename # file attached

### )

### print("Email Sent to "+reciver)

### app.py

### import streamlit as st

### import pandas as pd

### import time

### from datetime import datetime

### ts=time.time()

### date=datetime.fromtimestamp(ts).strftime("%d-%m-%Y")

### timestamp=datetime.fromtimestamp(ts).strftime("%H:%M-%S")

### from streamlit\_autorefresh import st\_autorefresh

### count = st\_autorefresh(interval=2000, limit=100, key="fizzbuzzcounter")

### if count == 0:

### st.write("Count is zero")

### elif count % 3 == 0 and count % 5 == 0:

### st.write("FizzBuzz")

### elif count % 3 == 0:

### st.write("Fizz")

### elif count % 5 == 0:

### st.write("Buzz")

### else:

### st.write(f"Count: {count}")

### df=pd.read\_csv("Attendance/Attendance\_" + date + ".csv")

### st.dataframe(df.style.highlight\_max(axis=0))

### Info.py

### EMAIL\_ID = 'smtp.freesmtpservers.com'

### PASSWORD = ''

# TESTING

## Chapter 7 TESTING

* 1. **Introduction**

Software testing is an investigation conducted to provide stack holders with information about the quality of the product or service under test. Testing has been defined as the process of analysing a software item to detect the differences between existing and required conditions and to evaluate the features of the software item. Software testing is the process used to assess the quality of computer software.

It involves operation of a system or application under controlled conditions and evaluating the results. The controlled conditions should include both normal and abnormal conditions. Testing should intentionally attempt to make things go wrong to determine if things happen when they should. It is oriented to ‘detection’.

Software testing has three main purposes:

* The verification process confirms that the software meet its technical specifications. A “Specification” is a description of a function in terms of a

measurable output value given a specific input value under specific preconditions.

* The validation process confirms that the software meets the business requirements.
* A defect is a variance between the expected and actual result. The defect’s ultimate source may be traced to a fault introduced in the specification, design, or development phases. Not all the defects will necessarily result in failures.

There are two types of software testing:

* Black box testing-internal system design is not considered in this type of testing. Tests are based on requirements and functionality.
* White box testing-this testing is based on knowledge of thaw internal logic of an application’s code. Also known as glass box testing.

Internal software and code working should be known for this type of testing. Tests are based on coverage of code statements, branches, paths and conditions. A test case is a software testing document, which consists of event, action, input, output, expected result and actual result. Clinically defined a test case is an input and an expected result. This can be pragmatic as ‘for condition x your derived

result is y’; Where as other test cases described in more detail the input scenario and what results might be expected. It can occasionally be a series of steps but one with expected results or expected outcome. A test case should also contain a place for the actual result.

White box testing is applicable at the unit, integration and system levels of the software testing process.

## Testing Objectives

* Finding defects which may get created by the programmer while developing the software
* Gaining confidence in and providing information about the level of quality
* To prevent defects
* To make sure that the end results meets the business and user requirement.
* To ensure that it satisfies the BRS that is Business Requirement Specification and SRS that is System Requirement Specification

## Testing Methods

System testing is the stage of implementation. This is to check whether system works accurately and efficiently before live operation commences. Testing is vital to the success of the system. The candidate system is subject to a variety of tests: online response ,volume, stress, recovery, security and usability tests. A series of tests are performed for the proposed system is ready for user acceptance testing.

## Testing Steps

* + - Unit testing
    - Integration testing
    - Validation testing
    - Output testing
    - User acceptance testing

### Unit Testing

Unit testing focuses efforts on the smallest unit of software design. This is known as module testing. The modules are tested separately. The test is carried out during programming stage itself. In this step, each module is found to be working satisfactory as regards to the expected output from the module.

### Integration Testing

Data can be lost across an interface. One module can have an adverse effect on another , sub functions, when combined, may not be linked in desired manner in major functions. Integration testing is a systematic approach for constructing the program structure, while at the same time conducting test to uncover errors associated within the interface. The objectives is to take modules and builds program structure. All the modules are combined and tested as a whole.

### Validation

At the culmination of the integration testing, software is completely assembled as a package. Interfacing errors have been uncovered and corrected and a final series of software test begin in validation testing. Validation testing can be defined in many ways, but a simple definition is that the validation succeeds when the software functions in a manner that is expected by the customer. After validation test has been conducted, one of the three possible conditions exists.

* The function or performance characteristics confirm to specification and are accepted.
* A deviation from specification is uncovered and a deficiency lists is created.
* Proposed system under consideration has been tested by using validation test and found to be working satisfactory.

### Output testing

After performing the validation testing, the next step is output testing of the proposed system, since no system could be useful if it does not produce the required output in a specific format. The output format on the screen is found to be correct. The format was designed in the system design time according to the user

needs. For the hard copy also; the output comes as per the specified requirements by the user. Hence output testing did not result in any correction for the system.

### User acceptance testing

User acceptance of a system is the key factor for the success of any system. The system under consideration is tested for the user acceptance by constantly keeping in touch with the prospective system users at the time of developing and making changes whenever required.

This is done in regard to the following point:

* Input screen design
* Output screen design
* Online message should be guide to the user
* Format of reports and other output

### Login Forms

|  |  |  |  |
| --- | --- | --- | --- |
| **SI**  **No.** | **Test Condition** | **Expected Result** | **Result** |
| **01** | **If User clicks on Sign in button without entering username and password.** | **Please fill out this field** | **Successful** |
| **02** | **If user id or password is blank .** | **Please fill out this field** | **Successful** |

* + 1. **Student Face capture form**

|  |  |  |  |
| --- | --- | --- | --- |
| **SI**  **No.** | **Test Condition** | **Expected Result** | **Result** |
| **01** | **If the name is empty** | **Please fill out this field** | **Successful** |

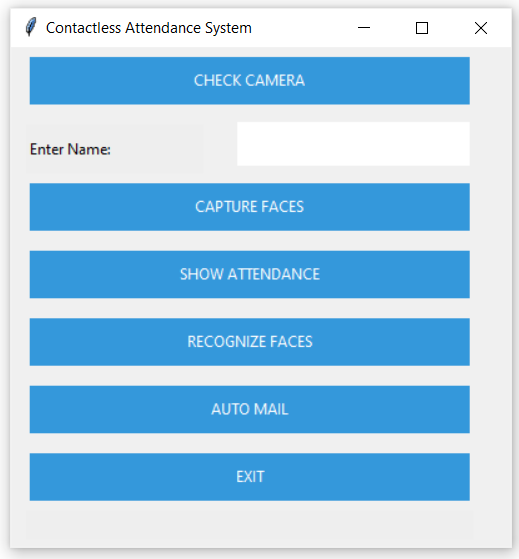
# USER INTERFACE

## Chapter 8 USER INTERFACE

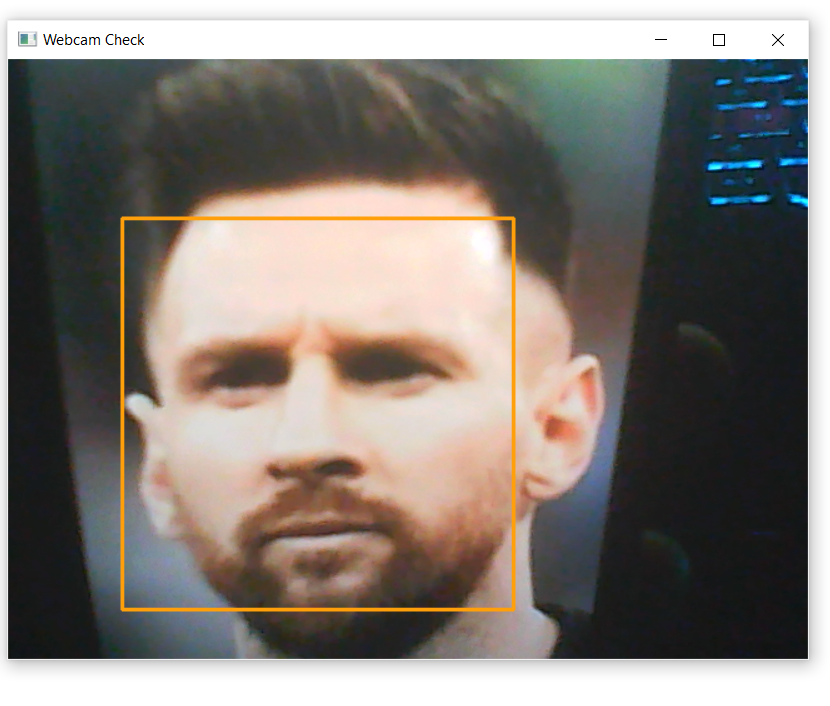
## Home page

## 

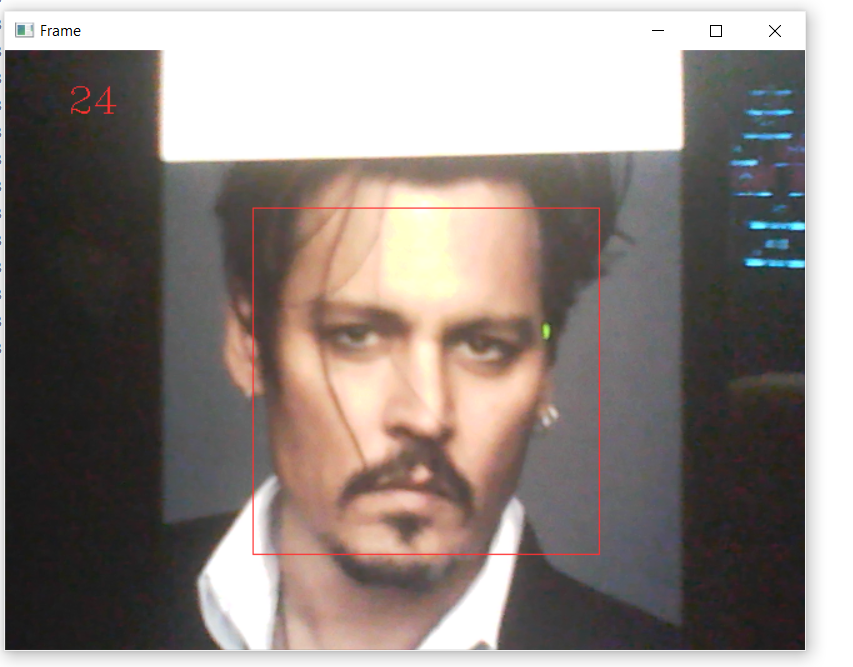
**Admin home**



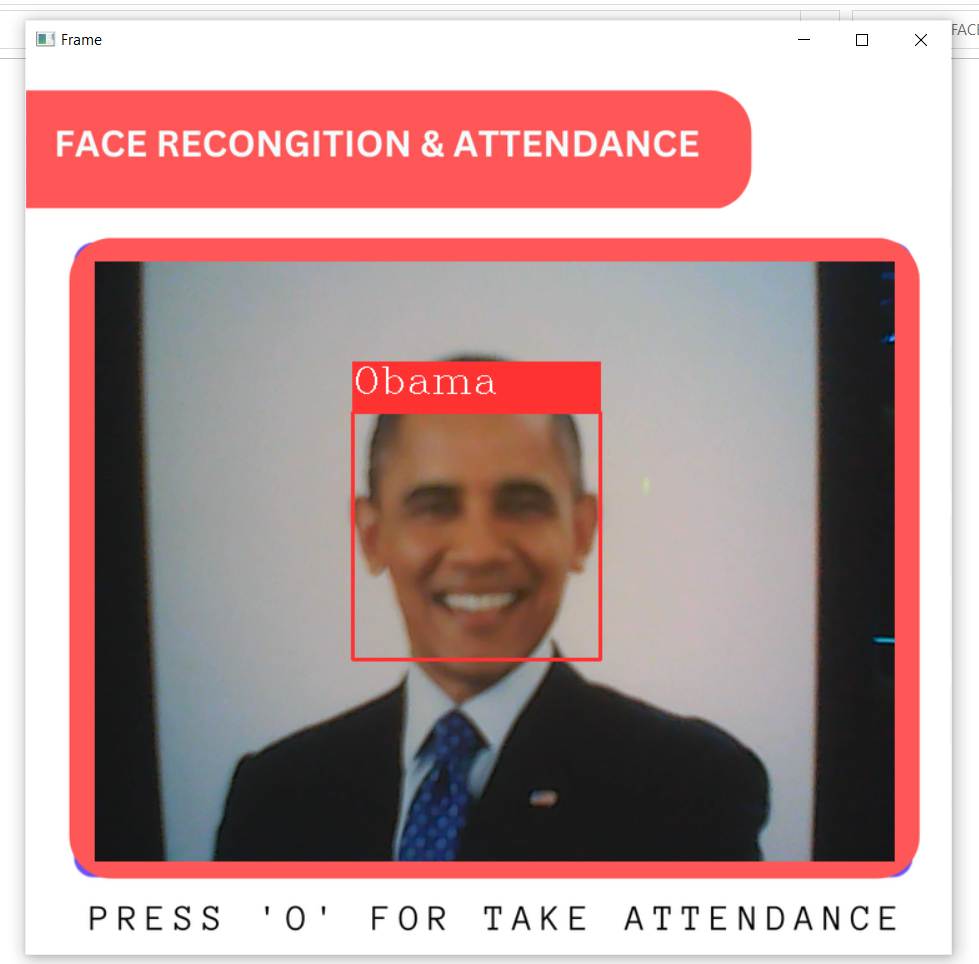
**Check camera**

****

**Capture faces**

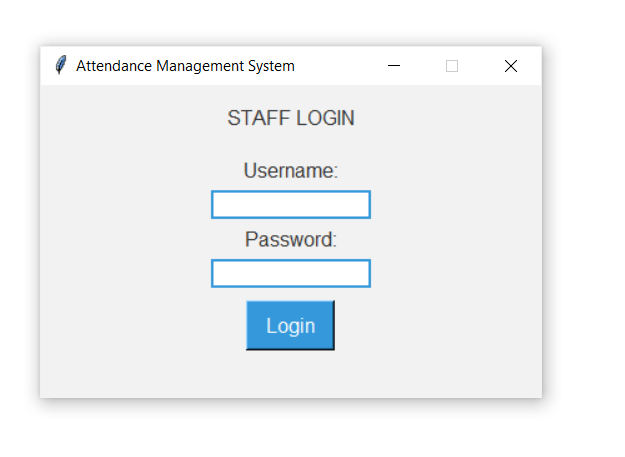
****

**Recognize faces**

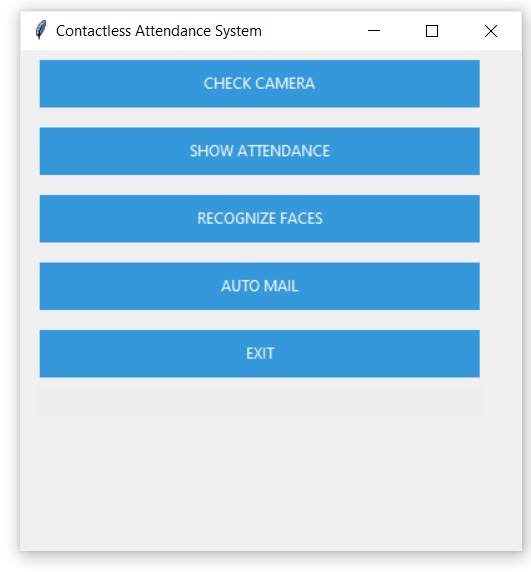
****

**View Attendance**

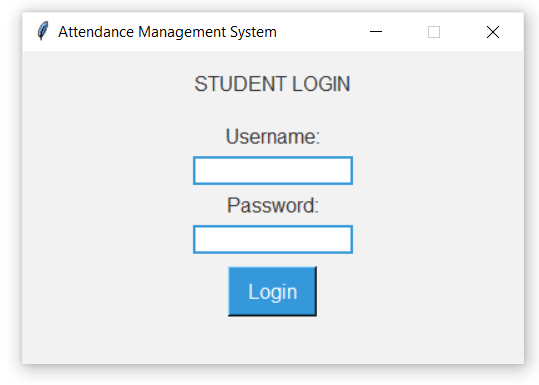
**Staff login**

****

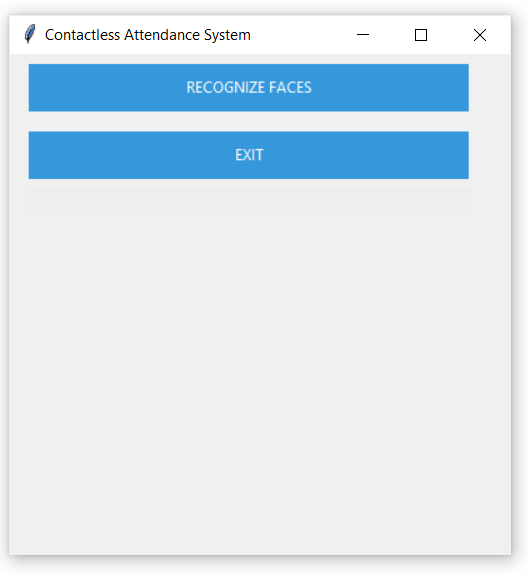
**Staff home**

****

**Student login**

****

**Student home**

****