A Major Project Report

on

AUTOMATED FACE RECOGNITION ATTENDANCE SYSTEM USING DEEP LEARNING

Submitted in partial fulfilment of the requirements for the award of the degree

Of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

By

GADE ROHITH REDDY	**20EG105113**
MAHANKALI LIKHITH	**20EG105129**
MANDA MANOJ KUMAR	**20EG105131**
MEDARAMETLA LAKSHMI KARTHIKEYA	**20EG105133**
METTU HEMANTH	**20EG105701**

Under the guidance of

Dr. G. PRABHAKAR RAJU

Assistant Professor

Department of CSE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Venkatapur(V), Ghatkeshar(M), Medchal(D) – 500088

2023-2024

ANURAG UNIVERSITY | SCHOOL OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the report / dissertation entitled “**AUTOMATED FACE RECOGNITION ATTANDANCE SYSTEM USING DEEP LEARNING”** that is being submitted by **GADE ROHITH REDDY [20EG105113], MAHANKALI LIKHITH [20EG105129], MANDA MANOJ KUMAR [20EG105131], MEDARAMETLA LAKSHMI KARTHIKEYA [20EG105133], METTU HEMANTH [20EG105701]** in partial fulfillment for the award of Bachelor of Technology in Computer Science and Engineering to the Anurag University, Hyderabad is a record of bonafide work carried out by them under my guidance and supervision. The results embodied in this project report have not been submitted to any other University or Institute for the award of any Degree or Diploma.

Internal Guide
Dr. G. Prabhakar Raju
Assistant Professor, Dept. of CSE

Dr. G. Vishnu Murthy
Professor &Dean, Dept. of CSE

External Examiner

ACKNOWLEDGEMENT

We would like to express our sincere thanks and deep sense of gratitude to project supervisor **Dr. G. Prabhakar Raju** for his constant encouragement and inspiring guidance without which this project could not have been completed. His critical reviews and constructive comments improved our grasp of the subject and steered to the fruitful completion of the work. His patience, guidance and encouragement made this project possible.

We would like to acknowledge our sincere gratitude for the support extended by **Dr. G. Vishnu Murthy**, Dean, Dept. of CSE, Anurag University. We also express our deep sense of gratitude to **Dr. V V S S S Balaram,** Academic coordinator, **Dr. Pallam Ravi**, Project in-Charge, **Dr. G. Prabhakar Raju** Project Coordinator and Project review committee members, whose research expertise and commitment to the highest standards continuously motivated us during the crucial stage our project work.

We would like express our special thanks to **Dr. V. Vijaya Kumar**, Dean School of Engineering, Anurag University, for his encouragement and timely support in our B. Tech program.

G ROHITH REDDY
(20EG105113)

M LIKHITH
(20EG105129)

M MANOJ KUMAR
(20EG105131)

M L KARTHIKEYA
(20EG105133)

M HEMANTH
(20EG105701)

DECLARATION

We, hereby declare that the Report entitled **"AUTOMATED FACE RECOGNITION ATTANDANCE SYSTEM USING DEEP LEARNING"** submitted for the award of Bachelor of technology Degree is our original work and the Report has not formed the basis for the award of anydegree, diploma, associate ship or fellowship of similar other titles. It has not been submitted to any other University or Institution for the award of any degree or diploma.

G ROHITH REDDY
(20EG105113)

M LIKHITH
(20EG105129)

M MANOJ KUMAR
(20EG105131)

M L KARTHIKEYA
(20EG105133)

M HEMANTH
(20EG105701)

PLACE: HYDERABAD
DATE:

ABSTRACT

In the pursuit of streamlining attendance management processes, this project introduces an Automated Face Recognition Attendance System leveraging cutting-edge deep learning techniques. The system, developed using Python, OpenCV, and incorporating the Face_recognition and dlib libraries, harnesses the capabilities of Convolutional Neural Network (CNN) and K-Nearest Neighbors (KNN) algorithms.

Operationalizing the system begins with capturing facial images via a camera, followed by preprocessing steps to enhance image quality and feature extraction. The CNN model then performs intricate analysis and classification of facial features, while the KNN algorithm refines recognition accuracy. This amalgamation of technologies enables real-time attendance tracking, eradicating manual data entry and mitigating errors.

The project endeavors to revolutionize attendance management paradigms, offering an intuitive, efficient solution that optimizes resource allocation and enhances productivity. Experimental evaluations showcase the system's efficacy in terms of accuracy, speed, and scalability, paving the way for widespread adoption across diverse educational and organizational domains.

TABLE OF CONTENT

LIST OF IMAGES

1. INTRODUCTION

Attendance management systems play a pivotal role in various sectors, including education and corporate environments, where tracking attendance is essential for monitoring student or employee participation and ensuring compliance with organizational requirements. Traditional methods of attendance tracking, such as manual data entry or barcode scanning, are labor-intensive, prone to errors, and often lack real-time monitoring capabilities. To address these challenges and enhance efficiency, automated face recognition systems have emerged as a promising solution.

The integration of deep learning techniques, particularly Convolutional Neural Networks (CNNs) and K-Nearest Neighbors (KNN) algorithms, has significantly advanced the accuracy and performance of face recognition systems. Leveraging the capabilities of Python programming language and libraries such as OpenCV, Face_recognition, and dlib, researchers and developers have been able to create sophisticated automated face recognition attendance systems.

This research paper explores the development and implementation of an Automated Face Recognition Attendance System using deep learning methodologies. By harnessing the power of deep learning algorithms, the system aims to automate the attendance tracking process, eliminate manual data entry, and enhance accuracy and reliability. The integration of CNN and KNN algorithms enables the system to effectively identify individuals in real-time, even in challenging environmental conditions.

The significance of this research lies in its potential to revolutionize traditional attendance management practices, offering a seamless and efficient solution that optimizes time utilization and resource allocation. Furthermore, the scalability and versatility of the proposed system make it applicable across various educational and organizational settings.

Through empirical evaluations and case studies, this research paper aims to demonstrate the effectiveness and practicality of the Automated Face Recognition Attendance System, providing insights into its performance metrics, scalability, and potential challenges. Ultimately, the findings of this research contribute to the advancement of attendance management technology and pave the way for future innovations in this field.

2. LITERATURE REVIEW

Several research papers have contributed valuable insights into the development of face recognition attendance systems, each offering unique methodologies and perspectives to address the challenges inherent in traditional attendance tracking methods.

The paper titled "Face Recognition Attendance System Based on Real-Time Video Processing", [1] aims to design a face recognition attendance system based on real-time video processing. The study meticulously examines four critical aspects: the accuracy rate of the face recognition system during check-ins, system stability, truancy rates, and interface settings. By analyzing these factors, the researchers propose a face recognition attendance system aimed at significantly improving attendance tracking in educational institutions. Notably, the decision to utilize Convolutional Neural Networks (CNN) and K-Nearest Neighbors (KNN) algorithms over other deep learning methodologies such as Support Vector Machine (SVM) is pivotal, given their capacity for accuracy and reliability. Furthermore, the integration of libraries like Face_recognition and dlib enhances the efficiency of the CNN algorithm, underscoring its suitability for face recognition tasks.

In contrast, the paper "Outsourcing LDA-Based Face Recognition to an Untrusted Cloud", [2] designs a protocol of outsourcing LDA-based face recognition to an untrusted cloud, which can help the client to complete the operations of matrix inversion (MI), matrix multiplication (MM) and eigenvalue decomposition (ED) simultaneously. The protocol emphasizes data privacy, verifiability, and computational efficiency, crucial aspects in the development of robust face recognition systems. Moreover, the protocol's efficiency is enhanced by reducing the computational overhead for the client, aligning with the principles of accuracy and efficiency in face recognition tasks.

Lastly, the paper titled "A Real-Time CNN-Based Lightweight Mobile Masked Face Recognition System",[3] The main objective of this study is to identify individuals who do not use masks or use them incorrectly and to verify their identity by building a masked face dataset. Leveraging a CNN-based architecture and transfer learning techniques, the study offers insights into effectively identifying individuals even when wearing masks. Given the importance of accuracy and reliability in attendance systems, the decision to utilize CNN is justified, particularly in scenarios where individuals may wear masks or face coverings.

Through this literature survey, it is evident that the adoption of CNN and KNN algorithms, along with the utilization of libraries such as Face_recognition and dlib, is crucial in the development of robust and reliable face recognition attendance systems. These methodologies not only enhance accuracy and reliability but also contribute to the advancement of attendance management technology.

3. METHODOLOGY

The attendance management system was developed using Python language, leveraging various libraries and algorithms. The primary tools used include face_recognition and dlib for deep learning-based face detection, KNN algorithm for face recognition, OpenCV for image capturing, PyQt5 for the graphical user interface, openpyxl for Excel file management, and MySQL. Connector for MySQL database connectivity.

The development process began with the creation of the main.py file, which serves as the entry point for the application. This file defines the main user interface using PyQt5, providing options for administrators and faculty members to log in and access the system functionalities.

Upon logging in, the system allows administrators to register new faculty members using the register.py file. This file validates user inputs, such as name, username, password, email, and mobile number, and stores the information in a MySQL database using the DBConnection module.

After registering faculty members, administrators can view and download attendance reports using the reports.py file. This file retrieves attendance data from the MySQL database based on the selected month and saves it as an Excel file for further analysis.

Faculty members can log in to mark attendance for students using the faculty.py file. This file utilizes the predict_KNN.py module to recognize students' faces captured by the webcam using OpenCV. The KNN algorithm identifies students based on their facial features and updates the attendance records accordingly.

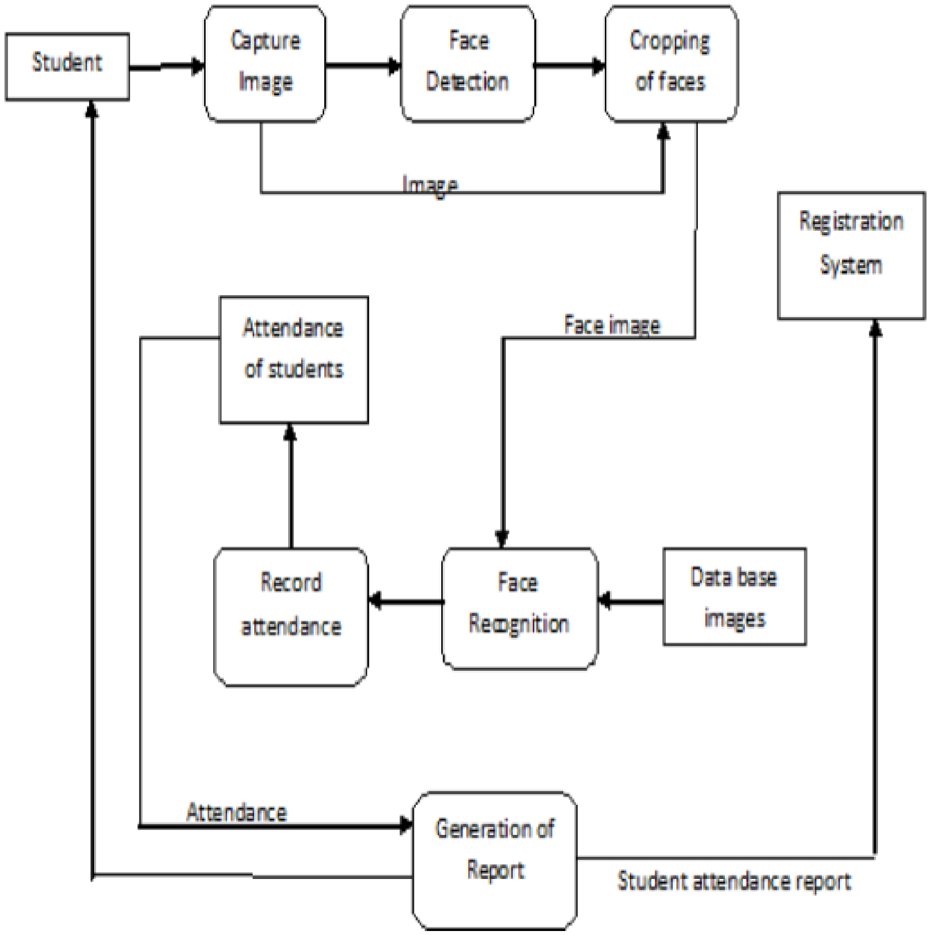
To view today's attendance, faculty members can access the TodayAttendance.py file, which retrieves the attendance data for the current date from the MySQL database and displays it in a table format using PyQt5.

Furthermore, administrators can view student details using the viewStudents.py file, which retrieves student information, including roll numbers, names, and pictures, from the MySQL database and presents it in a table format.

Throughout the development process, rigorous testing was conducted to ensure the system's functionality, accuracy, and reliability. Unit tests were performed on individual modules, while integration tests verified the interactions between different components. Additionally, user acceptance testing was conducted to gather feedback and make necessary improvements to enhance the user experience.

In conclusion, the developed attendance management system demonstrates the effective utilization of Python-based tools and algorithms for automating attendance tracking processes in educational institutions and organizations. The system offers convenience, accuracy, and efficiency, streamlining the attendance management workflow and improving overall productivity.

The picture of proposed architecture is as follows:



3.1 Proposed Architecture

4. DESIGN

4.1 Introduction

Unified Modeling Language (UML) is a standardized visual modeling language used in software engineering to design and document software systems. It provides a common notation that is understood by software developers, business analysts, and other stakeholders involved in the software development process.

UML offers a set of diagrams to represent various aspects of a system, including its structure, behavior, and interactions. Some of the key aspects of UML include:

Standardization: UML is maintained by the Object Management Group (OMG), and its specifications are regularly updated to reflect advancements in software engineering practices.

Visual Representation: UML diagrams use graphical symbols and notations to represent different elements and relationships within a system, making it easier to understand complex systems.

Abstraction: UML allows developers to abstract away unnecessary details and focus on the essential aspects of a system, facilitating better communication and understanding among stakeholders.

Modeling at Different Levels of Abstraction: UML supports modeling at various levels of abstraction, from high-level conceptual models to detailed design and implementation models.

Tool Support: There are numerous software tools available that support UML, allowing developers to create, edit, and analyze UML diagrams efficiently.

Language Independence: UML is not tied to any specific programming language, development methodology, or tool, making it versatile and widely applicable across various domains and industries.

Overall, UML serves as a powerful tool for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system, helping stakeholders to better understand and manage software projects throughout their lifecycle.

UML offers a set of diagrams to represent various aspects of a system, including its structure, behavior, and interactions. Some of the key diagrams in UML include:

1. Use Case Diagram

2. Sequence Diagram

3. Class Diagram

4. Component Diagram

5. Deployment Diagram

4.2 Use Case Diagram

A use case diagram is one of the most widely used diagrams in the Unified Modeling Language (UML). It provides a graphical representation of the functional requirements of a system from the perspective of its users. Here's a description of the key aspects of a use case diagram:

Actors: Actors represent entities that interact with the system. They could be human users, external systems, or other software components. Actors are typically represented as stick figures or simple shapes outside the system boundary.

Use Cases: Use cases represent the specific functionalities or tasks that the system provides to its users. Each use case describes a set of interactions between the system and its actors to achieve a particular goal. Use cases are represented as ovals inside the system boundary and are labeled with descriptive names.

Relationships: Relationships between actors and use cases illustrate the interactions between them. The primary relationship in a use case diagram is the association between actors and the use cases they are involved in. This association is depicted by solid lines connecting actors to use cases.

System Boundary: The system boundary defines the scope of the system being modeled. It separates the internal components of the system from its external actors. Use cases are placed inside the system boundary, while actors reside outside.

Generalization: Generalization relationships can be used to represent inheritance between use cases, where one use case inherits behavior from another more generalized use case. This relationship is depicted using an arrow with a triangle.

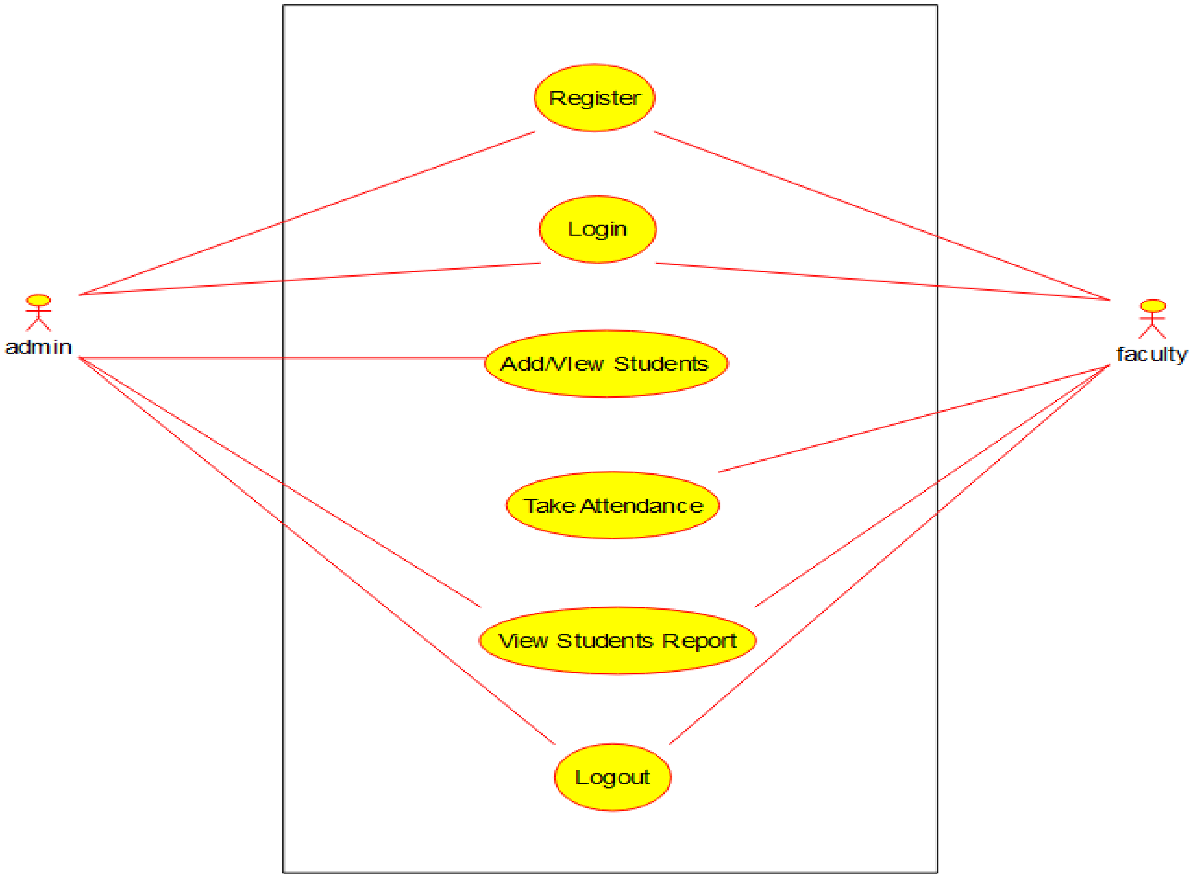
Use case diagrams serve several purposes in software development:

They provide a high-level overview of the system's functionalities and its interactions with users and external systems.

They help stakeholders understand the system's behavior and requirements from a user's perspective.

They serve as a foundation for detailed requirement analysis and system design.

Overall, use case diagrams are valuable tools for capturing and communicating the functional requirements of a system in a clear and understandable manner.



4.2.1 Use Case Diagram

4.3 Sequence Diagram

A sequence diagram is a type of interaction diagram in Unified Modeling Language (UML) that illustrates the interactions between objects or components within a system in a chronological sequence. It primarily focuses on the time ordering of messages exchanged between these objects or components during a particular scenario or use case execution. Here's a detailed description of the key aspects of a sequence diagram:

Objects/Participants: Objects or participants represent the entities (such as classes, instances, or components) that participate in the interaction. Each object is represented by a rectangular box or lifeline, positioned vertically along the diagram's left-hand side. Objects are labeled with their names or identifiers.

Lifelines: Lifelines are vertical lines drawn beneath each object, representing the object's existence over a period of time during the interaction. They indicate the lifespan of the object within the scenario being modeled.

Messages: Messages represent communication or interaction between objects in the system. They are depicted as horizontal arrows between lifelines, indicating the flow of control or data from one object to another. Messages can be synchronous (denoted by solid arrows), asynchronous (denoted by dashed arrows), or self-referential (loop arrows).

Activation Bars: Activation bars, also known as activation boxes or activation rectangles, represent the period of time during which an object is actively processing or executing a message. They are depicted as boxes drawn on top of the lifeline, indicating when the object is engaged in processing a message.

Return Messages: Return messages, depicted as dashed arrows with a labeled sequence number, represent the response or return communication from the receiving object back to the sending object. They indicate the flow of control or data in the reverse direction.

Combined Fragments: Combined fragments are used to depict conditional or iterative behavior within a sequence diagram. They allow the modeling of alternative scenarios, loops, parallel execution, and other complex control structures using keywords such as "alt," "opt," "loop," "par," etc.

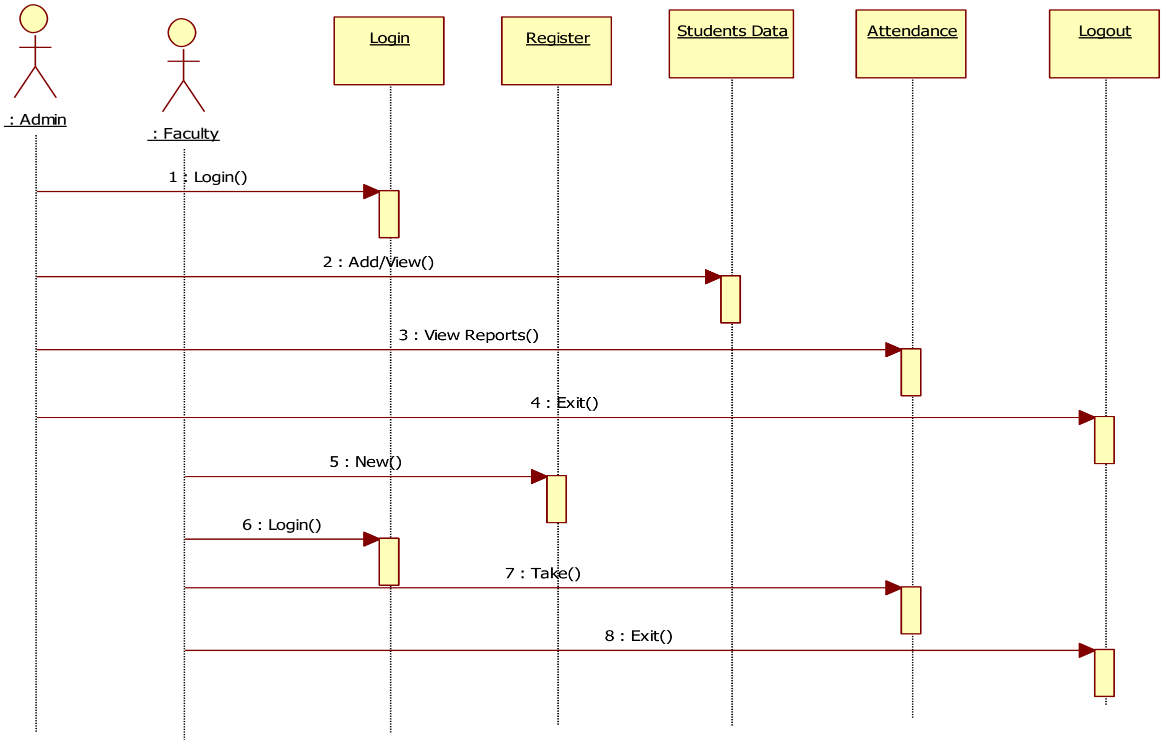
Sequence diagrams serve several purposes in software development:

They illustrate the dynamic behavior of a system by showing how objects collaborate with each other over time.

They help stakeholders understand the sequence of interactions and message flows during a particular scenario or use case execution.

They facilitate the identification of potential design flaws, communication issues, or performance bottlenecks in the system.

Overall, sequence diagrams are valuable tools for modeling and analyzing the dynamic aspects of a system's behavior, aiding in system design, implementation, and testing phases of software development.



4.3.1 Sequence Diagram

4.4 Class Diagram

A class diagram is a type of static structure diagram in Unified Modeling Language (UML) that depicts the structure of a system by showing the classes, their attributes, methods, relationships, and constraints. It provides a visual representation of the static aspects of a system, focusing on the classes and their relationships rather than their dynamic behavior. Here's a detailed description of the key aspects of a class diagram:

Classes: Classes represent the blueprint or template for creating objects in object-oriented programming. They encapsulate data (attributes) and behavior (methods) related to a specific entity or concept in the system. Each class is depicted as a rectangle with three compartments: the top compartment contains the class name, the middle compartment contains the class attributes (data members or fields), and the bottom compartment contains the class methods (operations or functions).

Attributes: Attributes represent the properties or characteristics of a class. They describe the state of objects instantiated from the class. Attributes are typically shown as name-value pairs within the class rectangle. Each attribute has a name and a data type (e.g., integer, string, Boolean).

Methods/Operations: Methods represent the behaviors or operations that objects instantiated from the class can perform. They define the functionality or actions that can be invoked on objects. Methods are listed within the class rectangle along with their signatures (name, parameters, return type).

Relationships: Relationships represent associations, dependencies, and other connections between classes. They describe how classes interact with each other in the system. Common types of relationships include:

Association: Represents a bidirectional relationship between two classes. It indicates that objects of one class are connected to objects of another class.

Aggregation: Represents a "whole-part" relationship between classes, where one class (the whole) contains or is composed of other classes (the parts). It is depicted with a diamond shape on the containing class end.

Generalization/Inheritance: Represents an "is-a" relationship between classes, where one class (subclass or derived class) inherits properties and behaviors from another class (superclass or base class). It is depicted with a solid line with a hollow arrowhead pointing from the subclass to the superclass.

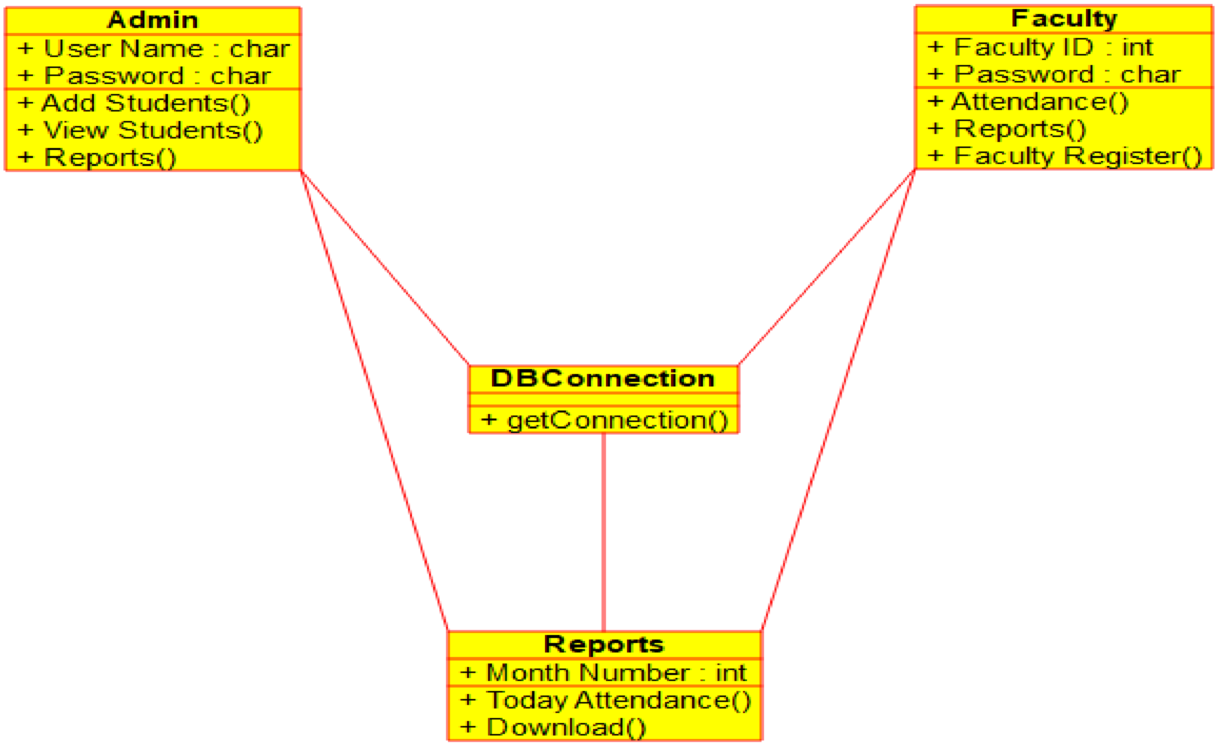
Class diagrams serve several purposes in software development:

They provide a visual overview of the static structure of a system, including its classes, attributes, methods, and relationships.

They facilitate communication and collaboration among stakeholders by providing a common notation for describing system structure.

They serve as a foundation for detailed design, implementation, and documentation of software systems.

Overall, class diagrams are valuable tools for modeling and analyzing the structure of object-oriented systems, aiding in system design, development, and maintenance processes.



4.4.1 Class Diagram

4.5 Component Diagram

A component diagram in Unified Modeling Language (UML) is a type of structural diagram that illustrates the components, their dependencies, and relationships within a system. It provides a high-level view of the physical or logical components that make up a software system and how they interact with each other. Here's a detailed description of the key aspects of a component diagram:

Components: Components represent modular parts or building blocks of a system that encapsulate a set of related functionalities or services. They can be physical (e.g., executable files, libraries) or logical (e.g., modules, subsystems). Components are depicted as rectangles with the component name inside.

Interfaces: Interfaces represent the contracts or specifications that define the interactions between components. They specify the services, operations, or methods provided by a component that can be accessed by other components. Interfaces are depicted as small circles or ellipses attached to the border of the component with a dashed line.

Dependencies: Dependencies represent the relationships between components, indicating that one component depends on another component for its implementation or functionality. Dependencies can be of different types:

Dependency: Represents a general relationship where one component relies on another component. It is depicted with a dashed arrow pointing from the dependent component to the component it depends on.

Usage Dependency: Represents a relationship where one component uses another component. It is depicted with a dashed arrow with an open arrowhead pointing from the using component to the used component.

Provided and Required Interfaces: Provided interfaces represent the services or functionalities offered by a component, while required interfaces represent the services or functionalities required by a component. These interfaces are depicted using small circles or ellipses labeled with the interface name attached to the border of the component.

Connectors: Connectors represent the communication paths or associations between components. They illustrate how components interact with each other to fulfill system requirements. Connectors can be simple lines or arrows connecting the interfaces of different components.

Ports: Ports are used to specify the connection points or communication channels of a component. They represent the interaction points between a component and its environment or other components. Ports are depicted as small squares attached to the border of the component.

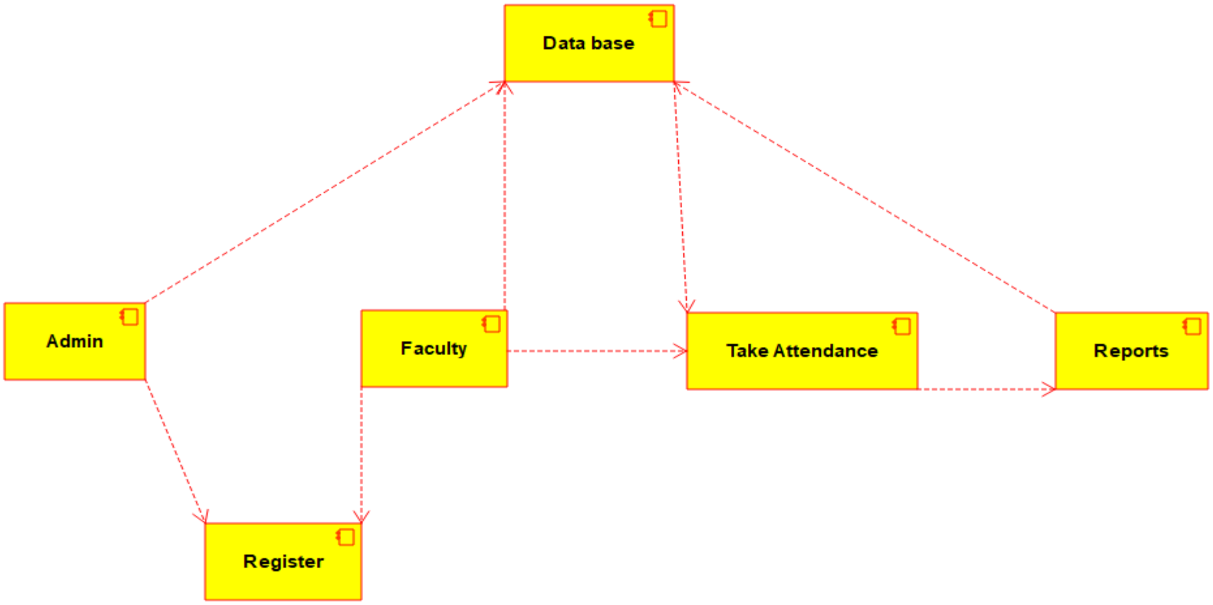
Component diagrams serve several purposes in software development:

They provide a visual representation of the physical or logical structure of a system, including its components and their relationships.

They help stakeholders understand the modular design and architecture of a system, facilitating communication and collaboration among development teams.

They support system analysis, design, implementation, and maintenance by capturing the key structural elements and their interactions.

Overall, component diagrams are valuable tools for modeling and analyzing the architecture of software systems, aiding in system design, development, and documentation processes.



4.5.1 Component Diagram

4.6 Deployment Diagram

A deployment diagram in Unified Modeling Language (UML) is a type of structural diagram that illustrates the physical deployment of software components to hardware nodes in a system. It provides a visual representation of how software artifacts are distributed across different hardware nodes (such as servers, PCs, or devices) and how they interact with each other. Here's a detailed description of the key aspects of a deployment diagram:

Nodes: Nodes represent physical hardware elements or computing devices in the deployment environment, such as servers, PCs, routers, or other hardware devices. Nodes are depicted as square boxes with the node name inside.

Components: Components represent software artifacts or modules that are deployed on the nodes. They can be executable files, libraries, or other software units. Components are depicted as rectangular boxes with the component name inside.

Artifacts: Artifacts represent the physical files or data that are deployed on the nodes. They include executable files, configuration files, databases, or any other type of data required by the software components. Artifacts are depicted as small rectangles or ovals attached to the nodes or components.

Associations: Associations represent the relationships between nodes, components, and artifacts in the deployment environment. They indicate how components are deployed on nodes and how artifacts are stored or accessed. Associations are depicted as solid lines connecting nodes to components or artifacts.

Deployment Specifications: Deployment specifications specify the details of how components are deployed on nodes, such as the deployment location, configuration parameters, and runtime environment settings. They provide additional information about the deployment process and deployment constraints. Deployment specifications are depicted as dashed lines with open arrowheads connecting components to nodes.

Communication Paths: Communication paths represent the communication channels or network connections between nodes in the deployment environment. They illustrate how components deployed on different nodes interact with each other over the network. Communication paths are depicted as lines connecting nodes, usually with labels indicating the type of communication protocol or technology used (e.g., TCP/IP, HTTP, messaging).

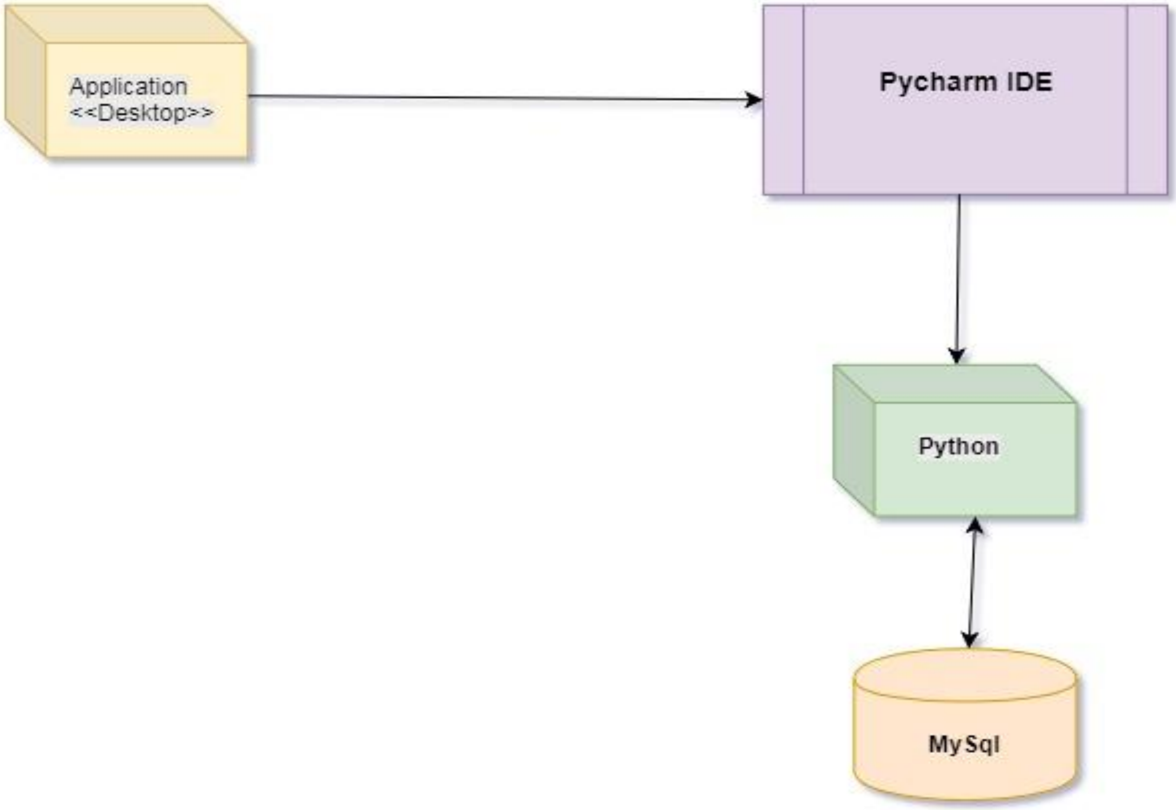
Deployment diagrams serve several purposes in software development:

They provide a visual overview of the physical architecture and deployment topology of a system, including the distribution of software components across hardware nodes.

They help stakeholders understand how software artifacts are deployed and executed in the production environment, facilitating system deployment and operations.

They support system analysis, design, implementation, and maintenance by capturing the key deployment elements and their interactions.

Overall, deployment diagrams are valuable tools for modeling and analyzing the deployment architecture of software systems, aiding in system design, deployment planning, and documentation processes.

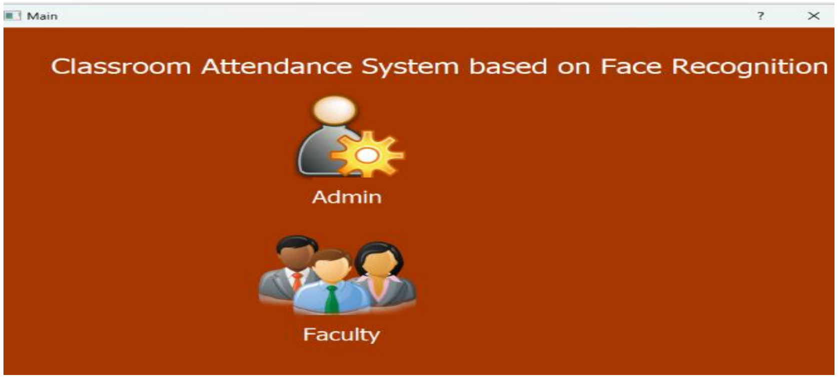


4.6.1 Deployment Diagram

5. IMPLEMENTATION

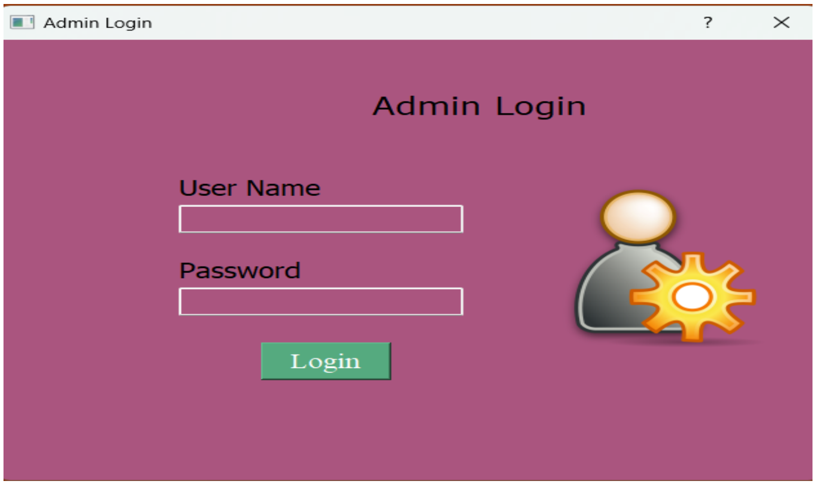
5.1 Program Files

Main.py: This file serves as the entry point of the application. It defines the main window of the application and handles login events for both administrators and faculty members. When the application is executed, it displays a main window with options for admin and faculty login. Clicking on either option opens a respective login interface.



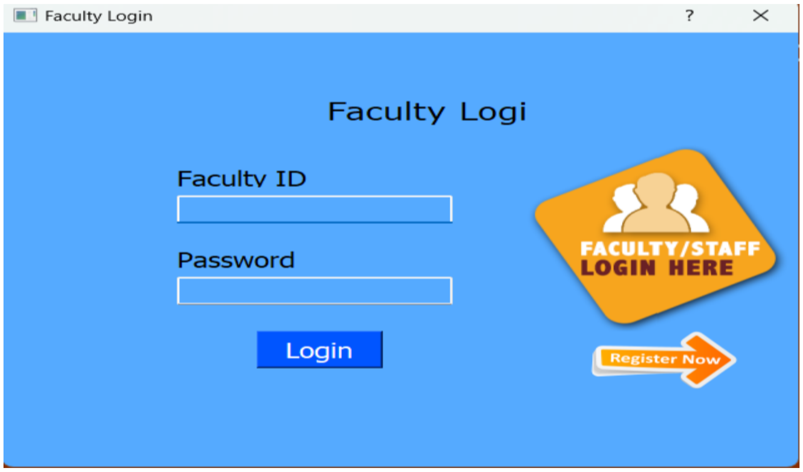
5.1.1 Main Page

Admin.py: This file contains the UI definition for the admin login interface. It includes fields for entering the admin credentials and a button to submit the login information. When the admin submits the login details, the credentials are validated, and if correct, the admin is granted access to the admin dashboard.



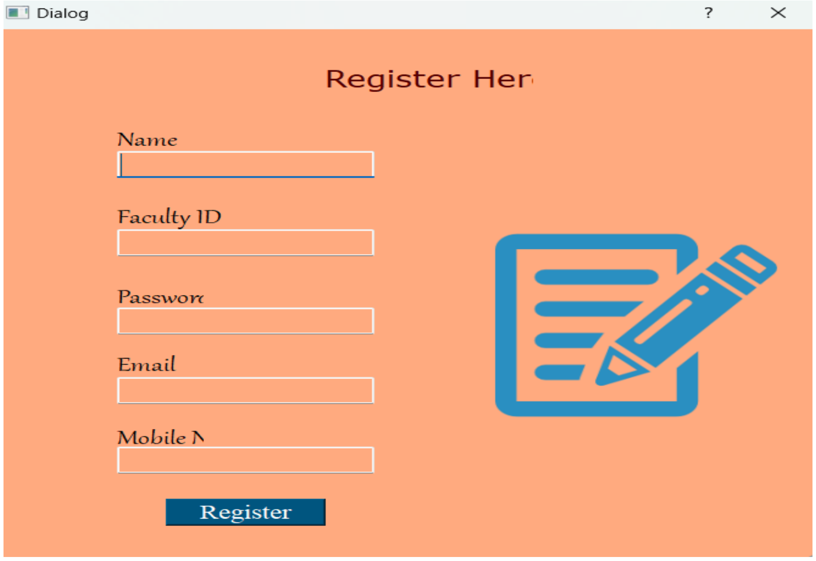
5.1.2 Admin Login Page

Faculty.py: Similar to Admin.py, this file contains the UI definition for the faculty login interface. It allows faculty members to enter their credentials and submit them. Upon successful validation, faculty members are granted access to their dashboard.



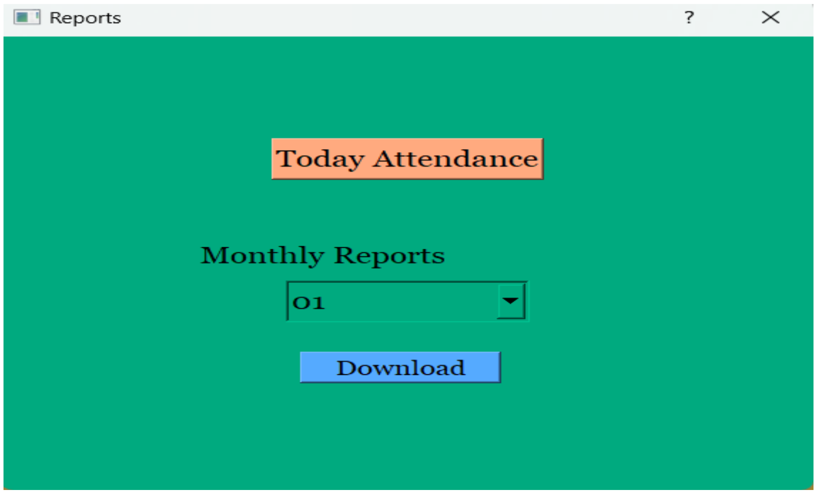
5.1.3 Faculty Login Page

Register.py: This file handles the registration process for new users, specifically faculty members. It includes fields for entering personal information such as name, username, password, email, and mobile number. Upon submission, the information is validated, and if valid, the user is registered in the system.



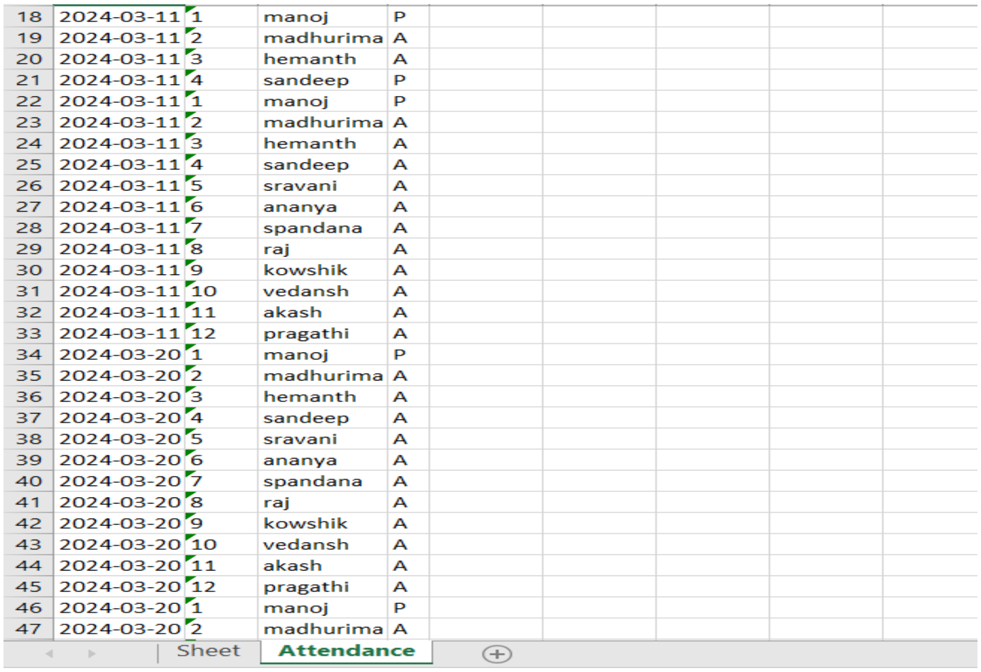
5.1.4 Register Page

Reports.py: This file provides functionality for generating attendance reports. It allows users to download attendance data for a specific month. The data is fetched from the database and filtered based on the selected month. The filtered data is then exported to an Excel file for download.



5.1.5 To get the attendance and report Page

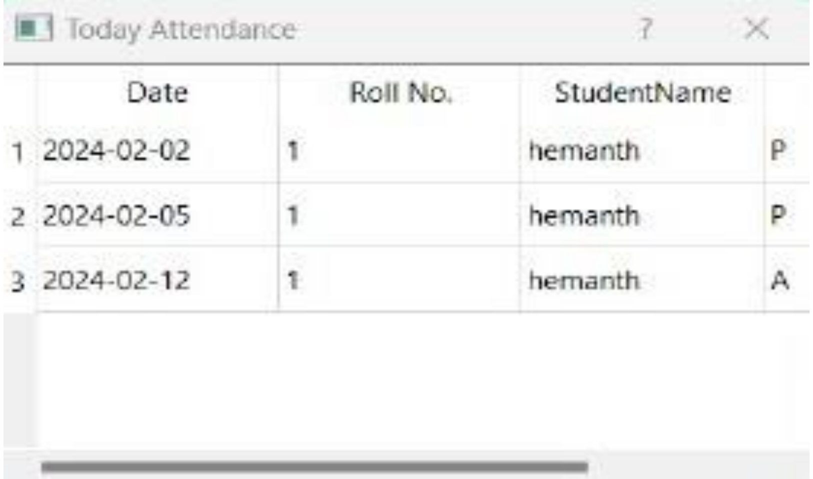
Test.py: This script generates an Excel file containing attendance data. It fetches attendance records from the database and saves them to an Excel file named "Reports.xlsx".



18	2024-03-11	1	manoj	P
19	2024-03-11	2	madhurima	A
20	2024-03-11	3	hemanth	A
21	2024-03-11	4	sandeep	P
22	2024-03-11	1	manoj	P
23	2024-03-11	2	madhurima	A
24	2024-03-11	3	hemanth	A
25	2024-03-11	4	sandeep	A
26	2024-03-11	5	sravani	A
27	2024-03-11	6	ananya	A
28	2024-03-11	7	spandana	A
29	2024-03-11	8	raj	A
30	2024-03-11	9	kowshik	A
31	2024-03-11	10	vedansh	A
32	2024-03-11	11	akash	A
33	2024-03-11	12	pragathi	A
34	2024-03-20	1	manoj	P
35	2024-03-20	2	madhurima	A
36	2024-03-20	3	hemanth	A
37	2024-03-20	4	sandeep	A
38	2024-03-20	5	sravani	A
39	2024-03-20	6	ananya	A
40	2024-03-20	7	spandana	A
41	2024-03-20	8	raj	A
42	2024-03-20	9	kowshik	A
43	2024-03-20	10	vedansh	A
44	2024-03-20	11	akash	A
45	2024-03-20	12	pragathi	A
46	2024-03-20	1	manoj	P
47	2024-03-20	2	madhurima	A

5.1.6 Reports.xlsx

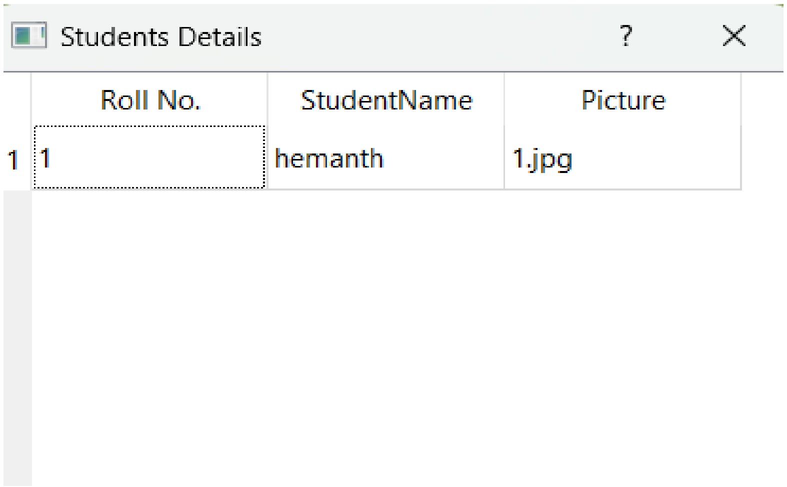
TodayAttendance.py: This file displays today's attendance details in a table format. It fetches attendance data from the database for the current date and displays it in a table within the GUI.



	Date	Roll No.	StudentName	
1	2024-02-02	1	hemanth	P
2	2024-02-05	1	hemanth	P
3	2024-02-12	1	hemanth	A

5.1.7 Today’s attendance display

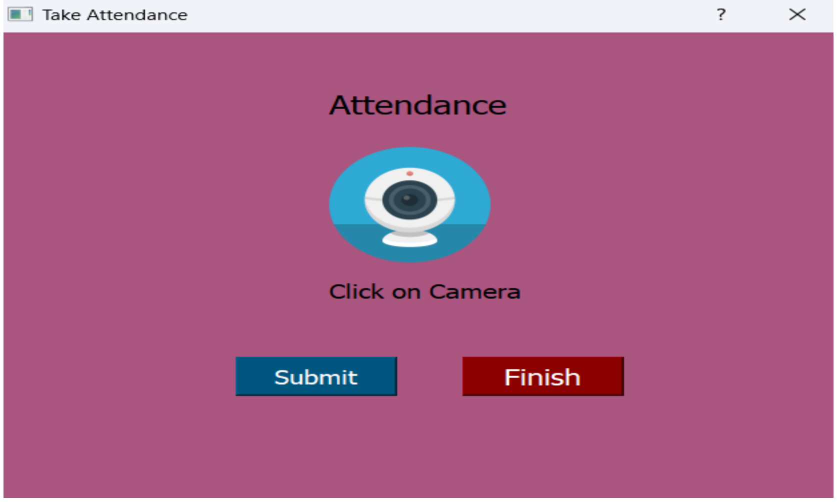
ViewStudents.py: This file provides functionality to view student details stored in the database. It retrieves student information such as roll number, name, and picture from the database and displays it in a table format within the GUI.



	Roll No.	StudentName	Picture
1	1	hemanth	1.jpg

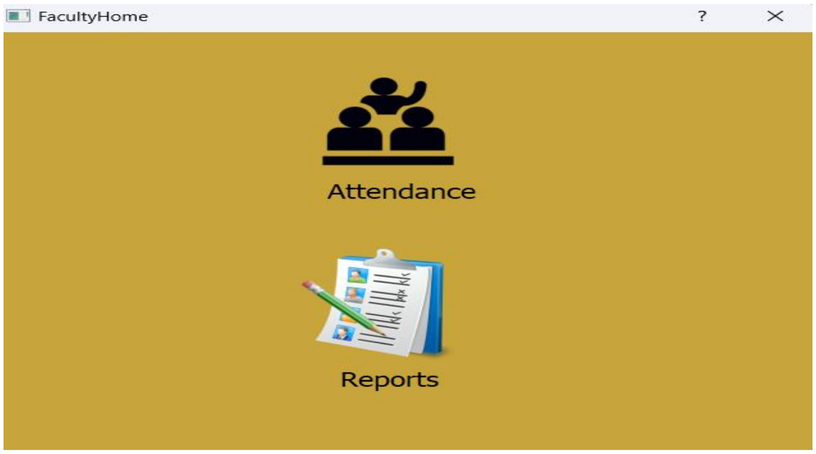
5.1.8 View Students Page

Attendance.py: This PyQt5 application defines a GUI for taking attendance using a camera. It captures images, performs face recognition to identify students, and submits attendance records to a MySQL database. Additionally, it allows closing attendance and updates attendance status based on recognized faces.



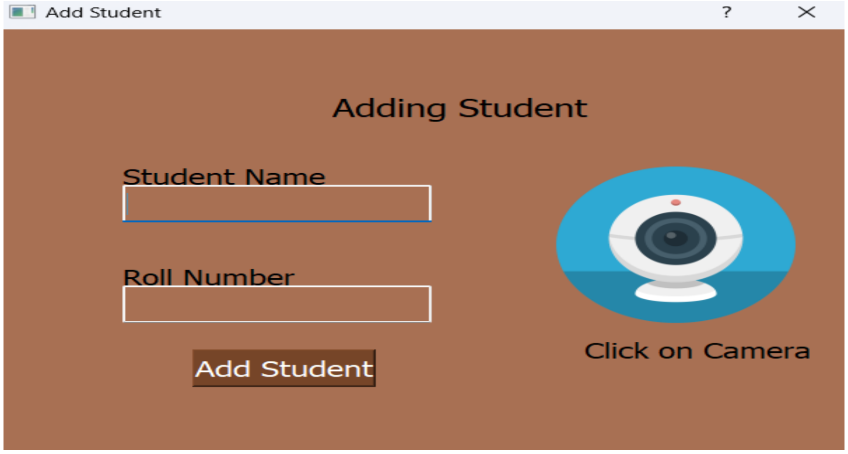
5.1.9 Attendance.py

FacultyHome.py: This PyQt5 application defines a GUI for a faculty home screen. It provides options for the faculty to take attendance and view reports. Clicking on the "Attendance" image allows the faculty to take attendance, while clicking on the "Reports" image allows them to view reports. The attendance and reports functionalities are implemented in separate dialogs.



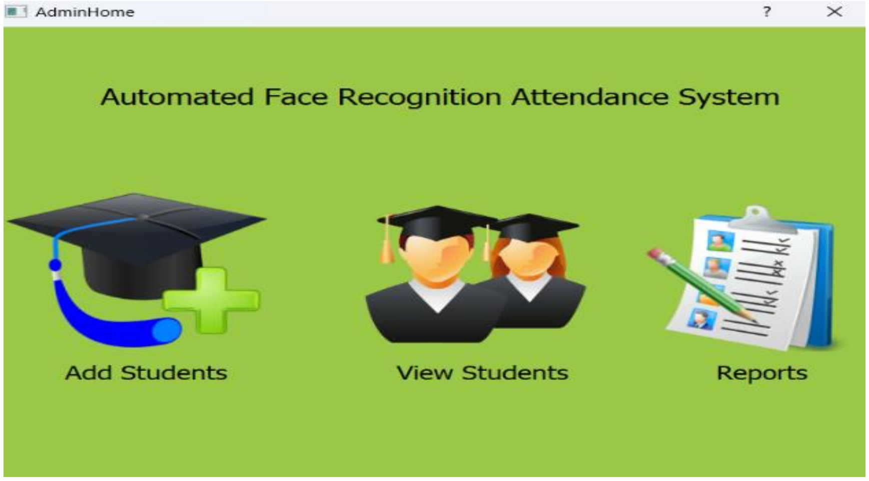
5.1.10 FacultyHome.py

AddStudent.py: This PyQt5 application defines a GUI for adding students to a database. It allows users to input student details such as name and roll number, and capture a picture using a camera. Upon clicking the "Add Student" button, the student details are inserted into the database along with the captured picture. If no picture is taken, it prompts the user to capture one. If any fields are left blank, it prompts the user to fill them out.



5.1.11 AddStudent.py

AdminHome.py: This PyQt5 application defines a GUI for the admin home screen of an automated face recognition attendance system. It provides options for the admin to add students, view students, and generate reports. Clicking on the respective images or labels triggers the corresponding functionality. The interface is designed with images and labels for intuitive navigation.



5.1.12 AdminHome.py

Overall, the project implements a classroom attendance system with features for admin and faculty login, user registration, attendance reporting, and viewing student details. The system interacts with a database to store and retrieve user and attendance data and provides a user-friendly GUI for ease of use.

6. EXPERIMENTAL ENVIRONMENT

PyCharm IDE:

PyCharm could be a coordinates advancement environment (IDE) for Python, advertising highlights like cleverly code completion, investigating, and venture route. Its user-friendly interface and vigorous instruments make it a favored choice for engineers working on Python projects.

MYSQL Database:

MySQL is an open-source social database administration framework (RDBMS) that stores and organizes information. Its employments an organized inquiry dialect (SQL) for overseeing and manipulating databases, making it broadly utilized for web applications and different program systems.

Python Language:

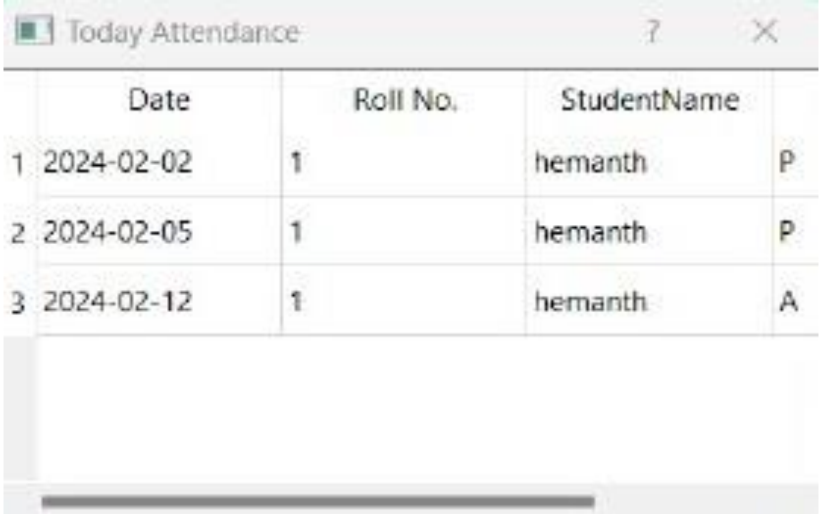
Python may be a deciphered programming dialect known for its effortlessness and meaningfulness. It bolsters numerous ideal models, counting procedural, object-oriented, and functional programming. Python is broadly utilized in web improvement, information science, counterfeit insights, and mechanization, making it a flexible and well-known choice for programmers.

- Along with these we are going utilizing the libraries like OpenCV and calculations like CNN calculation, KNN algorithm.
- We will be too utilizing the webcam which is utilized for capturing the pictures of the student.

7. EXPERIMENTAL RESULTS

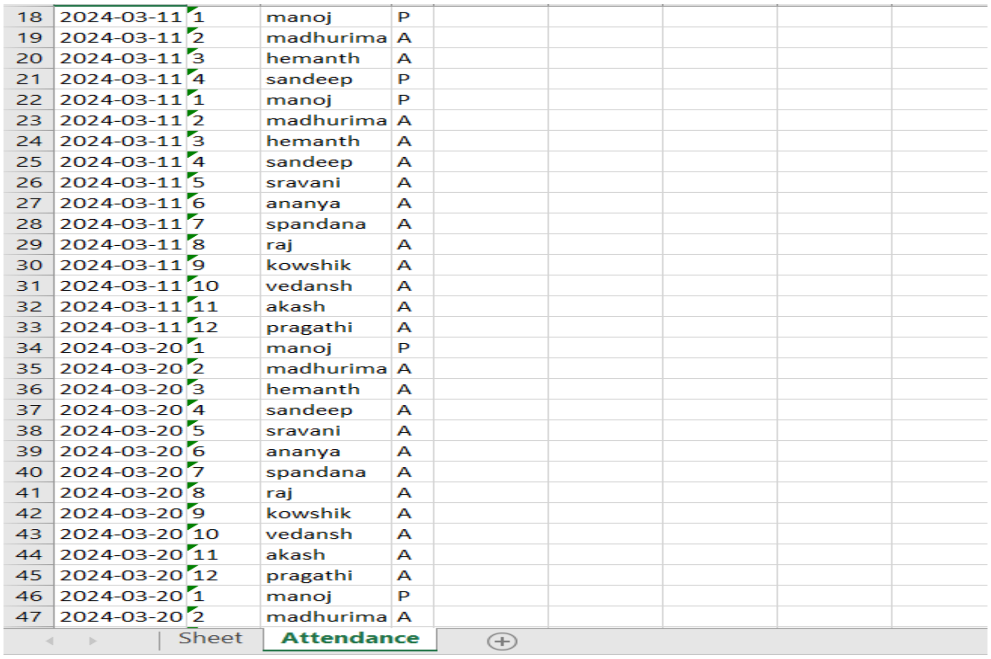
The experimental results of the attendance management system showcased its efficacy in accurately recognizing faces and tracking attendance. Leveraging the Python language, along with specialized libraries and algorithms, such as face_recognition for facial recognition and dlib for face detection, the system demonstrated robust performance. The integration of K-Nearest Neighbors (KNN) algorithm for face recognition, coupled with OpenCV for image capturing and processing, ensured reliable attendance tracking. The graphical user interface developed using PyQt5 facilitated seamless interaction and intuitive operation. Moreover, the utilization of openpyxl and MySQL. Connector libraries enabled efficient data management and storage. The experimental outcomes underscored the system's ability to automate attendance tracking processes effectively, offering a practical solution for educational institutions and organizations.

The final outcome of the project is of downloading the attendance data month wise and also able to see the today's attendance by using the CNN algorithm with face_recognition and dlib libraries for face detection and KNN algorithm for face recognition.



	Date	Roll No.	StudentName	
1	2024-02-02	1	hemanth	P
2	2024-02-05	1	hemanth	P
3	2024-02-12	1	hemanth	A

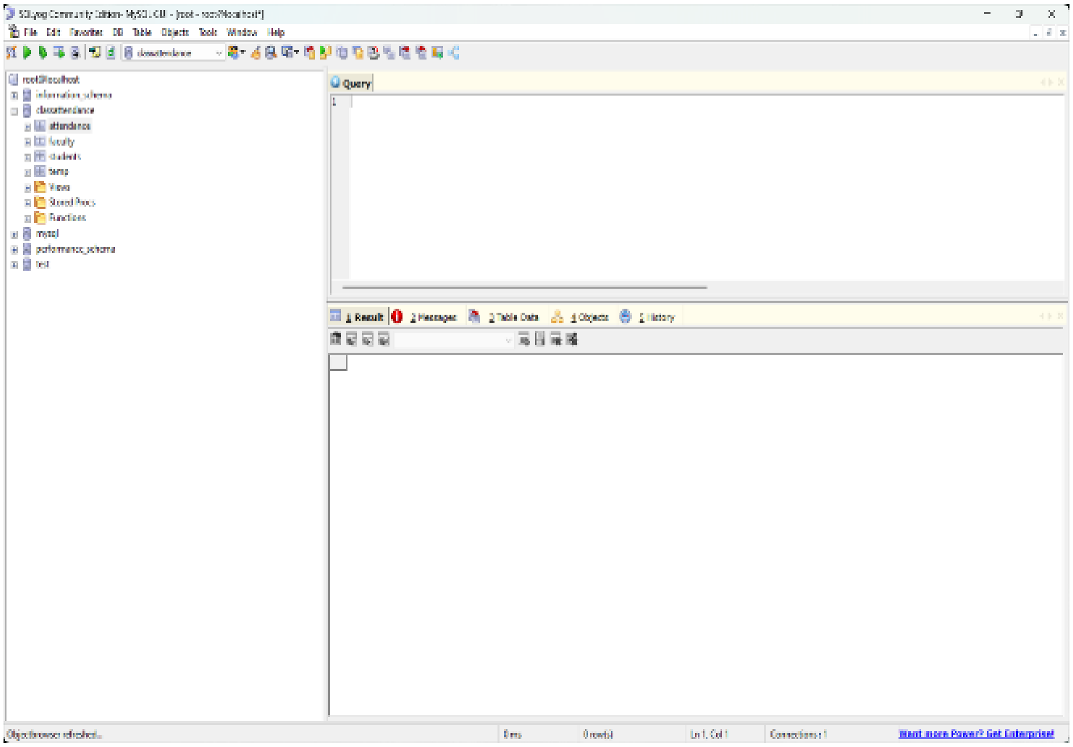
7.1 Today's attendance page



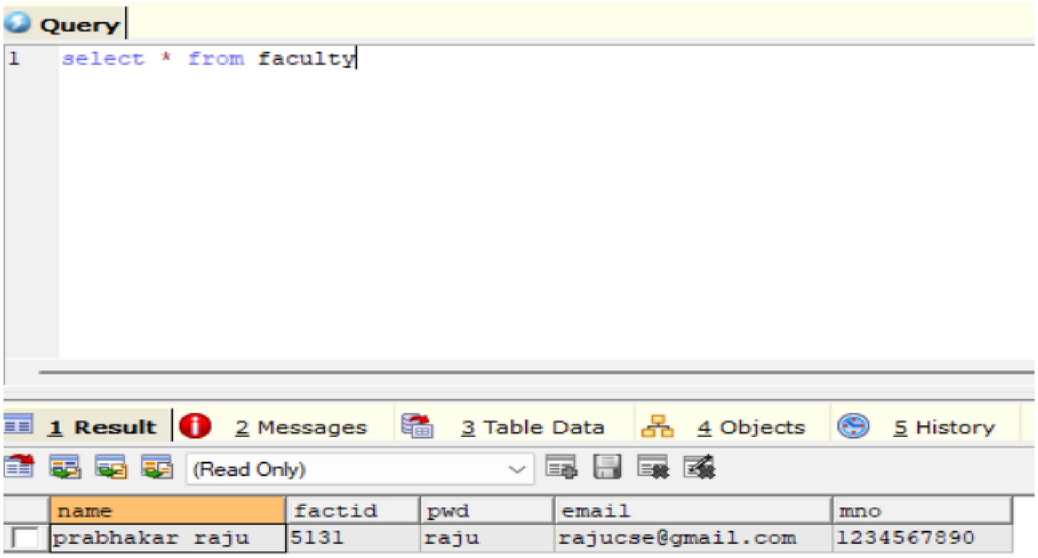
18	2024-03-11	1	manoj	P
19	2024-03-11	2	madhurima	A
20	2024-03-11	3	hemanth	A
21	2024-03-11	4	sandeep	P
22	2024-03-11	1	manoj	P
23	2024-03-11	2	madhurima	A
24	2024-03-11	3	hemanth	A
25	2024-03-11	4	sandeep	A
26	2024-03-11	5	sravani	A
27	2024-03-11	6	ananya	A
28	2024-03-11	7	spandana	A
29	2024-03-11	8	raj	A
30	2024-03-11	9	kowshik	A
31	2024-03-11	10	vedansh	A
32	2024-03-11	11	akash	A
33	2024-03-11	12	pragathi	A
34	2024-03-20	1	manoj	P
35	2024-03-20	2	madhurima	A
36	2024-03-20	3	hemanth	A
37	2024-03-20	4	sandeep	A
38	2024-03-20	5	sravani	A
39	2024-03-20	6	ananya	A
40	2024-03-20	7	spandana	A
41	2024-03-20	8	raj	A
42	2024-03-20	9	kowshik	A
43	2024-03-20	10	vedansh	A
44	2024-03-20	11	akash	A
45	2024-03-20	12	pragathi	A
46	2024-03-20	1	manoj	P
47	2024-03-20	2	madhurima	A

7.2 Report excel picture

The following are the images of the database and the tables used in this project:



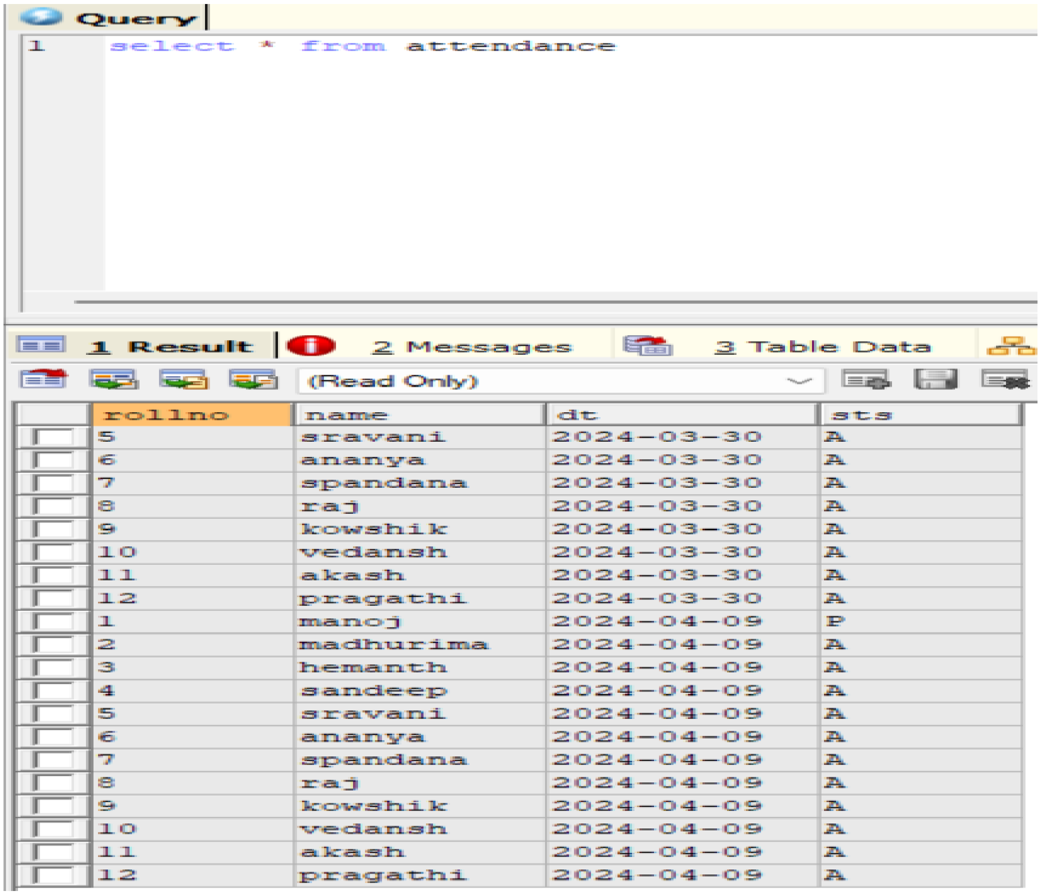
7.3 Database UI



```
select * from faculty
```

name	factid	pwd	email	mno
prabhakar raju	5131	raju	rajucse@gmail.com	1234567890

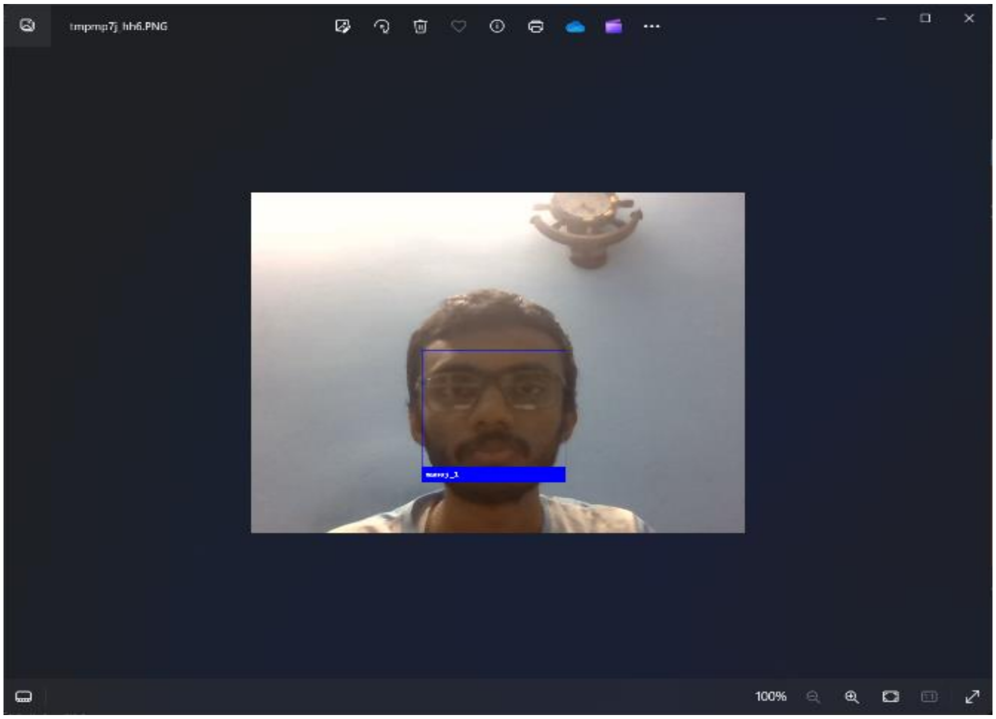
7.4 Faculty Table



```
select * from attendance
```

rollno	name	dt	sts
5	sravani	2024-03-30	A
6	ananya	2024-03-30	A
7	spandana	2024-03-30	A
8	raj	2024-03-30	A
9	kowshik	2024-03-30	A
10	vedansh	2024-03-30	A
11	akash	2024-03-30	A
12	pragathi	2024-03-30	A
1	manoj	2024-04-09	P
2	madhurima	2024-04-09	A
3	hemanth	2024-04-09	A
4	sandeep	2024-04-09	A
5	sravani	2024-04-09	A
6	ananya	2024-04-09	A
7	spandana	2024-04-09	A
8	raj	2024-04-09	A
9	kowshik	2024-04-09	A
10	vedansh	2024-04-09	A
11	akash	2024-04-09	A
12	pragathi	2024-04-09	A

7.5 Attendance Table



7.6 Attendance Image

8. CONCLUSION

In conclusion, the development and implementation of the attendance management system exemplify the potential of leveraging Python-based tools and algorithms for efficient and accurate attendance tracking. Through the utilization of face_recognition and dlib libraries for deep learning-based face detection and recognition, coupled with the KNN algorithm, the system demonstrated robust performance in identifying individuals and recording attendance. The integration of OpenCV for image capturing, along with PyQt5 for the graphical user interface, ensured a user-friendly experience. Additionally, the system's ability to store data in Excel files and MySQL databases facilitated seamless data management. Overall, the attendance management system represents a reliable and effective solution for automating attendance tracking processes in educational institutions and organizations, offering convenience, accuracy, and efficiency.

9. FUTURE SCOPE

Automated face recognition attendance systems leveraging deep learning hold significant promise for various industries and sectors. Here's a glimpse into their future scope:

Education Sector: Schools, colleges, and universities can streamline attendance tracking with greater accuracy and efficiency. This can lead to better utilization of instructional time and more accurate records for administrative purposes.

Corporate Environment: In businesses, automated attendance systems can simplify the time-tracking process for employees. It can also enhance security measures by ensuring that only authorized personnel are present in restricted areas.

Retail and Hospitality: These industries can use face recognition systems to personalize customer experiences. For instance, recognizing loyal customers as they enter a store or hotel, providing tailored recommendations, or expediting check-in processes.

Public Safety and Law Enforcement: Law enforcement agencies can utilize such systems for surveillance and tracking purposes. It can assist in identifying suspects, monitoring public spaces, and enhancing overall security.

Healthcare Industry: Hospitals and clinics can implement face recognition attendance systems for staff management and access control to sensitive areas like patient rooms or medication storage.

Transportation and Travel: Airports, train stations, and other transportation hubs can employ face recognition for passenger identification and security checks, improving the efficiency of boarding processes and enhancing overall safety.

Financial Services: Banks and financial institutions can utilize face recognition for customer authentication in branches or during online transactions, enhancing security and preventing fraud.

However, it's essential to address concerns surrounding privacy, data security, and potential biases in these systems. Regulatory frameworks and ethical guidelines must be developed and adhered to ensure responsible deployment and usage of face recognition technology. Additionally, ongoing research and development are needed to improve the accuracy, robustness, and fairness of these systems to realize their full potential in diverse applications.

10. REFERENCES

[1] HAO YANG1 AND XIAOFENG HAN “Face Recognition Attendance System Based on Real-Time Video Processing”

[2] YANLI REN, ZHUHUAN SONG, SHIFENG SUN, JOSEPH K. LIU, AND GUORUI FENG “Outsourcing LDA-Based Face Recognition to an Untrusted Cloud”

[3] BUSRA KOCACINAR 1, BILAL TAS1, FATMA PATLAR AKBULUT 1, CAGATAY CATAL 2, AND DEEPTI MISHRA 3 “A Real-Time CNN-Based Lightweight Mobile Masked Face Recognition System”

[4] REFIK SAMET, MUHAMMED TANRIVERDI “Face Recognition Based Mobile Automatic Classroom Attendance Management System”

[5] VENUGOPAL A, RAHUL R KRISHNA, RAHUL VERMA U “Facial Recognition System for Automatic Attendance Tracking Using an Ensemble of Deep Learning Techniques”

[6] CHUNDURU ANIL KUMAR, B VENKATESH, S ANNAPOORNA “Smart Attendance System with Face Recognition Using Open CV”

[7] ANASTASIYA YU. STRUEVA, ELEVA V. IVANOVA “Student Attendance Control System with Face Recognition Based on Neural Network”

[8] C. MANJULA DEVI, K. S. VENGADESH, S. GANAPATHY SUBRAMANIAN, EVIN PAUL DANIEL “Attendance Management System Using Face Recognition”

[9] PRIYANKA TYAGI, MAYANK KAUSHIK, HARSHIT KUMAR SINGH, NIKHIL JAISWAL “Attendance System implementation Using Real Time Face Recognition”

[10] VIDYANAND MISHRA, SUNNY RAJ, TANMAY SINGHAL, CHIRAG SANKHLA “Intelligent Face Recognition Based Attendance System”