Facial Recognition Software Report

CREATED BY,

Akash Singh

Vellore Institute of Technology, Vellore

August 2023

Version 1.0

Table of Contents

1. Introduction

1.1 Scope and Purpose

The facial recognition software is made up of a login system attached to a database which is made in SQLite as well as the GUI is made in Python. It is complete with the admin access in which you can register new user download the logs as well as delete a previously used user from the database. To recognise the face, I have trained a model which uses the face shape as well as face features like the eyes nose and lips to detect the face which works in low lighting as well as when the full face is not visible improving its accuracy.

Use Cases:

Time and Attendance Tracking:

• Workplaces: Companies can use facial recognition to track employee attendance, eliminating the need for traditional time clocks or manual time tracking.

• Educational Institutions: Schools and universities can automate attendance tracking, making the process more accurate and efficient.

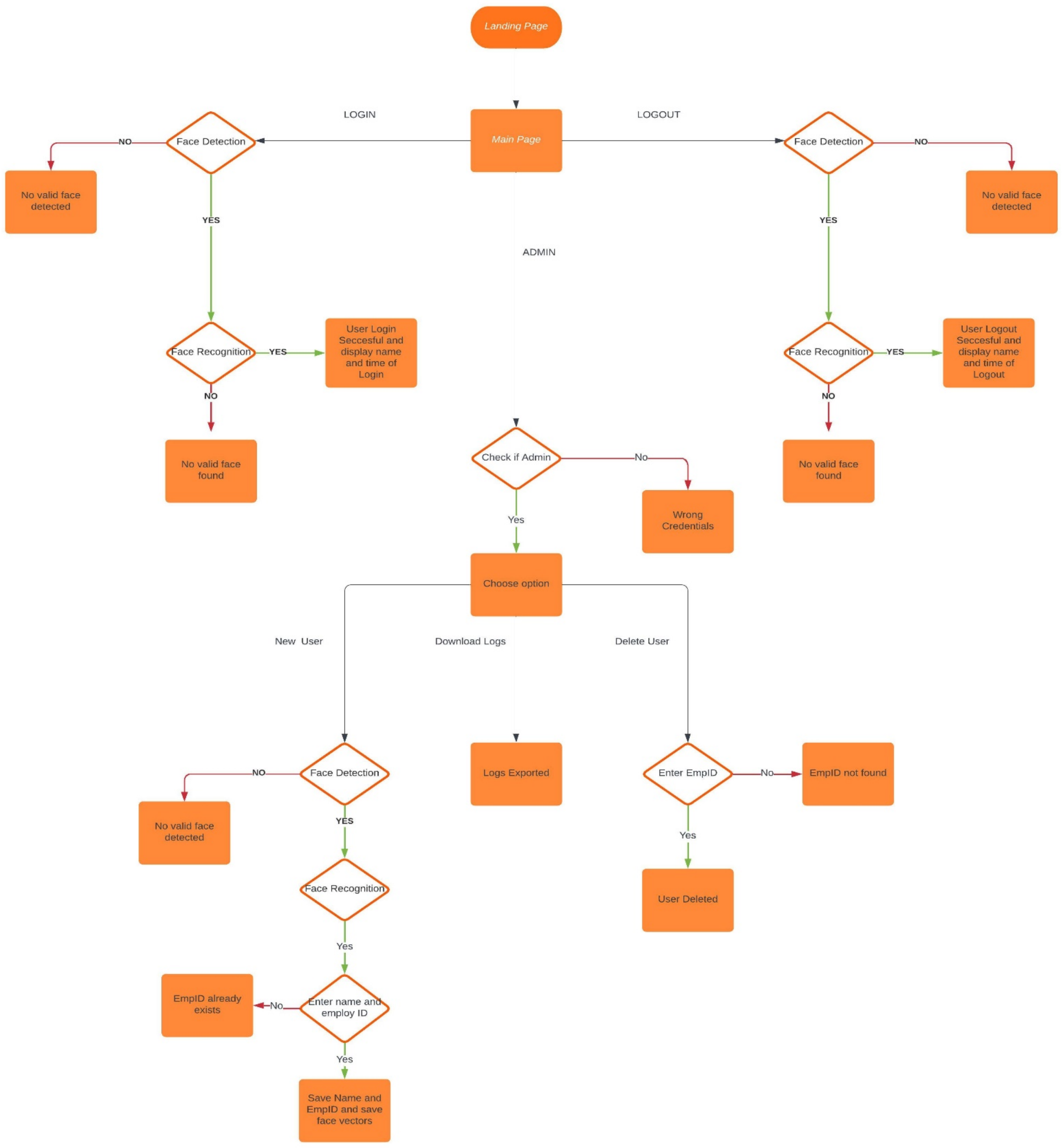
Automated Payment and Transactions:

• Retail and Restaurants: Facial recognition can be linked to payment methods, allowing customers to make secure payments without using physical cards or cash.

• Public Transportation: Facial recognition can facilitate contactless fare payments for buses, trains, and subways.

1.2 Process Overview

The current features of the software are when we launched the program it first takes you to the main window where you have 3 options the login, logout as well as the admin option when clicking on the login option it checks and detects the face of the person in front of the camera and logs in according to the registered username at the current time and similarly for the logout function detects the face and logs out for the current time which is also depicted in the database. When clicking the admin function button, it prompts you for the admin password and username which is hard coded into the code which makes it very hard to scam, alter which when entered it takes you to another window where we have 3 options which are register new user download logs and delete user. The register new user button then takes the user to another page where it captures a photo of the user and prompts to input the username as well as the employee Id which will be the main criteria to identify the user you can press enter and as the user will successfully be registered. Download logs option gives the admin privilege to download all the logs into an excel file where it can be viewed in an easier way making it convenient to look at the current logs allows to delete a particular user entering the user ID or the employee ID.



Flow Chart for Process followed.

2.Software/ Libraries Used

For writing the main code Python language has been used, for the GUI i have used the Tkinter library which makes it easy to create a GUI as well as to integrate it with the database. The database used is SQL Lite 3 which is a robust software make sure that the data is stored properly and correctly and
cannot be changed by any 3rd party. CV2 and DLib libraries have been used to operate the facial recognition system which make sure accurate facial recognition as well as detection improving the accuracy. Pandas' library has been used to export the logs into a excel file. NumPy is used to create an array of the facial recognition vectors.

Python: Python is a popular high-level programming language used for a variety of tasks, from web development to scientific computing. It is praised for its readability and adaptability.

OpenCV: Using a variety of functions and algorithms, OpenCV is a potent computer vision library for Python that makes it possible to perform operations like object detection, feature extraction, and image and video processing.

NumPy: The backbone of Python's numerical computing, NumPy offers support for multi-dimensional arrays and matrices as well as a number of mathematical operations that can be performed on these objects.

Tkinter: Python's de facto GUI (Graphical User Interface) toolkit, facilitating the creation of windows, dialogs, and interactive elements, making it simpler to craft user-friendly applications.

Dlib: A powerful toolkit for computer vision and machine learning tasks that includes features like facial recognition, landmark prediction, and deep learning integration. It makes complex applications simple to use.

Sqlite3: A lightweight Python built-in library that makes it easier to integrate and work with SQLite databases, providing a quick and effective way to manage structured data.

Pandas: A powerful tool for data manipulation that offers data structures like Data Frames to organise and analyse data as well as tools to effectively handle missing data, combine datasets, and perform aggregations..

PIL (Python Imaging Library): Now known as Pillow, this library provides extensive support for opening, editing, and saving a wide range of image file formats, simplifying image processing tasks in Python programmes.

3. Code Overview

The code file has been provided along with all the dependencies.

The external files used are:

- Dlib face recognition resnet model
- Haarcascade frontal face
- Dlib shape predictor

These are stored on the system and the path of these files are provided in the code.

The location where the logs are exported to is also provided in the code. Now going over the different function used-

__init__(self): This is the App class's constructor method. It establishes connections to the database and various resources, including Haarcascades and dlib models, and initialises a number of attributes. It also sets up the main window interface.

main(self): After the user clicks the "Proceed" button in the initial window, this function creates the application's main window. It sets up the primary user interface, from which users can access admin features and log in and out.

reg(self): When the "Admin" button in the main window is clicked, this function is activated. The administrator can register new users by taking pictures of their faces in the newly created window.

captureface(self, username, empid): This function is responsible for capturing a user's face image and registering it in the database. It checks if the provided employee ID exists, captures a face image using the webcam, and computes the face descriptor to store in the database.

add_webcam(self, label): This function starts processing frames to display them on the specified label after initialising the webcam capture.

process_webcam(self): This function's main loop for processing webcam frame data is called process_webcam(self). It takes pictures, finds faces, and configures the GUI to show the pictures with rectangles around faces.

checkadmin(self): When the "Admin" button in the main window is clicked, the function checkadmin(self) is invoked. The administrator must enter a username and password in a new window that opens in order to access admin features.

check(self): This function is called when the administrator tries to log in by entering their username and password. It checks if the entered credentials are correct and opens the admin panel if they are.

admin(self): This function creates the admin panel window where the administrator can manage users, including registering new users, downloading logs, and deleting users.

deluser(self): This function is called from the admin panel when the administrator wants to remove a user from the database. It opens a window where the administrator can enter the ID of the user to be deleted.

delete(self): This function deletes the user with the entered ID from the database.

login(self): This function is called when the "Login" button is clicked in the main window. It captures the current frame and face descriptor and checks if any registered user's face matches the captured one for a successful login.

logout(self): This function is similar to the **login** function but is used for logging out users.

add_img_to_label(self, label): This function updates the label with the most recent captured image during user registration.

start(self): This function starts the main event loop of the Tkinter application, allowing the GUI to respond to user interactions.

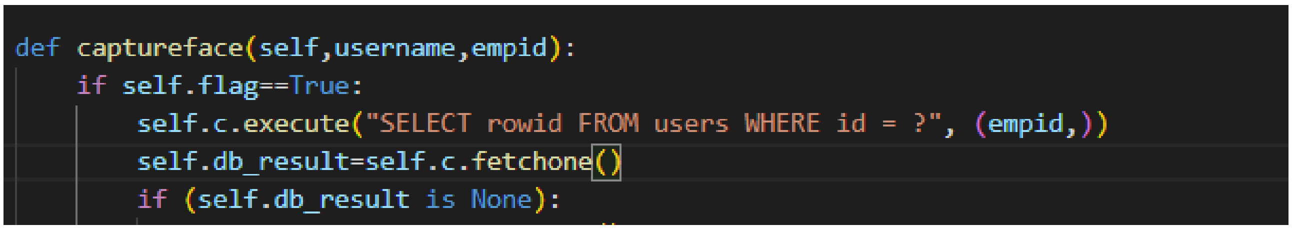
accept_register_new_user(self): This function is called when the "Register" button is clicked in the admin panel for registering new users.

logs(self): This function exports the logs of login and logout times to an Excel file.

user(self): This function exports user data (IDs and usernames) to an Excel file.

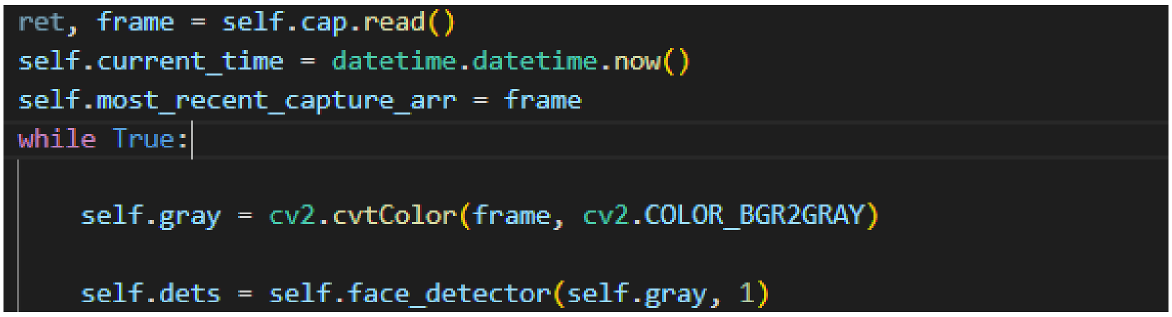
4. Facial Recognition Overview

Going through the capture face function-



```
def captureface(self,username,empid):
    if self.flag==True:
        self.c.execute("SELECT rowid FROM users WHERE id = ?", (empid,))
        self.db_result=self.c.fetchone()
        if (self.db_result is None):
```

- The self.flag variable indicates whether a valid face has been detected by the system.
- The code checks if the provided employee ID (empid) already exists in the database. It does this by executing a SELECT query on the users table to fetch the row ID for the given employee ID.
- If the query result (db_result) is None, it means that the provided employee ID doesn't exist in the database, which implies that this is a new user.

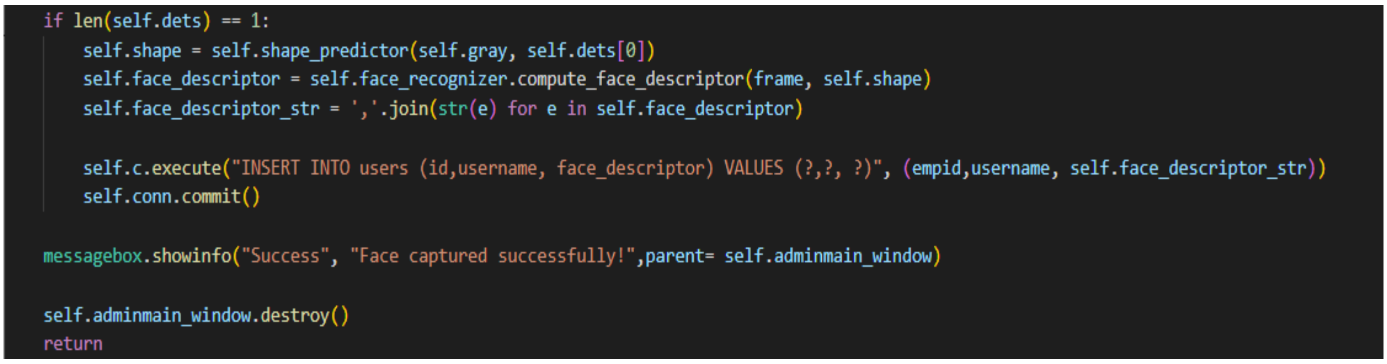


```
ret, frame = self.cap.read()
self.current_time = datetime.datetime.now()
self.most_recent_capture_arr = frame
while True:

    self.gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    self.dets = self.face_detector(self.gray, 1)
```

- If it's a new user, the code captures a frame from the webcam using cap.read() and stores it in the frame variable. It also records the current timestamp as current_time.
- A grayscale version of the frame is created using cv2.cvtColor().
- The face_detector (a dlib object) is used to detect faces in the grayscale frame. The 1 parameter in self.face_detector(self.gray, 1) specifies that the frame should be upsampled once to detect smaller faces.



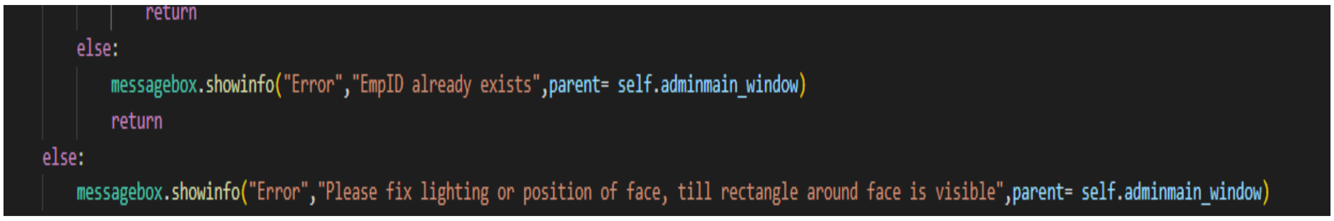
```
if len(self.dets) == 1:
    self.shape = self.shape_predictor(self.gray, self.dets[0])
    self.face_descriptor = self.face_recognizer.compute_face_descriptor(frame, self.shape)
    self.face_descriptor_str = ','.join(str(e) for e in self.face_descriptor)

    self.c.execute("INSERT INTO users (id,username, face_descriptor) VALUES (?,?, ?)", (empid,username, self.face_descriptor_str))
    self.conn.commit()

messagebox.showinfo("Success", "Face captured successfully!",parent= self.adminmain_window)

self.adminmain_window.destroy()
return
```

- If exactly one face is detected in the frame (len(self.dets) == 1), the facial landmarks for the detected face are computed using the shape_predictor.
- The face_recognizer is then used to compute the face descriptor (a numerical representation of the face) using the captured frame and the detected facial landmarks.
- The calculated face descriptor is converted to a string format and inserted into the users table of the database along with the provided employee ID and username.
- The success message is displayed using a message box, indicating that the face has been successfully captured and registered.
- The adminmain_window is destroyed, which closes the registration window.



```
            return
        else:
            messagebox.showinfo("Error","EmpID already exists",parent= self.adminmain_window)
            return
    else:
        messagebox.showinfo("Error","Please fix lighting or position of face, till rectangle around face is visible",parent= self.adminmain_window)
```

- If no face or more than one face is detected in the frame, an error message is displayed using a message box, indicating that the user should adjust the lighting or position of their face until a rectangle around the face is visible.
- If the provided employee ID already exists in the database (db_result is not None), an error message is displayed indicating that the employee ID already exists.
- If self.flag is not True, it means that a valid face hasn't been detected, so an error message is displayed indicating that the user should fix the lighting or position of their face.

5. GUI Overview

The library used to create the GUI of the software is Tkinter and to add more customizations I have used CustomTkinter imported and installed from GitHub.

Explaining the various aspects of GUI throughout the code-

Importing Libraries: You start by importing various libraries, including **tkinter**, **cv2** (OpenCV), **PIL** (Python Imaging Library), **sqlite3**, and others. These libraries provide functionalities for GUI development, image processing, face detection/recognition, and database operations.

Class Definition (App): You define a class named **App** that encapsulates the entire application. This class contains methods and attributes to manage different aspects of the GUI application.

Initializing the Application: The **__init__** method serves as the constructor for the **App** class. It initializes various attributes and sets up the main window (**first_window**) of the application.

GUI Elements: Throughout the script, you create various GUI elements such as labels, buttons, and entry fields using the **tkinter** library. For example:

- **tk.Label**: Used to display images and text labels.
- **customtkinter.CTkButton**: A custom button widget that you've defined.
- **tk.Entry**: Used to capture user input.

Methods and Functionality:

- **main**: Sets up the main window of the application with buttons like "Login," "Logout," and "Admin." It also includes webcam integration to display live camera feed.
- **login**, **logout**: These methods attempt to recognize a user's face based on the saved face descriptors. If successful, they perform login or logout actions.
- **checkadmin**, **admin**: These methods handle administrative functionalities, such as adding new users, deleting users, and logging their activities.
- **reg**, **captureface**, **accept_register_new_user**: These methods are used to register new users by capturing their face data and saving it to the database.
- **deluser**, **delete**: Handles the process of deleting a user's record from the database.
- **logs**: Exports log data to an Excel file.

- **start**: Initiates the main event loop of the application, allowing it to respond to user interactions.

Webcam Integration: The application uses the **cv2** library (OpenCV) to capture frames from the webcam and process them for face recognition and registration.

Database Interaction: The application interacts with an SQLite database to store user information and log data.

User Interface: The UI elements are arranged on different windows (**first_window**, **main_window**, **admin_newwindow**, etc.) with specific layouts and designs.

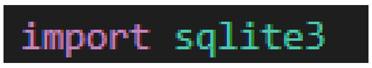
Custom Widgets: You've defined a custom button widget **customtkinter.CTkButton** with specific styling and behaviour.

Conditional Logic: The code uses conditional statements to handle various scenarios, such as successful or unsuccessful face recognition and user authentication.

Loops and Event Handling: The application utilizes loops (like **after**) to continuously update the GUI elements and respond to user events.

6. Database Overview

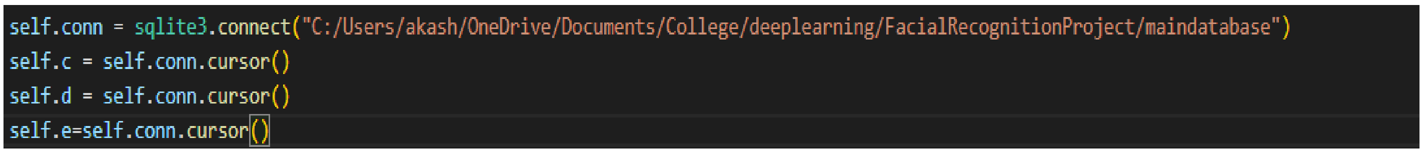
Importing the Module:



```
import sqlite3
```

This line imports the **sqlite3** module, which provides the necessary functions and classes for working with SQLite databases.

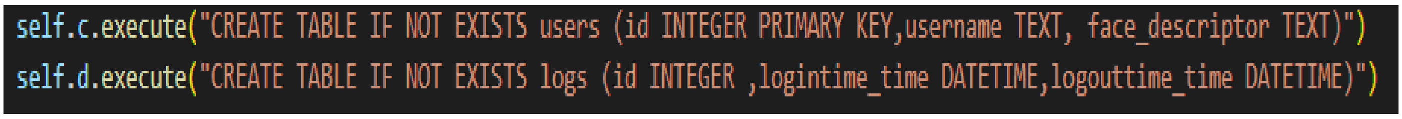
Connecting to a Database:



```
self.conn = sqlite3.connect("C:/Users/akash/OneDrive/Documents/College/deeplearning/FacialRecognitionProject/maindatabase")
self.c = self.conn.cursor()
self.d = self.conn.cursor()
self.e=self.conn.cursor()
```

This line establishes a connection to the SQLite database named "maindatabase" located at the specified file path. If the database file doesn't exist, it will be created. The connection object (**self.conn**) is used to interact with the database.

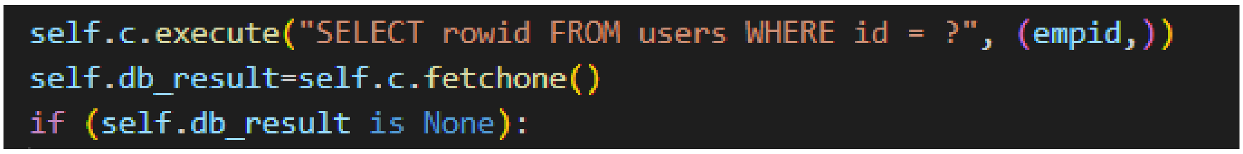
Creating Tables:



```
self.c.execute("CREATE TABLE IF NOT EXISTS users (id INTEGER PRIMARY KEY,username TEXT, face_descriptor TEXT)")
self.d.execute("CREATE TABLE IF NOT EXISTS logs (id INTEGER ,logintime_time DATETIME,logouttime_time DATETIME)")
```

These lines create two tables if they don't already exist in the database. The **users** table is created to store user information, and the **logs** table is created to store login and logout timestamps.

Executing SQL Queries:



```
self.c.execute("SELECT rowid FROM users WHERE id = ?", (empid,))
self.db_result=self.c.fetchone()
if (self.db_result is None):
```

Here, an SQL query is executed to select the row identifier (**rowid**) from the **users** table where the **id** matches the provided **empid**. The query is parameterized to avoid SQL injection by passing the **empid** as a parameter to the query. The **fetchone()** method retrieves the first result of the query.

Inserting Data:

```
self.c.execute("INSERT INTO users (id,username, face_descriptor) VALUES (?,?, ?)", (empid,username, self.face_descriptor_str))
self.conn.commit()
```

This code segment inserts new user information (id, username, face descriptor) into the **users** table. The **execute()** method is used to execute the SQL INSERT statement. The question marks in the statement are placeholders for values that are passed as parameters in the second argument. After making changes to the database, the **commit()** method is called to save the changes.

Fetching Data:

```
self.c.execute("SELECT rowid FROM users WHERE id = ?", (self.delusername.get(),))
self.del_result=self.c.fetchone()
```

This code segment executes an SQL SELECT query to fetch all rows from the **users** table. The **fetchall()** method retrieves all rows returned by the query as a list of tuples. Each tuple corresponds to a row in the table and contains the values of the specified columns.

Deleting Data:

```
self.c.execute('DELETE FROM users WHERE id=?',(self.delusername.get(),))
self.conn.commit()
```

This code segment deletes a user's information from the **users** table based on the provided user ID. The SQL DELETE statement is executed with the **id** value as a parameter. The **commit()** method is called afterward to apply the change to the database.

Closing the Connection: It's a good practice to close the database connection once you're done using it. You can close the connection using:

```
self.conn.close()
```

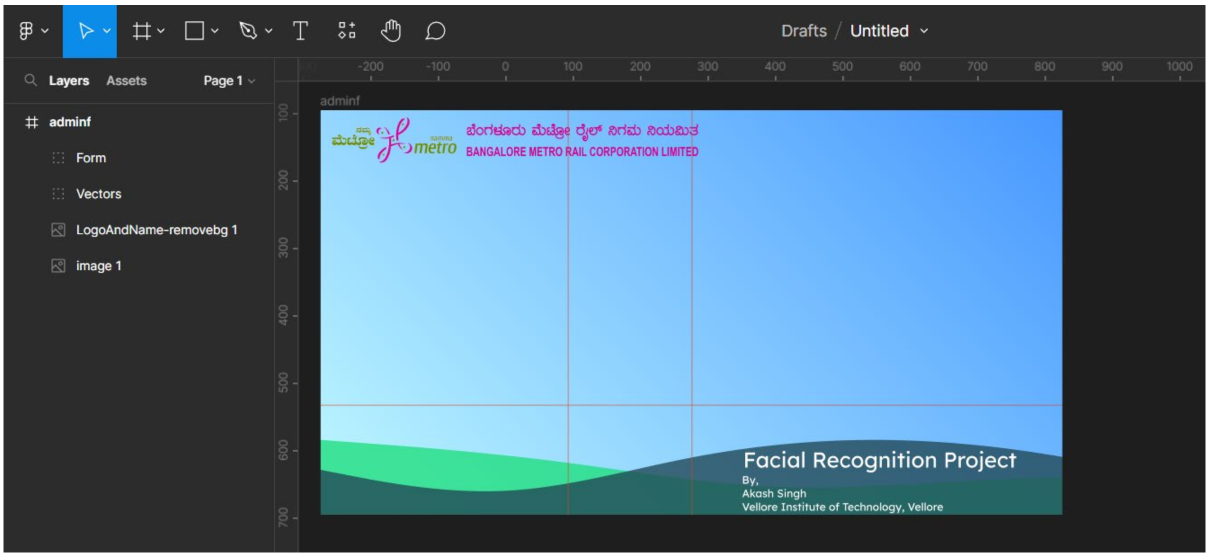
7. UI/ UX Overview

Figma has been used to create the background of each window as it provides great customization and visual effects. The colour scheme chosen is shades of Green, Purple and Blue as these are the BMRCL logo colours and are synonymous with the company.

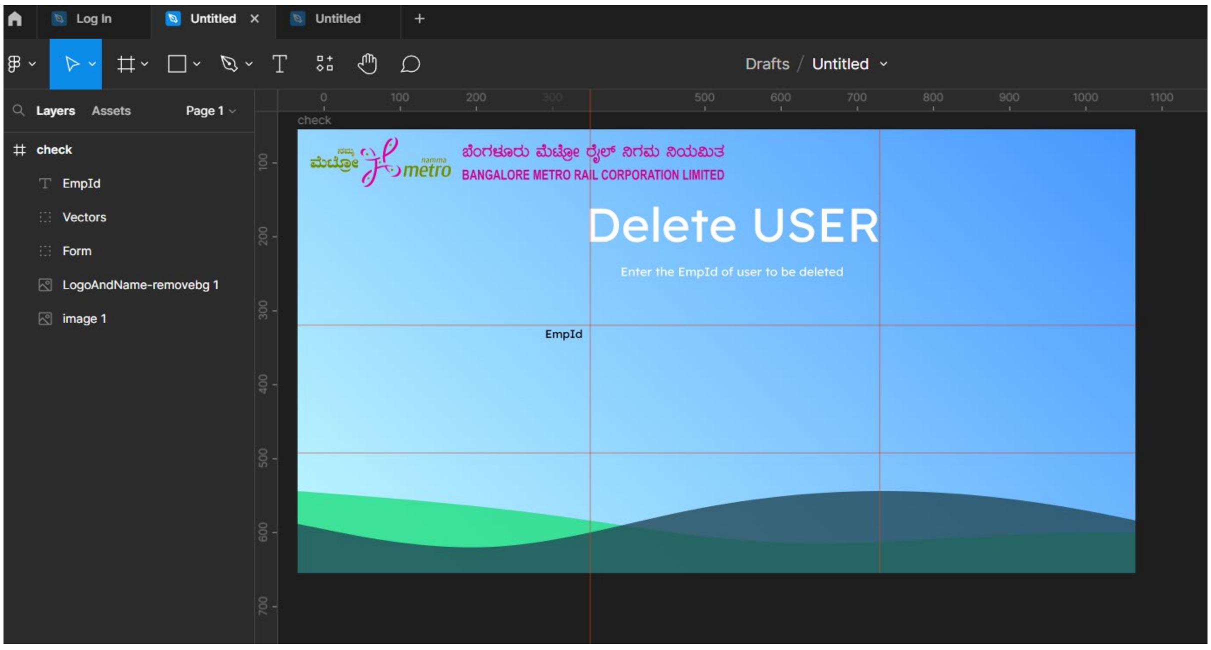
For the background I created a simple gradient of blue which two wave vectors at the bottom of the window to add more character to the design.

The BMRCL logo is places in the top left corner of the window which gives the design a clean look.

The font used is Lexend Deca and is used to create the text which can be seen throughout the Program.



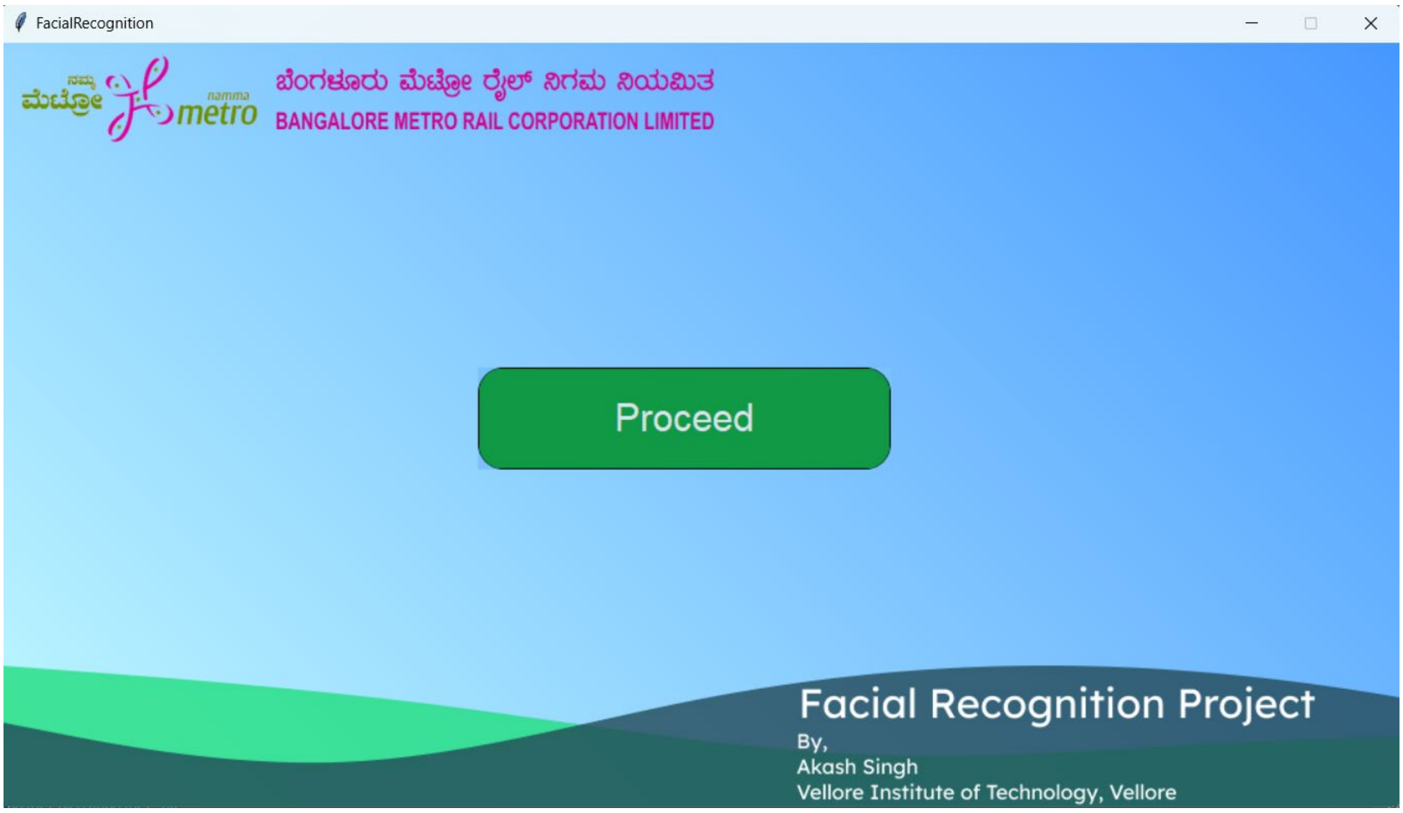
Facial Recognition Project
By,
Akash Singh
Vellore Institute of Technology, Vellore



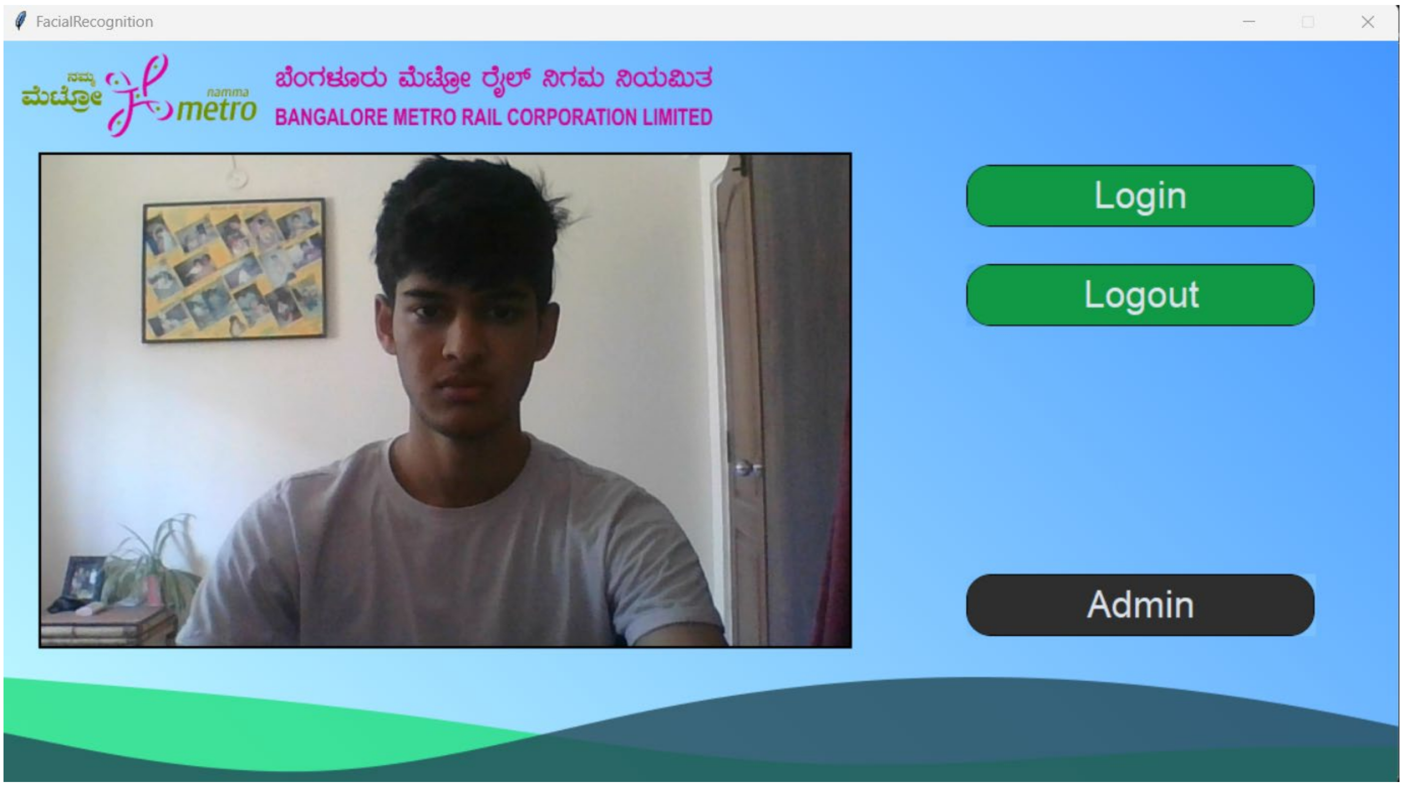
Delete USER
Enter the EmpId of user to be deleted
EmpId

8. Running the Software

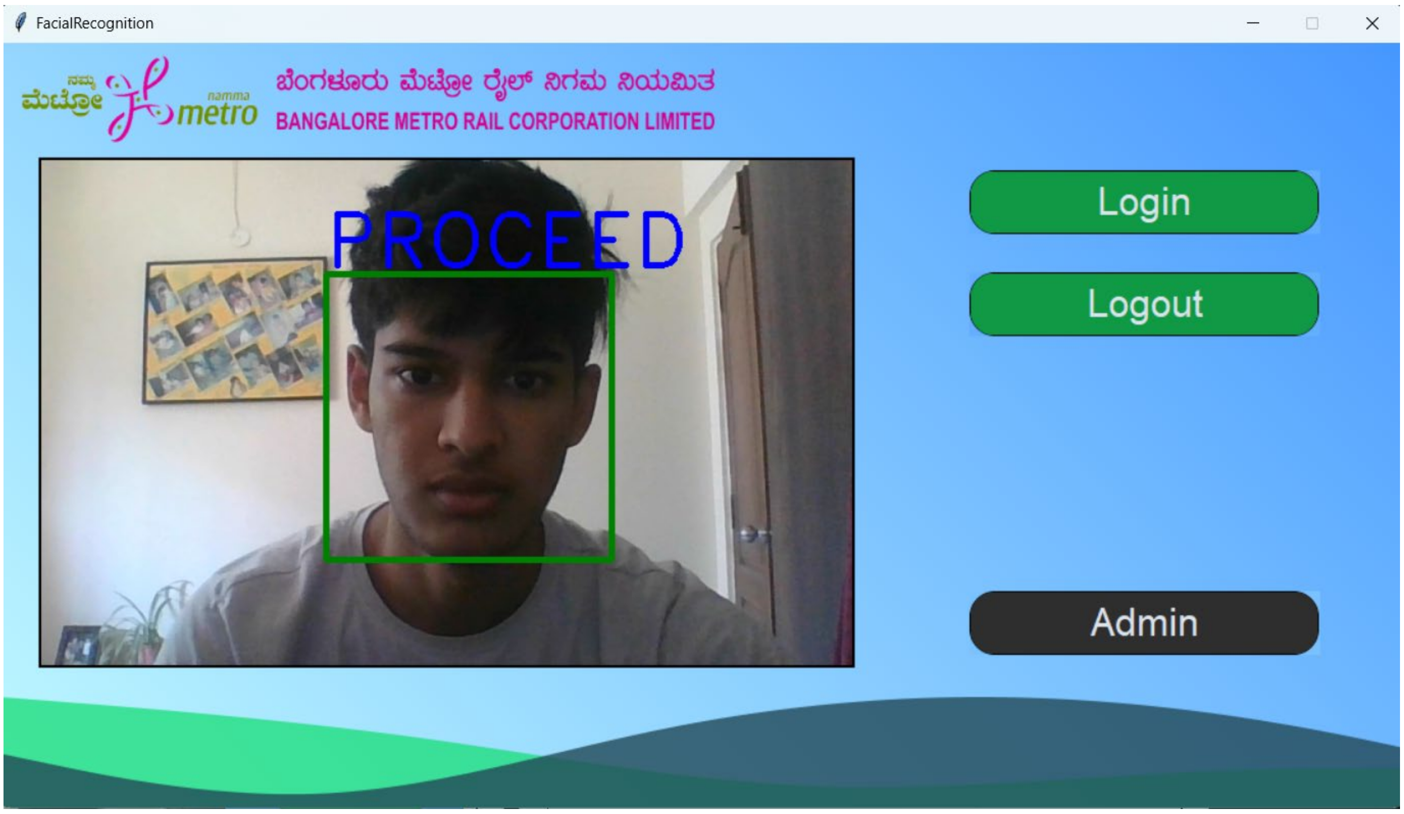
Landing page


FacialRecognition
ಬೆಂಗಳೂರು ಮೆಟ್ರೋ ರೈಲ್ ನಿಗಮ ನಿಯಮಿತ
BANGALORE METRO RAIL CORPORATION LIMITED
Proceed
Facial Recognition Project
By,
Akash Singh
Vellore Institute of Technology, Vellore

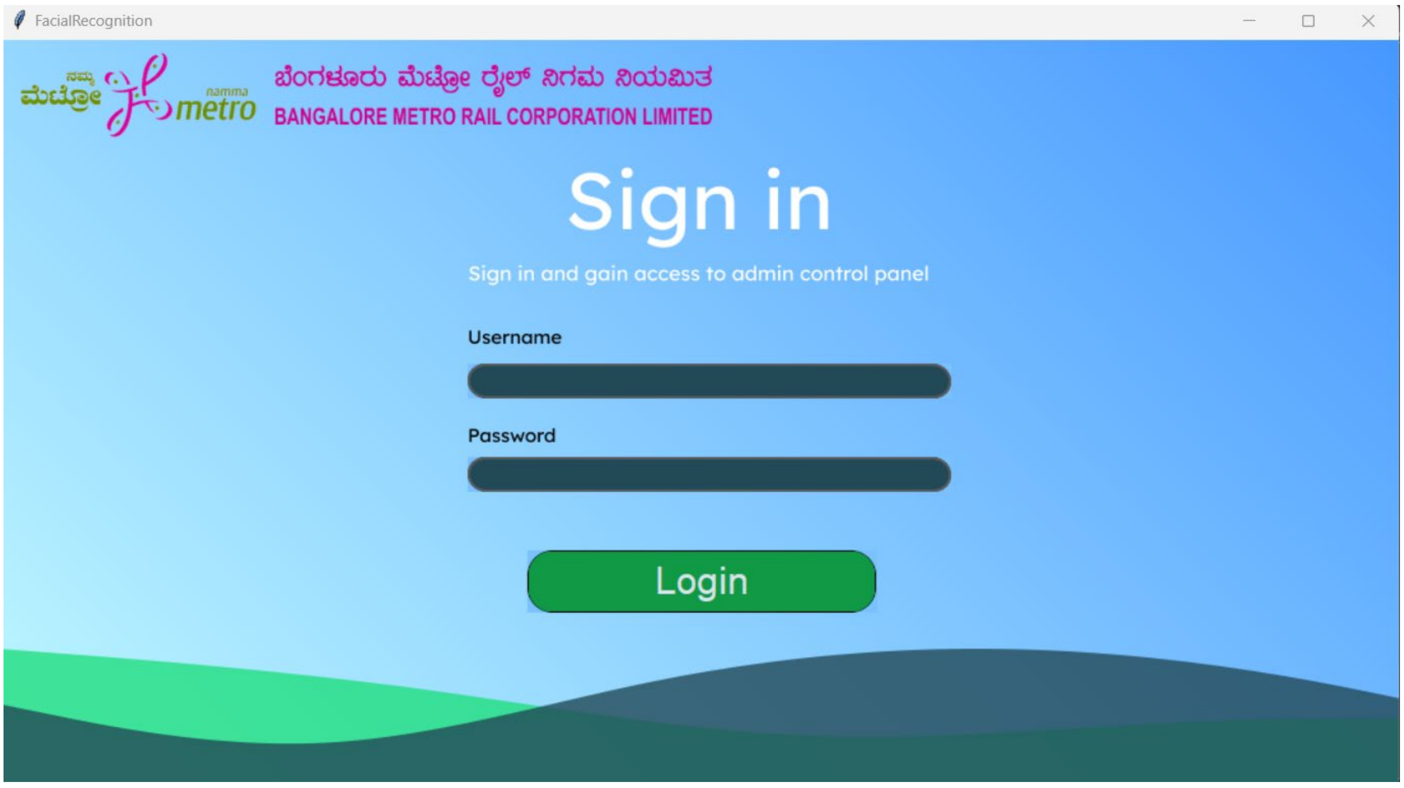
Main Window


FacialRecognition
ಬೆಂಗಳೂರು ಮೆಟ್ರೋ ರೈಲ್ ನಿಗಮ ನಿಯಮಿತ
BANGALORE METRO RAIL CORPORATION LIMITED
Login
Logout
Admin

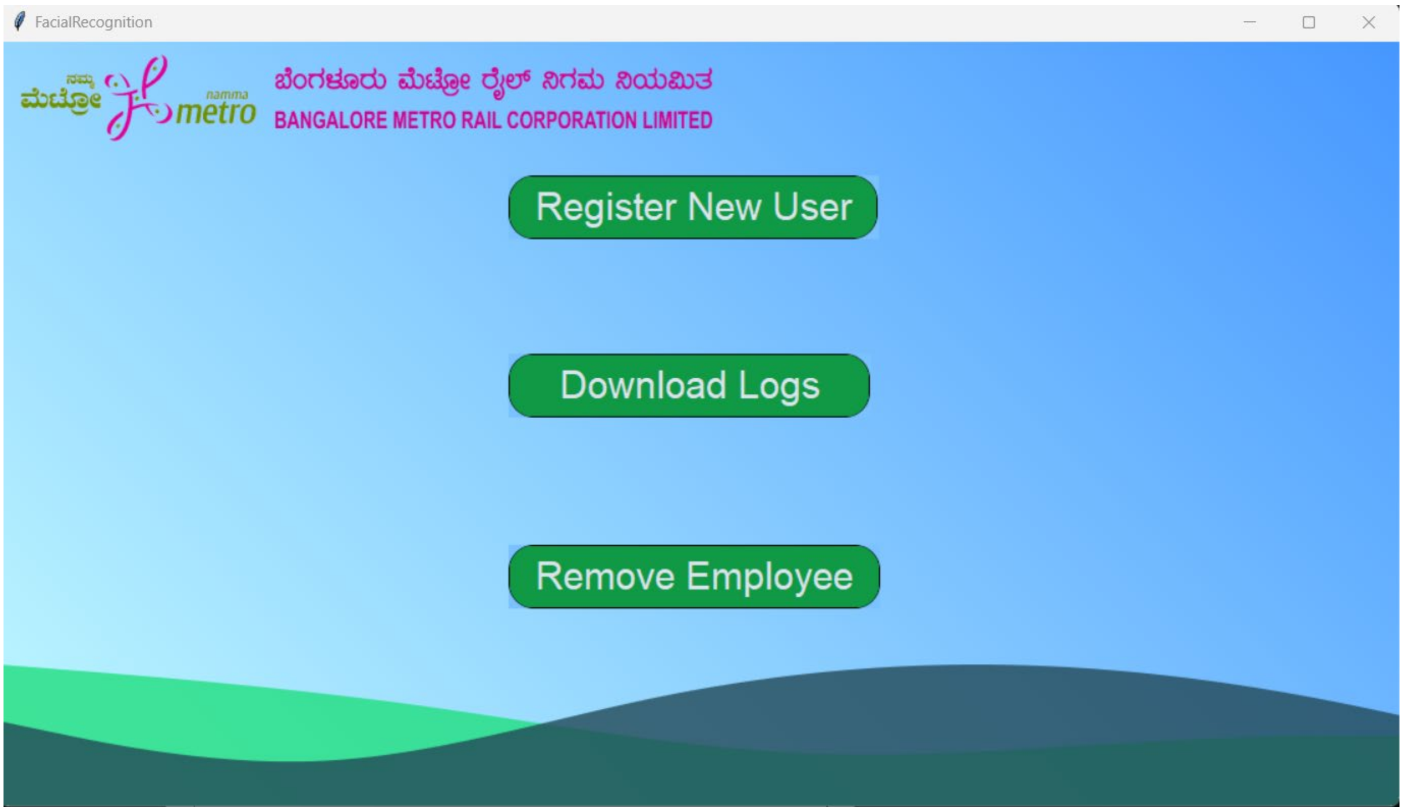
Facial Detection


FacialRecognition
ಬೆಂಗಳೂರು ಮೆಟ್ರೋ ರೈಲ್ ನಿಗಮ ನಿಯಮಿತ
BANGALORE METRO RAIL CORPORATION LIMITED
PROCEED
Login
Logout
Admin

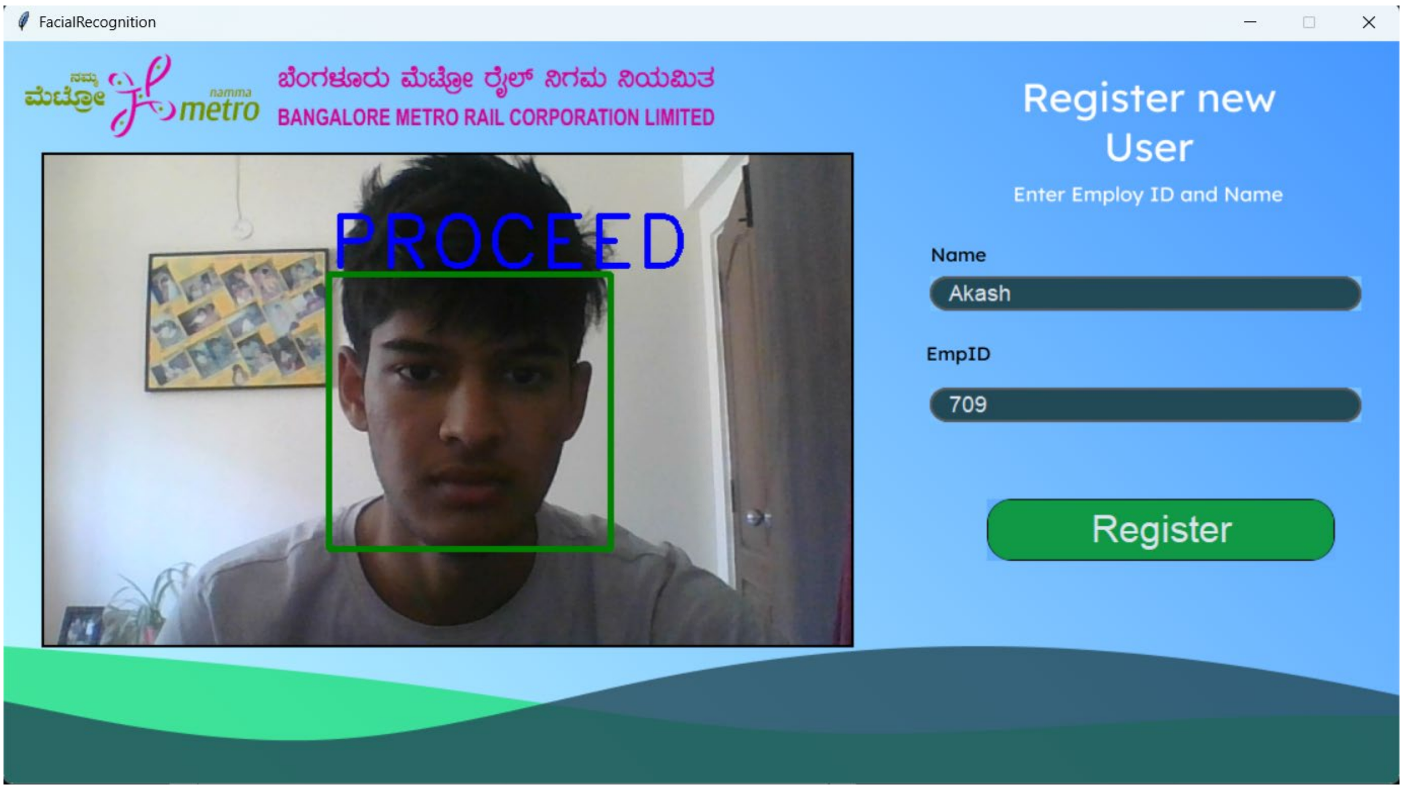
Admin Login


FacialRecognition
ಬೆಂಗಳೂರು ಮೆಟ್ರೋ ರೈಲ್ ನಿಗಮ ನಿಯಮಿತ
BANGALORE METRO RAIL CORPORATION LIMITED
Sign in
Sign in and gain access to admin control panel
Username
Password
Login

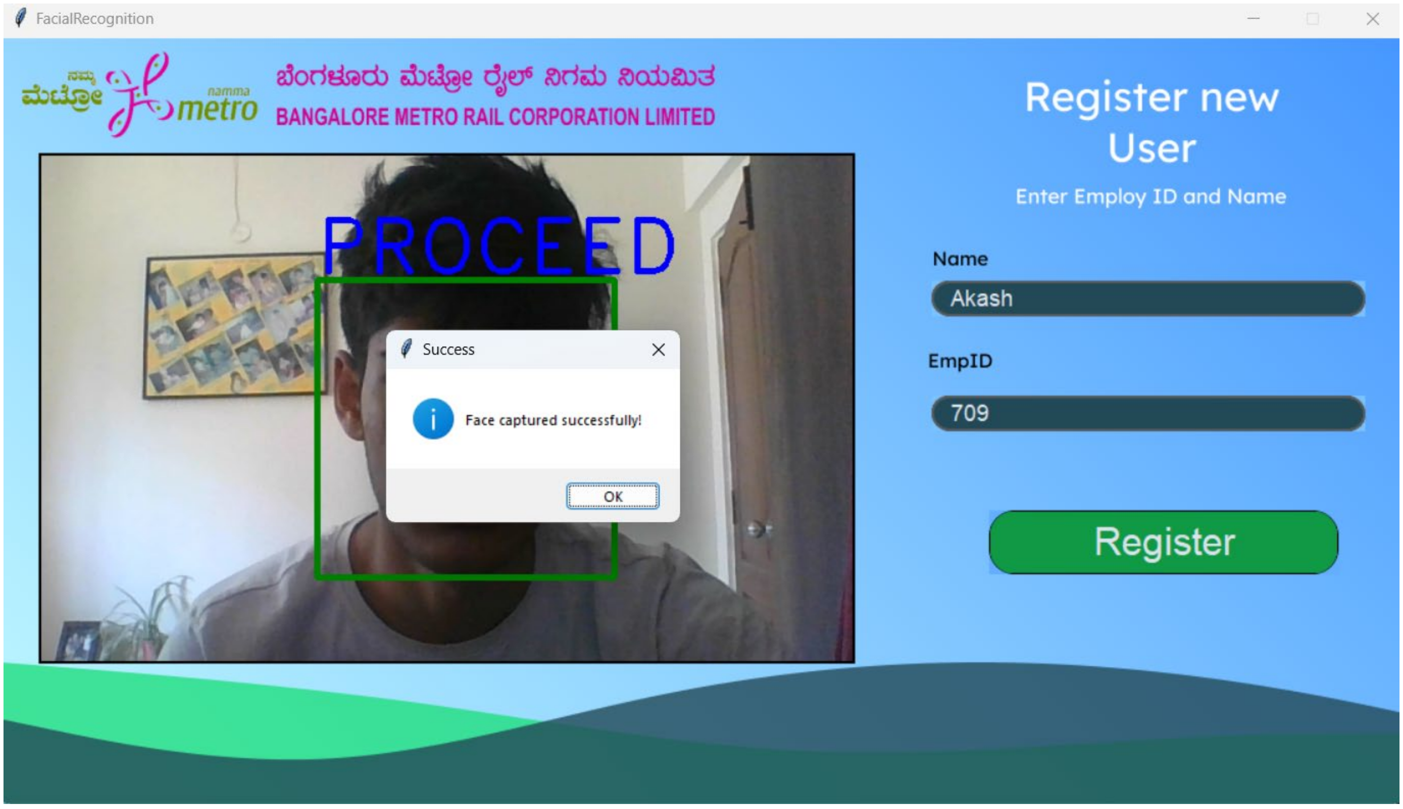
Admin Options


FacialRecognition
ಬೆಂಗಳೂರು ಮೆಟ್ರೋ ರೈಲ್ ನಿಗಮ ನಿಯಮಿತ
BANGALORE METRO RAIL CORPORATION LIMITED
Register New User
Download Logs
Remove Employee

Register New User



Name: Akash
EmpID: 709

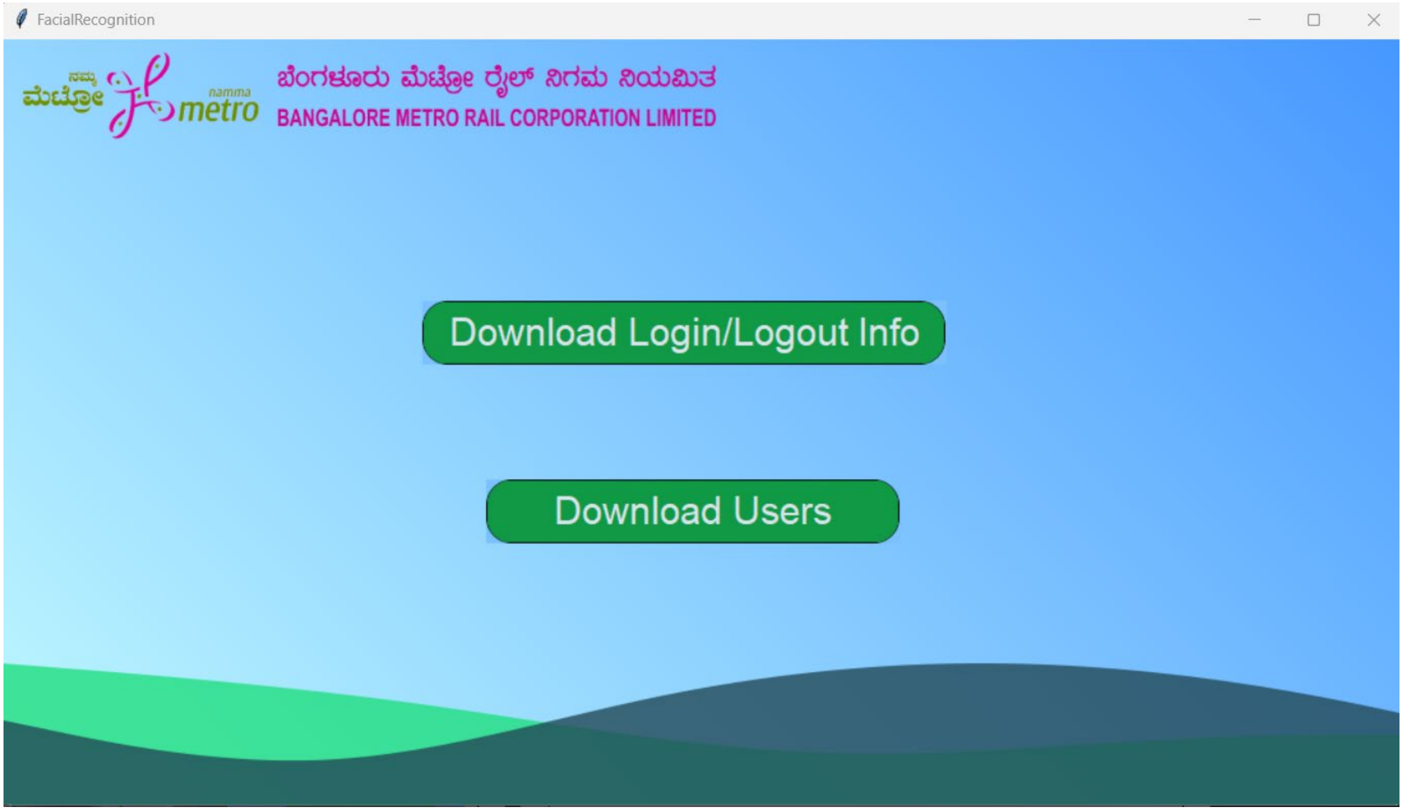


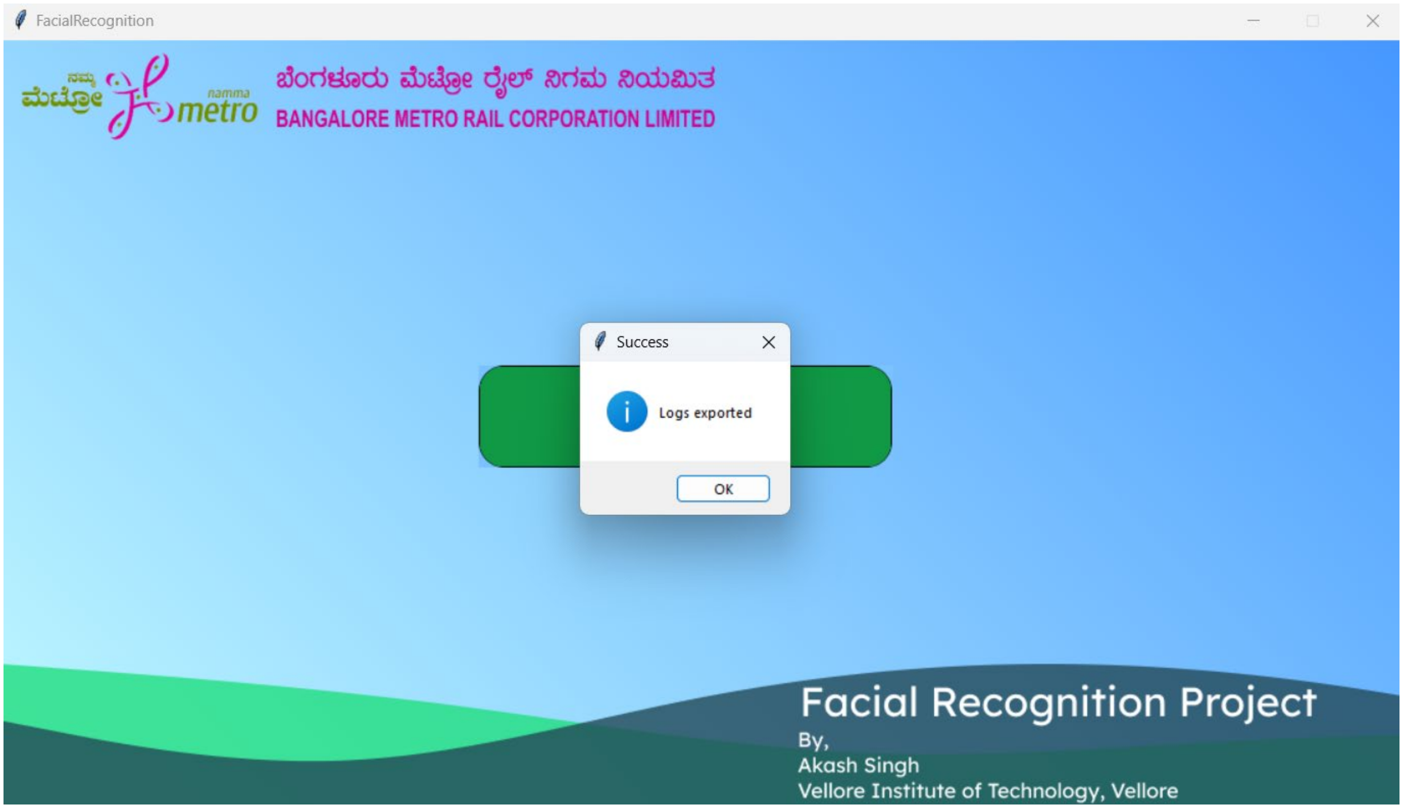
Face captured successfully!

Name: Akash

EmpID: 709

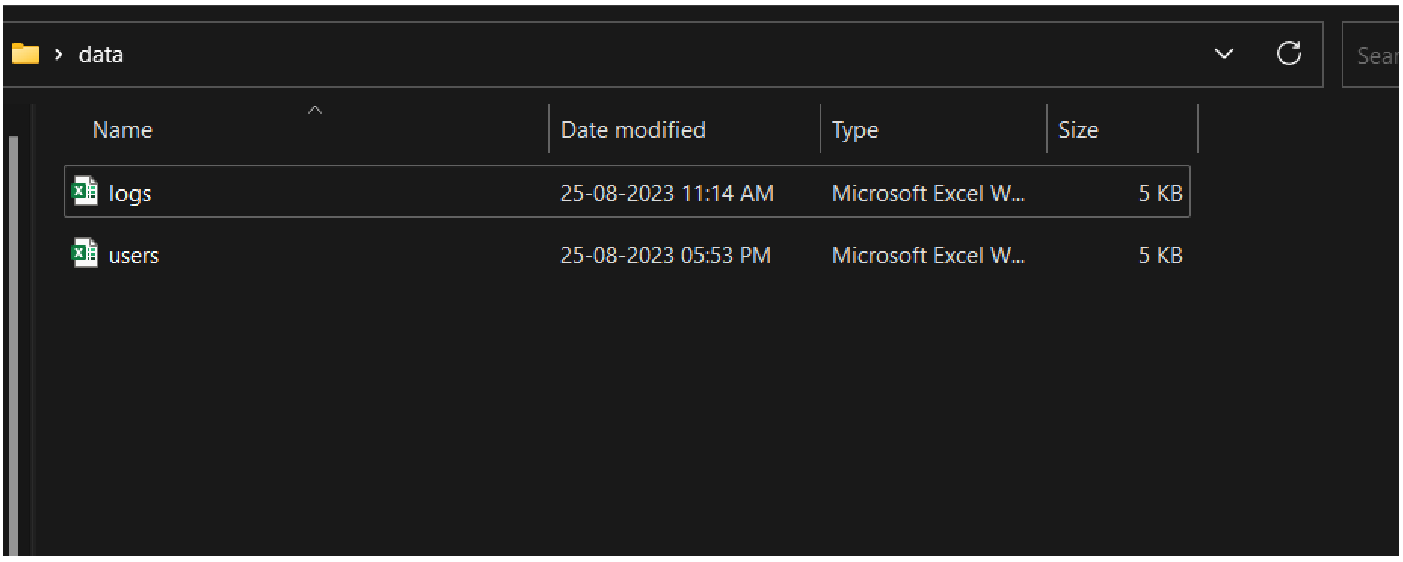
Download Logs


FacialRecognition
ಬೆಂಗಳೂರು ಮೆಟ್ರೋ ರೈಲ್ ನಿಗಮ ನಿಯಮಿತ
BANGALORE METRO RAIL CORPORATION LIMITED
Download Login/Logout Info
Download Users


FacialRecognition
ಬೆಂಗಳೂರು ಮೆಟ್ರೋ ರೈಲ್ ನಿಗಮ ನಿಯಮಿತ
BANGALORE METRO RAIL CORPORATION LIMITED
Success
OK
Facial Recognition Project
By,
Akash Singh
Vellore Institute of Technology, Vellore

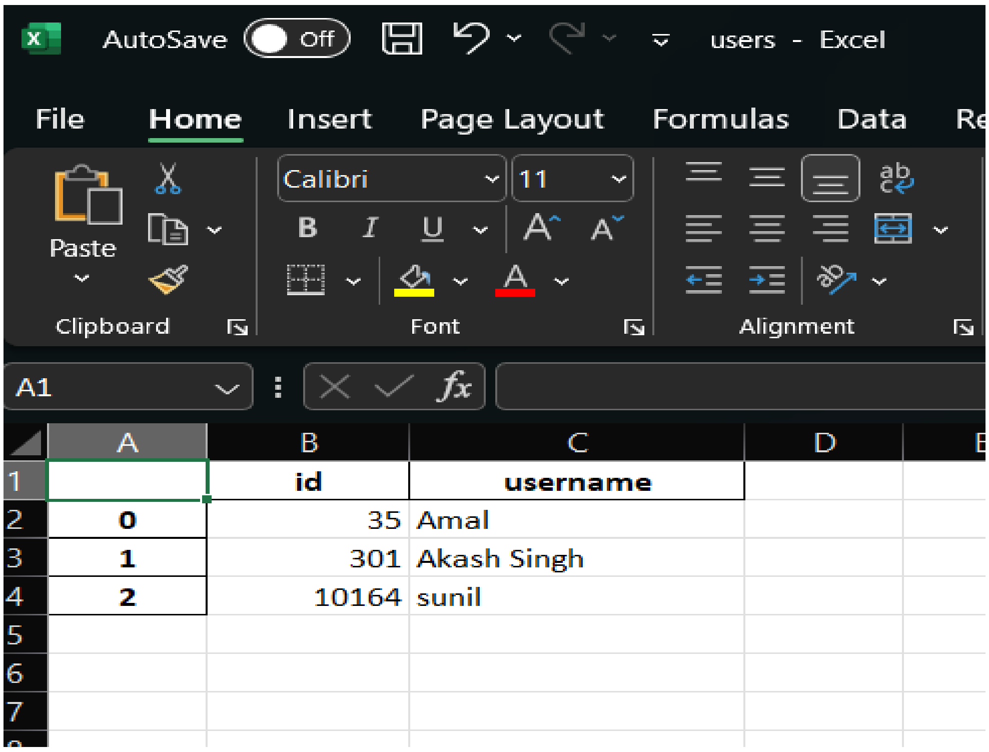
Logs exported

Data Stored as Excel



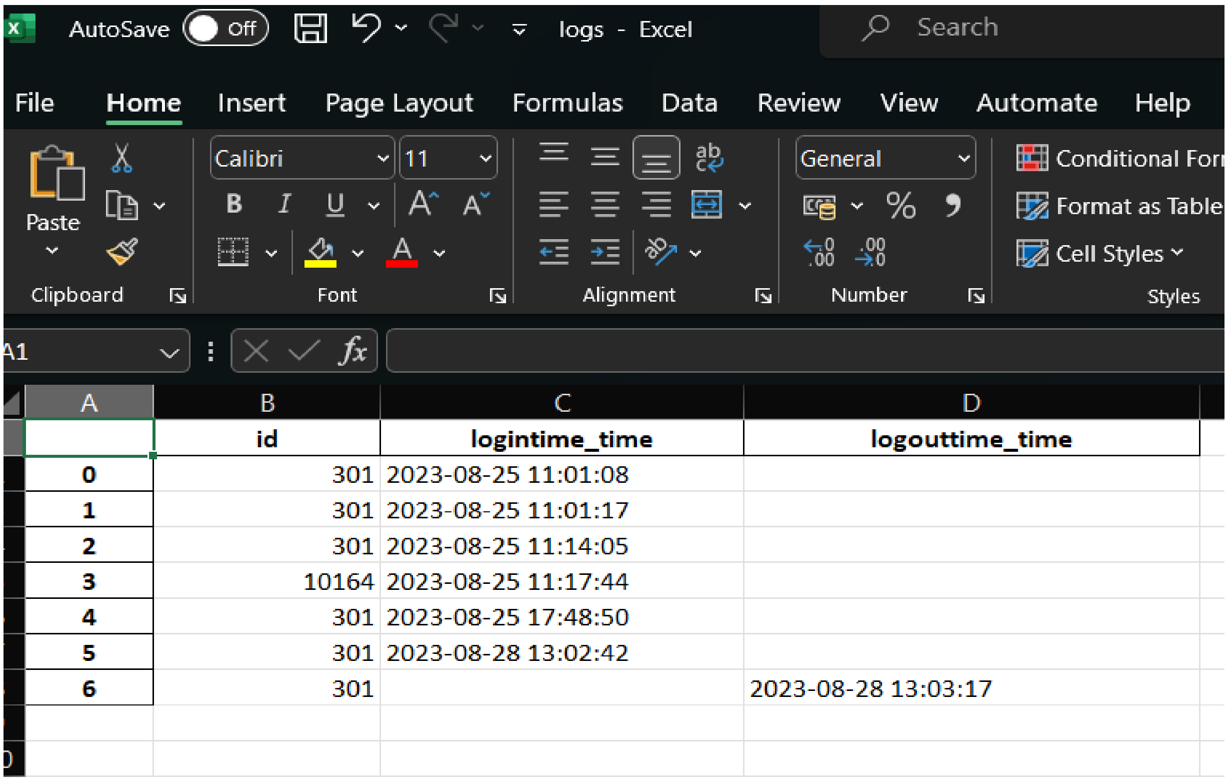
Name	Date modified	Type	Size
logs	25-08-2023 11:14 AM	Microsoft Excel W...	5 KB
users	25-08-2023 05:53 PM	Microsoft Excel W...	5 KB

Users



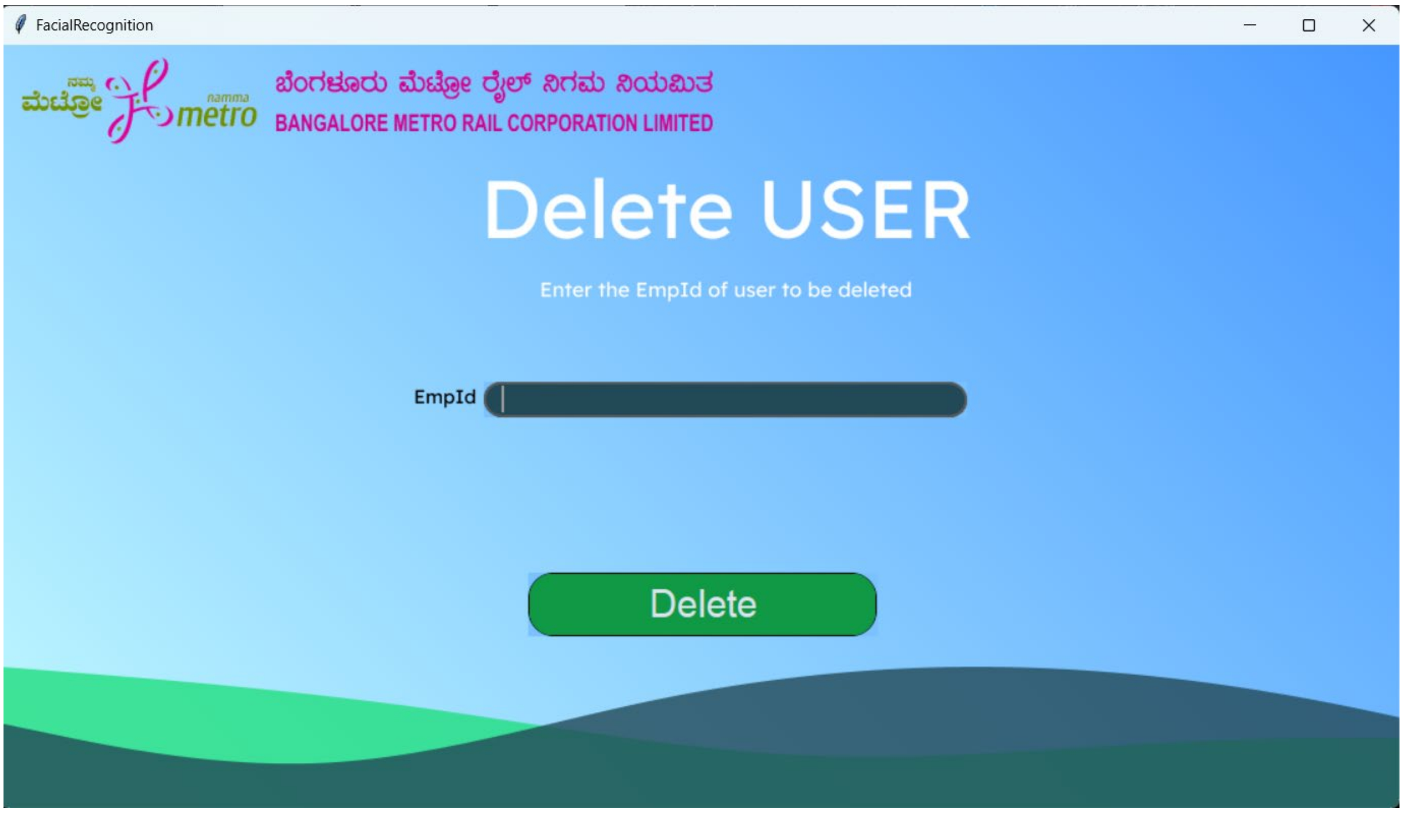
	id	username
0	35	Amal
1	301	Akash Singh
2	10164	sunil

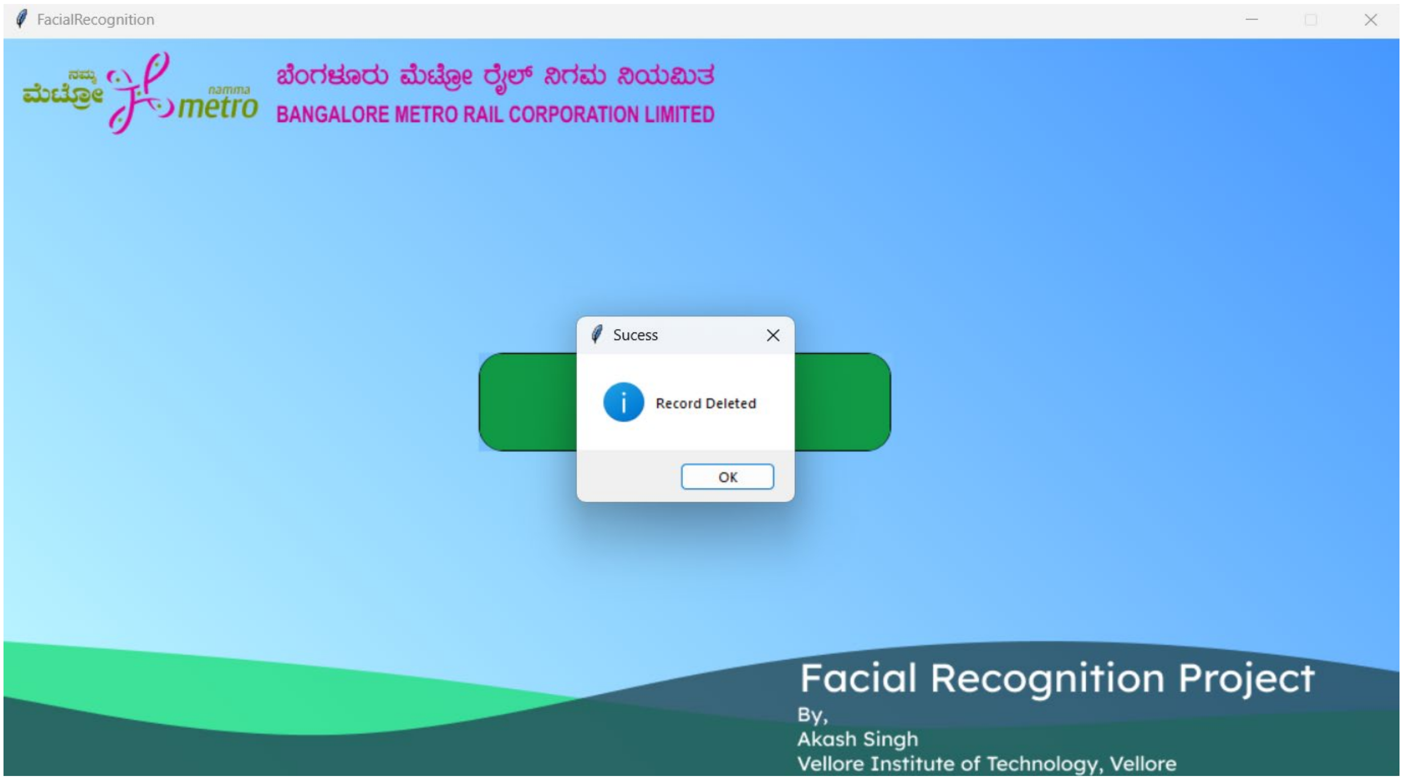
Login/Logout Information



	id	logintime_time	logouttime_time
0	301	2023-08-25 11:01:08	
1	301	2023-08-25 11:01:17	
2	301	2023-08-25 11:14:05	
3	10164	2023-08-25 11:17:44	
4	301	2023-08-25 17:48:50	
5	301	2023-08-28 13:02:42	
6	301		2023-08-28 13:03:17

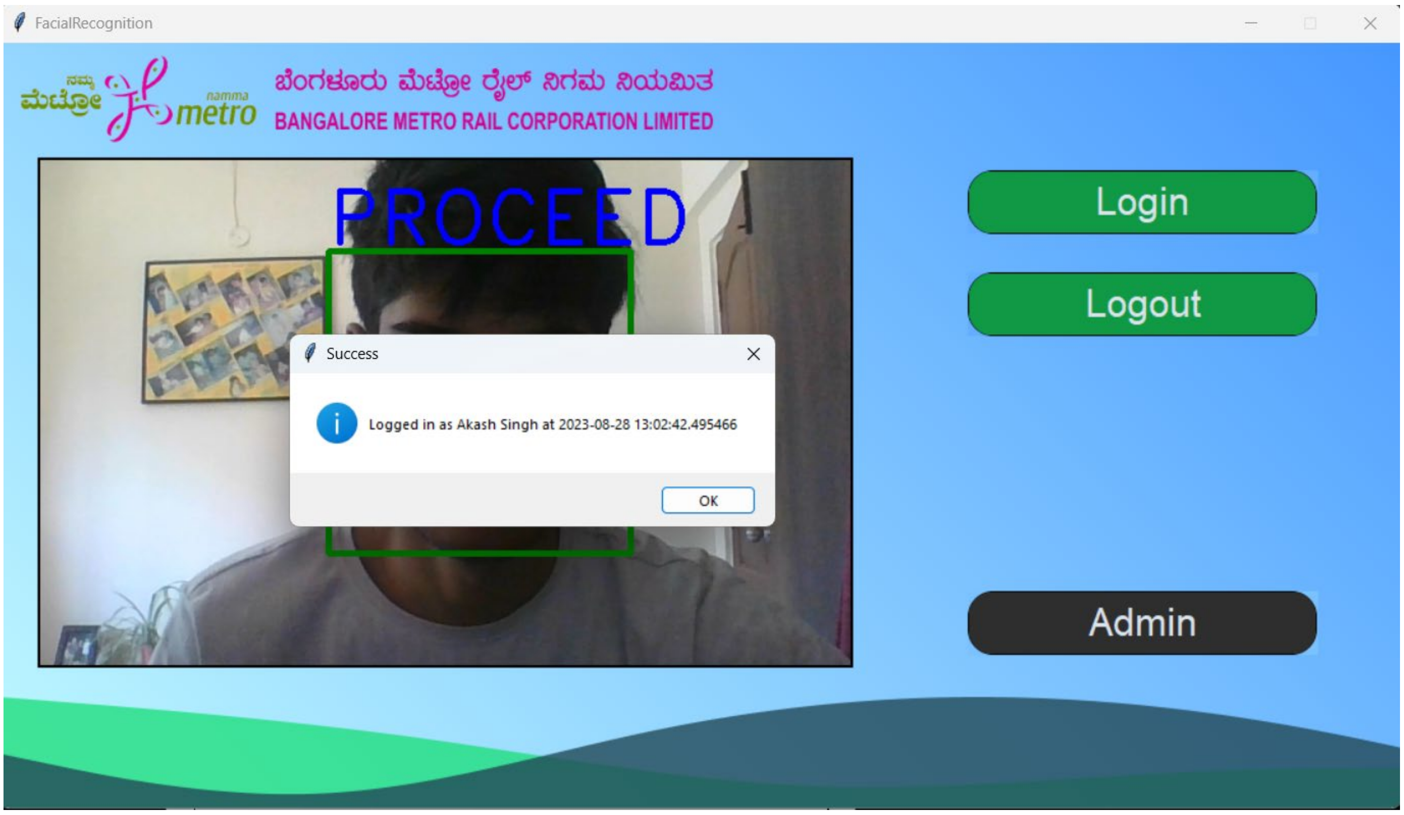
Delete Employee


FacialRecognition
ಬೆಂಗಳೂರು ಮೆಟ್ರೋ ರೈಲ್ ನಿಗಮ ನಿಯಮಿತ
BANGALORE METRO RAIL CORPORATION LIMITED
Delete USER
Enter the EmpId of user to be deleted
EmpId
Delete


FacialRecognition
ಬೆಂಗಳೂರು ಮೆಟ್ರೋ ರೈಲ್ ನಿಗಮ ನಿಯಮಿತ
BANGALORE METRO RAIL CORPORATION LIMITED
Sucess
OK
Facial Recognition Project
By,
Akash Singh
Vellore Institute of Technology, Vellore

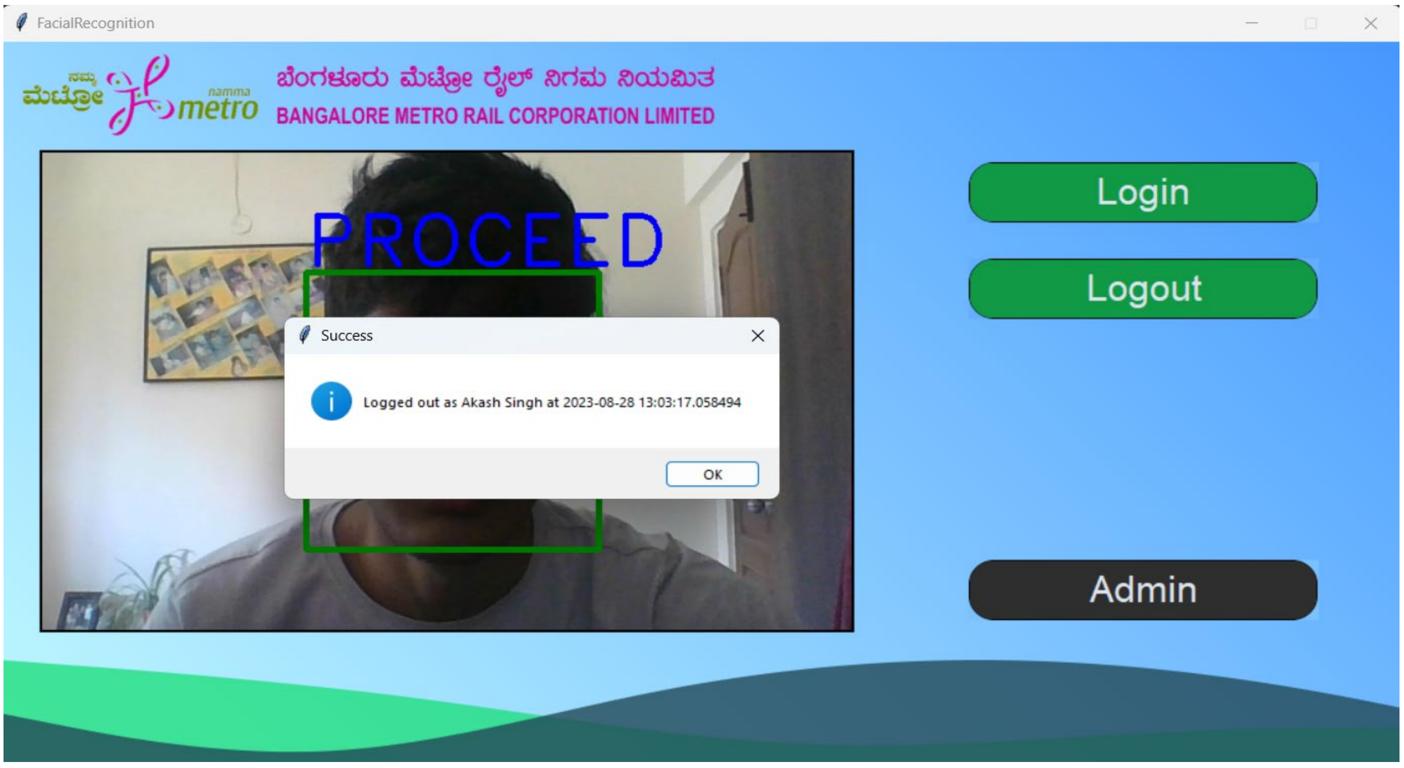
Record Deleted

Login


FacialRecognition
ಬೆಂಗಳೂರು ಮೆಟ್ರೋ ರೈಲ್ ನಿಗಮ ನಿಯಮಿತ
BANGALORE METRO RAIL CORPORATION LIMITED
PROCEED
Login
Logout
Admin
Success
OK

Logged in as Akash Singh at 2023-08-28 13:02:42.495466

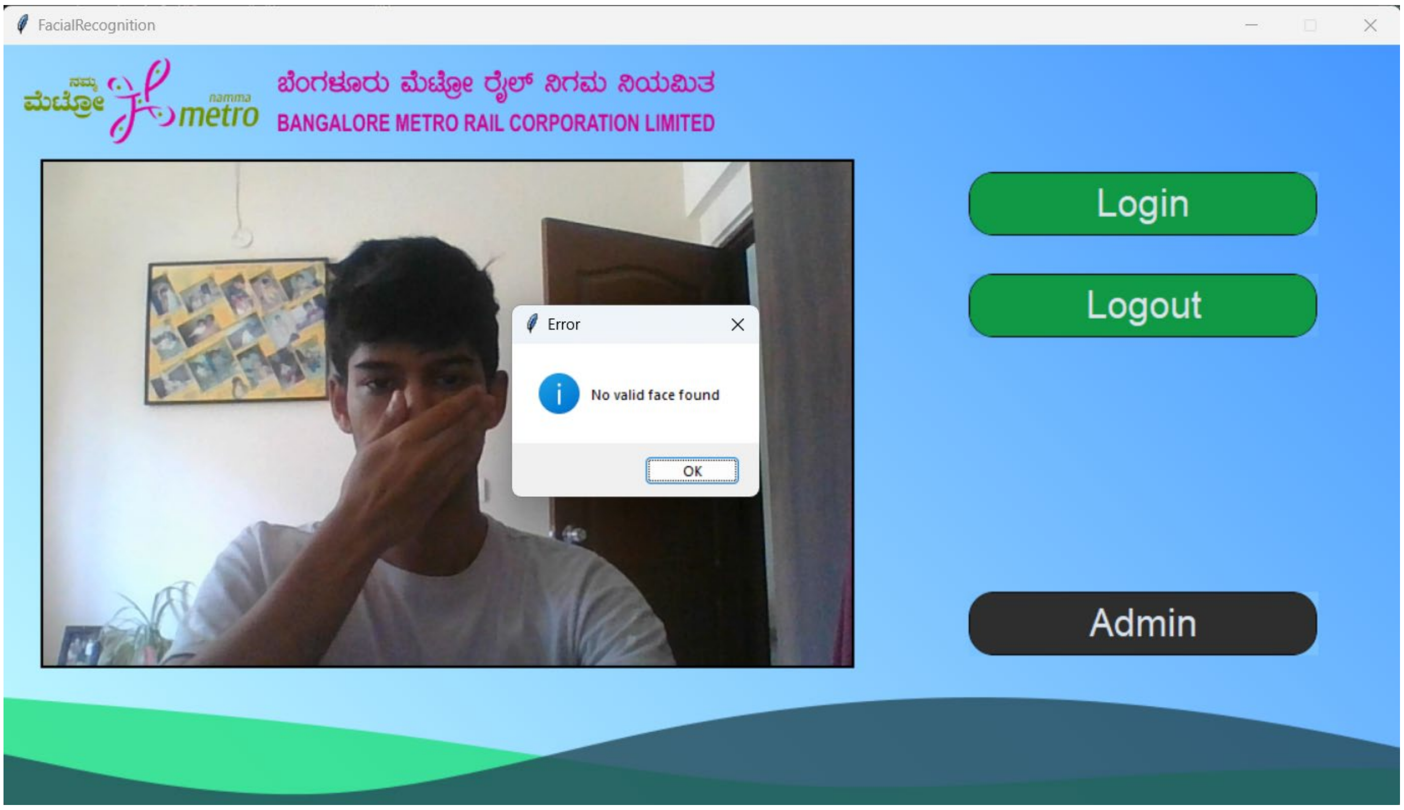
Logout


FacialRecognition
ಬೆಂಗಳೂರು ಮೆಟ್ರೋ ರೈಲ್ ನಿಗಮ ನಿಯಮಿತ
BANGALORE METRO RAIL CORPORATION LIMITED
PROCEED
Login
Logout
Admin
Success
OK

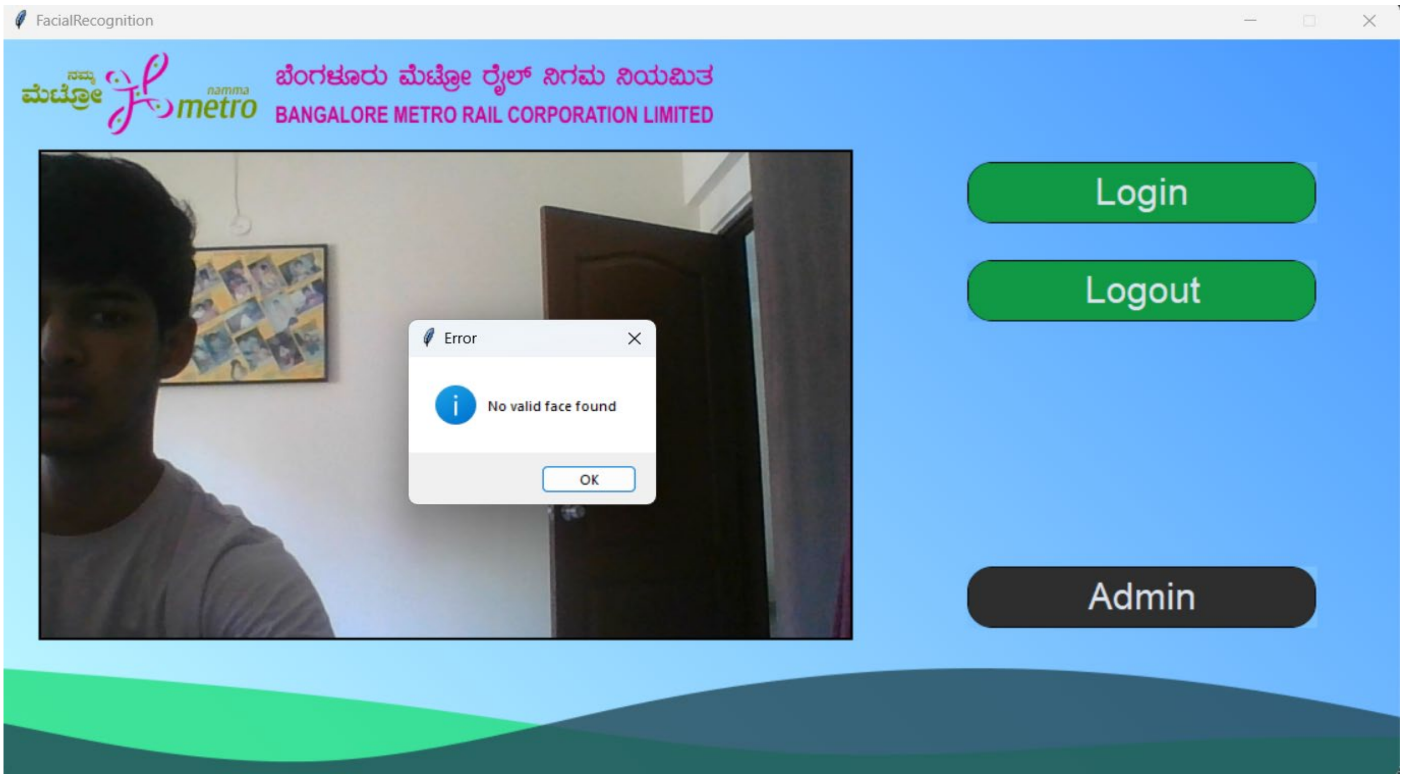
Logged out as Akash Singh at 2023-08-28 13:03:17.058494

9. Error Handling

No Face Found Error if no face is detected or recognized

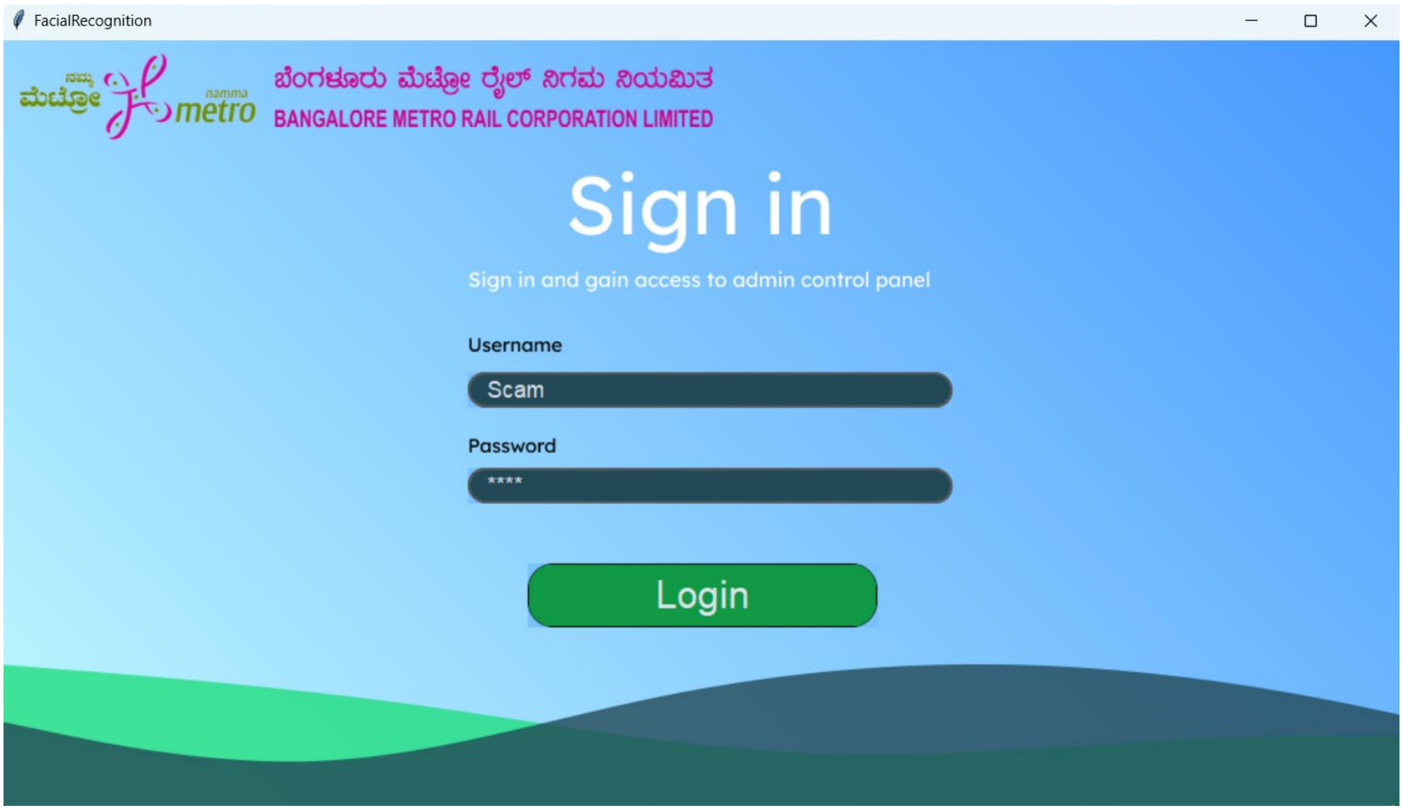

FacialRecognition
ಬೆಂಗಳೂರು ಮೆಟ್ರೋ ರೈಲ್ ನಿಗಮ ನಿಯಮಿತ
BANGALORE METRO RAIL CORPORATION LIMITED
Login
Logout
Admin
Error
OK

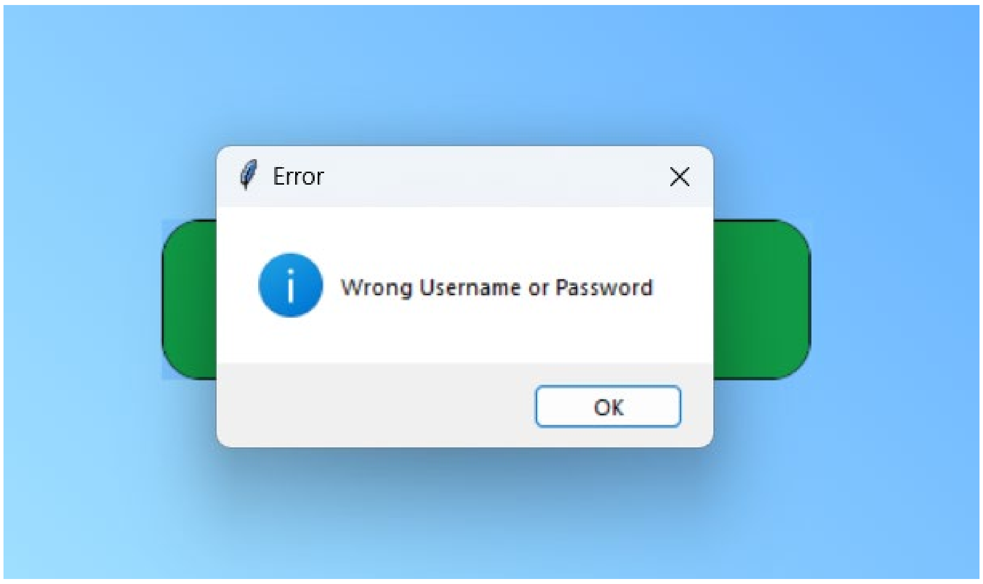
No valid face found


FacialRecognition
ಬೆಂಗಳೂರು ಮೆಟ್ರೋ ರೈಲ್ ನಿಗಮ ನಿಯಮಿತ
BANGALORE METRO RAIL CORPORATION LIMITED
Login
Logout
Admin
Error
OK

No valid face found

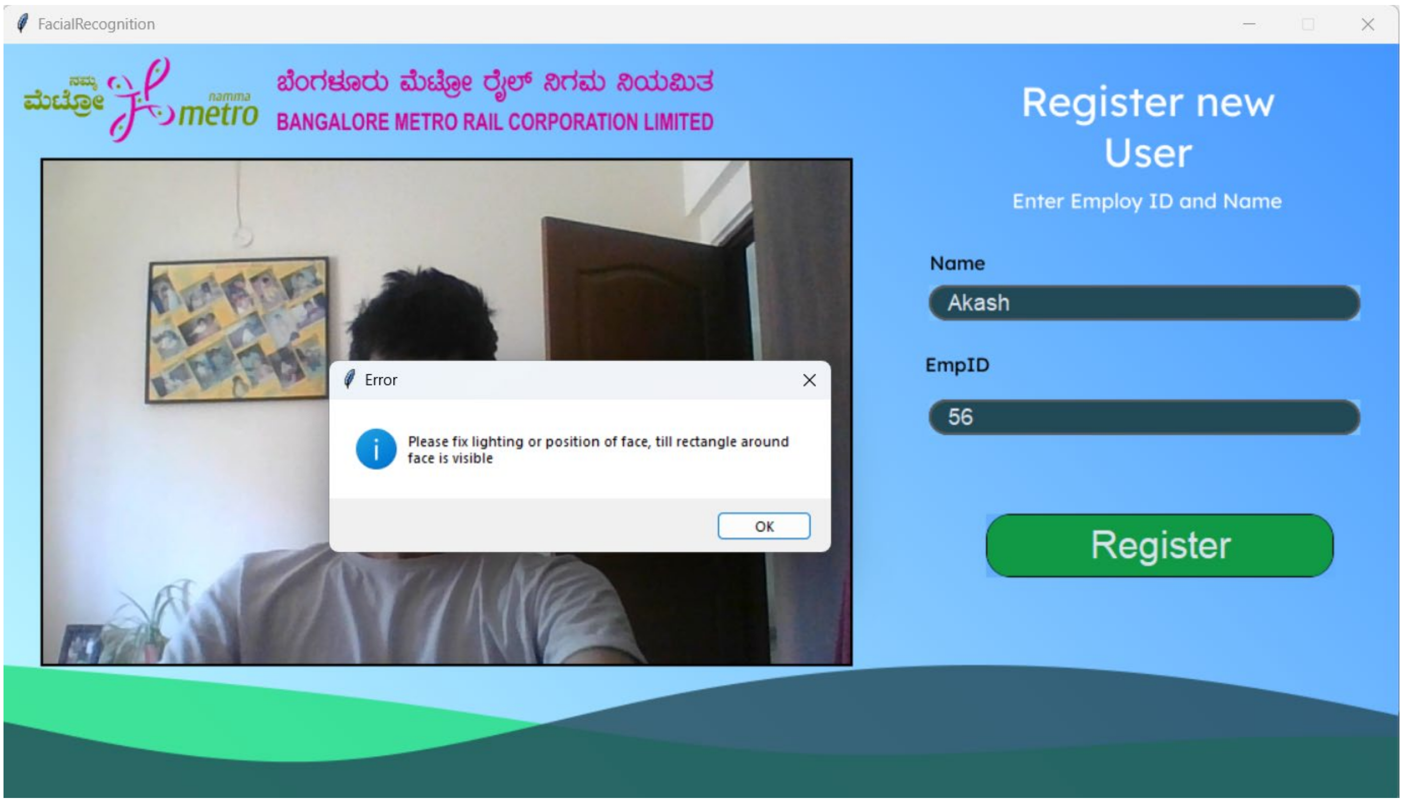
Wrong Admin Username/Password


FacialRecognition
ಬೆಂಗಳೂರು ಮೆಟ್ರೋ ರೈಲ್ ನಿಗಮ ನಿಯಮಿತ
BANGALORE METRO RAIL CORPORATION LIMITED
Sign in
Sign in and gain access to admin control panel
Username
Scam
Password
****
Login



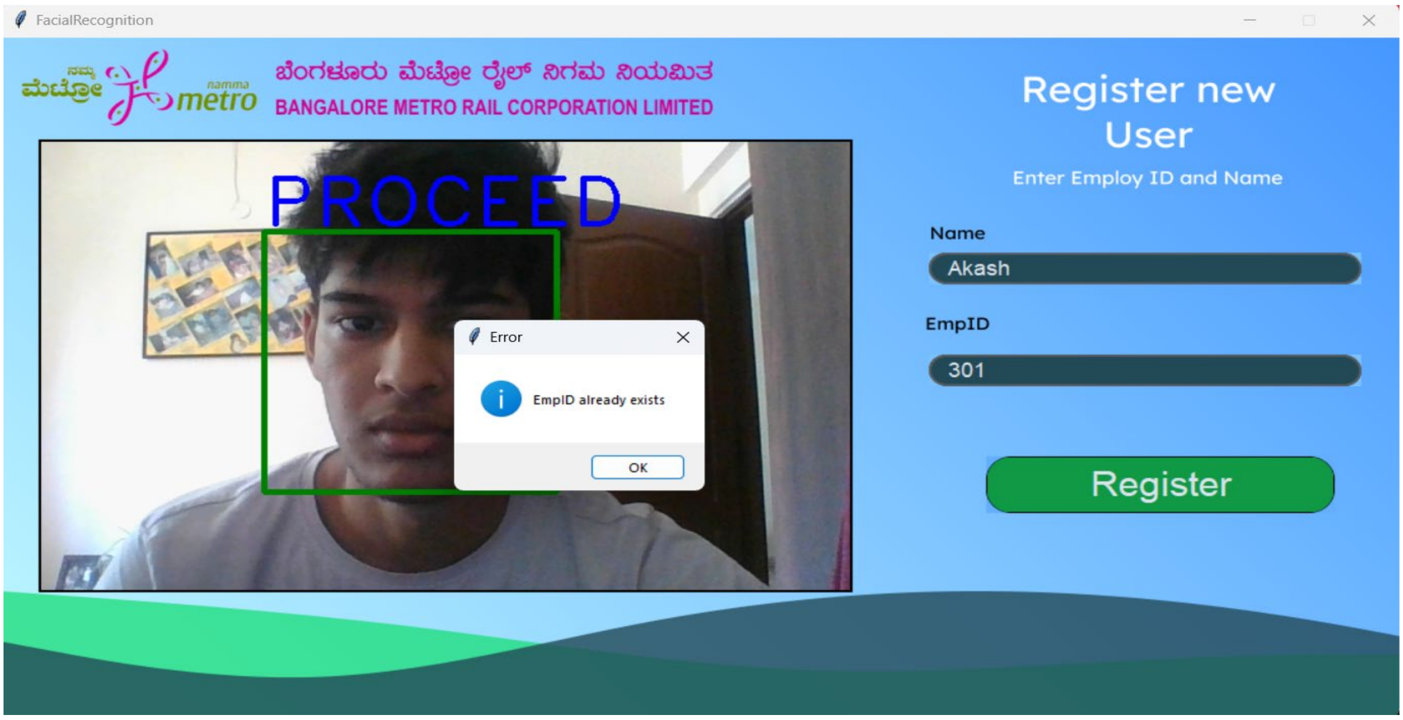
Wrong Username or Password

Bad Lighting or Head Position



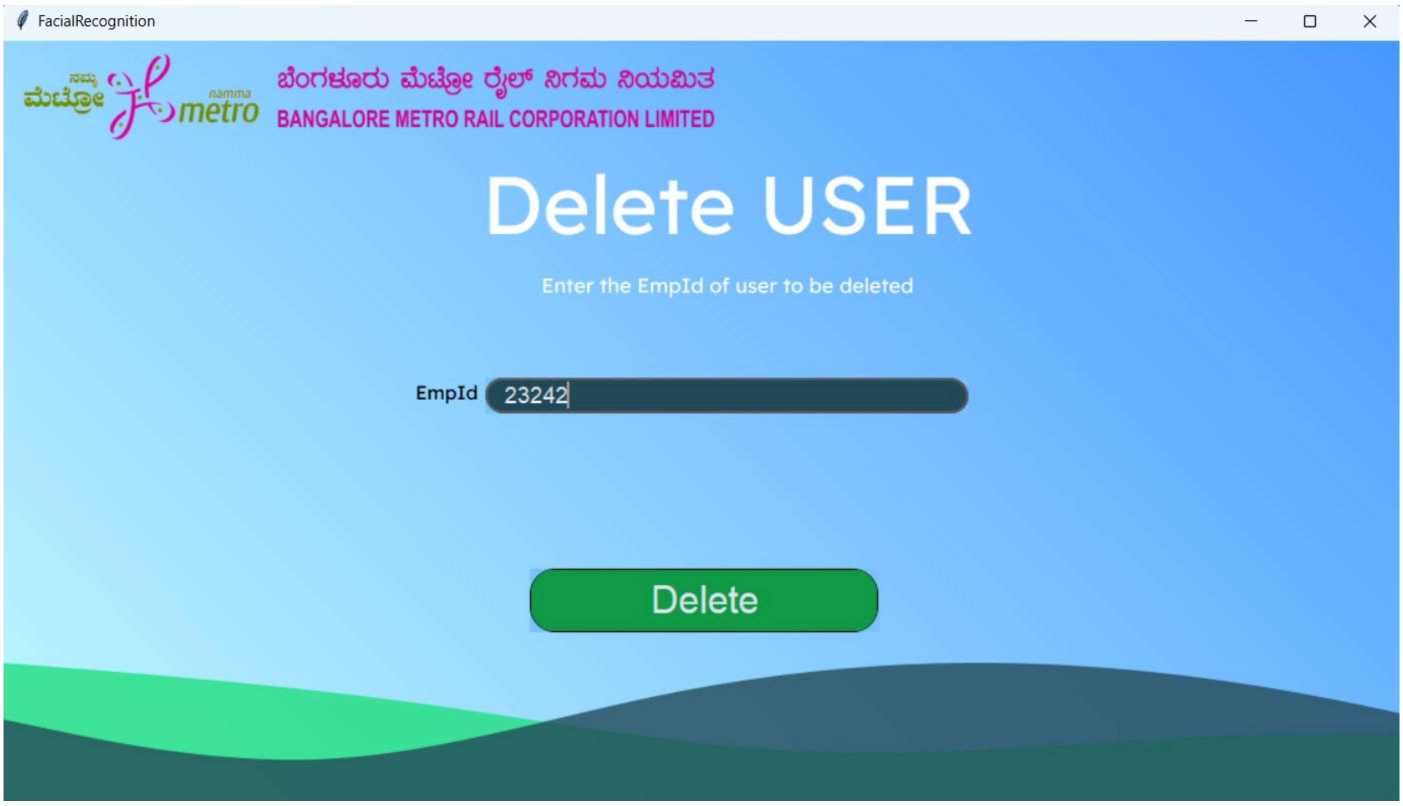
Please fix lighting or position of face, till rectangle around face is visible

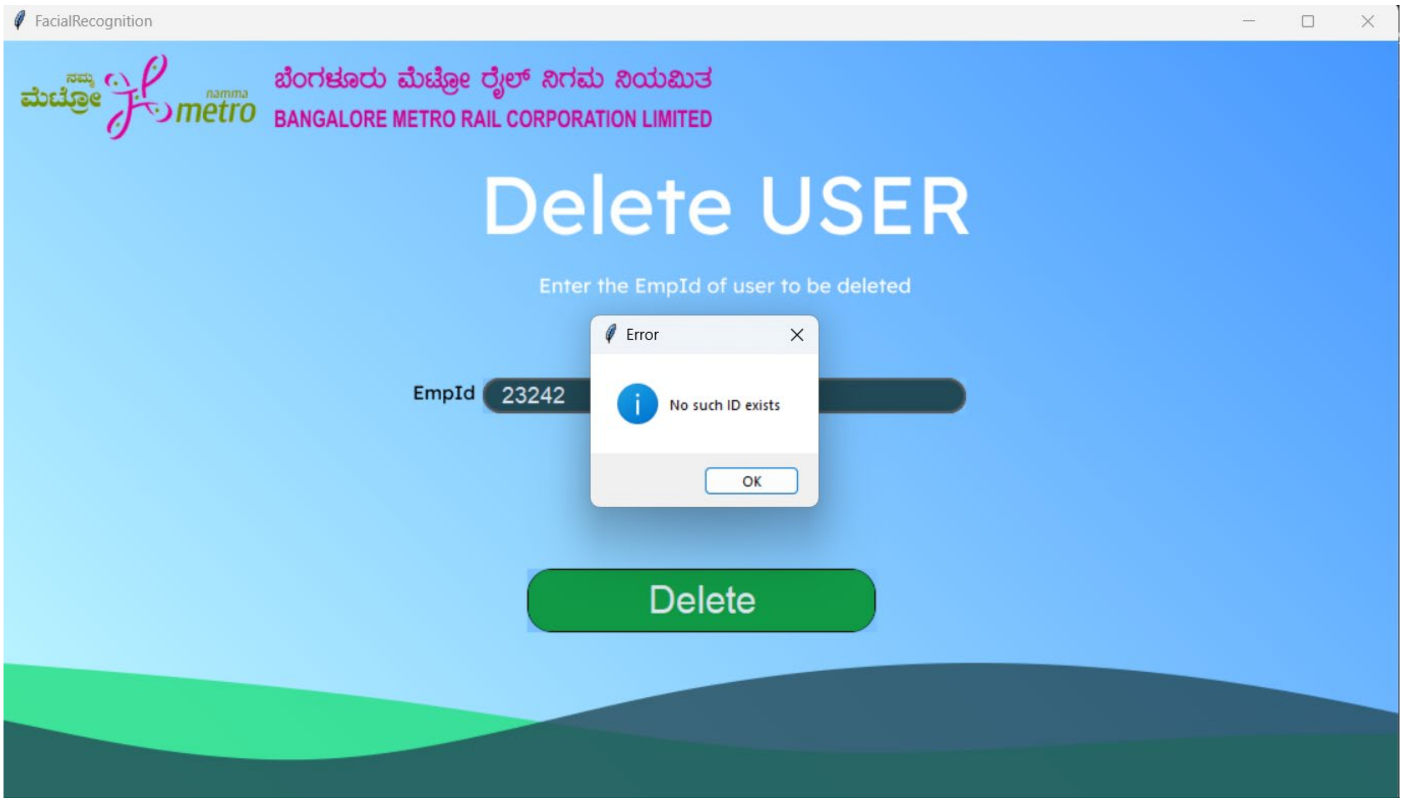
EmpId Already Exists in Database



EmpID already exists

Employee Id Does Not Exist


FacialRecognition
ಬೆಂಗಳೂರು ಮೆಟ್ರೋ ರೈಲ್ ನಿಗಮ ನಿಯಮಿತ
BANGALORE METRO RAIL CORPORATION LIMITED
Delete USER
Enter the EmpId of user to be deleted
EmpId
23242
Delete


FacialRecognition
ಬೆಂಗಳೂರು ಮೆಟ್ರೋ ರೈಲ್ ನಿಗಮ ನಿಯಮಿತ
BANGALORE METRO RAIL CORPORATION LIMITED
Delete USER
Enter the EmpId of user to be deleted
EmpId
23242
Error
OK
Delete

No such ID exists

10. Skills Acquired

- Before taking on this project, I did a lot of research on Machine Learning and AI as well as went in depth into computer vision which I found very fascinating. I did a course on Tensor Flow which taught me about various libraries I used in this project like NumPy and Pandas.
- As I planned to do the whole code using python I first brushed up on my python skills before starting the project.
- After starting I created my first version of the program without a good-looking UI and it was just a barebone implementation of the idea. Using Dlib to capture the face as well as to creating the face vectors which are stored and iterated through every time someone tries to login/logout.
- After implementing the login/logout system I focussed on learning SQLITE3 as this would be used as my database in the project. I have good understanding of SQL and its various queries.
- Once all these were in place, I also picked up on how to handle errors using various commands and logics. Making sure the software is robust.
- After all of this I turned towards making the software more user friendly and easy to use. For this I used Figma to design the backgrounds as well as to create a basic framework for each window. For the various buttons, labels, and inputs I used TKinter.
- Throughout the whole projects I even learned about various topics like OOPS in python, Arrays, Strings etc which further improved my knowledge.
- Visiting the Operational Control Centre of BMRCL also was an amazing experience I saw video analytics being used in a real-world environment and on a very large scale. It was fascinating to observe the technology used in modern metro.
- Completing this project in the short span of less than a month first appeared challenging and a daunting task however as I got more involved in the project I realized proper time management as well as it improved my problem solving skills.
- This project has made me very interested in the world of machine learning and computer vision and I will do various other projects in this field.