**Report for Semester Training**

**on**

**“FACE RECOGNITION BASED ATTENDANCE MANAGEMENT SYSTEM”**

*Submitted in partial fulfillment of requirements for the award of the degree*

***Bachelor of Technology***

***in***

***Computer Science & Engineering***

To

**IKG Punjab Technical University, Jalandhar**

**Submitted By**

Name:

Roll no.:

Semester: 8th

Batch: IT

**Under the Guidance**

Mr. Jagbir Singh Gill

Assistant Professor



**Department of Computer Science & Engineering**

**CGC – College of Engineering, Landran**

**Mohali, Punjab – 140307**

**March 2024**

**CERTIFICATE**

This is to certify that the work presented in this Project entitled ““FACE RECOGNITION BASED ATTENDANCE MANAGEMENT SYSTEM”” is a bonafide record of the work done during the period from Jan – June 2024 at “FUTURE FINDERS” by <your name>

The project work is an authentic record of my own work and is carried out under the supervision and guidance of Guide <GUIDE NAME>, <X> Department. The matter presented in the report has not been submitted elsewhere, wholly or in part, for the award of any other degree or diploma.

Your name(19xxxxx)

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

<Guide name>

**Guide Name**

Department of <X>

<HOD NAME>

**HOD <X>**

<X> Sri Guru Granth Sahib World University, Fatehgarh Sahib

**ACKNOWLEDGEMENT**

I take this opportunity to express my sincere gratitude to the Principal, CGC College Of Engineering ,Landran Mohali,Punjab – 140307 for providing this opportunity to carry out the present work.

The constant guidance and encouragement received from Prof. (Dr.) <X>, Professor and Head, Department of <XYZ> Engineering, has been of great help in carrying our present work and helped us in completing this project with success.

I would like to express a deep sense of gratitude to “FUTURE FINDERS” team and my Project Guide Prof. <X>, <XYZ> department for the guidance and support in defining the design problem and towards the completion of my project work. Without their wise counsel and able guidance, it would have been impossible to complete the thesis in this manner.

I am also thankful to all the faculty and staff members of FUTURE FINDERS ORGANISATION for their intellectual support throughout the course of this work.

<Your name>(19xxxxx)

**ABSTRACT**

Face Recognition Based Attendance Management System implemented using Python's tkinter library for the GUI, OpenCV for image processing, and MySQL for database management. The system offers various functionalities including taking images of students for dataset creation, training the face recognition model, automatically filling attendance based on recognized faces, and manually filling attendance for subjects. The GUI provides options for taking images, training the model, automatically marking attendance for registered subjects, and manually entering attendance details. Additionally, an admin panel allows access to registered student details. The system aims to streamline attendance management processes by automating the recognition and recording of student attendance using facial recognition technology.

1. **GUI Development:** The system's main window is created using Tkinter with buttons and entry fields for user interaction.
2. **Image Capture:** Users can input their enrollment ID and name, then capture their images using the "Take Images" button. These images are saved in the "TrainingImage" folder along with their respective enrollment IDs.
3. **Training Model:** The captured images are used to train a face recognition model using the LBPH algorithm. The trained model is then saved in the "TrainingImageLabel" folder.
4. **Automatic Attendance:** Users can choose a subject and fill attendance automatically using face recognition. The system captures live video, detects faces, recognizes enrolled students, and marks attendance accordingly. The attendance data is saved in CSV files.
5. **Manual Attendance:** Users can also manually fill attendance by entering subject names and student details. This data is stored in a MySQL database and can be converted to CSV format for further analysis.
6. **Admin Panel:** There's an admin login feature to view registered students' details stored in a CSV file.



**Table of Contents**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **CONTENTS** | **Page No.** |
| 1 | Certificate |  |
| 2 | Acknowledgement |  |
| 3 | Abstract |  |
| 4 | Introduction |  |
| 5 | 1.1 Introduction Profiles |  |
| 6 | 1.1.1 Project Profile |  |
| 7 | 1.1.2 Company Profile |  |
| 8 | 1.1.3 About Course |  |
| 9 | 1.2 Introduction of Project |  |
| 10 | 1.3 Problem Statement |  |
| 11 | 1.4 Project Objective |  |
| 12 | 1.5 Feasibility Study |  |
| 13 | Literature Review |  |
| 14 | Methodology |  |
| 15 | Project Setup |  |
| 16 | Implementation |  |
| 17 | Screenshots of the project |  |
| 18 | Conclusions and Future Work |  |
| 19 | References |  |

**INTRODUCTION**

Face Recognition Based Attendance Management System using Python's Tkinter for GUI, OpenCV for face detection and recognition, and MySQL for database management. This system offers various functionalities such as taking student images for dataset creation, training the recognition model, automatically filling attendance based on recognized faces, and manually filling attendance.

Users can input student enrollment numbers and names through the GUI, and the system captures images of the students, saves them in a designated folder, and records their details in a CSV file. The training function utilizes the images to train a LBPH (Local Binary Patterns Histograms) face recognition model, which is then saved for future use.

For automatic attendance, the system uses the trained model to recognize faces in real-time, matches them with the enrolled students, and records attendance data. The attendance is then stored in a CSV file and a MySQL database.

Additionally, the system provides an admin panel for accessing student details and a manual attendance filling option for cases where face recognition may not be feasible. Overall, this project aims to streamline the attendance management process by automating it through face recognition technology while also providing manual intervention capabilities when necessary.

**Project Profile**

The project aims to develop an attendance management system using face recognition technology. It allows users to take images of students, train the system with their faces, and then automatically recognize and record attendance based on facial recognition.

The project profile outlines the development of an Attendance Management System titled "FAMS - Face Recognition Based Attendance Management System". This system is designed to automate the attendance recording process through facial recognition technology. It enables users to register students by capturing their images, which are then used to train the face recognition model. Upon completion of training, the system can automatically recognize and record student attendance based on their faces. Additionally, manual attendance filling options are available for flexibility. The technology stack includes Python, leveraging libraries such as OpenCV for face detection and recognition, Tkinter for GUI development, and MySQL for database management (though not explicitly implemented in the provided code). The user interface is intuitive, with buttons and input fields facilitating interaction. Administrators have access to a panel for managing student details and viewing attendance records. The project follows a modular approach, ensuring scalability and maintainability, with error handling mechanisms to enhance reliability.

1. **Key Features:**

* Face image capture for student enrollment
* Training the system with captured images for face recognition
* Automatic attendance recording using face recognition technology
* Manual attendance filling option
* Administrator panel for managing student details and viewing attendance records

1. **Functional Flow:**

* The user can register students by capturing their images and entering their enrollment numbers and names.
* These images are then used to train the face recognition model.
* For attendance, the system automatically recognizes students' faces using the trained model and records their attendance along with the date and time.
* There's also a manual option available for filling attendance.
* Administrators can access a panel to view registered students' details and attendance records.

1. **User Interface:**

* The user interface is developed using tkinter, providing a user-friendly experience for interaction with the system.
* Buttons and input fields are provided for various functionalities such as capturing images, training the model, taking attendance, and accessing the administrator panel.

1. **Project Management:**

* The project follows a modular approach, with different functions defined for specific tasks such as image capture, training, and attendance recording.
* Error handling mechanisms are implemented to ensure smooth user experience and system reliability.
* The code structure is organized into functions and logical blocks for better readability and maintenance.

# About Us

# Future Finders is a ground-breaking platform that develops young Indian talent that is motivated to advance and forge successful careers in IT. We provide a variety of courses to help you launch your career and locate the employer that will assist you as you rise to the top. Our expertise with the latest tools and techniques, and the experience of our professional experts help us deliver high-end services to our esteemed clients. Future Finders educates students and developers about the most recent technologies that are now popular. We are working on distributing informational know-how and offering clients specialized services following global best practices. At Future Finders, we recognize your unrivalled skill and help you realize your creative ideas. Our sole goal is to provide students with cutting-edge practical skills that will enable them to swiftly and effectively adapt to the constantly evolving technologies found in the business world. At Future Finders, our goal is to raise educational standards via innovation in both quality and practical knowledge.



**MISSION**

* At Future Finders, our goal is to improve quality and practical skills while raising the bar for education. Future Finders' principal objective is to bridge the knowledge gap between what is being taught in schools and what the industry needs.

**VISION**

* Future Finders' vision is to provide students with cutting-edge practical skills so they can easily handle and swiftly acclimate to the constantly evolving technology in the corporate world.

**PHILOSOPHY**

# To impart hardcore practical quality training among students/developers about latest technologies trending today.

To share knowledge of information security and create awareness in the market. The solution to clients' as per the International standard practices and governance.

# To support good business practices through continual employee training and education

To equip a local team with a strong knowledge of international best practices and international expert support so as to provide practical advisories in the best interests of our clients

* JAVA
* FULL STACK
* C LANGUAGE
* C++ LANGUAGE
* CYBER SECURITY
* MERN STACK
* PYTHON
* DIGITAL MARKETING
* PHP
* NETWORKING
* ANDROID
* IOS
* .NET
* ORADE DBA

SOFTWARE TESTING

* ETHICAL HACKING
* MACHINE LEARNING
* BIG DATA HADOOP
* NODE JS
* ANGULAR JS
* CLOUD COMPUTING
* AUTO CAD
* NETWORKING
* ROBOTICS
* VLSI – VHDL
* PCB
* MATLAB
* EMBEDDED SYSTEM
* IOT
* PLC / SCADE
* ARDUINO
  + NX- CAM
  + AUTOCAD ME
  + SOLID WORKS
  + CATIA
  + CREO
  + CNC PROGRAMMING
  + ANSYS



* + - MARKETING
    - DIGITAL MARKETING
    - FINANCE
    - HR MANAGEMENT
  + AUTOCAD CE
  + PRIMA VERA
  + MX ROAD
  + 3DS MAX
  + REVIT ARCHITECTURE
  + STAAD PRO

CSE & IT

ECE & EE

MECHANICAL

CIVIL

MANAGEMENT

Why choose us:

 Experience

 Goal Oriented

 A Streamlined / Quality-Driven Process  Talented Designers & Expert Developers

 Our Websites & E-marketing Platforms are Easy to Manage  We Are Dedicated to Our Client’s Success

## We focus on imparting practical skills to the trainees & not just theoretical knowledge. The courses are designed in this way at FUTURE FINDERS correspond to the standards of the corporate divisions and industries. Only through the acquisition of practical skills you can handle the everlasting technologies that venture out in real-time situations.

Description: * At **FUTURE FINDERS,** Future Finders has committed staff members with organised learning curricula that can assist you in beginning your career in the most cutting-edge and successful industries.

Description: * **Quality of the Product:** Our software service sector has been maintaining the highest international standards of quality.

## Description: * Live projects: Working on active projects and working at a job aid in experiencing growth. In keeping with this, we assist you in developing greater self-assurance, which helps you succeed as quickly as feasible.

Description: * **Global certification:** The courses that Future Finders offers make sure that you are certified and eligible to grab the best opportunities in your career.

## Description: * Partnership: FUTURE FINDERS, considers every client a partner. From the initial stages, you are closely involved into the procedure of technical classification, development, and testing.

Description: * **Jobs & Career Prospective:** Future Finders views education and learning as investments in oneself. Consequently, the paths we have set forth ensure that your investment in us is a success

* In addition to a panel of eminent consultants and advisors, we have a dedicated pool of trained Developers and Trainer, investigators, working under the guidance of professional managers. **“A Ship is as good as the crew who sail her**.**”** Our Technical team of professionals handing, designing & delivering of projects has a strong presence in the North India & the US. Our engineers are already working on the latest technologies like **I-Phone & Android** Applications, **Robotics**, **VLSI-VHDL**, Embedded System, Networking and **Cloud computing.** Some of the key professionals and advisors are listed

### Mr. Bonish Singla: (Director)



* He is the backbone of FUTURE FINDERS, manage the company’s day to day affairs and a man with more than 9 years rich practical experience who believes in taking up new ventures and projects. He has been awarded many times for his exemplary work in process improvement for IT Service Delivery Domains. MASTERS in Computer applications and Certified from CU Certification. Holds total of 9 Years of rich experience including 5 Years in Information Security Implementation, Maintenance and Auditing and initial over 4 years of experience in Project Management, Client Relationship Management and Server, Desktop, and IT Service Delivery web designing.

### 

### Miss. Harjit Kaur :(Branch Manager)



* She has more than 5 years solid industrial experience in software companies and she is very innovative in her technical approach. She has completed her masters in MBA. She takes all the responsibilities and maintains staff by recruiting , selecting , orienting , and training employees and Accomplishes staff results by communicating job expectations , planning ,monitoring , and appraising job results.

### Miss. Harsimran: (HR)



* Human resources (HR) are the division of a business responsible for finding, recruiting, screening, and training job applicants. MBA in HR and marketing from (CU) Total of 5+ Years of rich experience HR departments also handle employee compensation, benefits, and terminations. HR departments must keep up to date with laws that can affect the company and its employees. She also assists with payroll management so employees receive their paychecks on time.

### 

### Miss. Isha Bala: (Technical Head)



* A technical lead, or tech lead, oversees the technical aspects of a software team. M. tech and diploma in (CSE) . She helps making architectural and design decisions, guiding team members, and supervising system modifications. Identify potential risk and forming contingency plan as fast as possible. Efficiently liaise with the team members, clients , and also the management .

### Miss. Nihirika: (Head Counselor)

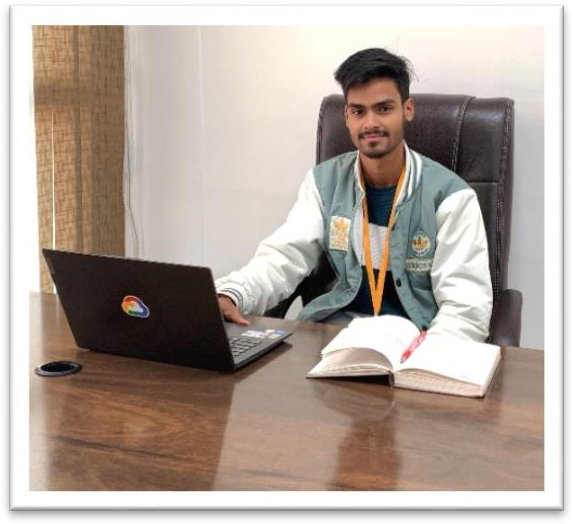
### 

* Head Counselor, under the general direction of the Head - Sales and Marketing, provides leadership and direction to the Counseling Department and assumes responsibilities in developing, implementing, and evaluating the Company counseling and guidance program that includes academic, career, personal/social development. She completed her degree in B .Tech

### M r. Jaspal Singh: (Civil &Mechanical Head)

* He is leader of the team of civil, mechanical, and electrical engineers and responsible for the planning and analysis of the aspect of the construction that involves mechanical works . He has more than 37 years of experience in industrial field. He is providing services as a technical trainer for more than 8 years. He did his B. Tech in Mechanical Engineering from PEC (Punjab Engineering College). He has been awarded many times for his brilliant services.

### Mr. Chetan Kalra: (Digital Marketing Head)



* B.tech (CSE) – IKG-PTU, Experienced digital marketing manager with extensive experience building, maintaining, and running successful digital marketing campaigns from past 4 years. Bringing forth broad marketing knowledge, coupled with focused campaign experience. Adept at creating and implementing client- centered, successful campaigns, aimed at improving brand awareness and presence. Collaborative and creative manager accomplished at managing digital marketing presence content. Experienced in leading teams of marketing professionals to meet and exceed digital marketing goals

**Mr. Shivam: (Java Developer)**

### 

* Having 5+ years of experience in analysis, design, development, testing & implementation of complex software applications. B. Tech in CSE (CGC College) Experience and involvement in designing, implementing, and evaluating end-to-end systems using several Java frameworks and technologies like J2EE.

### Miss. Archana: (Full Stack Developer)



* Full Stack Developer with 6+ years of hands-on experience designing, developing, and implementing applications and solutions using a range Of technologies and programming languages. B.tech (CSE) PU Certification Seeking to leverage broad development experience and hands-on technical



Plot NO F-465, K&B Tower, Second Floor, Phase 8B, Mohali I (+91) 98559-08009 I [www.futurefinders.in](http://www.futurefinders.in/)

### Mr. Ashwani: (Automation Executive)

### 

* (EE) – CGC College Automation engineer with 4+ years of experience in a variety of industries. Passionate for developing and implementing process improvements through the use of robotics, PLCs, and HMIs. Demonstrated ability to lead cross-functional teams in the design, development, and deployment of manufacturing and process automation solutions.

**Mr. Ayush: (PHP Developer)**



* PHP developer to manage our back-end services and ensure a seamless interchange of data between the server and our users. Bachelor's degree in computer programming, development and Certified from CU Certification PHP developer, responsible for developing and coding all server-side logic and required to maintain the central database and respond to requests from front-end developers

**1.1.3 About the course**

**MODULE 1:**

**History of Python**

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

**Features:**

* **Easy-to-learn −** Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
* **Easy-to-read −** Python code is more clearly defined and visible to the eyes.
* **Easy-to-maintain −** Python's source code is fairly easy-to-maintain.
* **A broad standard library −** Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
* **Interactive Mode −** Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
* **Portable −** Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
* **Extendable −** You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient**.**
* **Databases −** Python provides interfaces to all major commercial databases.
* **GUI Programming −** Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
* **Scalable −** Python provides a better structure and support for large programs than shell scripting.

### Good to know

* The most recent major version of Python is Python 3, which we shall be using in this tutorial. However, Python 2, although not being updated with anything other than security updates, is still quite popular.
* In this tutorial Python will be written in a text editor. It is possible to write Python in an Integrated Development Environment, such as Thonny, Pycharm, Netbeans or Eclipse which are particularly useful when managing larger collections of Python files.

### Python Syntax compared to other programming languages

* Python was designed for readability, and has some similarities to the English language with influence from mathematics.
* Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses.
* Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and classes. Other programming languages often use curly-brackets for this purpose.

The Python syntax defines a set of rules that are used to create Python statements while writing a Python Program. The Python Programming Language Syntax has many similarities to Perl, C, and Java Programming Languages. However, there are some definite differences between the languages.

## First Python Program:

Let us execute a Python "Hello, World!" Programs in different modes of programming.

Print(“Hello world”)

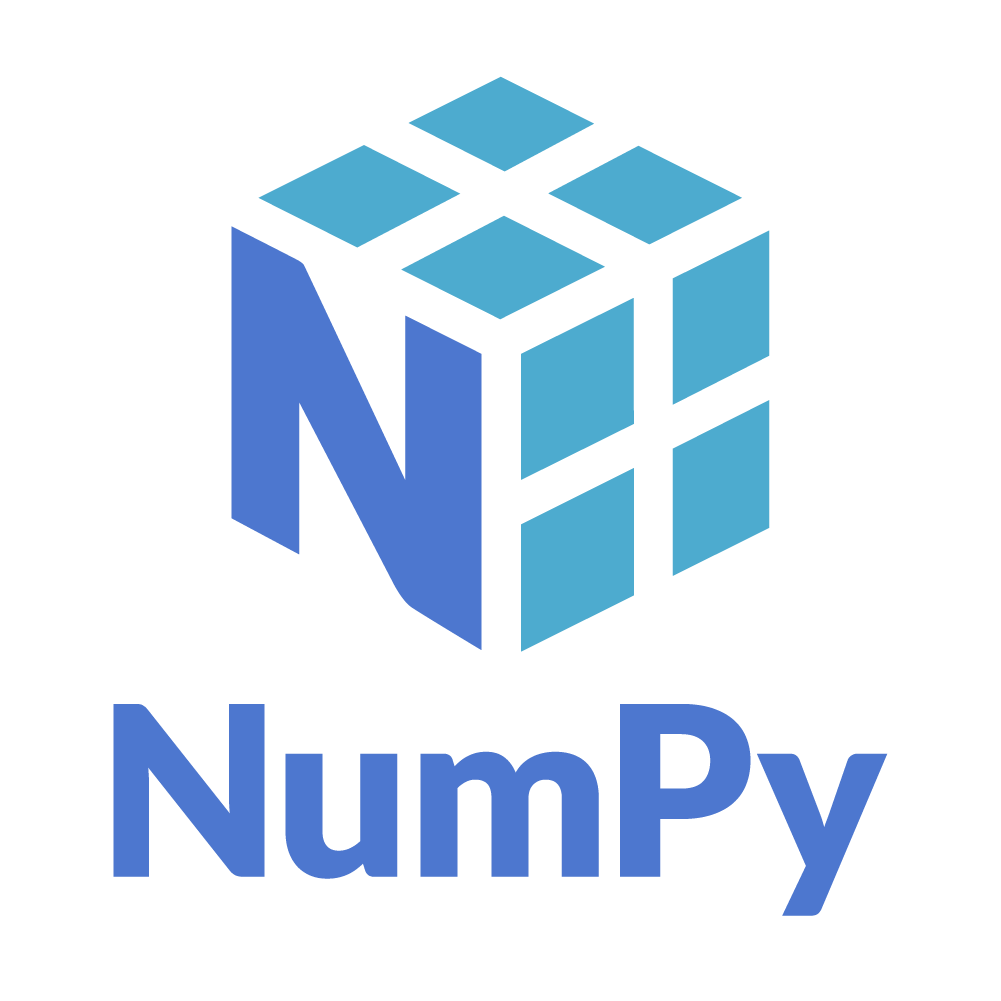
**Numpy Library:**

NumPy (Numerical Python) is an open-source library for Python that is used extensively in scientific computing and data analysis. It is a popular library that provides support for multi-dimensional arrays and matrices, as well as a wide range of mathematical functions for performing operations on these arrays.

One of the key features of NumPy is its powerful N-dimensional array object, which is designed to handle large datasets efficiently. This array object provides a flexible and convenient way to represent vectors, matrices, and other high-dimensional data structures. NumPy arrays can be indexed and sliced like regular Python lists, but they offer many additional capabilities, such as vectorized operations and broadcasting.

Here are some key features and functionalities of NumPy:

1. **Multi-dimensional arrays:** NumPy provides a powerful array object called ndarray, which represents multi-dimensional arrays of homogeneous data types. These arrays can be of any dimension and can hold elements of the same data type.
2. **Efficient operations:** NumPy provides a wide range of mathematical functions that operate element-wise on arrays, making it efficient and convenient for performing mathematical computations on large datasets.
3. **Broadcasting:** NumPy's broadcasting capability allows operations to be performed on arrays of different shapes, making it easy to write vectorized code without using explicit loops.
4. **Linear algebra operations:** NumPy includes functions for linear algebra operations such as matrix multiplication, matrix inversion, eigenvalue decomposition, and singular value decomposition (SVD).
5. **Random number generation:** NumPy provides a random module (numpy.random) that generates random numbers from various probability distributions.



**Pandas Library:**

Pandas is an open-source data analysis and manipulation library for the Python programming language. It is widely used in data science and analytics to manipulate and analyze data in various formats, such as CSV, Excel, SQL databases, and more. The library provides a variety of data structures and functions to simplify the process of data manipulation and analysis.

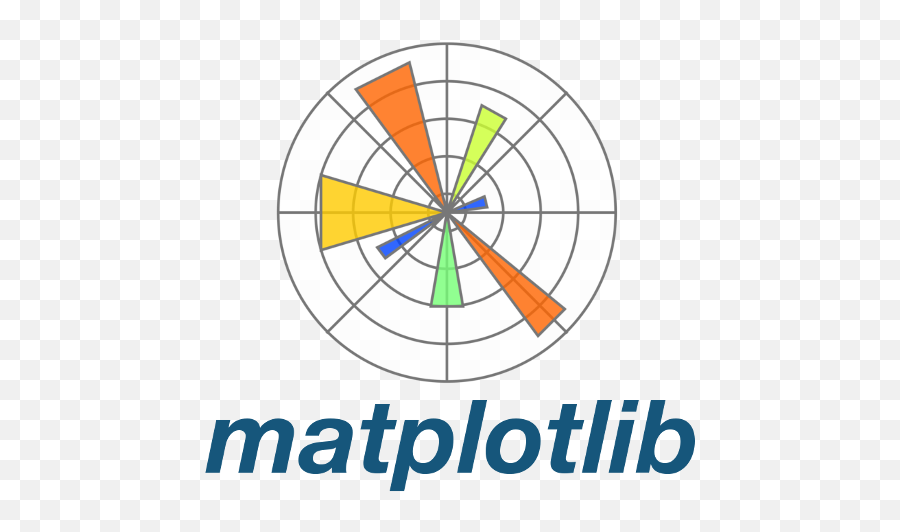
Some key features and functionalities of Pandas include:

1. **DataFrame:** The core data structure in Pandas is the DataFrame, which is a two-dimensional labeled data structure with columns of potentially different types. It can be thought of as a spreadsheet or SQL table, where data can be easily manipulated and analyzed.
2. **Series:** Pandas also provides the Series data structure, which is a one-dimensional labeled array capable of holding any data type (integers, strings, floating point numbers, Python objects, etc.). A DataFrame is essentially a collection of Series objects.
3. **Data manipulation:** Pandas provides a wide range of methods and functions for manipulating and transforming data, including filtering, sorting, grouping, merging, and reshaping operations. These operations allow users to clean and preprocess data for analysis efficiently.
4. **Data alignment and indexing:** Pandas supports powerful indexing and alignment functionality, enabling users to access and manipulate data based on labels rather than integer indices. This makes it easy to handle missing data and align data from different sources.
5. **Input/output tools:** Pandas provides functions to read data from various file formats such as CSV, Excel, SQL databases, and JSON, as well as the ability to write data back to these formats.
6. **Time series data:** Pandas has excellent support for working with time series data, including date/time indexing, time zone handling, resampling, and frequency conversion.
7. **Integration with other libraries:** Pandas integrates well with other Python libraries such as NumPy, Matplotlib, and scikit-learn, making it a key component of the Python data science ecosystem.



**Matplotlib Library:**

Matplotlib is an open-source data visualization library for the Python programming language. It is widely used in data science and analytics to create high-quality, publication-ready visualizations of data. The library provides a variety of functions and tools for creating different types of visualizations, such as line plots, scatter plots, histograms, and more.



**SciPy Library:**

Scipy is an open-source scientific computing library for Python that provides a wide range of tools and functions for solving various scientific and engineering problems. The library is built on top of NumPy, another popular Python library for numerical computing, and provides additional functionality for signal processing, optimization, interpolation, and more.

.

Scipy also provides support for signal processing, including filtering, Fourier transforms, and wavelet transforms. These tools are particularly useful in analyzing and manipulating signals from various sources, such as audio or biomedical signals.



**Machine Learning:**

Machine learning is a subfield of artificial intelligence that focuses on developing algorithms and statistical models that enable computers to learn from data without being explicitly programmed. In other words, machine learning algorithms learn patterns and relationships in data and use this knowledge to make predictions or decisions on new, unseen data.

There are three main types of machine learning:

* Supervised learning
* Unsupervised learning
* Reinforcement learning.

**Face Recognition Based Attendance Management System**

**1.2. Introduction of Project:**

The code provided implements a Face Recognition Based Attendance Management System using Python's Tkinter for GUI, OpenCV for face detection and recognition, and MySQL for database management. This system offers various functionalities such as taking student images for dataset creation, training the recognition model, automatically filling attendance based on recognized faces, and manually filling attendance.

Users can input student enrollment numbers and names through the GUI, and the system captures images of the students, saves them in a designated folder, and records their details in a CSV file. The training function utilizes the images to train a LBPH (Local Binary Patterns Histograms) face recognition model, which is then saved for future use.

For automatic attendance, the system uses the trained model to recognize faces in real-time, matches them with the enrolled students, and records attendance data. The attendance is then stored in a CSV file and a MySQL database.

Additionally, the system provides an admin panel for accessing student details and a manual attendance filling option for cases where face recognition may not be feasible. Overall, this project aims to streamline the attendance management process by automating it through face recognition technology while also providing manual intervention capabilities when necessary.

**1.3 Problem Statement:**

Face Recognition Based Attendance Management System using Python's Tkinter for the GUI, OpenCV for face recognition, and MySQL for database management. The system offers various functionalities such as registering students by capturing their images along with enrollment details, training the model with the collected images, automatically taking attendance using face recognition for a chosen subject, and manually filling attendance for any subject. Additionally, there's an admin panel for checking registered students' details and logging in for system access. The primary goal of this system is to automate the attendance process in educational institutions by leveraging facial recognition technology, enhancing efficiency, accuracy, and convenience while managing attendance records.

* **Registration of Students:** Allows users to input student enrollment numbers and names, which are then stored in a CSV file along with the corresponding images of the students' faces.
* **Taking Images for Datasets:** Enables capturing images of students' faces for training the face recognition model. The system detects faces using the Haar Cascade Classifier and saves the images along with the associated enrollment numbers.
* **Training the Model:** Trains a face recognition model using the captured images and associated enrollment numbers. The model is trained using the LBPH (Local Binary Patterns Histograms) algorithm.
* **Automatic Attendance**: Automatically recognizes students' faces in real-time using the trained model to mark their attendance. The system identifies students based on their faces and records their attendance in a CSV file.
* **Manually Fill Attendance:** Provides a manual option to input student enrollment numbers and names for filling attendance. Users can enter subject names and manually mark the attendance of students.
* **Admin Panel:** Offers an admin panel for user authentication. Admins can log in using a username and password to view registered students' details.

**1.4 Project Objective:**

The project objective seems to be the development of a Face Recognition Based Attendance Management System (FAMS). The system aims to automate the process of taking attendance by recognizing faces of students and recording their attendance in a database. The system offers the following features:

* **Registration of Students:** Allows users to register students by entering their enrollment numbers and names. This information is stored in a CSV file (StudentDetails.csv).
* **Taking Images for Datasets:** Provides functionality to capture images of students' faces for training the face recognition model. The captured images are saved in a directory named TrainingImage.
* **Training the Model:** Trains the face recognition model using the captured images of students' faces. The trained model is saved in a file named Trainner.yml in the TrainingImageLabel directory.
* **Automatic Attendance:** Automatically recognizes students' faces using the trained model and records their attendance in a CSV file based on the recognized subject. The attendance records are stored in files named according to the subject, date, and time.
* **Manual Attendance:** Allows manual entry of attendance for a specific subject by entering enrollment numbers and names of students. The manually entered attendance is stored in a CSV file and can be converted to a spreadsheet for further analysis.
* **Admin Panel:** Provides access to an admin panel where users can log in using a predefined username and password (pragya and pragya123 in this case) to view registered student details.

**1.5 Feasibility Study**

Feasibility study is an essential step in the software development process to determine whether a proposed system is feasible and viable within the constraints of time, cost, and resources. Here's how the above code can be analyzed in terms of feasibility:

1. **Technical Feasibility:**

* **Analysis:** The code utilizes libraries like tkinter for GUI, OpenCV for image processing, and Pandas for data manipulation. These libraries are commonly used and well-supported, indicating technical feasibility.
* **Conclusion:** The project seems technically feasible due to the availability of required tools and libraries.

1. **Operational Feasibility:**

* **Analysis:** The system aims to automate attendance management through face recognition, which could potentially enhance operational efficiency compared to manual methods.
* **Conclusion:** The system is operationally feasible as it addresses a real-world problem and offers automation benefits.

1. **Economic Feasibility:**

* **Analysis:** The code doesn't explicitly mention the costs involved in terms of development, hardware, or maintenance. However, it relies on open-source libraries and doesn't require specialized hardware beyond a standard computer with a webcam.
* **Conclusion:** The system appears economically feasible as it doesn't involve significant upfront costs and may lead to cost savings in the long run by streamlining attendance management.

1. **Schedule Feasibility:**

* **Analysis:** The code includes various functionalities like taking images, training the model, and filling attendance, each of which may require a different amount of time for implementation and testing.
* **Conclusion:** While the schedule feasibility isn't explicitly addressed in the code, it's feasible to implement the system within a reasonable timeframe considering the complexity of each functionality.

1. **Legal and Ethical Feasibility:**

* **Analysis:** The system deals with sensitive data such as student enrollment IDs and attendance records. It's essential to ensure compliance with data protection regulations (e.g., GDPR) and ethical considerations regarding privacy and consent.
* **Conclusion:** Legal and ethical feasibility considerations are crucial and need to be addressed through proper data handling practices, consent mechanisms, and compliance with relevant regulations.

**2. Literature Review**

Face Recognition Based Attendance Management System implemented using Python's Tkinter for the GUI, OpenCV for face detection and recognition, and Pandas for data manipulation. Here's a brief overview of the system's functionalities:

1. **Main Interface:** The system's main interface is created using Tkinter. It allows users to perform various tasks related to attendance management, such as taking images, training the model, filling attendance automatically or manually, and checking registered students.
2. **Taking Images:** Users can input enrollment numbers and names to capture images of individuals for creating a dataset. OpenCV is used to detect faces in real-time video captured from the webcam. These images are then saved in a directory along with corresponding enrollment numbers and names.
3. **Training Model:** After capturing images, users can train the face recognition model using the captured dataset. The LBPH (Local Binary Patterns Histograms) Face Recognizer from OpenCV is utilized for this purpose. The trained model is then saved for future use.
4. **Automatic Attendance:** The system allows for automatic attendance tracking. Users can select a subject, and the system will use the trained model to recognize faces in real-time. Detected faces are matched with the stored dataset to mark attendance automatically. Attendance records are saved in CSV files.
5. **Manually Fill Attendance:** Users can also manually fill attendance by entering enrollment numbers and names for a specific subject. The system provides error handling for empty fields and allows users to clear entries. Attendance records are stored in a MySQL database.
6. **Admin Panel:** There's an admin panel for authentication. Upon successful login, administrators can view registered students' details stored in a CSV file.

**3. Methodology**

**3.1 Libraries used in the project:**

**Tkinter:**

1. **tkinter:** Used for creating the graphical user interface (GUI).
2. **cv2**: OpenCV library for computer vision tasks.
3. **csv**: Used for reading and writing CSV files.
4. **os:** Operating system dependent functionality.
5. **numpy:** Used for numerical computations.
6. **PIL:** Python Imaging Library, used for opening, manipulating, and saving many different image file formats.
7. **pandas:** Used for data manipulation and analysis.
8. **datetime:** Used for manipulating dates and times.
9. **time:** Provides various time-related functions.
10. **pymysql:** Python MySQL database access library.

**MySQL**

MySQL is an open-source relational database management system (RDBMS) that is widely used for web-based applications. It's a popular choice among developers and organizations due to its reliability ease of use, and extensive community support.

**Here are some key features and aspects of MySQL:**

1. **Open Source:** MySQL is distributed under an open-source license, which means it's free to use, modify, and distribute. This makes it particularly attractive for startups, small businesses, and large enterprises alike.
2. **Relational Database:** MySQL is a relational database management system, meaning it organizes data into tables with rows and columns, and establishes relationships between them. This structure allows for efficient storage and retrieval of data.
3. **SQL Support:** MySQL uses Structured Query Language (SQL) for managing and manipulating data. SQL is a standard language for relational databases, making it widely understood and compatible with other systems.
4. **Scalability:** MySQL is designed to handle both small-scale and large-scale applications. It supports features like replication, clustering, and sharding, which allow you to scale your database horizontally and vertically as your application grows.
5. **Performance:** MySQL is known for its fast performance, especially when optimized properly. It supports various storage engines, each with its own performance characteristics, allowing you to choose the one that best fits your application's needs.
6. **Cross-Platform Compatibility:** MySQL runs on various operating systems, including Linux, Windows, and macOS. This makes it flexible and accessible for developers working in different environments.
7. **Community Support:** MySQL has a large and active community of developers, administrators, and users who contribute to its development, provide support, and share knowledge through forums, mailing lists, and online resources.
8. **Security:** MySQL offers robust security features to protect your data, including access controls, encryption, and auditing. It's important to follow best practices to ensure the security of your MySQL database.

**3.2 Steps Involved:**

1. **Planning:**

This initial phase involves determining what needs to be achieved with the software. For the Face Recognition Based Attendance Management System (FAMS), planning would include:

**Defining the scope:** Deciding what features the system should have, such as automatic attendance, manual attendance filling, and an admin panel.

1. **Gathering requirements:**

Identifying specific functionalities required, like capturing images, training the recognition model, and managing attendance records.

**Setting objectives:** Determining the goals of the system, such as improving attendance accuracy and efficiency.

1. **Analysis:**

In the analysis phase, the requirements gathered during planning are analyzed in detail. This involves:

* **Understanding user needs:** Identifying the needs of the users who will interact with the system, such as teachers, students, and administrators.
* **Assessing constraints:** Considering limitations like hardware compatibility, budget constraints, and time constraints.
* **Defining preferences:** Determining user preferences regarding system usability, performance, and features.

1. **Design:**

The design phase translates the requirements into a blueprint for the software. This includes:

* **Architectural design:** Deciding on the overall structure of the system, including how different components will interact.
* **User interface design:** Creating the graphical interface through which users will interact with the system, using tools like Tkinter in this case.
* **Detailed design:** Specifying how each feature will be implemented, such as capturing images using a webcam and storing them in a database.

1. **Implementation:**

In this phase, the actual code is written based on the design specifications. This involves:

* **Coding:** Writing the software using a programming language like Python.
* **Database setup:** Creating the necessary database tables and connections for storing data like images and attendance records.
* **Integration:** Combining different modules and components to create a working system.

1. **Testing:**

The testing phase ensures that the software functions correctly and meets the specified requirements. This includes:

* **Unit testing:** Testing individual components or modules in isolation to ensure they work as expected.
* **Integration testing:** Testing how different components work together to ensure they interact correctly.
* **System testing:** Testing the entire system as a whole to verify that it meets the requirements and functions as intended.

1. **Deployment:**

Once the software has been thoroughly tested and approved, it is deployed for use by end-users. This involves:

* **Distribution:** Making the software available to users, which could involve installing it on their computers or providing access via a web application.
* **Configuration:** Setting up the software to work in the user's environment, including any necessary customization or settings.

1. **Maintenance:**

After deployment, the software requires ongoing maintenance and support to ensure it remains functional and up-to-date. This includes:

* **Bug fixes:** Addressing any issues or problems that arise during use.
* **Updates:** Releasing new versions of the software to add features, improve performance, or address security vulnerabilities.
* **User support:** Providing assistance to users who encounter problems or have questions about how to use the software.

**4. Project Setup**

Face Recognition Based Attendance Management System implemented using Python with the help of various libraries such as tkinter, OpenCV, pandas, and pymysql. Here's a brief explanation of the project setup :

* **Main Window Setup:** The main window is created using tkinter, with a title "Face-Recognition-Based-Attendance-Management-System". It has buttons for various actions like taking images, training images, automatic attendance, manually filling attendance, and checking registered students.
* **Functionality Overview:**
* **take\_img():** This function is triggered when the "Take Images" button is clicked. It captures images from the webcam for enrollment and name provided by the user, and saves them in the "TrainingImage" folder.
* **trainimg():** This function trains the face recognition model using the images stored in the "TrainingImage" folder and saves the trained model as "Trainner.yml" in the "TrainingImageLabel" folder.
* **subjectchoose():** This function allows the user to select a subject and fill the attendance automatically using face recognition. It uses the trained model to recognize faces in real-time and records attendance accordingly.
* **manually\_fill():** This function allows the user to manually fill the attendance for a subject by entering enrollment and name for each student. The attendance data is then stored in a CSV file.
* **admin\_panel():** This function creates a login window for accessing registered student details. It checks the credentials and displays the student details in a separate window.
* **Database Integration:** The system seems to integrate with a MySQL database for storing attendance records. It creates tables for storing attendance data and inserts records into the database accordingly.
* **GUI Components:** Various GUI components like labels, entry fields, buttons, and notifications are used to create an interactive user interface for the system.

**Data Structures:**

Face Recognition Based Attendance Management System using Python's Tkinter library for GUI, OpenCV for face recognition, and CSV for data storage. Here are the main data structures:

* **Lists and Dictionaries:**
* **imagePaths:** A list containing the paths to the images used for training the face recognition model.
* **faceSamples:** A list containing the cropped face images used for training.
* **Ids:** A list containing corresponding IDs (labels) for the face images.
* **attendance:** A pandas DataFrame storing attendance records with columns for enrollment ID, name, date, and time.
* **CSV Files:**

StudentDetails.csv: CSV file storing student details including enrollment ID, name, date, and time.

Generated CSV files for attendance records, named based on the subject, date, and time of the attendance session.

* **Database Tables:**

Tables created in a MySQL database for storing attendance records, with columns for enrollment ID, name, date, and time.

* **Model File:**
* **Trainner.yml:** A YAML file storing the trained LBPH (Local Binary Patterns Histograms) face recognition model.
* **Window and GUI Components:**

Various Tkinter window objects like sb, MFW, windo, win, etc., for different GUI screens and components like buttons, labels, and entry fields.

* **OpenCV Objects:**
* **cam:** OpenCV VideoCapture object for accessing the webcam.
* **recognizer:** LBPH face recognizer object from OpenCV for training and recognizing faces.
* **detector:** OpenCV Cascade Classifier object for detecting faces in images.

1. **Computer System:**

A computer system is a basic requirement to set up the Face Recognition System. The computer system should have the following minimum configuration:

Processor: Intel Core i3 or higher

RAM: 4 GB or higher

Hard Disk: 250 GB or higher

Operating System: Windows 7 or higher, or Linux

1. **Python Programming Language:**

Python is a popular programming language that is used for a wide range of applications. Python is easy to learn, and it has a large community of developers who contribute to its development. To set up the supermarket navigation system, we need to install Python on our computer system. We can download the latest version of Python from the official website and follow the installation instructions.

1. **Tkinter:**

Tkinter is a standard GUI (Graphical User Interface) library in Python. It provides functions and classes for creating desktop applications with a graphical interface. In this code, tkinter is used to create the GUI for the attendance management system, including buttons, labels, and entry fields.

1. **cv2 (OpenCV):**

OpenCV (Open Source Computer Vision Library) is a popular library for computer vision and image processing tasks. In this code, cv2 is primarily used for face detection and recognition. It provides functions to detect faces in images or video streams and recognize faces based on pre-trained models or user-provided data.

1. **csv:**

The csv module provides functionality for reading and writing CSV (Comma-Separated Values) files. In this code, it's used to read and write attendance data to a CSV file. CSV files are commonly used for storing tabular data, where each row represents a record and columns represent different attributes or fields.

1. **os:**

The os module provides functions for interacting with the operating system, such as file operations (e.g., file manipulation, directory traversal) and environment variables. In this code, os is likely used for file handling tasks, such as checking file existence, creating directories, or accessing file paths.

1. **numpy (np):**

NumPy is a fundamental package for numerical computing in Python. It provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays efficiently. In this code, numpy may be used for processing image data or performing numerical computations related to face recognition.

1. **PIL:**

PIL (Python Imaging Library), now known as Pillow, is a library for opening, manipulating, and saving many different image file formats. It provides functions for basic image processing tasks, such as resizing, cropping, and converting between different image formats. In this code, PIL might be used for loading and preprocessing images before face detection or recognition.

1. **pandas (pd):**

Pandas is a powerful library for data manipulation and analysis in Python. It provides data structures like DataFrame and Series, along with functions for reading, writing, and analyzing structured data. In this code, pandas could be used for handling attendance data in a more structured and efficient manner, compared to standard Python data structures.

**Implementation**

1. **Imports:** The code imports necessary libraries such as tkinter for creating the GUI, cv2 for computer vision tasks, csv for handling CSV files, os for interacting with the operating system, numpy for numerical computations, PIL for image processing tasks, pandas for data manipulation, and datetime and time for handling date and time operations.
2. **Global Variables:** Global variables like Face\_reco\_fill and manually\_fill\_attendance are defined for database table names.
3. **Window Initialization:** The main window for the Face Recognition Based Attendance Management System (FAMS) is created using tkinter.
4. **Functions:**

* **manually\_fill:** Opens a window to manually fill attendance.
* err\_screen\_for\_subject, err\_screen1: Functions to display error messages for missing subject names.
* **fill\_attendance:** Allows manual entry of attendance data into the database and CSV file.
* **create\_csv:** Generates a CSV file for the manually filled attendance.
* **clear, clear1:** Functions to clear text entry fields.
* err\_screen, err\_screen1: Functions to display error messages for missing enrollment or name.
* **subjectchoose:** Allows the user to select a subject and fill attendance automatically using face recognition.
* **admin\_panel:** Provides a login panel for administrators to view registered students.
* **on\_closing:** Handles the closing event of the main window.

1. **GUI Elements:**

Labels, Entry fields, and Buttons for user interaction.

Messages and notifications displayed in the GUI.

1. **Face Recognition and Attendance Handling:**

Functions like take\_img, trainimg, and getImagesAndLabels are used for capturing images, training the face recognition model, and extracting face images and labels.

Automatic attendance filling and manual attendance filling functionalities are provided.

Database operations are performed for storing attendance data.

**Code:**

import tkinter as tk

from tkinter import \*

import cv2

import csv

import os

import numpy as np

from PIL import Image, ImageTk

import pandas as pd

import datetime

import time

Face\_reco\_fill = 'face\_reco\_fill'

manually\_fill\_attendance='manually\_fill\_attendance'

# Window is our Main frame of system

window = tk.Tk()

window.title("FAMS-Face Recognition Based Attendance Management System")

window.geometry('1280x720')

window.configure(background='grey80')

# GUI for manually fill attendance

def manually\_fill():

global sb

sb = tk.Tk()

# sb.iconbitmap('AMS.ico')

sb.title("Enter subject name...")

sb.geometry('580x320')

sb.configure(background='grey80')

def err\_screen\_for\_subject():

def ec\_delete():

ec.destroy()

global ec

ec = tk.Tk()

ec.geometry('300x100')

# ec.iconbitmap('AMS.ico')

ec.title('Warning!!')

ec.configure(background='snow')

Label(ec, text='Please enter your subject name!!!', fg='red',

bg='white', font=('times', 16, ' bold ')).pack()

Button(ec, text='OK', command=ec\_delete, fg="black", bg="lawn green", width=9, height=1, activebackground="Red",

font=('times', 15, ' bold ')).place(x=90, y=50)

def fill\_attendance():

ts = time.time()

Date = datetime.datetime.fromtimestamp(ts).strftime('%Y\_%m\_%d')

timeStamp = datetime.datetime.fromtimestamp(ts).strftime('%H:%M:%S')

Time = datetime.datetime.fromtimestamp(ts).strftime('%H:%M:%S')

Hour, Minute, Second = timeStamp.split(":")

# Creatting csv of attendance

# Create table for Attendance

date\_for\_DB = datetime.datetime.fromtimestamp(ts).strftime('%Y\_%m\_%d')

global subb

subb = SUB\_ENTRY.get()

DB\_table\_name = str(subb + "\_" + Date + "\_Time\_" +

Hour + "\_" + Minute + "\_" + Second)

import pymysql.connections

# Connect to the database

try:

global cursor

connection = pymysql.connect(

host='localhost', user='root', password='123456', db='manually\_fill\_attendance')

cursor = connection.cursor()

except Exception as e:

print(e)

sql = "Attand " + manually\_fill\_attendance + """

(ID INT NOT NULL AUTO\_INCREMENT,

ENROLLMENT varchar(100) NOT NULL,

NAME VARCHAR(50) NOT NULL,

DATE VARCHAR(20) NOT NULL,

TIME VARCHAR(20) NOT NULL,

PRIMARY KEY (ID)

);

"""

try:

cursor.execute(sql) # for create a table

except Exception as ex:

print(ex) #

if subb == '':

err\_screen\_for\_subject()

else:

sb.destroy()

MFW = tk.Tk()

# MFW.iconbitmap('AMS.ico')

MFW.title("Manually attendance of " + str(subb))

MFW.geometry('880x470')

MFW.configure(background='grey80')

def del\_errsc2():

errsc2.destroy()

def err\_screen1():

global errsc2

errsc2 = tk.Tk()

errsc2.geometry('330x100')

# errsc2.iconbitmap('AMS.ico')

errsc2.title('Warning!!')

errsc2.configure(background='grey80')

Label(errsc2, text='Please enter Student & Enrollment!!!', fg='black', bg='white',

font=('times', 16, ' bold ')).pack()

Button(errsc2, text='OK', command=del\_errsc2, fg="black", bg="lawn green", width=9, height=1,

activebackground="Red", font=('times', 15, ' bold ')).place(x=90, y=50)

def testVal(inStr, acttyp):

if acttyp == '1': # insert

if not inStr.isdigit():

return False

return True

ENR = tk.Label(MFW, text="Enter Enrollment", width=15, height=2, fg="black", bg="grey",

font=('times', 15))

ENR.place(x=30, y=100)

STU\_NAME = tk.Label(MFW, text="Enter Student name", width=15, height=2, fg="black", bg="grey",

font=('times', 15))

STU\_NAME.place(x=30, y=200)

global ENR\_ENTRY

ENR\_ENTRY = tk.Entry(MFW, width=20, validate='key',

bg="white", fg="black", font=('times', 23))

ENR\_ENTRY['validatecommand'] = (

ENR\_ENTRY.register(testVal), '%P', '%d')

ENR\_ENTRY.place(x=290, y=105)

def remove\_enr():

ENR\_ENTRY.delete(first=0, last=22)

STUDENT\_ENTRY = tk.Entry(

MFW, width=20, bg="white", fg="black", font=('times', 23))

STUDENT\_ENTRY.place(x=290, y=205)

def remove\_student():

STUDENT\_ENTRY.delete(first=0, last=22)

# get important variable

def enter\_data\_DB():

ENROLLMENT = ENR\_ENTRY.get()

STUDENT = STUDENT\_ENTRY.get()

if ENROLLMENT == '':

err\_screen1()

elif STUDENT == '':

err\_screen1()

else:

time = datetime.datetime.fromtimestamp(

ts).strftime('%H:%M:%S')

Hour, Minute, Second = time.split(":")

Insert\_data = "INSERT INTO " + DB\_table\_name + \

" (ID,ENROLLMENT,NAME,DATE,TIME) VALUES (0, %s, %s, %s,%s)"

VALUES = (str(ENROLLMENT), str(

STUDENT), str(Date), str(time))

try:

cursor.execute(Insert\_data, VALUES)

except Exception as e:

print(e)

ENR\_ENTRY.delete(first=0, last=22)

STUDENT\_ENTRY.delete(first=0, last=22)

def create\_csv():

import csv

cursor.execute("select \* from " + DB\_table\_name + ";")

csv\_name = 'C:\\Users\\welcome\\Desktop\\face\\Face-Recognition-Attendance-System-main\\Attendance\\Manually Attendance+'.csv

with open(csv\_name, "w") as csv\_file:

csv\_writer = csv.writer(csv\_file)

csv\_writer.writerow(

[i[0] for i in cursor.description]) # write headers

csv\_writer.writerows(cursor)

O = "CSV created Successfully"

Notifi.configure(text=O, bg="Green", fg="white",

width=33, font=('times', 19, 'bold'))

Notifi.place(x=180, y=380)

import csv

import tkinter

root = tkinter.Tk()

root.title("Attendance of " + subb)

root.configure(background='grey80')

with open(csv\_name, newline="") as file:

reader = csv.reader(file)

r = 0

for col in reader:

c = 0

for row in col:

# i've added some styling

label = tkinter.Label(root, width=18, height=1, fg="black", font=('times', 13, ' bold '),

bg="white", text=row, relief=tkinter.RIDGE)

label.grid(row=r, column=c)

c += 1

r += 1

root.mainloop()

Notifi = tk.Label(MFW, text="CSV created Successfully", bg="Green", fg="white", width=33,

height=2, font=('times', 19, 'bold'))

c1ear\_enroll = tk.Button(MFW, text="Clear", command=remove\_enr, fg="white", bg="black", width=10,

height=1,

activebackground="white", font=('times', 15, ' bold '))

c1ear\_enroll.place(x=690, y=100)

c1ear\_student = tk.Button(MFW, text="Clear", command=remove\_student, fg="white", bg="black", width=10,

height=1,

activebackground="white", font=('times', 15, ' bold '))

c1ear\_student.place(x=690, y=200)

DATA\_SUB = tk.Button(MFW, text="Enter Data", command=enter\_data\_DB, fg="black", bg="SkyBlue1", width=20,

height=2,

activebackground="white", font=('times', 15, ' bold '))

DATA\_SUB.place(x=170, y=300)

MAKE\_CSV = tk.Button(MFW, text="Convert to CSV", command=create\_csv, fg="black", bg="SkyBlue1", width=20,

height=2,

activebackground="white", font=('times', 15, ' bold '))

MAKE\_CSV.place(x=570, y=300)

def attf():

import subprocess

subprocess.Popen(

r'explorer /select,"C:\Users\Pragya Singh\PycharmProjects\Attendace\_management\_system\Attendance\Manually Attendance\-------Check atttendance-------"')

attf = tk.Button(MFW, text="Check Sheets", command=attf, fg="white", bg="black",

width=12, height=1, activebackground="white", font=('times', 14, ' bold '))

attf.place(x=730, y=410)

MFW.mainloop()

SUB = tk.Label(sb, text="Enter Subject : ", width=15, height=2,

fg="black", bg="grey80", font=('times', 15, ' bold '))

SUB.place(x=30, y=100)

global SUB\_ENTRY

SUB\_ENTRY = tk.Entry(sb, width=20, bg="white",

fg="black", font=('times', 23))

SUB\_ENTRY.place(x=250, y=105)

fill\_manual\_attendance = tk.Button(sb, text="Fill Attendance", command=fill\_attendance, fg="black", bg="SkyBlue1", width=20, height=2,

activebackground="white", font=('times', 15, ' bold '))

fill\_manual\_attendance.place(x=250, y=160)

sb.mainloop()

# For clear textbox

def clear():

txt.delete(first=0, last=22)

def clear1():

txt2.delete(first=0, last=22)

def del\_sc1():

sc1.destroy()

def err\_screen():

global sc1

sc1 = tk.Tk()

sc1.geometry('300x100')

# sc1.iconbitmap('AMS.ico')

sc1.title('Warning!!')

sc1.configure(background='grey80')

Label(sc1, text='Enrollment & Name required!!!', fg='black',

bg='white', font=('times', 16)).pack()

Button(sc1, text='OK', command=del\_sc1, fg="black", bg="lawn green", width=9,

height=1, activebackground="Red", font=('times', 15, ' bold ')).place(x=90, y=50)

# Error screen2

def del\_sc2():

sc2.destroy()

def err\_screen1():

global sc2

sc2 = tk.Tk()

sc2.geometry('300x100')

# sc2.iconbitmap('AMS.ico')

sc2.title('Warning!!')

sc2.configure(background='grey80')

Label(sc2, text='Please enter your subject name!!!', fg='black',

bg='white', font=('times', 16)).pack()

Button(sc2, text='OK', command=del\_sc2, fg="black", bg="lawn green", width=9,

height=1, activebackground="Red", font=('times', 15, ' bold ')).place(x=90, y=50)

# For take images for datasets

def take\_img():

l1 = txt.get()

l2 = txt2.get()

if l1 == '':

err\_screen()

elif l2 == '':

err\_screen()

else:

try:

cam = cv2.VideoCapture(0)

detector = cv2.CascadeClassifier(

'haarcascade\_frontalface\_default.xml')

Enrollment = txt.get()

Name = txt2.get()

sampleNum = 0

while (True):

ret, img = cam.read()

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

faces = detector.detectMultiScale(gray, 1.3, 5)

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

cv2.rectangle(img, (x, y), (x + w, y + h), (255, 0, 0), 2)

# incrementing sample number

sampleNum = sampleNum + 1

# saving the captured face in the dataset folder

cv2.imwrite("TrainingImage/ " + Name + "." + Enrollment + '.' + str(sampleNum) + ".jpg",

gray)

print("Images Saved for Enrollment :")

cv2.imshow('Frame', img)

# wait for 100 miliseconds

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

break

#

# # break if the sample number is morethan 100

elif sampleNum > 70:

break

cam.release()

cv2.destroyAllWindows()

ts = time.time()

Date = datetime.datetime.fromtimestamp(ts).strftime('%Y-%m-%d')

Time = datetime.datetime.fromtimestamp(ts).strftime('%H:%M:%S')

row = [Enrollment, Name, Date, Time]

with open('StudentDetails\StudentDetails.csv', 'a+') as csvFile:

writer = csv.writer(csvFile, delimiter=',')

writer.writerow(row)

csvFile.close()

res = "Images Saved for Enrollment : " + Enrollment + " Name : " + Name

Notification.configure(

text=res, bg="SpringGreen3", width=50, font=('times', 18, 'bold'))

Notification.place(x=250, y=400)

except FileExistsError as F:

f = 'Student Data already exists'

Notification.configure(text=f, bg="Red", width=21)

Notification.place(x=450, y=400)

# for choose subject and fill attendance

def subjectchoose():

def Fillattendances():

sub = tx.get()

now = time.time() # For calculate seconds of video

future = now + 20

if time.time() < future:

if sub == '':

err\_screen1()

else:

recognizer = cv2.face.LBPHFaceRecognizer\_create() # cv2.createLBPHFaceRecognizer()

try:

recognizer.read("TrainingImageLabel\Trainner.yml")

except:

e = 'Model not found,Please train model'

Notifica.configure(

text=e, bg="red", fg="black", width=33, font=('times', 15, 'bold'))

Notifica.place(x=20, y=250)

harcascadePath = "haarcascade\_frontalface\_default.xml"

faceCascade = cv2.CascadeClassifier(harcascadePath)

df = pd.read\_csv("StudentDetails\StudentDetails.csv")

cam = cv2.VideoCapture(0)

font = cv2.FONT\_HERSHEY\_SIMPLEX

col\_names = ['Enrollment', 'Name', 'Date', 'Time']

attendance = pd.DataFrame(columns=col\_names)

while True:

ret, im = cam.read()

gray = cv2.cvtColor(im, cv2.COLOR\_BGR2GRAY)

faces = faceCascade.detectMultiScale(gray, 1.2, 5)

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

global Id

Id, conf = recognizer.predict(gray[y:y + h, x:x + w])

if (conf < 70):

print(conf)

global Subject

global aa

global date

global timeStamp

Subject = tx.get()

ts = time.time()

date = datetime.datetime.fromtimestamp(

ts).strftime('%Y-%m-%d')

timeStamp = datetime.datetime.fromtimestamp(

ts).strftime('%H:%M:%S')

aa = df.loc[df['Enrollment'] == Id]['Name'].values

global tt

tt = str(Id) + "-" + aa

En = '15624031' + str(Id)

attendance.loc[len(attendance)] = [

Id, aa, date, timeStamp]

cv2.rectangle(

im, (x, y), (x + w, y + h), (0, 260, 0), 7)

cv2.putText(im, str(tt), (x + h, y),

font, 1, (255, 255, 0,), 4)

else:

Id = 'Unknown'

tt = str(Id)

cv2.rectangle(

im, (x, y), (x + w, y + h), (0, 25, 255), 7)

cv2.putText(im, str(tt), (x + h, y),

font, 1, (0, 25, 255), 4)

if time.time() > future:

break

attendance = attendance.drop\_duplicates(

['Enrollment'], keep='first')

cv2.imshow('Filling attedance..', im)

key = cv2.waitKey(30) & 0xff

if key == 27:

break

ts = time.time()

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

timeStamp = datetime.datetime.fromtimestamp(

ts).strftime('%H:%M:%S')

Hour, Minute, Second = timeStamp.split(":")

fileName = "Attendance/" + Subject + "\_" + date + \

"\_" + Hour + "-" + Minute + "-" + Second + ".csv"

attendance = attendance.drop\_duplicates(

['Enrollment'], keep='first')

print(attendance)

attendance.to\_csv(fileName, index=False)

# Create table for Attendance

date\_for\_DB = datetime.datetime.fromtimestamp(

ts).strftime('%Y\_%m\_%d')

DB\_Table\_name = str(

Subject + "\_" + date\_for\_DB + "\_Time\_" + Hour + "\_" + Minute + "\_" + Second)

import pymysql.connections

# Connect to the database

try:

global cursor

connection = pymysql.connect(

host='localhost', user='root', password='12345', db='Face\_reco\_fill')

cursor = connection.cursor()

except Exception as e:

print(e)

sql = "CREATE TABLE " + Face\_reco\_fill + """

(ID INT NOT NULL AUTO\_INCREMENT,

ENROLLMENT varchar(100) NOT NULL,

NAME VARCHAR(50) NOT NULL,

DATE VARCHAR(20) NOT NULL,

TIME VARCHAR(20) NOT NULL,

PRIMARY KEY (ID)

);

"""

# Now enter attendance in Database

insert\_data = "INSERT INTO " + DB\_Table\_name + \

" (ID,ENROLLMENT,NAME,DATE,TIME) VALUES (0, %s, %s, %s,%s)"

VALUES = (str(Id), str(aa), str(date), str(timeStamp))

try:

cursor.execute(sql) # for create a table

# For insert data into table

cursor.execute(insert\_data, VALUES)

except Exception as ex:

print(ex) #

M = 'Attendance filled Successfully'

Notifica.configure(text=M, bg="Green", fg="white",

width=33, font=('times', 15, 'bold'))

Notifica.place(x=20, y=250)

cam.release()

cv2.destroyAllWindows()

import csv

import tkinter

root = tkinter.Tk()

root.title("Attendance of " + Subject)

root.configure(background='grey80')

cs = 'C:/Users/Pragya Singh/PycharmProjects/Attendace\_management\_system/' + fileName

with open(cs, newline="") as file:

reader = csv.reader(file)

r = 0

for col in reader:

c = 0

for row in col:

# i've added some styling

label = tkinter.Label(root, width=10, height=1, fg="black", font=('times', 15, ' bold '),

bg="white", text=row, relief=tkinter.RIDGE)

label.grid(row=r, column=c)

c += 1

r += 1

root.mainloop()

print(attendance)

# windo is frame for subject chooser

windo = tk.Tk()

# windo.iconbitmap('AMS.ico')

windo.title("Enter subject name...")

windo.geometry('580x320')

windo.configure(background='grey80')

Notifica = tk.Label(windo, text="Attendance filled Successfully", bg="Green", fg="white", width=33,

height=2, font=('times', 15, 'bold'))

def Attf():

import subprocess

subprocess.Popen(

r'explorer /select,"C:\Users\Pragya Singh\PycharmProjects\Attendace\_management\_system\Attendance\-------Check atttendance-------"')

attf = tk.Button(windo, text="Check Sheets", command=Attf, fg="white", bg="black",

width=12, height=1, activebackground="white", font=('times', 14, ' bold '))

attf.place(x=430, y=255)

sub = tk.Label(windo, text="Enter Subject : ", width=15, height=2,

fg="black", bg="grey", font=('times', 15, ' bold '))

sub.place(x=30, y=100)

tx = tk.Entry(windo, width=20, bg="white",

fg="black", font=('times', 23))

tx.place(x=250, y=105)

fill\_a = tk.Button(windo, text="Fill Attendance", fg="white", command=Fillattendances, bg="SkyBlue1", width=20, height=2,

activebackground="white", font=('times', 15, ' bold '))

fill\_a.place(x=250, y=160)

windo.mainloop()

def admin\_panel():

win = tk.Tk()

# win.iconbitmap('AMS.ico')

win.title("LogIn")

win.geometry('880x420')

win.configure(background='grey80')

def log\_in():

username = un\_entr.get()

password = pw\_entr.get()

if username == 'pragya':

if password == 'pragya123':

win.destroy()

import csv

import tkinter

root = tkinter.Tk()

root.title("Student Details")

root.configure(background='grey80')

cs = 'C:/Users/Pragya Singh/PycharmProjects/Attendace\_management\_system/StudentDetails/StudentDetails.csv'

with open(cs, newline="") as file:

reader = csv.reader(file)

r = 0

for col in reader:

c = 0

for row in col:

# i've added some styling

label = tkinter.Label(root, width=10, height=1, fg="black", font=('times', 15, ' bold '),

bg="white", text=row, relief=tkinter.RIDGE)

label.grid(row=r, column=c)

c += 1

r += 1

root.mainloop()

else:

valid = 'Incorrect ID or Password'

Nt.configure(text=valid, bg="red", fg="white",

width=38, font=('times', 19, 'bold'))

Nt.place(x=120, y=350)

else:

valid = 'Incorrect ID or Password'

Nt.configure(text=valid, bg="red", fg="white",

width=38, font=('times', 19, 'bold'))

Nt.place(x=120, y=350)

Nt = tk.Label(win, text="Attendance filled Successfully", bg="Green", fg="white", width=40,

height=2, font=('times', 19, 'bold'))

# Nt.place(x=120, y=350)

un = tk.Label(win, text="Enter username : ", width=15, height=2, fg="black", bg="grey",

font=('times', 15, ' bold '))

un.place(x=30, y=50)

pw = tk.Label(win, text="Enter password : ", width=15, height=2, fg="black", bg="grey",

font=('times', 15, ' bold '))

pw.place(x=30, y=150)

def c00():

un\_entr.delete(first=0, last=22)

un\_entr = tk.Entry(win, width=20, bg="white", fg="black",

font=('times', 23))

un\_entr.place(x=290, y=55)

def c11():

pw\_entr.delete(first=0, last=22)

pw\_entr = tk.Entry(win, width=20, show="\*", bg="white",

fg="black", font=('times', 23))

pw\_entr.place(x=290, y=155)

c0 = tk.Button(win, text="Clear", command=c00, fg="white", bg="black", width=10, height=1,

activebackground="white", font=('times', 15, ' bold '))

c0.place(x=690, y=55)

c1 = tk.Button(win, text="Clear", command=c11, fg="white", bg="black", width=10, height=1,

activebackground="white", font=('times', 15, ' bold '))

c1.place(x=690, y=155)

Login = tk.Button(win, text="LogIn", fg="black", bg="SkyBlue1", width=20,

height=2,

activebackground="Red", command=log\_in, font=('times', 15, ' bold '))

Login.place(x=290, y=250)

win.mainloop()

# For train the model

def trainimg():

recognizer = cv2.face.LBPHFaceRecognizer\_create()

global detector

detector = cv2.CascadeClassifier("haarcascade\_frontalface\_default.xml")

try:

global faces, Id

faces, Id = getImagesAndLabels("TrainingImage")

except Exception as e:

l = 'please make "TrainingImage" folder & put Images'

Notification.configure(text=l, bg="SpringGreen3",

width=50, font=('times', 18, 'bold'))

Notification.place(x=350, y=400)

recognizer.train(faces, np.array(Id))

try:

recognizer.save("TrainingImageLabel\Trainner.yml")

except Exception as e:

q = 'Please make "TrainingImageLabel" folder'

Notification.configure(text=q, bg="SpringGreen3",

width=50, font=('times', 18, 'bold'))

Notification.place(x=350, y=400)

res = "Model Trained" # +",".join(str(f) for f in Id)

Notification.configure(text=res, bg="olive drab",

width=50, font=('times', 18, 'bold'))

Notification.place(x=250, y=400)

def getImagesAndLabels(path):

imagePaths = [os.path.join(path, f) for f in os.listdir(path)]

# create empth face list

faceSamples = []

# create empty ID list

Ids = []

# now looping through all the image paths and loading the Ids and the images

for imagePath in imagePaths:

# loading the image and converting it to gray scale

pilImage = Image.open(imagePath).convert('L')

# Now we are converting the PIL image into numpy array

imageNp = np.array(pilImage, 'uint8')

# getting the Id from the image

Id = int(os.path.split(imagePath)[-1].split(".")[1])

# extract the face from the training image sample

faces = detector.detectMultiScale(imageNp)

# If a face is there then append that in the list as well as Id of it

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

faceSamples.append(imageNp[y:y + h, x:x + w])

Ids.append(Id)

return faceSamples, Ids

window.grid\_rowconfigure(0, weight=1)

window.grid\_columnconfigure(0, weight=1)

# window.iconbitmap('AMS.ico')

def on\_closing():

from tkinter import messagebox

if messagebox.askokcancel("Quit", "Do you want to quit?"):

window.destroy()

window.protocol("WM\_DELETE\_WINDOW", on\_closing)

message = tk.Label(window, text="Face-Recognition-Based-Attendance-Management-System", bg="black", fg="white", width=50,

height=3, font=('times', 30, ' bold '))

message.place(x=80, y=20)

Notification = tk.Label(window, text="All things good", bg="Green", fg="white", width=15,

height=3, font=('times', 17))

lbl = tk.Label(window, text="Enter Enrollment : ", width=20, height=2,

fg="black", bg="grey", font=('times', 15, 'bold'))

lbl.place(x=200, y=200)

def testVal(inStr, acttyp):

if acttyp == '1': # insert

if not inStr.isdigit():

return False

return True

txt = tk.Entry(window, validate="key", width=20, bg="white",

fg="black", font=('times', 25))

txt['validatecommand'] = (txt.register(testVal), '%P', '%d')

txt.place(x=550, y=210)

lbl2 = tk.Label(window, text="Enter Name : ", width=20, fg="black",

bg="grey", height=2, font=('times', 15, ' bold '))

lbl2.place(x=200, y=300)

txt2 = tk.Entry(window, width=20, bg="white",

fg="black", font=('times', 25))

txt2.place(x=550, y=310)

clearButton = tk.Button(window, text="Clear", command=clear, fg="white", bg="black",

width=10, height=1, activebackground="white", font=('times', 15, ' bold '))

clearButton.place(x=950, y=210)

clearButton1 = tk.Button(window, text="Clear", command=clear1, fg="white", bg="black",

width=10, height=1, activebackground="white", font=('times', 15, ' bold '))

clearButton1.place(x=950, y=310)

AP = tk.Button(window, text="Check Registered students", command=admin\_panel, fg="black",

bg="SkyBlue1", width=19, height=1, activebackground="white", font=('times', 15, ' bold '))

AP.place(x=990, y=410)

takeImg = tk.Button(window, text="Take Images", command=take\_img, fg="black", bg="SkyBlue1",

width=20, height=3, activebackground="white", font=('times', 15, ' bold '))

takeImg.place(x=90, y=500)

trainImg = tk.Button(window, text="Train Images", fg="black", command=trainimg, bg="SkyBlue1",

width=20, height=3, activebackground="white", font=('times', 15, ' bold '))

trainImg.place(x=390, y=500)

FA = tk.Button(window, text="Automatic Attendance", fg="black", command=subjectchoose,

bg="SkyBlue1", width=20, height=3, activebackground="white", font=('times', 15, ' bold '))

FA.place(x=690, y=500)

quitWindow = tk.Button(window, text="Manually Fill Attendance", command=manually\_fill, fg="black",

bg="SkyBlue1", width=20, height=3, activebackground="white", font=('times', 15, ' bold '))

quitWindow.place(x=990, y=500)

window.mainloop()

import cv2

import numpy as np

recognizer = cv2.createLBPHFaceRecognizer()

recognizer.read('TrainingImageLabel/trainner.yml')

cascadePath = "haarcascade\_frontalface\_default.xml"

faceCascade = cv2.CascadeClassifier(cascadePath)

font = cv2.FONT\_HERSHEY\_SIMPLEX

cam = cv2.VideoCapture(0)

while True:

ret, im = cam.read()

gray = cv2.cvtColor(im, cv2.COLOR\_BGR2GRAY)

faces = faceCascade.detectMultiScale(gray, 1.2, 5)

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

Id, conf = recognizer.predict(gray[y:y+h, x:x+w])

# # else:

# # Id="Unknown"

# cv2.rectangle(im, (x-22,y-90), (x+w+22, y-22), (0,255,0), -1)

cv2.rectangle(im, (x, y), (x + w, y + h), (0, 260, 0), 7)

cv2.putText(im, str(Id), (x, y-40), font, 2, (255, 255, 255), 3)

# cv2.putText(im, str(Id), (x + h, y), font, 1, (0, 260, 0), 2)

cv2.imshow('im', im)

if cv2.waitKey(10) & 0xFF == ord('q'):

break

cam.release()

cv2.destroyAllWindows()

import cv2

import os

import numpy as np

from PIL import Image

#

# recognizer = cv2.face.LBPHFaceRecognizer\_create()

recognizer = cv2.face.LBPHFaceRecognizer\_create()

detector = cv2.CascadeClassifier("haarcascade\_frontalface\_default.xml")

def getImagesAndLabels(path):

# get the path of all the files in the folder

imagePaths = [os.path.join(path, f) for f in os.listdir(path)]

# create empth face list

faceSamples = []

# create empty ID list

Ids = []

# now looping through all the image paths and loading the Ids and the images

for imagePath in imagePaths:

# loading the image and converting it to gray scale

pilImage = Image.open(imagePath).convert('L')

# Now we are converting the PIL image into numpy array

imageNp = np.array(pilImage, 'uint8')

# getting the Id from the image

Id = int(os.path.split(imagePath)[-1].split(".")[1])

# extract the face from the training image sample

faces = detector.detectMultiScale(imageNp)

# If a face is there then append that in the list as well as Id of it

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

faceSamples.append(imageNp[y:y+h, x:x+w])

Ids.append(Id)

return faceSamples, Ids

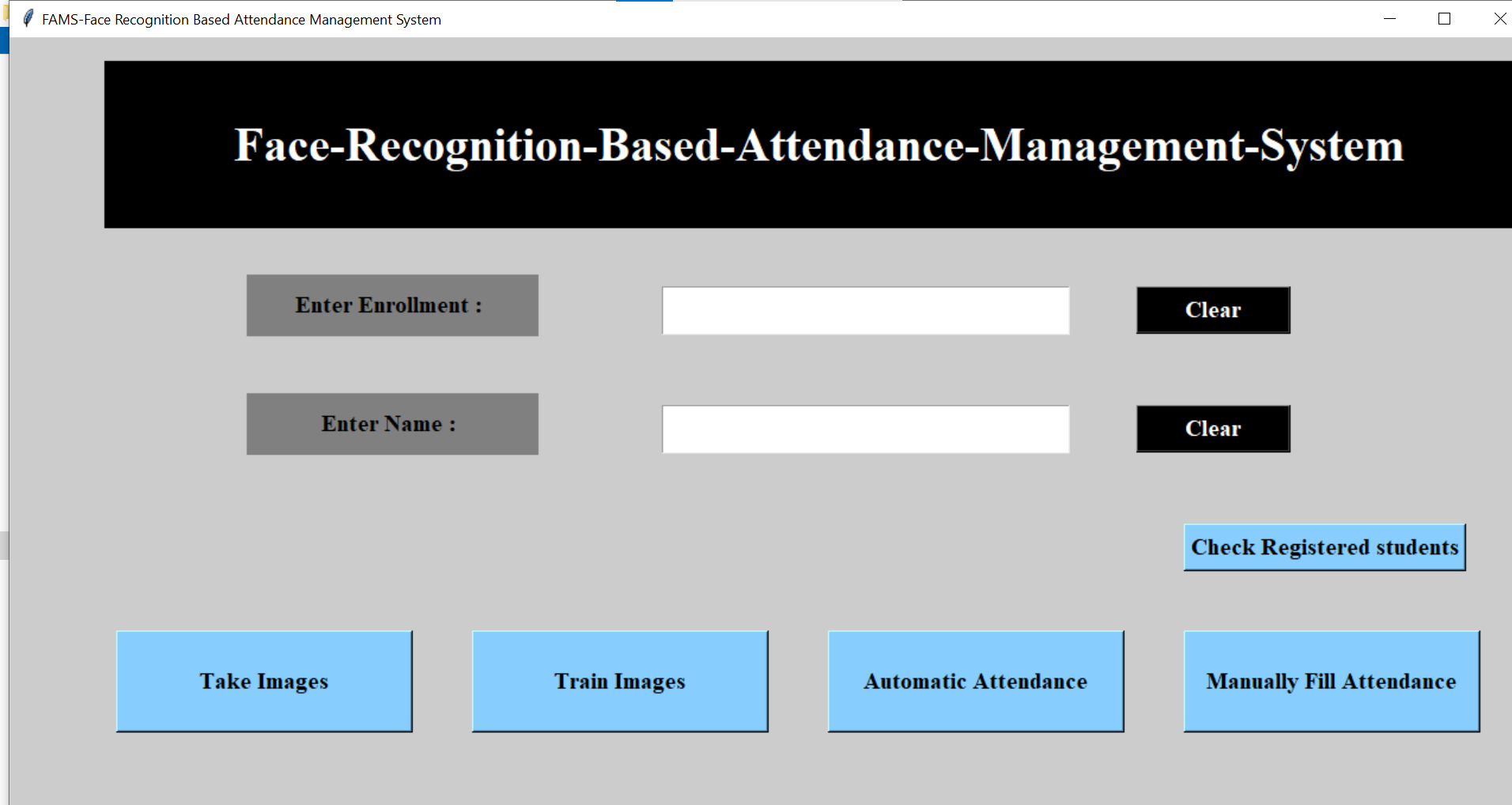
faces, Ids = getImagesAndLabels('TrainingImage')

recognizer.train(faces, np.array(Ids))

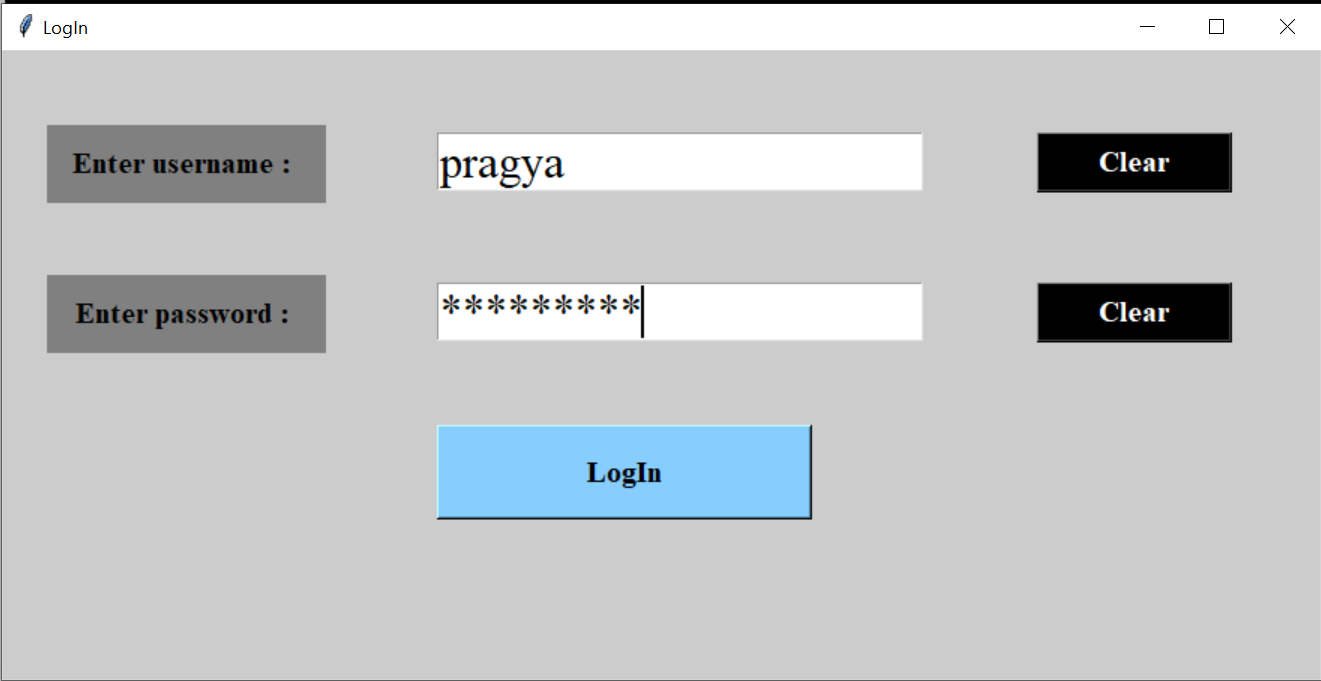
recognizer.save('TrainingImageLabel/trainner.yml')

**7. Screenshots of the Project:**

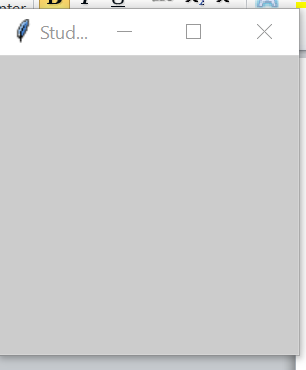
**Figure 1**



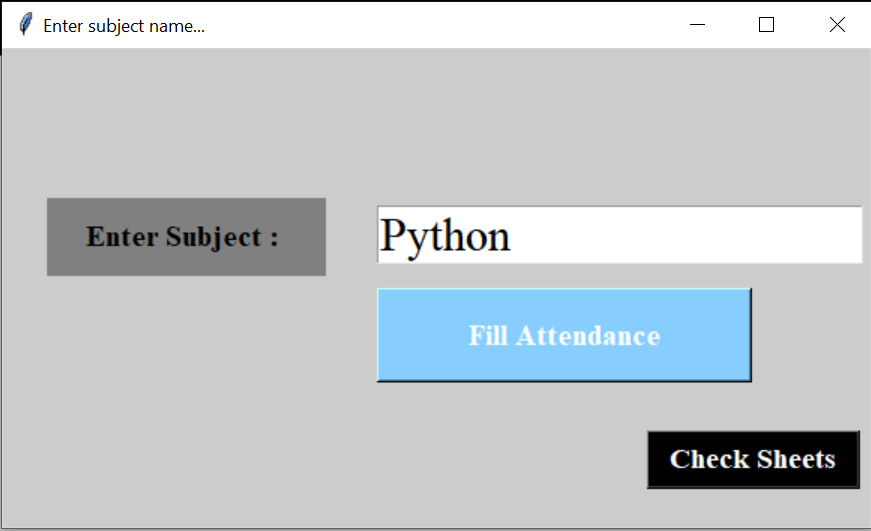
**Figure 2**

****

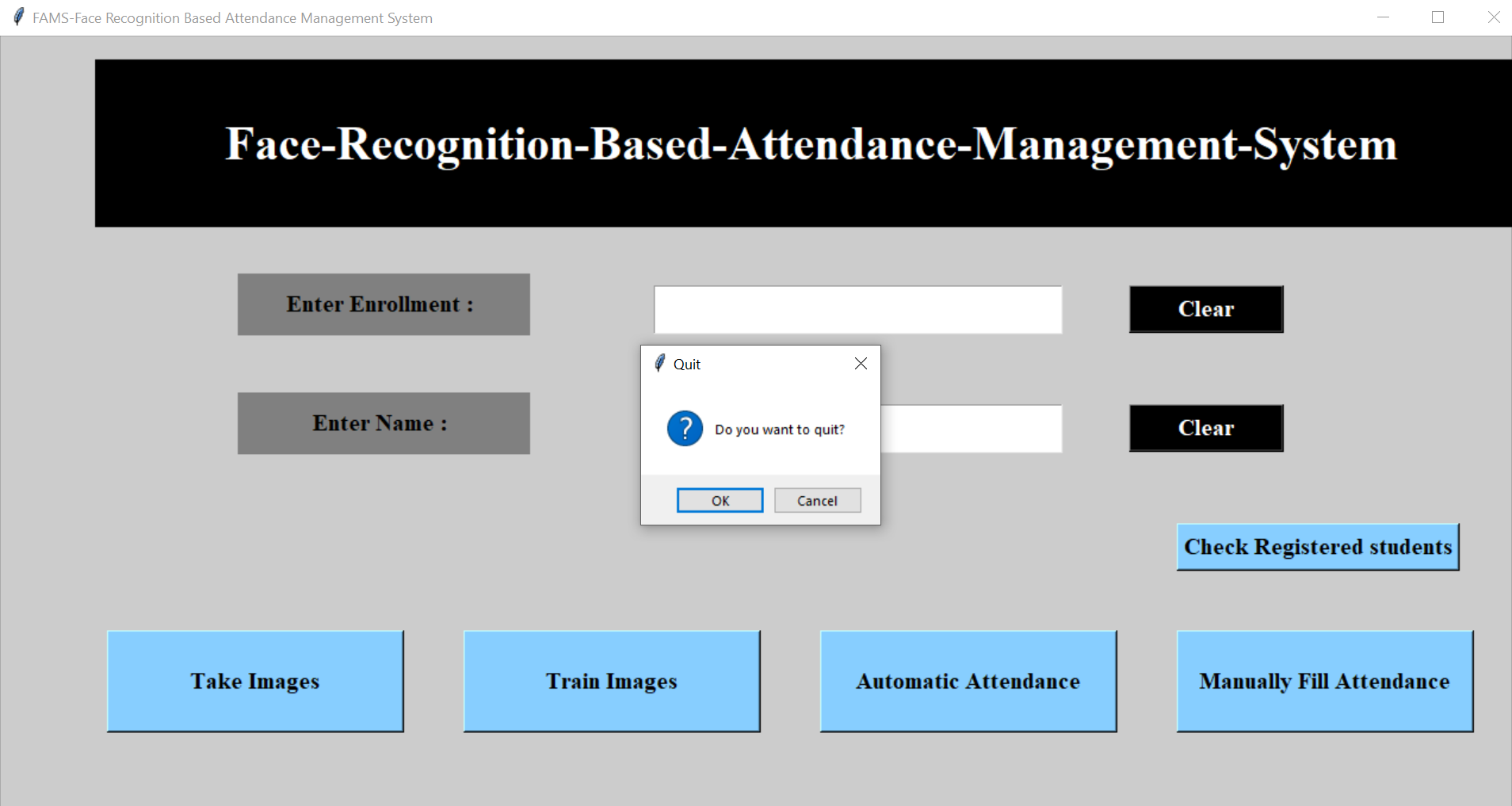
**Figure 3**



**Figure 4**



**Figure 5**



**Conclusion and Future Work:**

Face Recognition Based Attendance Management System implemented in Python, leveraging libraries such as tkinter, OpenCV, and pandas. It includes a graphical user interface (GUI) with functionalities for capturing images using the webcam, training a LBPH (Local Binary Patterns Histograms) Face Recognizer model, and marking attendance automatically or manually. The system integrates with a MySQL database to store attendance records dynamically, creating tables based on subject and date-time. Error handling is implemented for various scenarios, ensuring a smoother user experience. However, to make the system more robust and production-ready, several areas require further attention. These include improving the accuracy of the face recognition model, enhancing the user interface for better usability, implementing security measures such as authentication, and exploring scalability and integration with other systems. With additional refinement and development, the system has the potential to become a valuable tool for automating attendance management in educational or organizational settings.

**Future Work of the project:**

Here are five potential future improvements or additions to the project:

1. **Enhanced User Interface:** Improve the user interface of the application by adding more interactive features, better layout designs, and intuitive controls to enhance user experience.
2. **Real-time Face Recognition:** Implement real-time face recognition capabilities to recognize faces instantly as they appear in the camera feed, enabling immediate attendance tracking without the need for manual intervention.
3. **Attendance Reports:** Develop a feature to generate detailed attendance reports for each subject or class, including statistics such as attendance percentages, late arrivals, and absences. These reports can be exported in various formats for further analysis.
4. **Multi-platform Support:** Extend the application's compatibility to run on multiple platforms such as mobile devices (iOS, Android) and web browsers, allowing users to access attendance data from anywhere and on any device.
5. **Integration with Student Management Systems:** Integrate the attendance management system with existing student management systems used by educational institutions to streamline data flow, automate administrative tasks, and ensure data consistency across different platforms.

**References:**

1. **OpenCV:** Open Source Computer Vision Library - OpenCV is used for image processing and computer vision tasks in Python.

**Website:** https://opencv.org/

1. **NumPy -** NumPy is a fundamental package for scientific computing with Python.

**Website:** https://numpy.org/

1. **Pillow (PIL Fork) -** Pillow is a Python Imaging Library that adds image processing capabilities to Python interpreter.

**Website:** https://python-pillow.org/

1. **Pandas -** Pandas is a fast, powerful, flexible, and easy-to-use open-source data analysis and manipulation tool.

**Website:** https://pandas.pydata.org/

1. **datetime -** datetime is a module in Python that supplies classes for manipulating dates and times.

**Python Documentation:** https://docs.python.org/3/library/datetime.html

**time -** time is a module in Python that provides various time-related functions.

Python Documentation: https://docs.python.org/3/library/time.html

1. **tkinter -** tkinter is the standard GUI (Graphical User Interface) library for Python.

**Python Documentation:** https://docs.python.org/3/library/tkinter.html

1. **pymysql -** pymysql is a Python MySQL client library.
2. **CSV -** CSV module provides classes to read and write tabular data in CSV format.

**Python Documentation:** <https://docs.python.org/3/library/csv.html>