

FACE RECOGNITION SYSTEM

A PROJECT REPORT

Submitted by

PAREKH VED HEMANTKUMAR

C.V. Raman Global University, Bhubaneswar

Year – 3rd, Branch – CSE (AI & ML)

Towards completion of Summer Internship

as

Deep Learning Intern

(1st June 2024 – 15th July 2024)

at



INDIAN INSTITUTE OF TECHNOLOGY, BHUBANESWAR

Under the guidance of

DR. YOGESH G. BHUMKAR

School of Mechanical Sciences

IIT Bhubaneswar

TABLE OF CONTENTS

1. Introduction	02
2. Objectives	02
3. Working	03
4. Testing	11
5. Logical Explanation	14
6. Code Explanation	15
7. Conclusion	17
8. Future Prospects	17

INTRODUCTION

The concept of Face Recognition has become an integral part of modern technologies. From mobile phone cameras to surveillance systems for security, this underlying software have made our lives more secure and convenient by providing robust authentication identities. These systems leverage the unique physiological traits of a person's face, making them highly reliable and difficult to replicate compared to traditional methods such as passwords or PINs.

Face Recognition System built in this project is an advanced and reliable software/app which enables quick detection, storing, processing and recognition of faces in real-time. This system / software enhances the user authentication and attendance updation quality.

Need for Face Recognition Systems:

- **Security:** Traditional security measures like passwords and PINs are susceptible to being forgotten, stolen, or hacked. Face recognition provides a more secure alternative as it relies on unique facial features that are difficult to forge.
- **Convenience:** Face recognition offers a quick and user-friendly way to authenticate users without the need for memorizing complex passwords. It simplifies the process of access control in various applications.
- **Automation:** With the rise of automation in various sectors, face recognition systems can streamline processes such as attendance tracking, access control, and customer verification, reducing the need for manual intervention.
- **Attendance Tracking:** Face Recognition Systems can also be used for tracking or updating the attendance of employees or students. Although, it does not guarantee superiority as compared to biometric system, it is still robust and cheap with ability to be used in multiple domains as well.

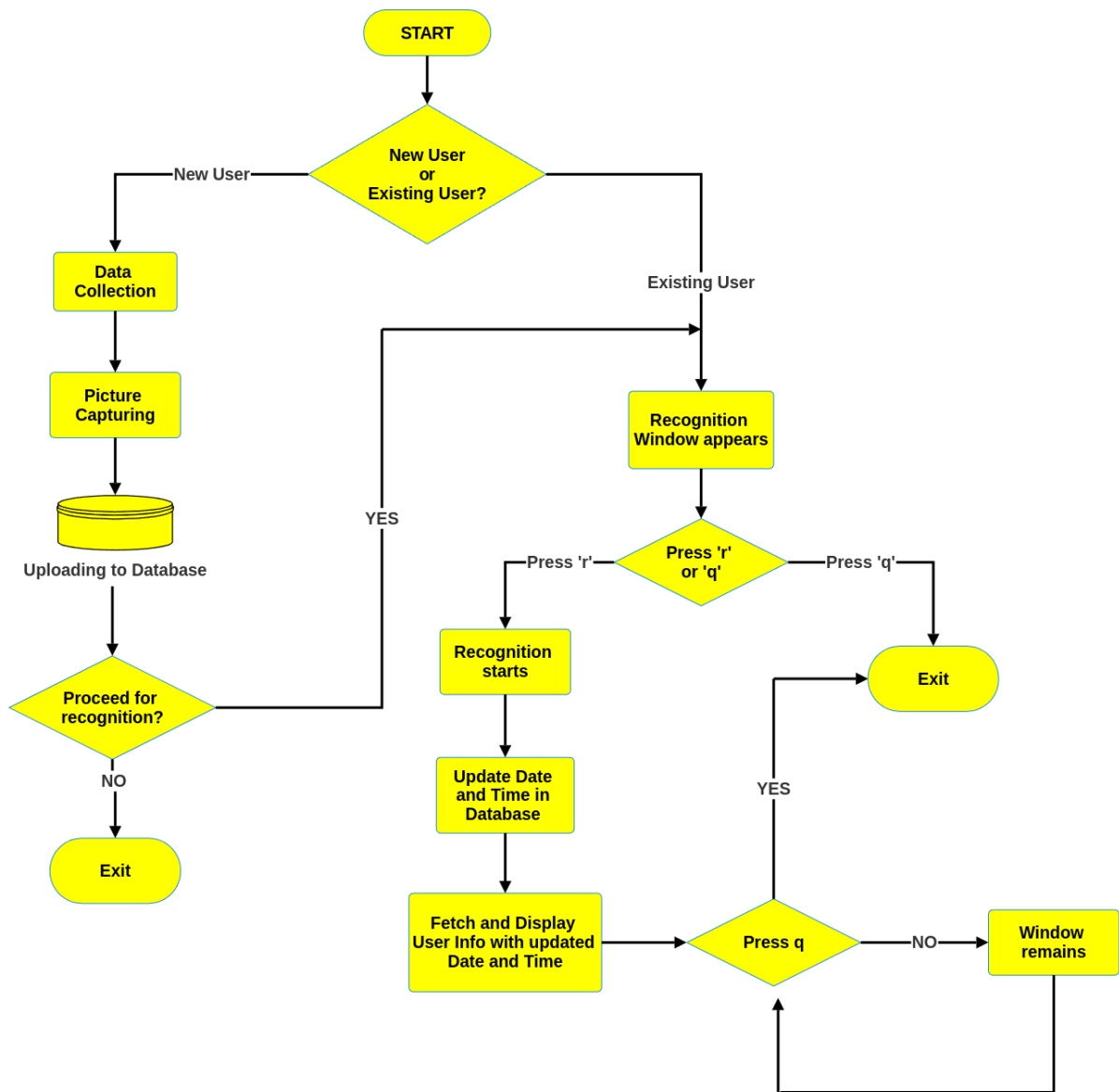
OBJECTIVES

The primary objectives of the mentioned project are as follows:

- To develop a robust and reliable Face Recognition System with a user-friendly interface, which can be used in entry gates of an institution/company.
- Provide features to capture user's data (including face) and store them securely in the database.
- Perform real-time recognition of faces in live video-graphical window and display the details of the corresponding user along with the updated date and time.

WORKING

The following figure depicts the workflow of the mentioned system.



START

To initiate the software, we have to run the main python script which consists of all the code.

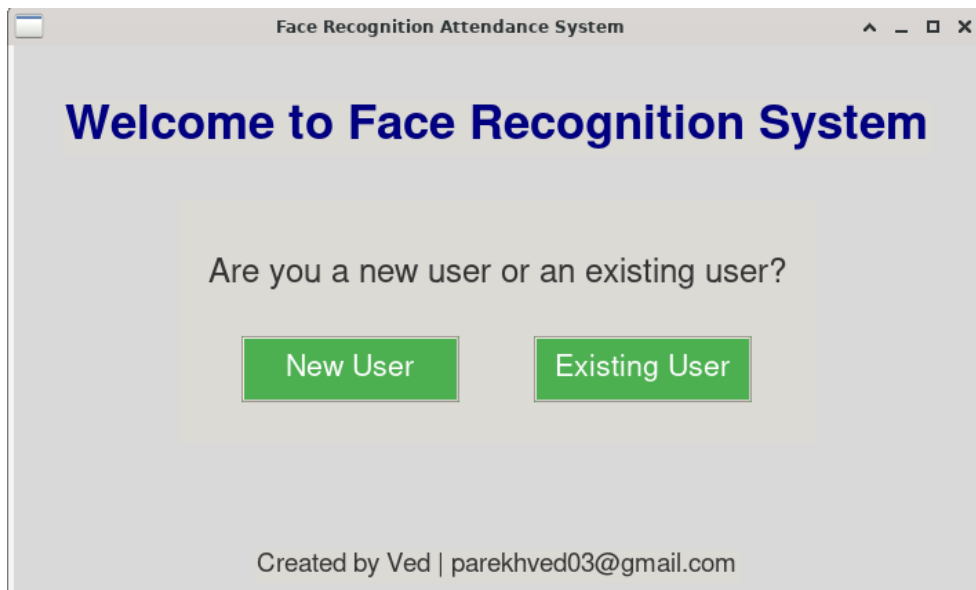
In Terminal – python main.py

In IDE – Click on run button

In future, the system can be integrated with IoT Devices like Jetson Nano. In that case, additional functionality or special mechanisms like Button or touch facility in display can be provided to run the software.

NEW USER OR EXISTING USER CHOICE

Once the code has been initiated, a window will appear asking the user whether he/she is a new user or an existing user of this system.

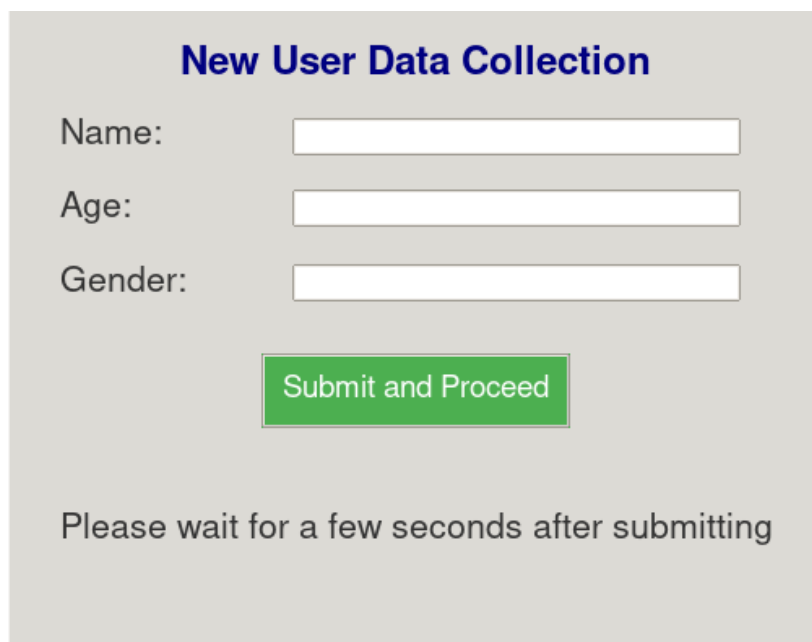


The screenshot shows a window titled "Face Recognition Attendance System". The main heading is "Welcome to Face Recognition System" in bold blue text. Below it, the question "Are you a new user or an existing user?" is displayed. There are two green buttons: "New User" and "Existing User". At the bottom, it says "Created by Ved | parekhved03@gmail.com".

NEW USER

On clicking the New User button, another window will appear which will collect the data from the user. Presently, the data includes **Name**, **Age**, **Gender**.

In future, more attributes of the user can be captured according to the usage. For example, for student attendance system, the **Registration No.** can be collected as well, through this data collection window.



The screenshot shows a window titled "New User Data Collection". It contains three input fields: "Name:", "Age:", and "Gender:". Below these fields is a green button labeled "Submit and Proceed". At the bottom, there is a message: "Please wait for a few seconds after submitting".

After filling out the details, the user is required to click on Submit and Proceed button, and

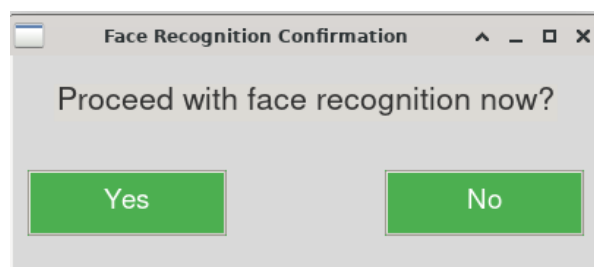
wait for a few seconds for another window to appear which will capture the image/photo of the user.

The photo of the user is required to be captured so that it can be stored in the database which would help in real-time recognition and to display the corresponding details.



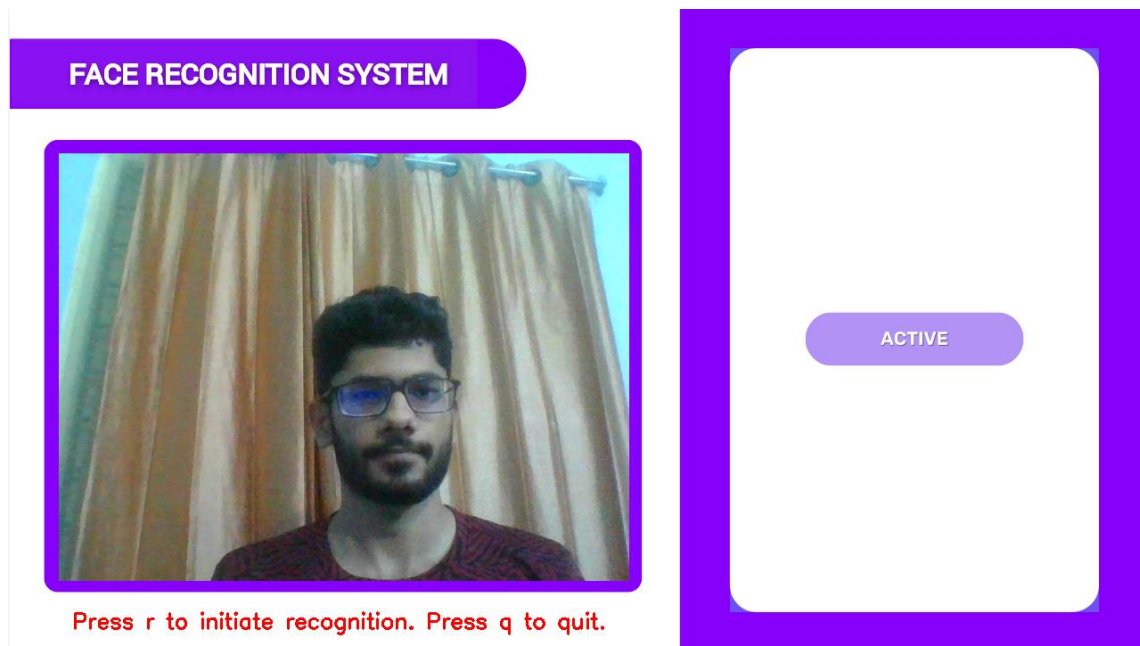
The window gives two options to users. If the users want to capture their image, they are required to **press 'c'**. Else they can **press 'q'** if they wish to quit.

By pressing 'q', the system or software would close automatically. Whereas, after pressing 'c', another window pops up, asking the users if they want to proceed with the Recognition now (since their data has been captured and stored in the database).



On clicking NO, the software would close/quit.

On clicking YES, the main recognition window would appear.

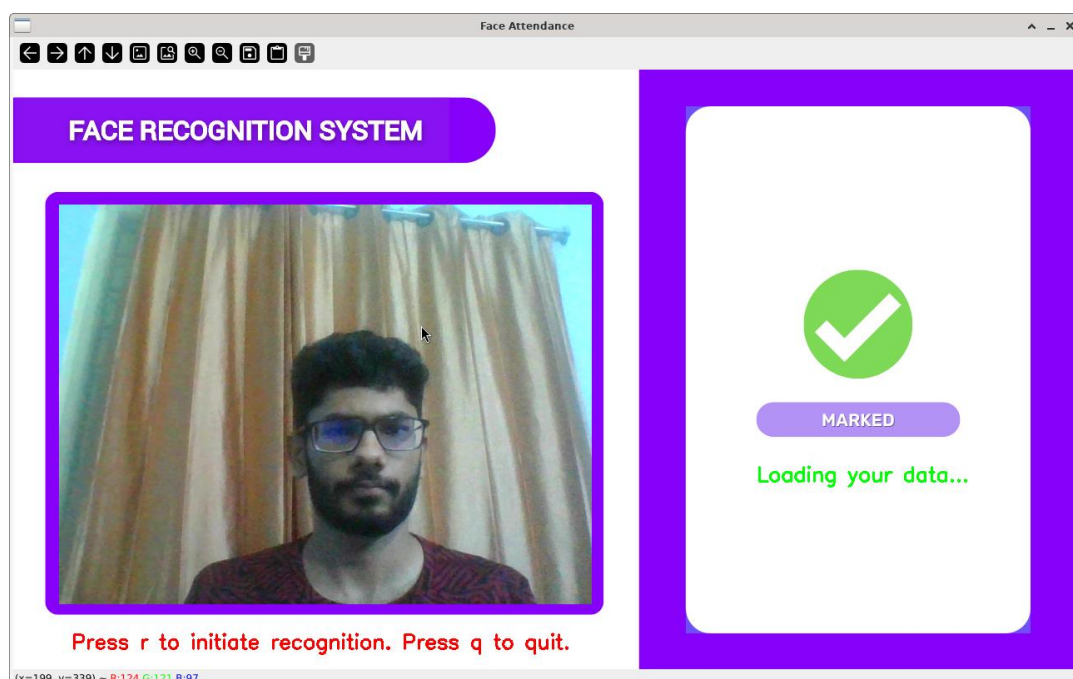


ACTIVE signifies that the system is in active state.

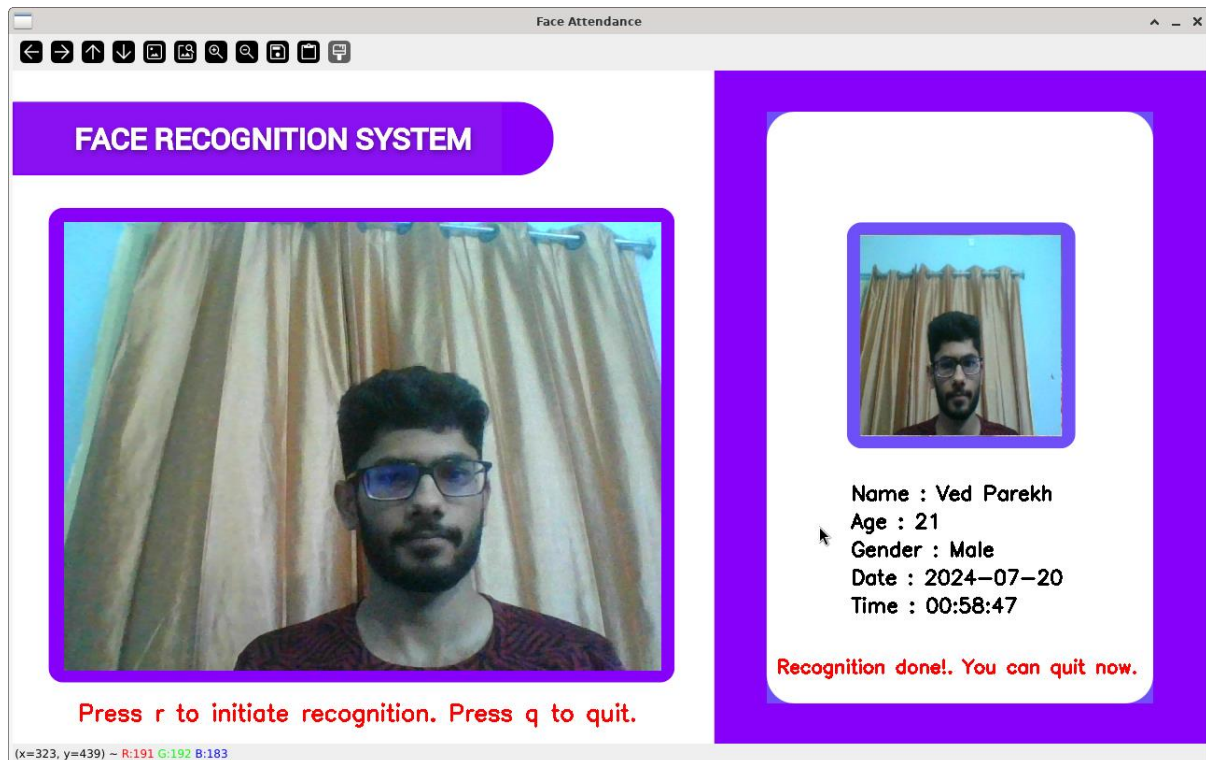
Through this window, the user can perform two operations:

1. Perform recognition/attendance tracking by pressing 'r' (case insensitive).
2. Quit the software by pressing 'q' (case insensitive).

On pressing 'r', the actual recognition starts in the backend of the software, and once the face has been recognized, the following window appears for a few seconds which would indicate that the face of user has been recognized and attendance has been marked/updated in the database.



After a few seconds the final window with user details will be displayed



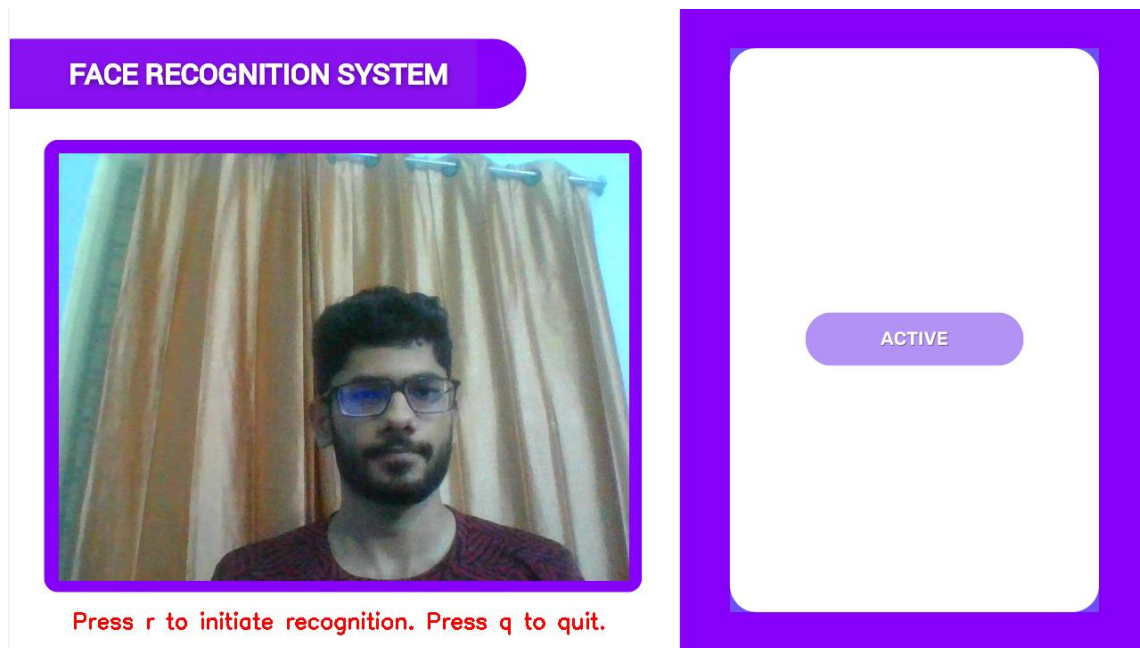
The right portion of the main recognition window will consist of the details extracted from the database.

The Date and Time gets updated every time the software the user runs this software and presses 'r' for recognition / attendance updation.

This window will remain as it is, until the user presses 'q'. This has been done intentionally so that the users can take as much time as they want, to check the details displayed in the window.

EXISTING USER

On clicking the Existing User button, the main recognition window would appear directly without going through any registration process.

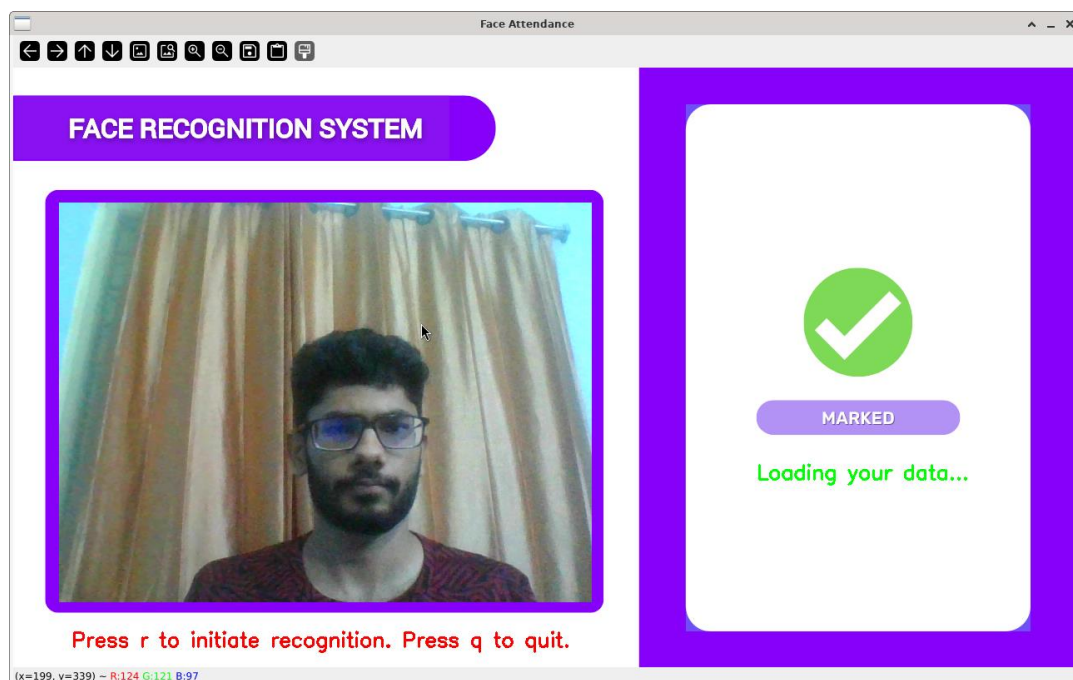


ACTIVE signifies that the system is in active state.

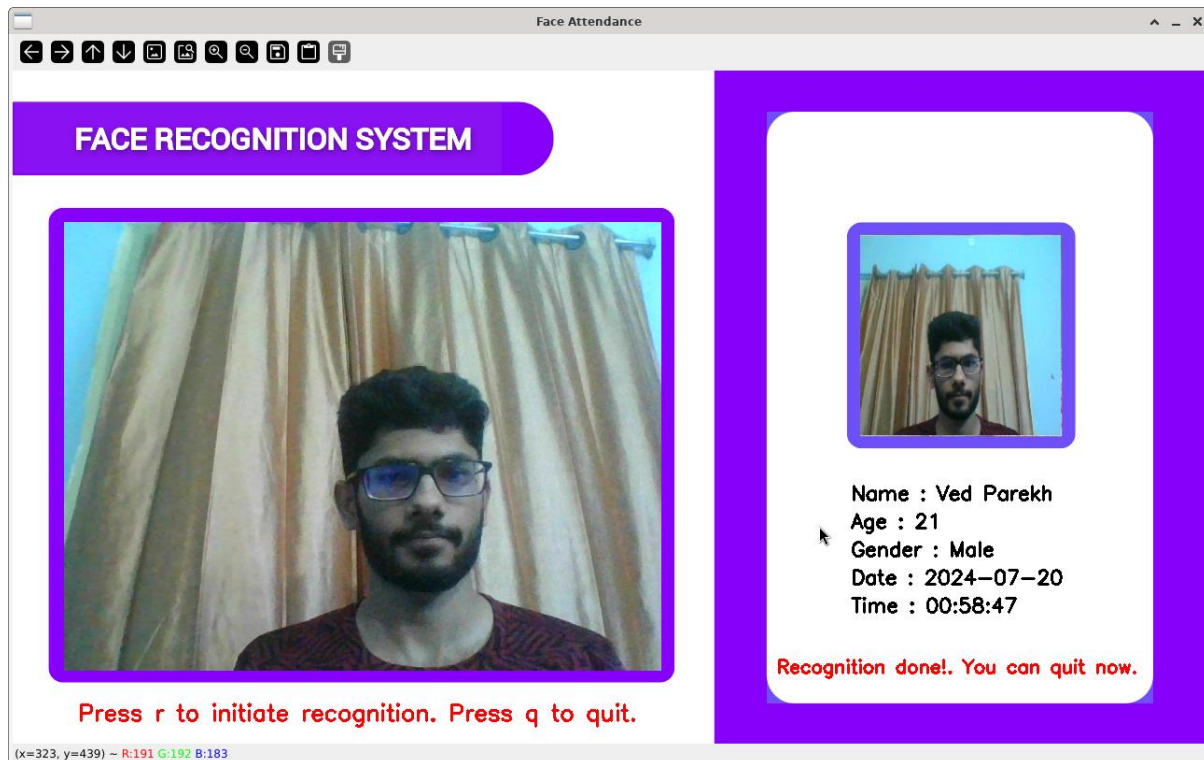
Through this window, the user can perform two operations:

1. Perform recognition/attendance tracking by pressing 'r' (case insensitive).
2. Quit the software by pressing 'q' (case insensitive).

On pressing 'r', the actual recognition starts in the backend of the software, and once the face has been recognized, the following window appears for a few seconds which would indicate that the face of user has been recognized and attendance has been marked/updated in the database.



After a few seconds the final window with user details will be displayed



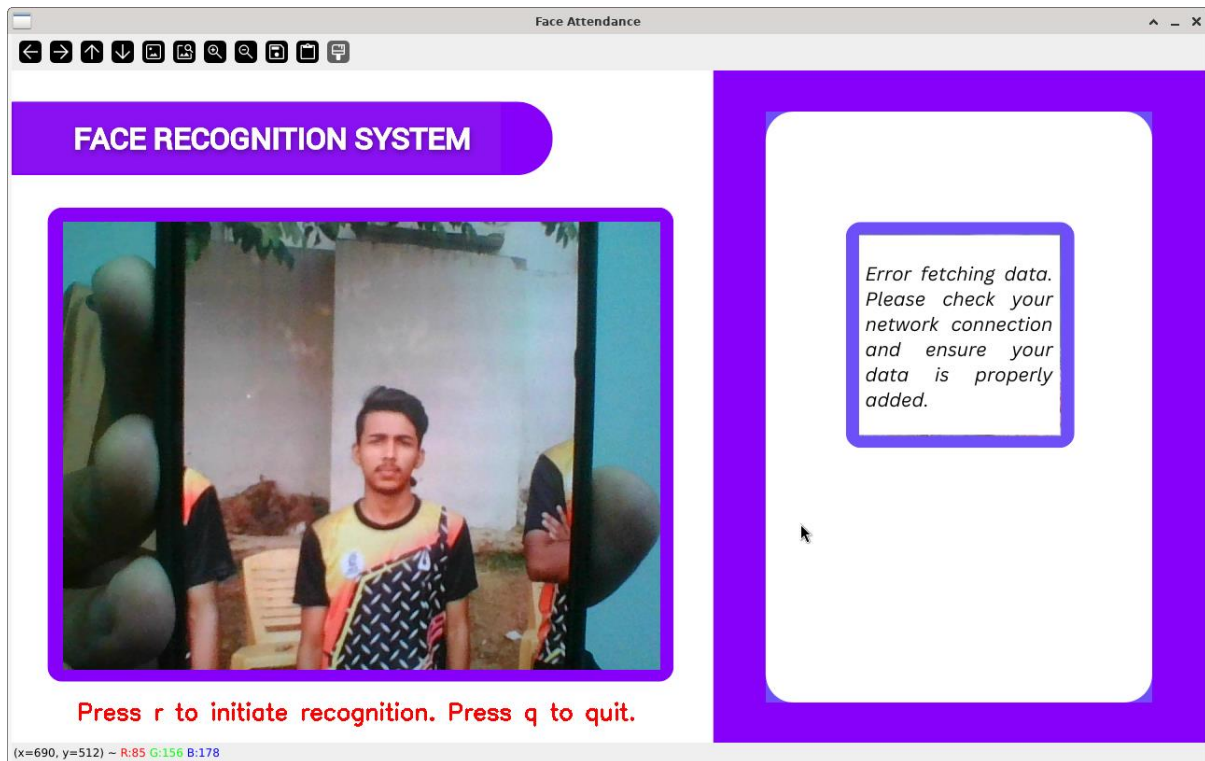
The right portion of the main recognition window will consist of the details extracted from the database.

The Date and Time gets updated every time the software the user runs this software and presses 'r' for recognition / attendance updation.

This window will remain as it is, until the user presses 'q'. This has been done intentionally so that the users can take as much time as they want, to check the details displayed in the window.

Notes:

1. Due to poor network connectivity, the second window displaying "Marked. Loading your data" might not appear. Instead, the final details window might be shown directly.
2. If the face is not recognized during the real-time live video, pressing 'r' multiple times will have no effect. It is recommended to use the system in a well-lit area where the face is visible to the camera.
3. Poor network connectivity or data deletion from the database can result in data not being displayed in the final window. In such cases, the following will be observed:

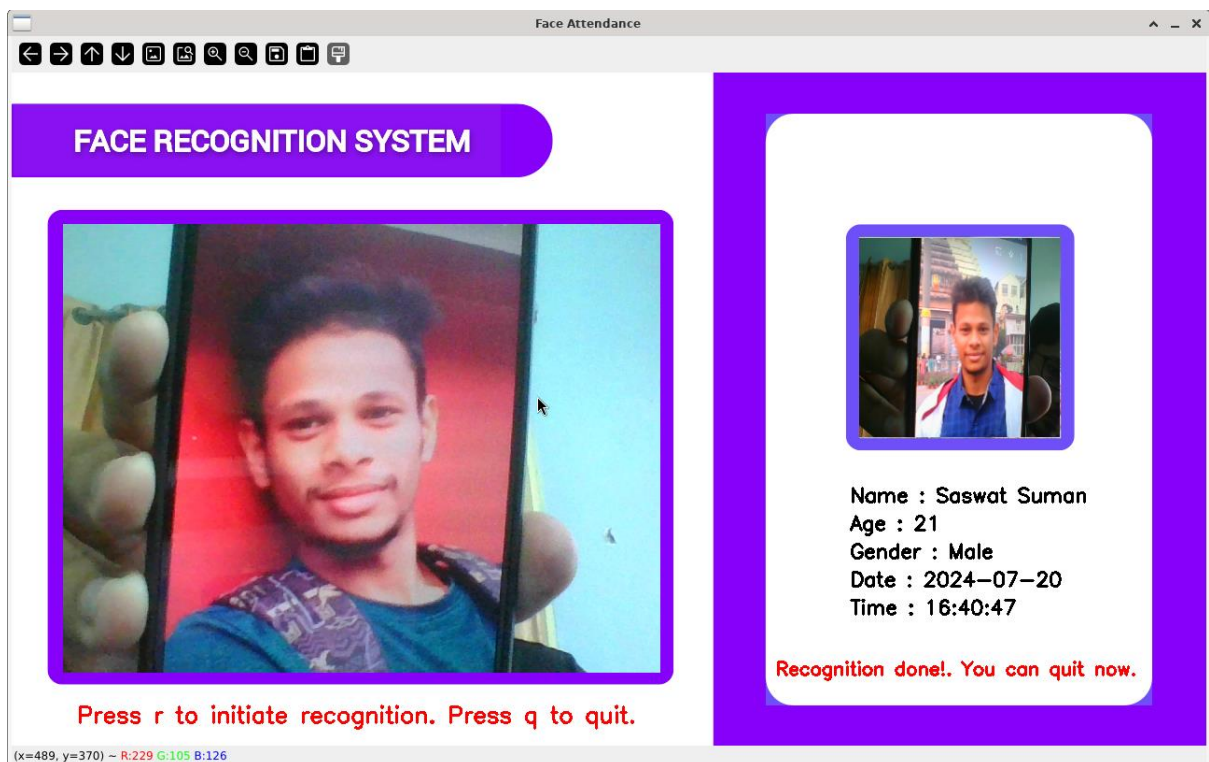


4. If an existing user clicks the new user button and provides their details again, it will update their data (including Age, Gender, Picture, and some backend processes). Therefore, existing users who wish to update their data should use the 'New User' functionality.

TESTING


The developed software was rigorously tested on a diverse group of individuals under various conditions. The system demonstrated high accuracy and reliability in recognizing faces, regardless of variations in lighting, background, and facial expressions. This extensive testing ensures the robustness and effectiveness of the face recognition system in real-world scenarios.

The following figures depict the results when tested on different people:




Face Attendance

FACE RECOGNITION SYSTEM



Press r to initiate recognition. Press q to quit.

(x=548, y=450) ~ R:109 G:94 B:58

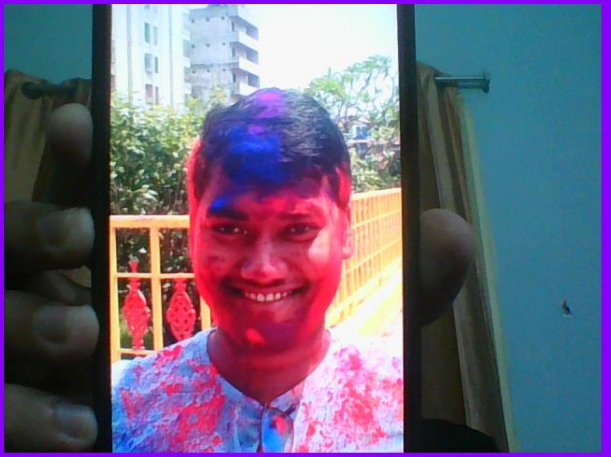


Name : Vasu Parekh
Age : 28
Gender : Male
Date : 2024-07-20
Time : 23:41:35

Recognition done!. You can quit now.


Face Attendance

FACE RECOGNITION SYSTEM



Press r to initiate recognition. Press q to quit.

(x=466, y=468) ~ R:248 G:248 B:248



Name : Sawan Hansdah
Age : 21
Gender : Male
Date : 2024-07-20
Time : 23:45:39

Recognition done!. You can quit now.

LOGICAL EXPLANATION

Overview

The project primarily leverages the **face_recognition** library, which provides a state-of-the-art model for recognizing faces in a frame and obtaining their encodings. This model boasts an impressive accuracy of 99.38%, making it highly reliable for face recognition tasks. The library works in by performing the following steps:

Working of face_recognition library:

The **face_recognition** library operates through a series of sophisticated steps:

1. **Encoding a Picture Using the HOG Algorithm:**
 - The first step involves encoding the input image using the Histogram of Oriented Gradients (HOG) algorithm. This algorithm simplifies the image, emphasizing edges and contours, which are crucial for identifying facial features. The goal is to find parts of the image that resemble a generic face HOG encoding.
2. **Face Landmark Detection and Pose Estimation:**
 - Once a potential face is located, the library identifies key landmarks on the face, such as the eyes, nose, and mouth. These landmarks are used to align and warp the face image so that the eyes and mouth are centered, providing a standardized pose for further processing.
3. **Feature Extraction Using a Neural Network:**
 - The centered face image is passed through a neural network trained to extract 128 unique measurements or features from the face. These features represent the face's unique characteristics.
4. **Face Comparison and Matching:**
 - The extracted features are compared with a database of previously measured faces. The person whose stored measurements are closest to the new face's measurements is identified as the match.

More detailed explanation of how the library was built and how face recognition works can be found [here](#).

Additional Libraries and Frameworks

Along with face_recognition library, the project uses several other python libraries and frameworks like:

- **Tkinter:** A standard GUI library for Python to create graphical user interfaces.
- **OpenCV (cv2):** An open-source computer vision library used for real-time computer vision applications.
- **Firebase Admin SDK:** For integrating with Firebase services like Realtime Database and Cloud Storage.
- **NumPy:** A library for numerical operations.
- **PIL (Pillow):** Python Imaging Library for image processing.
- **datetime:** To handle date and time operations.
- **sys:** To handle system-specific parameters and functions.

- **pickle**: To serialize and deserialize Python objects.

New User Registration

The following process is executed when a new user registers:

1. The user provides personal details such as name, age, and gender through a Tkinter interface.
2. The system captures the user's face using a webcam. The **face_recognition** library processes the captured image, detecting the face and generating its unique encoding.
3. The generated face encoding is stored/updated in a pickle file named "Encodings.p" alongside the user's name. This file acts as a local database for face recognition.
4. The user's details (name, age, gender) and profile picture are stored in Firebase's Realtime Database and Cloud Storage, respectively.

Existing User Registration

The following process is executed when an existing user registers:

1. The system captures the user's face from a webcam in real-time when the user presses 'r'. The **face_recognition** library detects the face and generates its encoding.
2. The generated encoding is compared with the encodings stored in the "Encodings.p" file. The comparison uses a similarity metric to find the closest match.
3. The system identifies the user by selecting the name corresponding to the closest encoding match from the pickle file.
4. The user's details are retrieved from the Firebase Realtime Database and displayed in the recognition window.

Firebase Integration

Firebase plays a crucial role in data storage and management:

- **Realtime Database**: Stores user details such as name, age, gender, date, and time of registration.
- **Cloud Storage**: Stores user profile pictures, ensuring secure and scalable image storage.

CODE EXPLANATION

(Kindly refer to **main.py** for the complete code)

Class **new_user**:

__init__ method:

- **Purpose**: Initializes the **new_user** class with the root window, Firebase database reference, Firebase storage bucket, known face encodings, and student names.
- **Function**: Sets the class attributes and calls **get_user_data** to start the new user registration process.

Put_text_with_background method:

- **Purpose**: Draws text with a background color on a given frame.
- **Parameters**: frame (image frame), text (string to display), position (coordinates for text), font, font_scale, text_color, bg_color, and thickness.

- **Function:** Calculates text size and position, draws a background rectangle, and overlays text.

get_user_data method:

- **Purpose:** Creates a new Tkinter window to collect new user data (name, age, gender).
- **Function:** Sets up the window with styled labels, entry fields, and a submit button. When the submit button is clicked, it calls `submit_and_proceed`.

Submit_and_proceed method:

- **Purpose:** Submits the user data to the Firebase database, captures a picture of the user, and stores it in Firebase storage.
- **Function:** Destroys the data collection window, captures an image from the webcam, overlays instructional text, uploads the captured image to Firebase storage, updates the database with the capture time, encodes the face, and saves the encodings. Finally, it calls `proceed_with_recognition`.

Proceed_with_recognition method:

- **Purpose:** Asks the user if they want to proceed with face recognition.
- **Function:** Creates a confirmation window with "Yes" and "No" buttons. If "Yes" is clicked, it creates an instance of `existing_user` to start face recognition. If "No" is clicked, it exits the program.

Center_window method:

- **Purpose:** Centers an OpenCV window on the screen.
- **Parameters:** `window_name` (name of the OpenCV window), `width` (width of the window), `height` (height of the window).
- **Function:** Calculates the screen dimensions and positions the window accordingly.

Class existing_user:

__init__ method:

- **Purpose:** Initializes the `existing_user` class with the root window, Firebase database reference, Firebase storage bucket, known face encodings, and student names.
- **Function:** Sets the class attributes and calls `recognize_face` to start face recognition.

Recognize_face method:

- **Purpose:** Captures video from the webcam and performs face recognition on existing users.
- **Function:** Continuously captures video frames, displays instructional text, compares captured face encodings with known encodings, and updates the database with the recognition time. If a face is recognized, it displays the recognized user's details and exits.

Main Application:

If `__name__ == "__main__"` block:

- **Purpose:** Initializes the main application, including Firebase, known encodings, and the main Tkinter window.
- **Function:** Loads known encodings from a pickle file (if available), creates the main Tkinter window with styled labels and buttons, and sets up the main loop. Clicking "New User" creates an instance of `new_user` for registration. Clicking "Existing User" creates an instance of `existing_user` for face recognition.

CONCLUSION

The Face Recognition System project successfully demonstrates the potential of leveraging deep learning techniques for real-time facial recognition and attendance tracking. This developed system showcases the ability to detect, store, process, recognize faces, and update data accordingly in the database, with high accuracy and reliability.

Key Achievements:

1. **High Accuracy and Reliability:** The system has undergone rigorous testing across various conditions, confirming its robustness and effectiveness. It maintains consistent performance despite variations in lighting, background, and facial expressions.
2. **User-Friendly Interface:** The project focuses on providing an intuitive and seamless user experience, simplifying the process of user registration and real-time face recognition.
3. **Security and Convenience:** By relying on unique physiological traits, the system offers a more secure alternative to traditional methods such as passwords or PINs. It also enhances user convenience, eliminating the need for memorizing complex passwords.
4. **Automation and Efficiency:** The system streamlines processes like attendance tracking and access control, reducing the need for manual intervention and increasing overall efficiency.

FUTURE PROSPECTS

1. Integration with Advanced Embedded Devices:

Utilize advanced embedded devices like the **Jetson Nano** for seamless real-time deployment. This integration would enhance the system's portability and make it more suitable for edge computing scenarios, reducing the reliance on a central server.

2. Web-Based Application:

Develop a **web-based application** to improve the user interface and accessibility. A web-based platform can offer more flexibility and ease of use, allowing users to access the system from any device with internet connectivity. However, careful consideration must be given to potential latency issues and the need for robust backend infrastructure to handle real-time face recognition efficiently.

3. Enhanced Security with Password-Based Credentials:

Implement **password-based credentials** for users in addition to face recognition. This multi-factor authentication approach would enhance security, ensuring that only authorized users can access sensitive information and functionalities within the system.