Table of Contents

Table of Figures

Table of Tables

Abstract:

The documentation shows the design and development of the face recognition system, which is used for access control in high-security places such as government buildings, security infrastructures, etc. This describes the research methodology, SDLC, design, implementation and testing of the system. Moreover, it raises concerns about ethical issues such as privacy and bias. This documentation accepts the security demands as well as individual rights by encouraging the development of access control for security buildings.

Introduction:

As the security measures should increase in the high-security buildings, a Facial recognition system for controlling access will be implemented so that there will be fewer threats and higher security in places like government buildings and other sensitive places. This system is specifically designed to control access to the security facilities.

Facial recognition uses advanced computer vision techniques and Machine learning algorithms for higher accuracy to identify people (Sunardi, 2022). This report discusses the features of how the system has been structured and what are the techniques, models and algorithms has been used. A detailed explanation of designing, implementation and testing will be shown, which includes the collection of data through webcam or device, processing of the data in the database and training of the model for facial identification.

For the accuracy of identifying people who are authorized to access the building can be done through testing, which will be done at last after design and implementation.

Moreover, the report studies the ethical aspects like violating people privacy by facial recognition. So, the paper also focuses on the governance structure and rules ensuring measures have been taken.

By addressing both technical as well as ethical concerns, the system will be developed for access control, which follows the demands of security by protecting individual rights.

Research methodology:

Aim: Developing an accurate and robust face recognition system that enhances access control in high-security buildings by taking ethical considerations of individuals.

Objectives:

1. Creating a user-friendly web application interface using the Streamlit library for the facial recognition system.

2. Designing the system architecture that will be secure and prevent unauthorized access.

3. Implementing computer vision techniques and Complex machine learning algorithms for the accuracy of detecting people in facial recognition.

4. Accuracy rates in matching the image in low lighting conditions, angles and expressions.

Research questions:

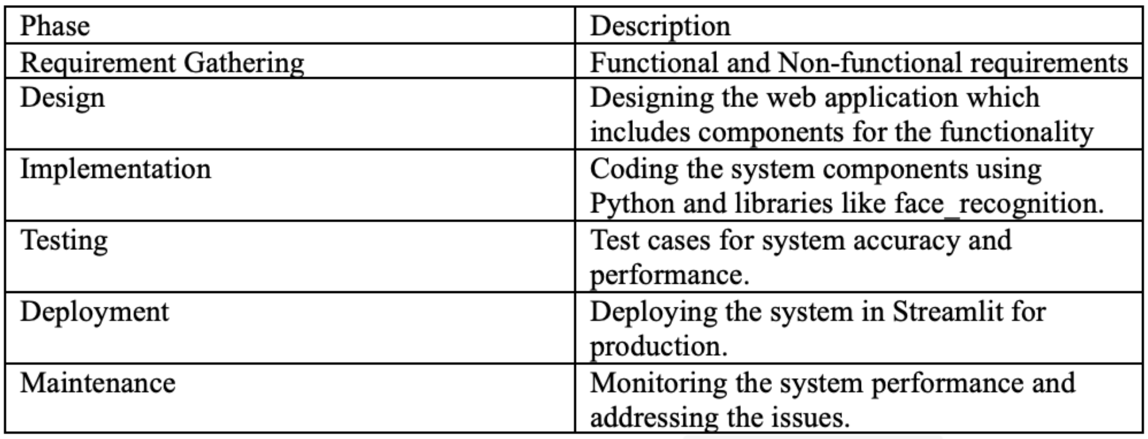
1. How can the system be designed to get accurate results in identifying the images in various conditions?

2. How did the Agile methodology contribute to the effective implementation and development of the system? (Jones, 2023)

3. How accurately does the ResNet algorithm work compared to the other algorithms? (Cantemir, 2024)

Software development life cycle

The documentation has followed the SDLC(Software development life cycle) process following several steps. Firstly, the requirement gathering (betsol, 2019) has been done where the functional and non-functional requirements have been identified. Also, the selection of the algorithm has been done by researching the articles, papers and websites. Then, designing of data acquisition, model training and integration has been done. The implementation happened using an agile method. Moreover, testing under different test cases has been done for the accurate results and analysis of the system. Below is the table that shows the documentation phases and the description.



Phase	Description
Requirement Gathering	Functional and Non-functional requirements
Design	Designing the web application which includes components for the functionality
Implementation	Coding the system components using Python and libraries like face_recognition.
Testing	Test cases for system accuracy and performance.
Deployment	Deploying the system in Streamlit for production.
Maintenance	Monitoring the system performance and addressing the issues.

Table 1: SDLC PHASES (Dhandapani, 2016)

Justification:

The Agile method has been used for this documentation, as it offers several benefits. The improvement and the adjustments have been done by taking the feedback of stakeholders, (Staff, 2023) which should be an iterative step in the Agile approach, as well as incremental nature gave the project accurate results, as several changes have been done after every feedback.

Agile Practice	Implementation
Iterative Dev	System developed iteratively in sprints, delivering components
Incremental Delivery	Working functionalities delivered after each sprint for feedback
Cross-functional Teams	Collaboration with stakeholders and subject experts
User-centric	Requirements captured as user stories from stakeholder perspectives
Adaptive Planning	Sprint planning based on feedback and evolving requirements
Continuous Improvement	Retrospectives to review progress and identify improvements
Frequent Inspection	Daily stand-ups to review progress and coordinate tasks
Continuous Integration	Regular code integration to shared repository
Stakeholder Engagement	Active stakeholder involvement throughout development
Embracing Change	Ability to adapt system based on insights during development

Table 2: AGILE Table (Mnkandla, 2008)

The above table shows the practices followed in this agile approach, In this method the system is represented in tiny stages called sprints. For every stakeholder feedback the working features will be delivered. There will be the collaboration of cross functional teams with stakeholders and experts within the whole process. Ensuring the user-centric approach requirements will be captured as user stories from the stakeholders. Adaptive planning will be done which is based on feedback. Achieving continuous improvement can be through meetings for review and identifying the improvements.

By following this approach, the system has been developed successfully.

Requirement gathering:

Before designing, requirement gathering helps the development easier, which can be divided into functional and non-functional requirements. (Solutions, 2023)

Functional requirements:

1. The system can take the images as input from different sources like web camera, and file upload.

2. The system can identify the faces of the visitors who are in the database and should be added to the visitor history for the logs every time they visit. (aws, n.d.)

3. The algorithm can extract the facial features like eyes, mouth, jaw, etc., from the identified face. (Jachak, 2024)

4. The application should handle multiple visitors simultaneously with accurate results.

5. The application should detect the faces in various conditions like different angles and different lighting conditions.

Non-functional requirements:

1. Depending on the application, the system can process the images in real time.

2. With minimum false positives and negatives, the system should work with high accuracy

3. The application should ensure that the visitor data should be confidential and that there should not be any misuse.

4. There should be a user-friendly interface with options like validating, adding and history of the visitors. (mobibob, 2010)

5. The application should be compatible (uxtweak, n.d.) with different operating systems and input devices.

Design:

1. The facial recognition system has been designed for giving protection the security by giving access only for the authorized people like staff, and visitor that are approved by the organisation.

This system includes the following features:

1. Visitor validation: This feature captures the images of the visitors through the live webcam and creates the history of their details.

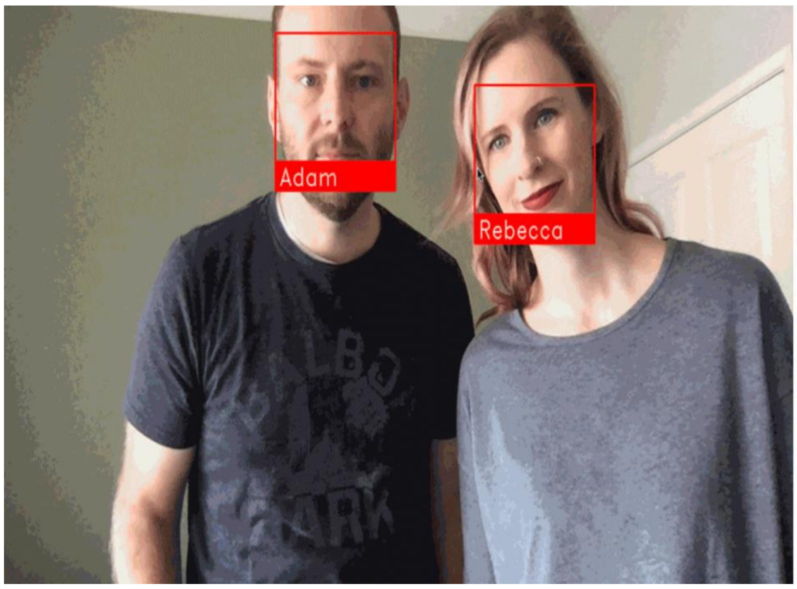
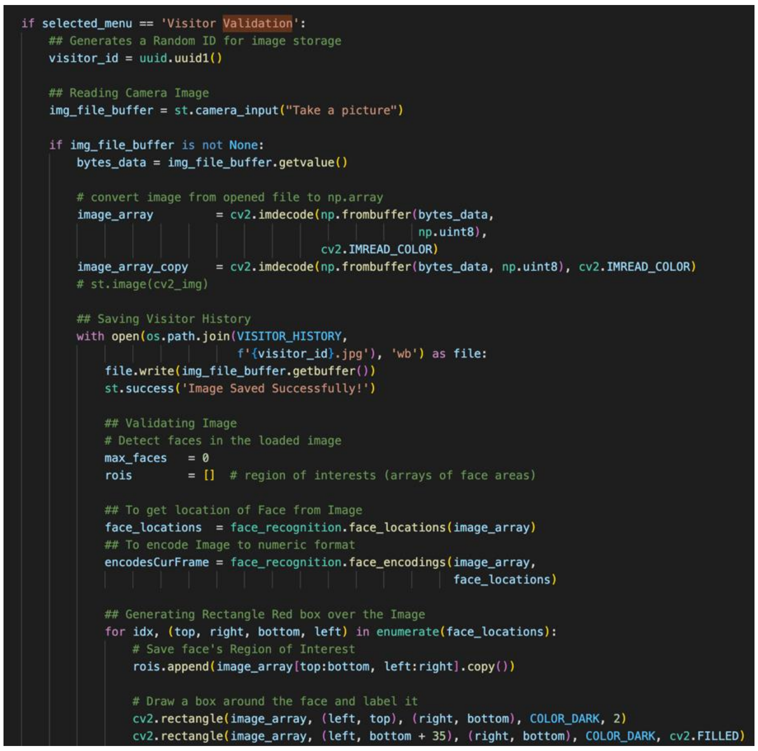


Figure 1: Validation

The above image shows how the model recognize the face and shows their identities which will be fetched from the database.

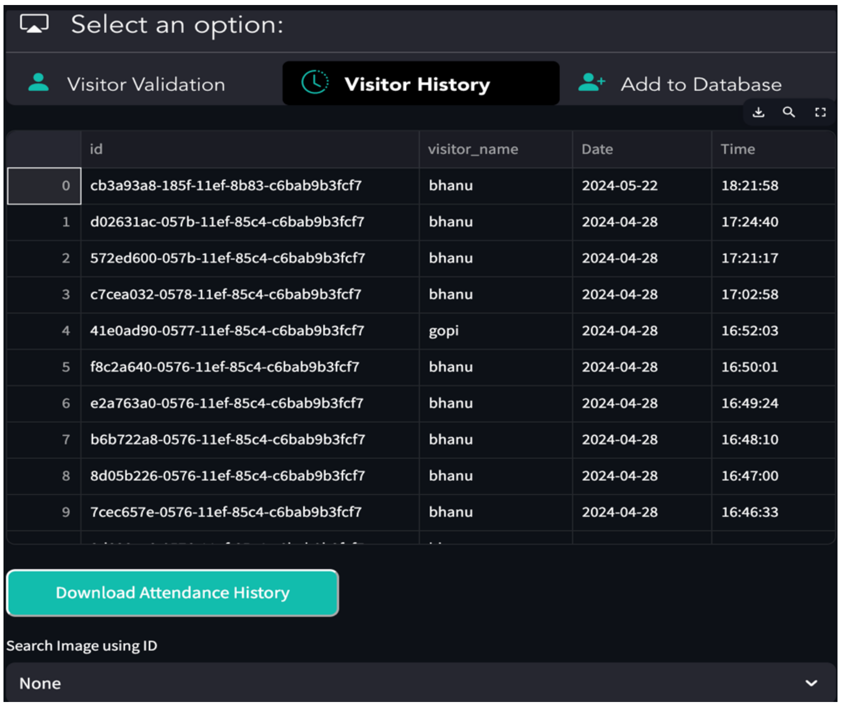


```
if selected_menu == 'Visitor Validation':
    ## Generates a Random ID for image storage
    visitor_id = uuid.uuid1()

    ## Reading Camera Image
    img_file_buffer = st.camera_input("Take a picture")

    if img_file_buffer is not None:
        bytes_data = img_file_buffer.getvalue()

        # convert image from opened file to np.array
        image_array         = cv2.imdecode(np.frombuffer(bytes_data,
                                                         np.uint8),
                                           cv2.IMREAD_COLOR)
        image_array_copy    = cv2.imdecode(np.frombuffer(bytes_data, np.uint8), cv2.IMREAD_COLOR)
        # st.image(cv2_img)

        ## Saving Visitor History
        with open(os.path.join(VISITOR_HISTORY,
                               f'{visitor_id}.jpg'), 'wb') as file:
            file.write(img_file_buffer.getbuffer())
            st.success('Image Saved Successfully!')

            ## Validating Image
            # Detect faces in the loaded image
            max_faces   = 0
            rois        = []  # region of interests (arrays of face areas)

            ## To get location of Face from Image
            face_locations  = face_recognition.face_locations(image_array)
            ## To encode Image to numeric format
            encodesCurFrame = face_recognition.face_encodings(image_array,
                                                              face_locations)

            ## Generating Rectangle Red box over the Image
            for idx, (top, right, bottom, left) in enumerate(face_locations):
                # Save face's Region of Interest
                rois.append(image_array[top:bottom, left:right].copy())

                # Draw a box around the face and label it
                cv2.rectangle(image_array, (left, top), (right, bottom), COLOR_DARK, 2)
                cv2.rectangle(image_array, (left, bottom + 35), (right, bottom), COLOR_DARK, cv2.FILLED)
```

Figure 2: Validation code

The above code is the validation code for the interface which gives the functionality for the system.

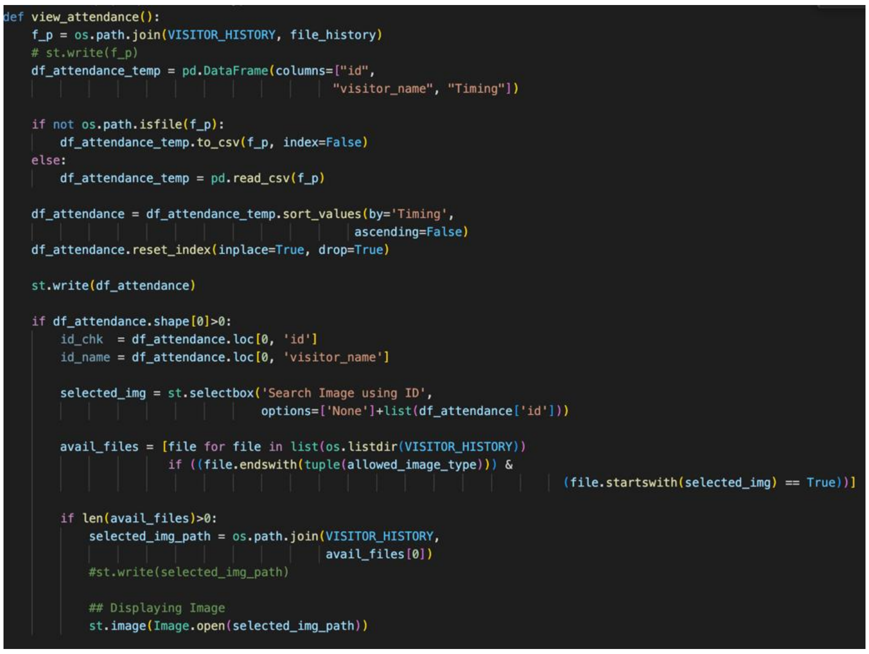
2. Visitor History: As the first feature recognizes the face, the face is either identified or not, it will be shown in this visitor history with time and date, which has been created automatically by the system.



	id	visitor_name	Date	Time
0	cb3a93a8-185f-11ef-8b83-c6bab9b3fcf7	bhanu	2024-05-22	18:21:58
1	d02631ac-057b-11ef-85c4-c6bab9b3fcf7	bhanu	2024-04-28	17:24:40
2	572ed600-057b-11ef-85c4-c6bab9b3fcf7	bhanu	2024-04-28	17:21:17
3	c7cea032-0578-11ef-85c4-c6bab9b3fcf7	bhanu	2024-04-28	17:02:58
4	41e0ad90-0577-11ef-85c4-c6bab9b3fcf7	gopi	2024-04-28	16:52:03
5	f8c2a640-0576-11ef-85c4-c6bab9b3fcf7	bhanu	2024-04-28	16:50:01
6	e2a763a0-0576-11ef-85c4-c6bab9b3fcf7	bhanu	2024-04-28	16:49:24
7	b6b722a8-0576-11ef-85c4-c6bab9b3fcf7	bhanu	2024-04-28	16:48:10
8	8d05b226-0576-11ef-85c4-c6bab9b3fcf7	bhanu	2024-04-28	16:47:00
9	7cec657e-0576-11ef-85c4-c6bab9b3fcf7	bhanu	2024-04-28	16:46:33

Figure 3: Visitor History

The above image is the history image of the visitor with date and time, and it also includes the person's names in the database. Moreover, the history can be downloaded from the "Download Attendance History" button.



```python
def view_attendance():
    f_p = os.path.join(VISITOR_HISTORY, file_history)
    # st.write(f_p)
    df_attendance_temp = pd.DataFrame(columns=["id",
                                              "visitor_name", "Timing"])

    if not os.path.isfile(f_p):
        df_attendance_temp.to_csv(f_p, index=False)
    else:
        df_attendance_temp = pd.read_csv(f_p)

    df_attendance = df_attendance_temp.sort_values(by='Timing',
                                                   ascending=False)
    df_attendance.reset_index(inplace=True, drop=True)

    st.write(df_attendance)

    if df_attendance.shape[0]>0:
        id_chk  = df_attendance.loc[0, 'id']
        id_name = df_attendance.loc[0, 'visitor_name']

        selected_img = st.selectbox('Search Image using ID',
                                    options=['None']+list(df_attendance['id']))

        avail_files = [file for file in list(os.listdir(VISITOR_HISTORY))
                       if ((file.endswith(tuple(allowed_image_type))) &
                                                                (file.startswith(selected_img) == True))]

        if len(avail_files)>0:
            selected_img_path = os.path.join(VISITOR_HISTORY,
                                             avail_files[0])
            #st.write(selected_img_path)

            ## Displaying Image
            st.image(Image.open(selected_img_path))
```

Figure 4: Visitor History code

The code above gives the functionality for the interface.

3. Add to database: In this section, the administrator can add the visitor by uploading the image or by clicking a picture through the webcam. The details of the person will be given in the same section when the image will be added to the database.

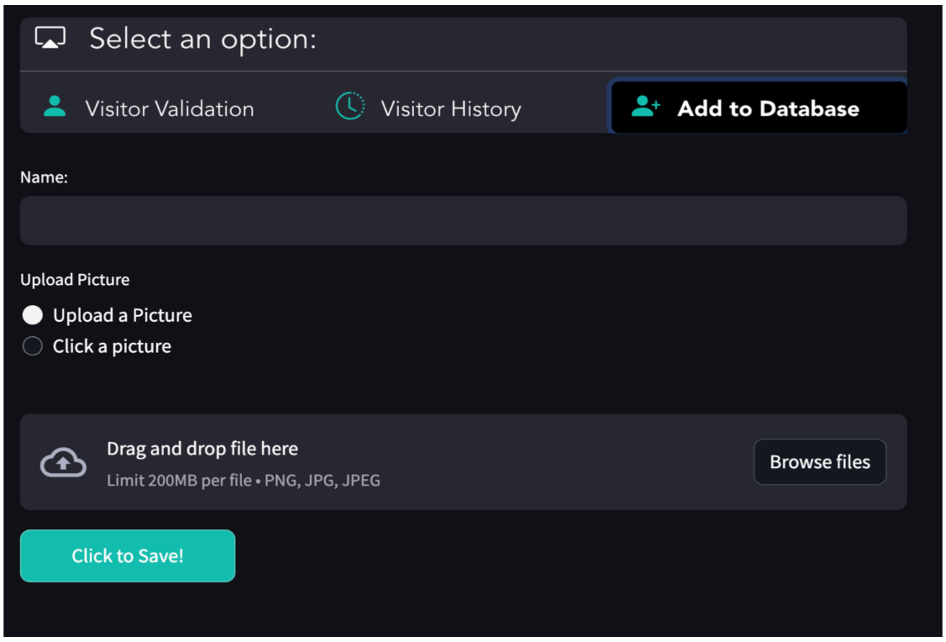
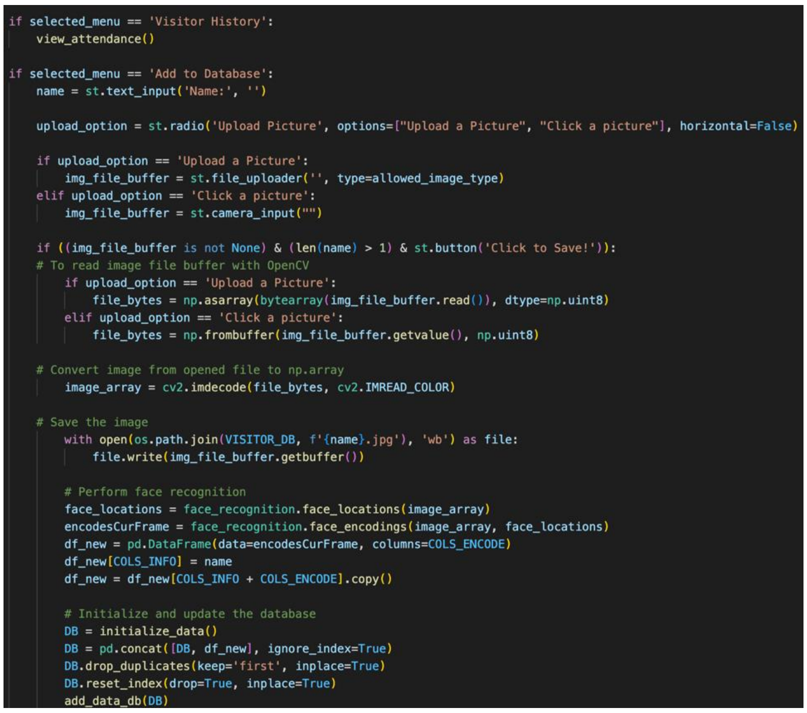


Figure 5: Adding in Database

The above shows the details that has to be enter for adding the visitor to the database.



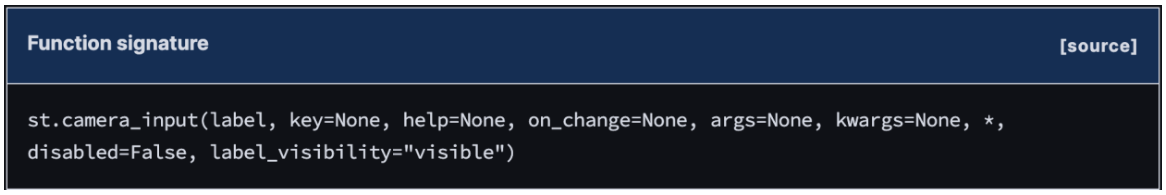
```python
if selected_menu == 'Visitor History':
    view_attendance()

if selected_menu == 'Add to Database':
    name = st.text_input('Name:', '')

    upload_option = st.radio('Upload Picture', options=["Upload a Picture", "Click a picture"], horizontal=False)

    if upload_option == 'Upload a Picture':
        img_file_buffer = st.file_uploader('', type=allowed_image_type)
    elif upload_option == 'Click a picture':
        img_file_buffer = st.camera_input("")

    if ((img_file_buffer is not None) & (len(name) > 1) & st.button('Click to Save!')):
    # To read image file buffer with OpenCV
        if upload_option == 'Upload a Picture':
            file_bytes = np.asarray(bytearray(img_file_buffer.read()), dtype=np.uint8)
        elif upload_option == 'Click a picture':
            file_bytes = np.frombuffer(img_file_buffer.getvalue(), np.uint8)

    # Convert image from opened file to np.array
        image_array = cv2.imdecode(file_bytes, cv2.IMREAD_COLOR)

    # Save the image
        with open(os.path.join(VISITOR_DB, f'{name}.jpg'), 'wb') as file:
            file.write(img_file_buffer.getbuffer())

        # Perform face recognition
        face_locations = face_recognition.face_locations(image_array)
        encodesCurFrame = face_recognition.face_encodings(image_array, face_locations)
        df_new = pd.DataFrame(data=encodesCurFrame, columns=COLS_ENCODE)
        df_new[COLS_INFO] = name
        df_new = df_new[COLS_INFO + COLS_ENCODE].copy()

        # Initialize and update the database
        DB = initialize_data()
        DB = pd.concat([DB, df_new], ignore_index=True)
        DB.drop_duplicates(keep='first', inplace=True)
        DB.reset_index(drop=True, inplace=True)
        add_data_db(DB)
```

Figure 6: Adding in Database code

The code above gives the functionality for each component in the interface.

The current system uses the device's built-in camera which can be accessed accordingly. There will be two techniques in this: Input technique, which includes uploading the images using the cam. Output includes giving visual results like identified faces, visitor history and alerts.

The current system's user interface has been developed using the Streamlit library that allows the camera to identify people. It also gives a great interface with some front-end functionalities.



Function signature [source]

```
st.camera_input(label, key=None, help=None, on_change=None, args=None, kwargs=None, *,
disabled=False, label_visibility="visible")
```

Figure 7: Function signature (Streamlit, 2024)

This function in Streamlit enables the camera to image recognition.

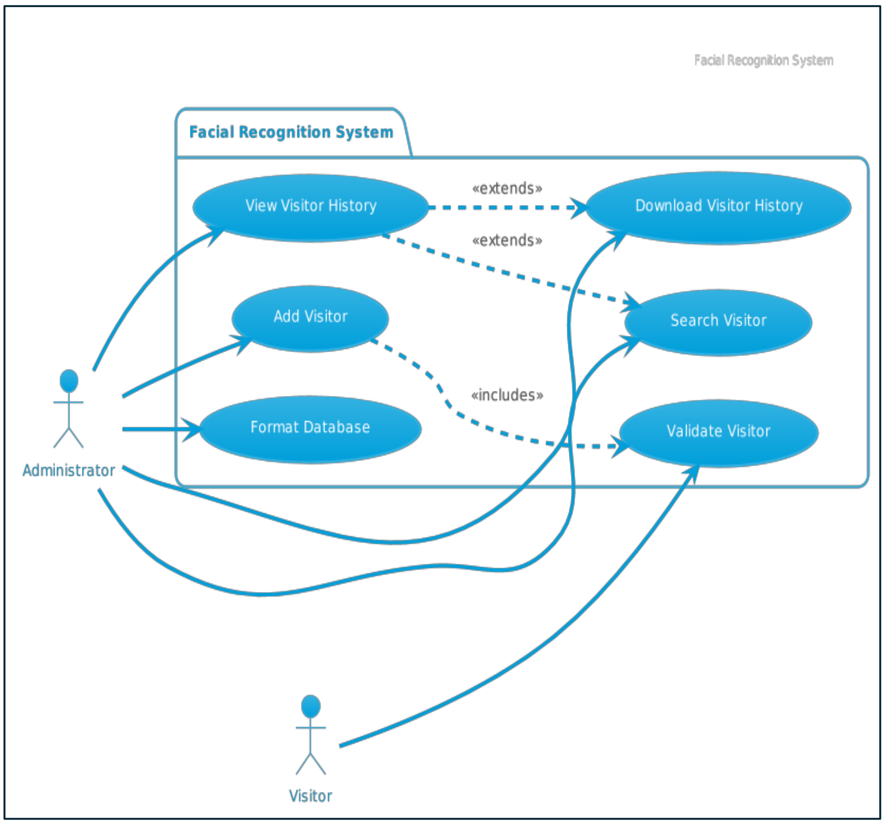


Figure : Use case diagram (Idrizi, n.d.)

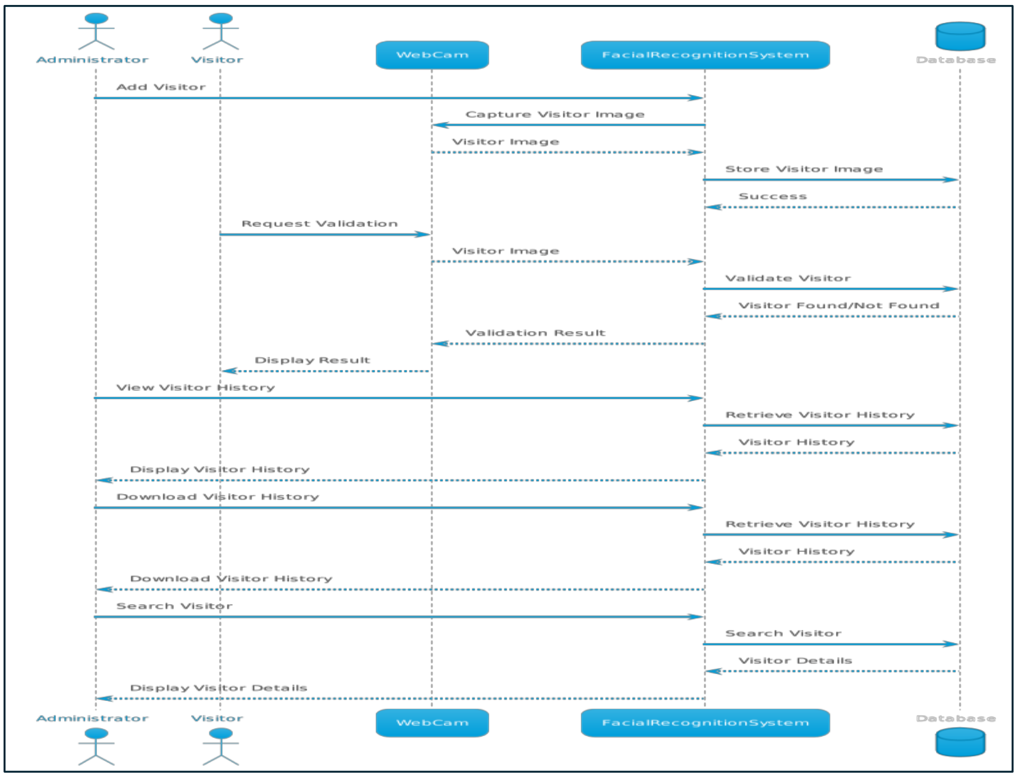


Figure : Sequence diagram (Ashritha, 2022)

Above are the UML diagrams for the facial recognition system that has the different purposes.

Use case shows the functional requirements of the system which demonstrates the interactions in between system and actor.

Sequence diagram shows the sequence interactions between the components overtime

Architecture:

The facial recognition system for high-security buildings needs to handle a large amount of data because of the large dataset that is going to increase as visitors visit the place.

The data of the visitor, like images, can be gathered through the webcam and user-uploaded images, which can be seen in the data acquisition phase. Then the data has been taken as input and processed for the face detection, alignment and extraction. The human faces in the images will be located by the face identification method. As there will be different orients of faces, the face alignment aligns the orientation into a standard scale that can be analysed for extraction of facial traits such as embeddings.

After the extraction, the retrieved face will be matched to the database to identify people. This matching/comparison will be done by the facial recognition algorithm, which is based on the facial recognition library. This library uses the Deep learning model like CNN architecture.

First the algorithm will calculate the similarities between the input face and database, If there is any match, it gives the data of the visitor form the database.

The visitor history has been kept for the record of visitor data like name, ID, time and data, which can be helpful for auditing.

The comprehensive facial recognition has been designed to fulfil the needs of high-security buildings by ensuring accuracy and performance.

Algorithm:

A. As there are plenty of algorithms for facial recognition, a suitable one has been taken for the current model. A Few other algorithms, like PCA and LDA, have some benefits, but they are not the correct approach for this system; let's see about those functionalities.

Principle Component Analysis: This method is one of the oldest widely used methods; it turns the image into a set of components (Paul, 2019) for capturing the variations in the photos. For the 'face space' new faces will be shown for any successful matches from the database. The main benefit of using PCA is simple concept and computational efficiency. However, in different lighting conditions, it may not work efficiently.

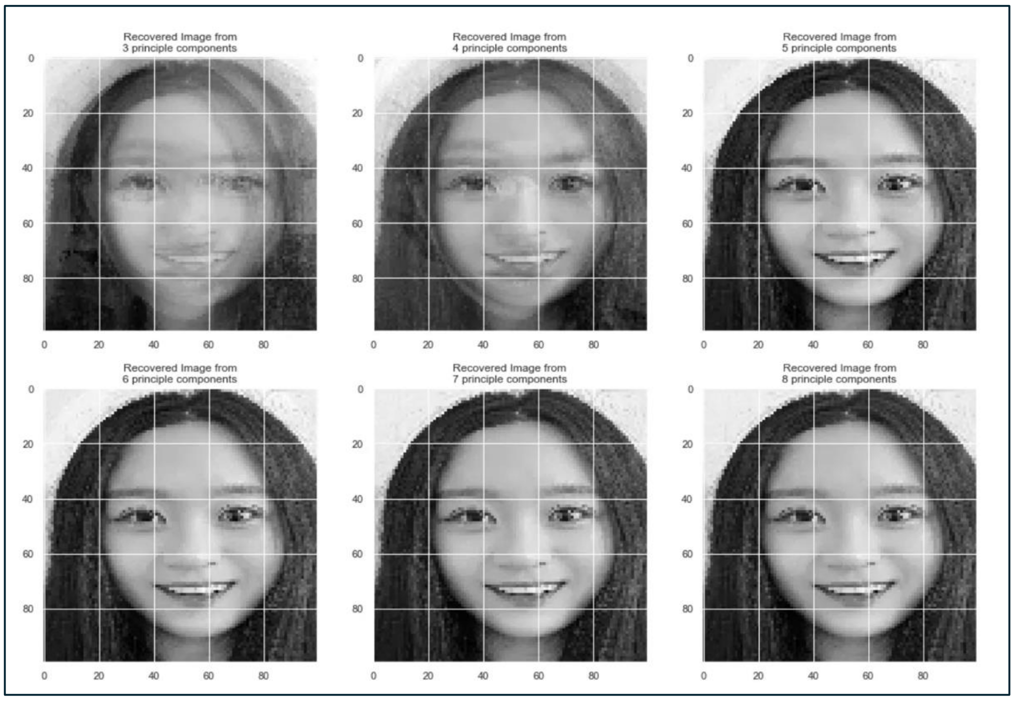

Recovered Image from 3 principle components
Recovered Image from 4 principle components
Recovered Image from 5 principle components
Recovered Image from 6 principle components
Recovered Image from 7 principle components
Recovered Image from 8 principle components
0
20
40
60
80

Figure 8: PCA-Image from different principle components (Shi, 2020)

Linear Discriminant Analysis: LDA works the same as PCA, where it decreases the large amount of face data by storing the unique features that makes each face differentiate to other.

In this manner, the LDA can work more efficiently in lighting conditions. However, this requires more computational power. (Begum, 2023)

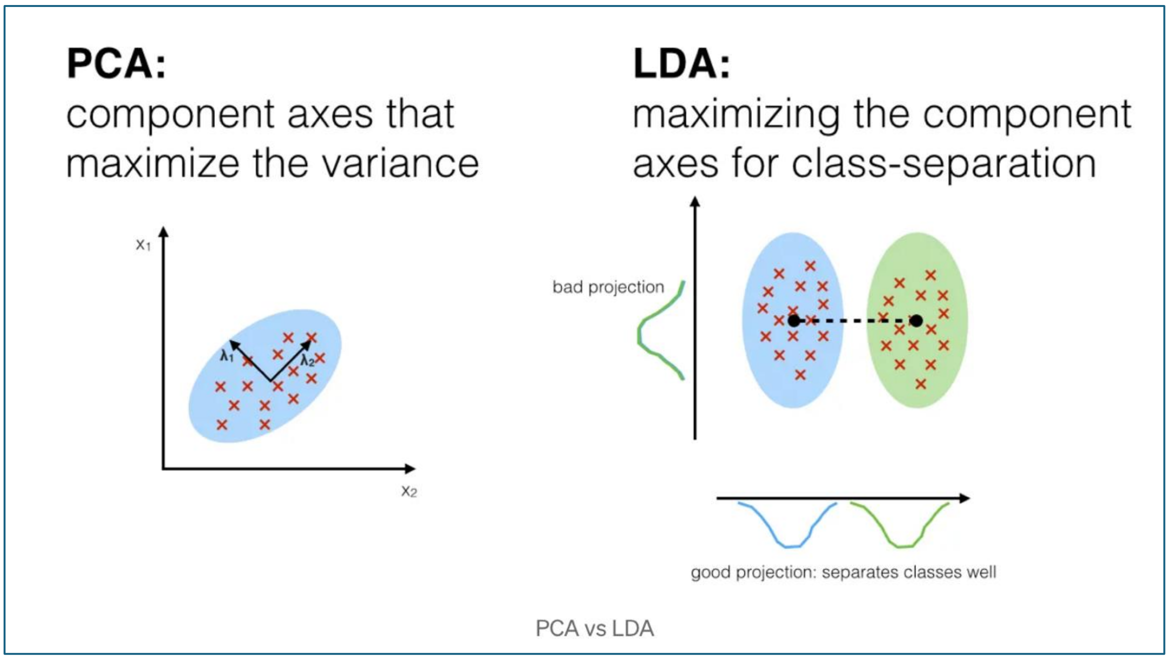

PCA:
component axes that maximize the variance
LDA:
maximizing the component axes for class-separation
x_1
x_2
λ_1
λ_2
bad projection
good projection: separates classes well
PCA vs LDA

Figure 9: PCA vs LDA (Dwivedi, 2018)

Deep Learning Approaches: For the last few years, CNN, which is an advanced computer technique, has been showing excellent performance in recognising faces. From the trained data, this model can learn automatically and also understand the complex structures and patterns in the faces. (Komlavi, 2024) This will be done without any code instructions. Some of the famous designs like VGGFace, FaceNet, and SphereNet will perform accurately and efficiently, but these will be complicated to train which requires a large amount of memory.

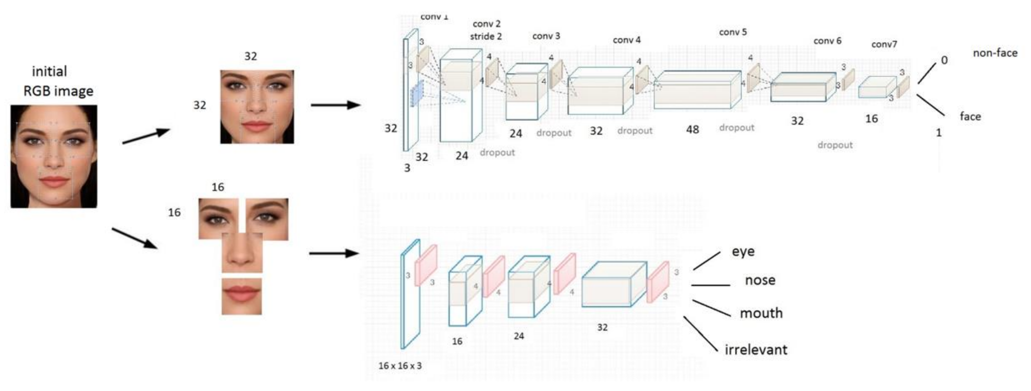


Figure 10: CNN trained for full face and facial parts detection (Triantafyllidou, 2017)

As quick adaption and accuracy are important things in the face recognition system, the deep learning-based model will be the appropriate one for this current system.

B. Deep learning models are the best for use as they work in lighting conditions with accuracy. So, ResNet is used as the primary algorithm for this web application. ResNet can be utilized from the Python library “face_recognition”, which employs deep learning models like CNN. The “face_recognition” library uses ResNet for face recognition and face encoding, and also, for landmark detection, CNN will be used. For this application, “face_recognition” has been used because of its simplicity, which offers high-level interfaces like face detection, encoding and comparison. The below image shows the architecture of the ResNet.

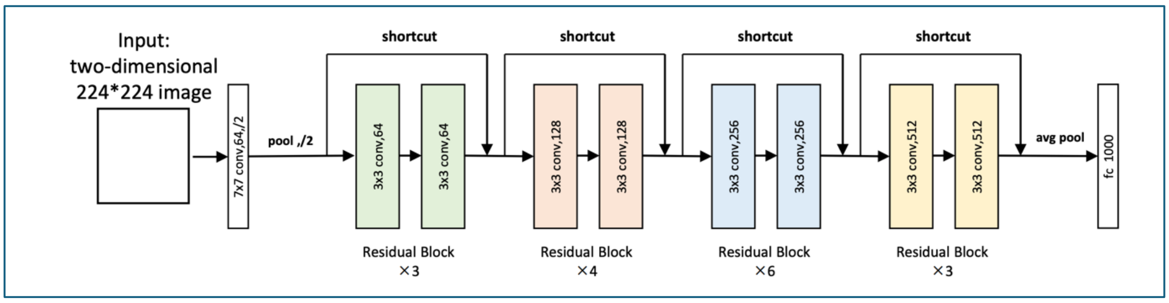


Figure 11: ResNet Architecture (Zhang, 2023)

Factors:

1. Accuracy: As this model has been trained on massive face images of the dataset, it can implement patterns that are complex and can construct the 128-dimensional face encodings, which gives an accurate identification.

2. Speed: Pre-trained ResNet can generate and compare the face encodings fastly without any training which makes the process fast.

3. Resource requirements: As this has limited resources, it is suitable for the system because it doesn’t need as much memory and power as large models.

4. Scalability: If there are multiple faces, it can be identified easily as it generates the face encodings and compares them using distance calculations.

The library uses the CNN for detecting facial landmarks like eyes, nose, mouth and jaw. As CNN is trained on a large dataset of these landmarks, it learns to connect the face pictures to those landmarks. Moreover, it automatically learns and extracts the features from the raw data, which makes it perfect for image processing.

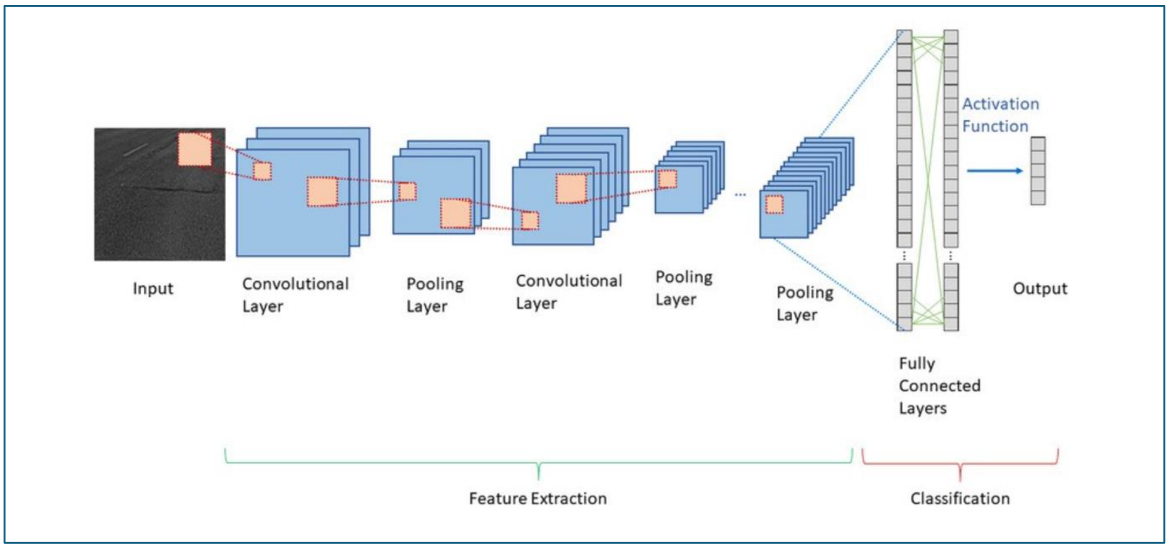


Figure 12: CNN Architecture (Moustafa, 2023)

As CNN and ResNet are trained on the large datasets using the supervised learning, ResNet can identify the faces and CNN can locate the facial landmarks.

Limitations:

1. Both give high performance. However, it will depend on what training data was given, either good or bad.

2. Every device can't use this model because of the limited resources.

3. There may be a change in the performance as the images will be in low lighting or low angles.

4. Sometimes, because of partial images of faces, detecting landmarks will not be accurate.

Overall, both CNN and ResNet works efficiently in different patterns.

The Deep learning models that are pre-trained on large datasets can handle variations in position, lighting and expressions. To increase the robustness, additional steps need to be taken.

For Pose variations: As the library can handle poses up to 20 degrees if it wants to handle more angles, advanced techniques need to be taken, like multi-view recognition models trained on different pose data, 3D landmark detection and estimating pose will help. Increasing the data of the images by adding different angles will help the model to be trained effectively and work accurately.

For Lightning conditions: The library works at normal lighting conditions; however, for different lightings, additional techniques can be followed like enhancement in contrast and histogram equalization.

For facial expressions: This library works accurately for the normal expression of the face; however, for different expressions or extreme expressions, the model should be trained with a large dataset having different expressions or using some methods like deep networks explicitly, which could improve the performance of the system.

Implementation:

A. Python and Streamlit:

Python is used as the programming language for this project as it is a popular language and has plenty of useful libraries for building ML and DL models and also CV applications like face recognition.

Let's see the benefits of using Python as a programming language:

Machine Learning Libraries: ML libraries like TensorFlow, Keras, and PyTorch can be seen in Python which gives advanced tools for developing the facial recognition models using DL techniques.

Computer Vision Libraries: CV libraries like OpenCV, Dlib and face_recognition have been used in this system for image processing, identification of faces and also extraction, which are the important components in face recognition. (AI, 2023)

Develop and Prototype: Quick Developing and Prototyping applications will be done because of python's simplicity and readability, which allows the changes in the phase of implementation.

Python's open-source library Streamlit has been chosen as a development environment because it is easy to use and can build interactive web applications easily. The benefits of using them are:

Easy integration with Python: Streamlit can work with Python without any issues. Moreover, it allows the existing libraries for facial recognition and data processing.

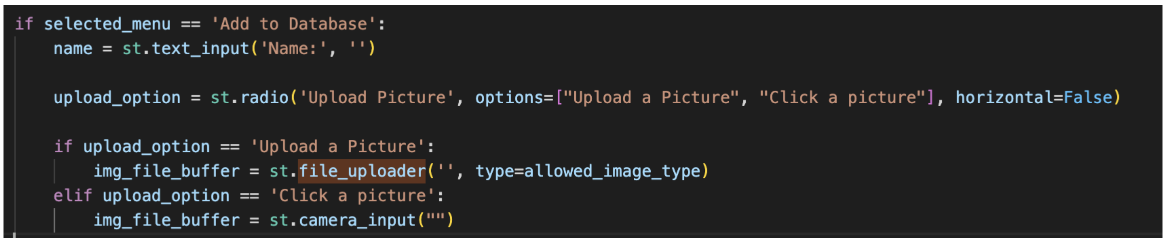
Interactive UI: Streamlit provides the frameworks that are responsive for creating the user interfaces, which include camera input, and file uploads.

Deployment: Streamlit applications can be deployed as web applications, which is ideal for deploying in production environments.

Overall, building a successful face recognition system can be achieved by combining both Python and Streamlit has the powers of development and deployment.

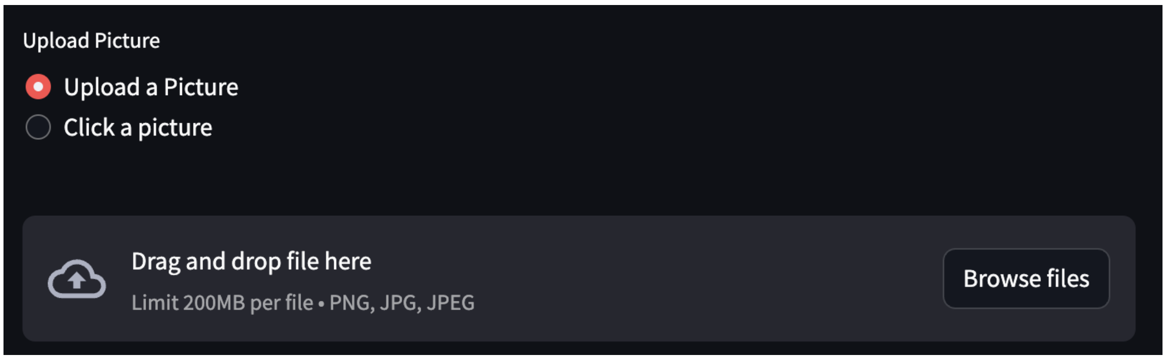
Implementation process:

Data Acquisition: The application takes the images in two ways: one is capturing the live image, and another is uploading the image from the device. This operation uses "Streamlit's file_uploader" and "camera_input" features.



```
if selected_menu == 'Add to Database':
    name = st.text_input('Name:', '')

    upload_option = st.radio('Upload Picture', options=["Upload a Picture", "Click a picture"], horizontal=False)

    if upload_option == 'Upload a Picture':
        img_file_buffer = st.file_uploader('', type=allowed_image_type)
    elif upload_option == 'Click a picture':
        img_file_buffer = st.camera_input("")
```

Figure 13: Code for data acquisition



Upload Picture

[✓ Upload a Picture]

Click a picture

Drag and drop file here

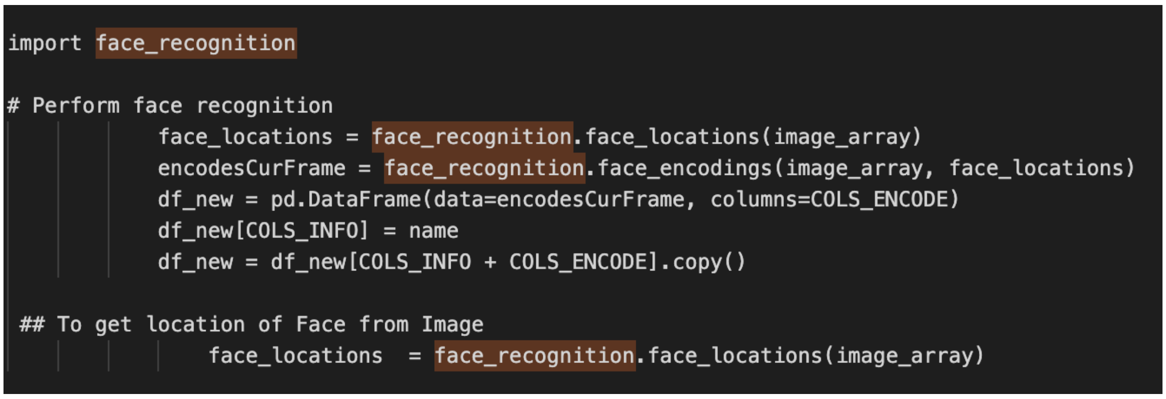
Limit 200MB per file • PNG, JPG, JPEG

Browse files

Figure 14: Interface for data acquisition

Pre-processing: After the acquisition, it goes through the following steps:

1. The face_recognition library identifies and locates the face part in the image.



```
import face_recognition

# Perform face recognition
            face_locations = face_recognition.face_locations(image_array)
            encodesCurFrame = face_recognition.face_encodings(image_array, face_locations)
            df_new = pd.DataFrame(data=encodesCurFrame, columns=COLS_ENCODE)
            df_new[COLS_INFO] = name
            df_new = df_new[COLS_INFO + COLS_ENCODE].copy()

 ## To get location of Face from Image
                face_locations  = face_recognition.face_locations(image_array)
```

Figure 15: face_recognition

2. If the faces are in different aligned positions, Face alignment makes the images normal and aligns them to a standard orientation.

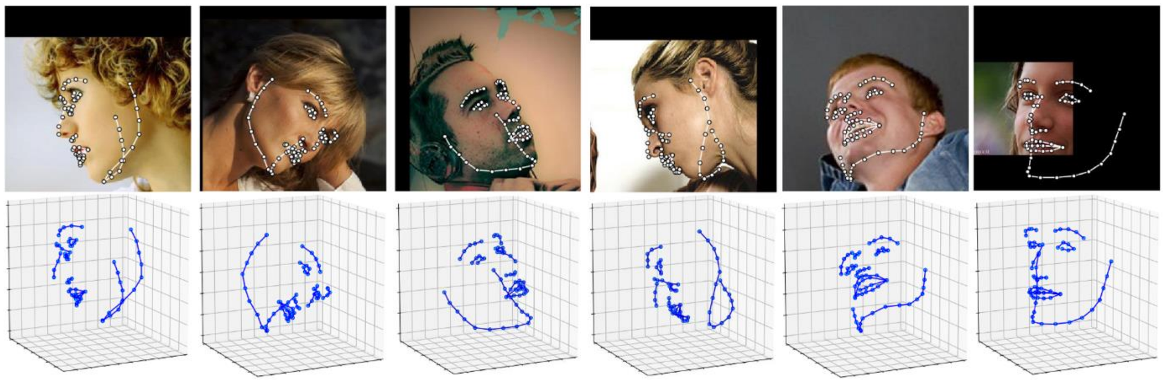
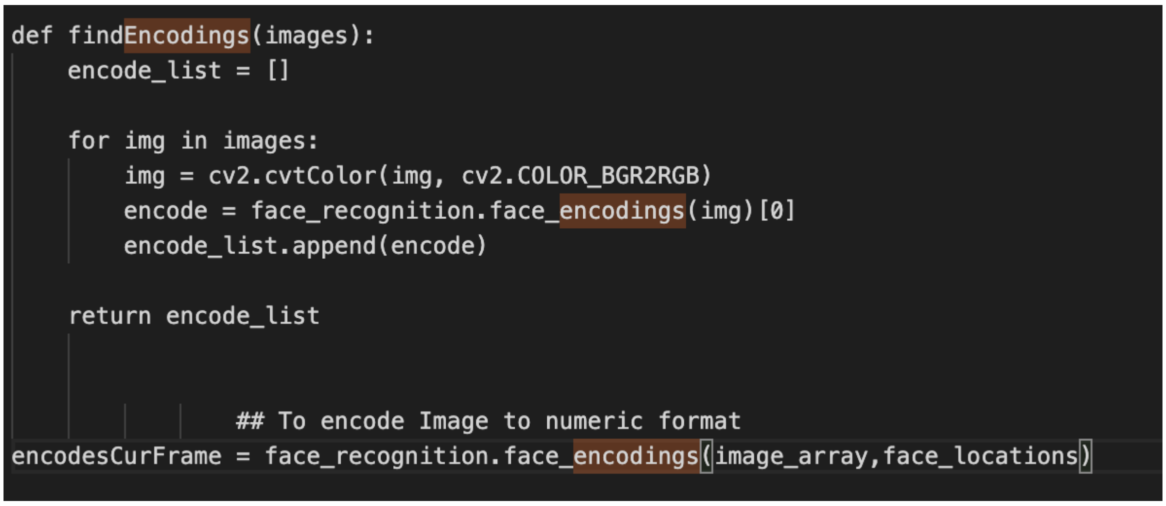


Figure 16: Face alignments

3. Face landmarks and embeddings will be extracted from the aligned faces with the help of the face_recognition library that has the face_encodings function.



```
def findEncodings(images):
    encode_list = []

    for img in images:
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        encode = face_recognition.face_encodings(img)[0]
        encode_list.append(encode)

    return encode_list


                ## To encode Image to numeric format
encodesCurFrame = face_recognition.face_encodings(image_array,face_locations)
```

Figure 17: Face encoding code

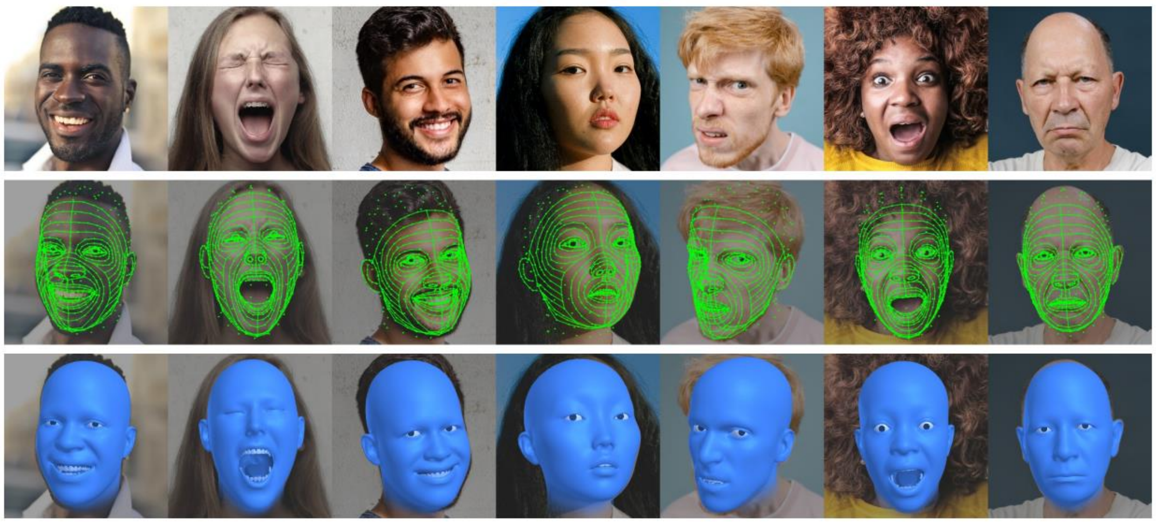
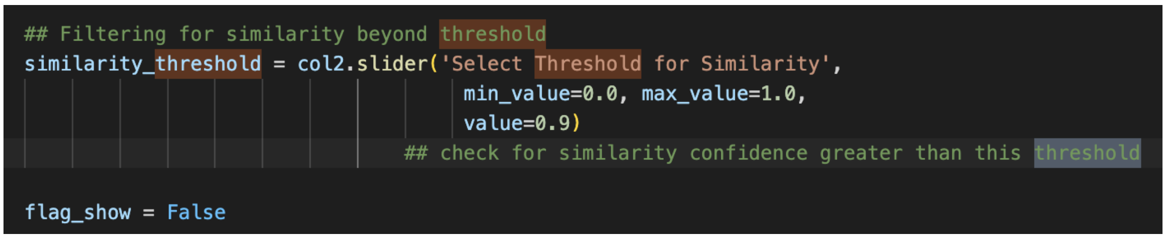


Figure 18: Face Encodings

Training and optimizing models:

The application uses the face_recogntion library that helps by applying pre-trained deep learning models like CNN to the face recognition application.

For better accuracy in matching the faces from the database, there is a defined parameter in the application called “Threshold”, which can be adjusted to identify the faces with the highest accuracy.



```
## Filtering for similarity beyond threshold
similarity_threshold = col2.slider('Select Threshold for Similarity',
                                    min_value=0.0, max_value=1.0,
                                    value=0.9)
                                ## check for similarity confidence greater than this threshold

flag_show = False
```

Figure 19: Threshold code

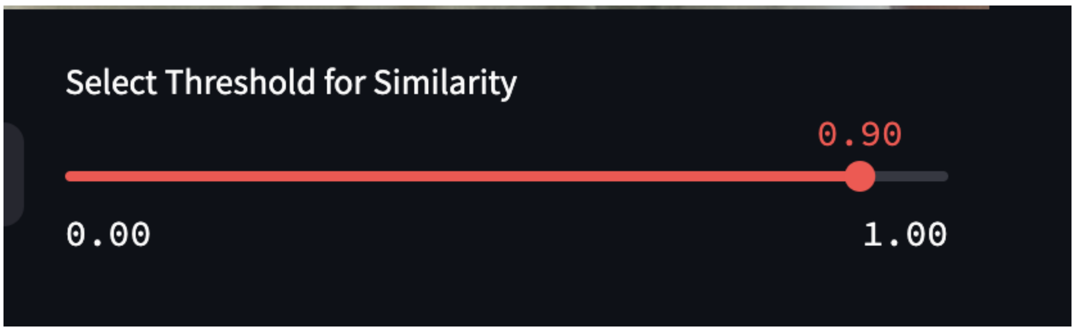


Figure 20: Threshold

Integration of User Interface and Security Protocols:

Using Streamlit, the user interface has been developed by creating three main options that are required for the application options: Visitor validation, Visitor history and Add to Database. These options allow the administrator to add the image of the visitor to the database, monitor the logs and validating the visitor.

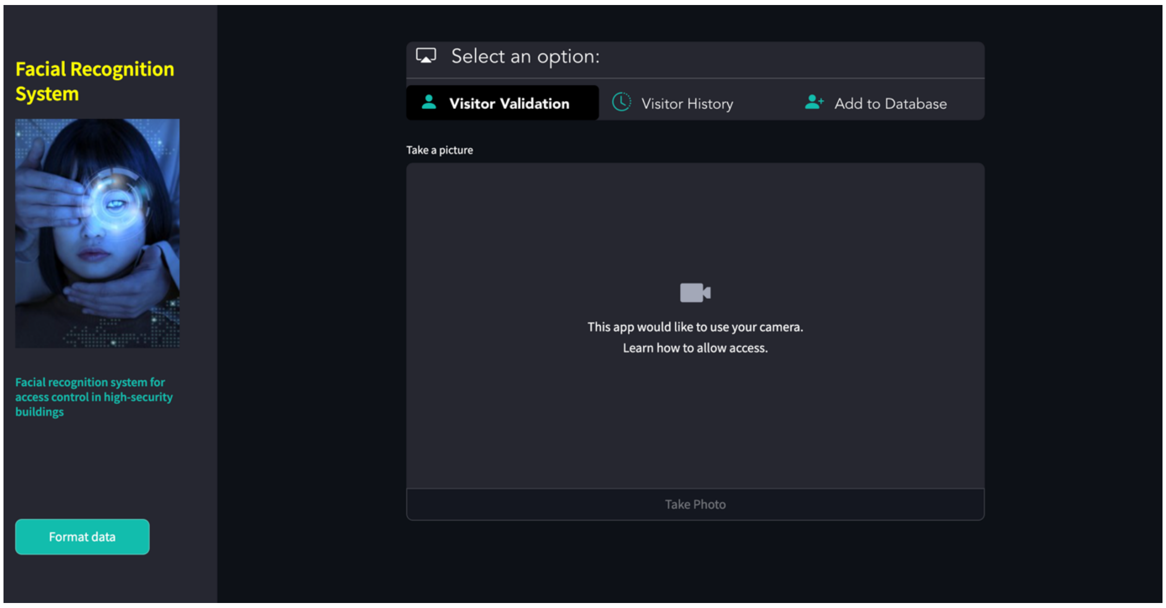
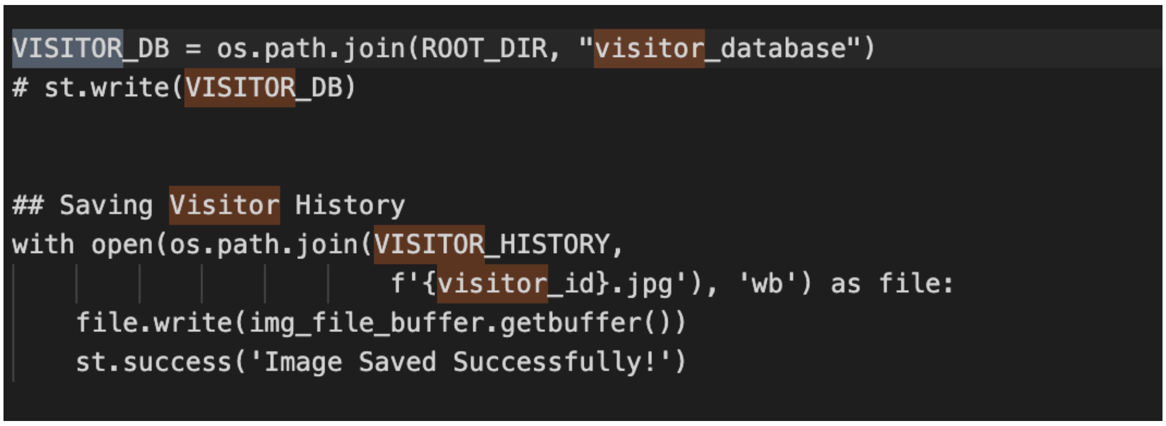

Facial Recognition System
Facial recognition system for access control in high-security buildings
Format data
Select an option:
Visitor Validation
Visitor History
Add to Database
Take a picture
This app would like to use your camera.
Learn how to allow access.
Take Photo

Figure 21: Visitor validation

Security protocols:

The database of the visitors can't be seen in the application, as they are sensitive information; those images have been stored in the "visitor_database" and "visitor_history", for which only the admin has permission to access.



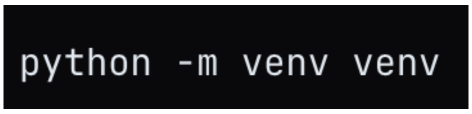
```
VISITOR_DB = os.path.join(ROOT_DIR, "visitor_database")
# st.write(VISITOR_DB)


## Saving Visitor History
with open(os.path.join(VISITOR_HISTORY,
                       f'{visitor_id}.jpg'), 'wb') as file:
    file.write(img_file_buffer.getbuffer())
    st.success('Image Saved Successfully!')
```

Figure 22: Visitor database

Overall, the implementation has data gathering, pre-processing, facial recognition, and integrating with Streamlit. With the help of these libraries and techniques, facial recognition has been successfully implemented.

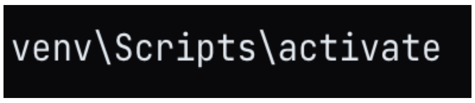
Steps to open the web application through terminal:

1. Create the virtual environment.



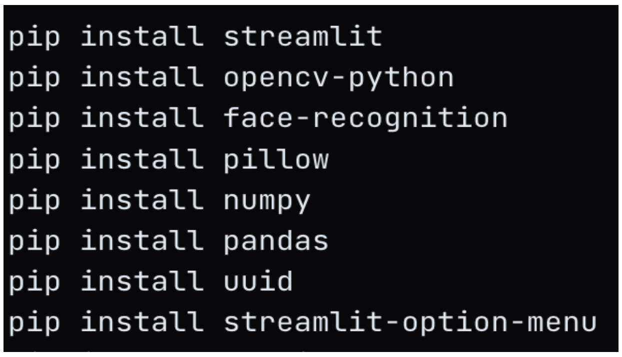
```
python -m venv venv
```

2. Activate the virtual environment.



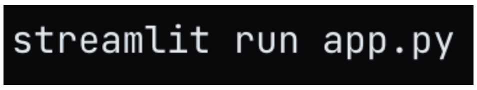
```
venv\Scripts\activate
```

3. Install the required libraries.



```
pip install streamlit
pip install opencv-python
pip install face-recognition
pip install pillow
pip install numpy
pip install pandas
pip install uuid
pip install streamlit-option-menu
```

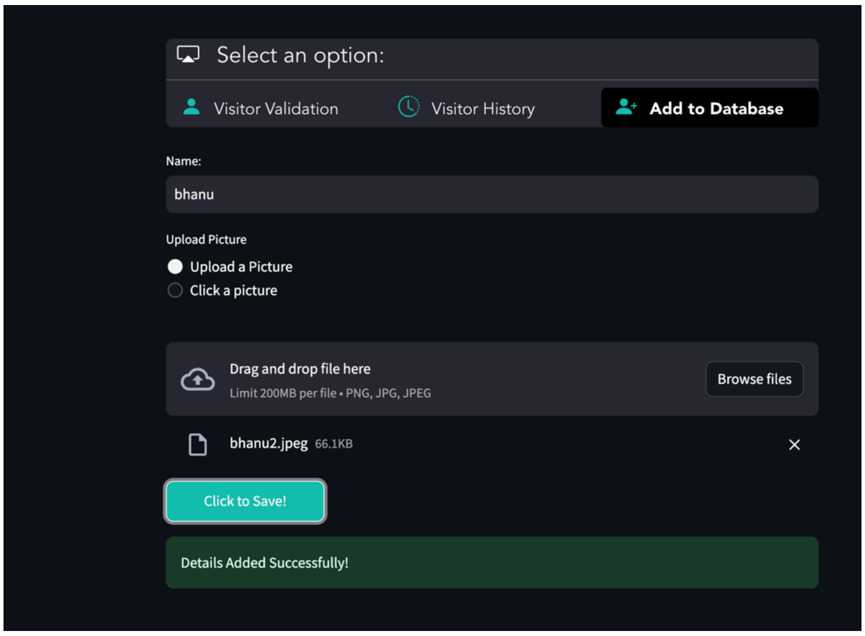
4. After successful installation, run the following command.



```
streamlit run app.py
```

Testing:

Test Case	Test Data	Expected Result
Adding a visitor to the database.	Entering name and uploading an image.	Image will be updated in the database and shows a successful message.



Name:

bhanu

bhanu2.jpeg 66.1KB

Details Added Successfully!

Figure 23: Adding database

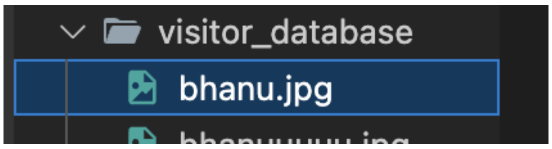
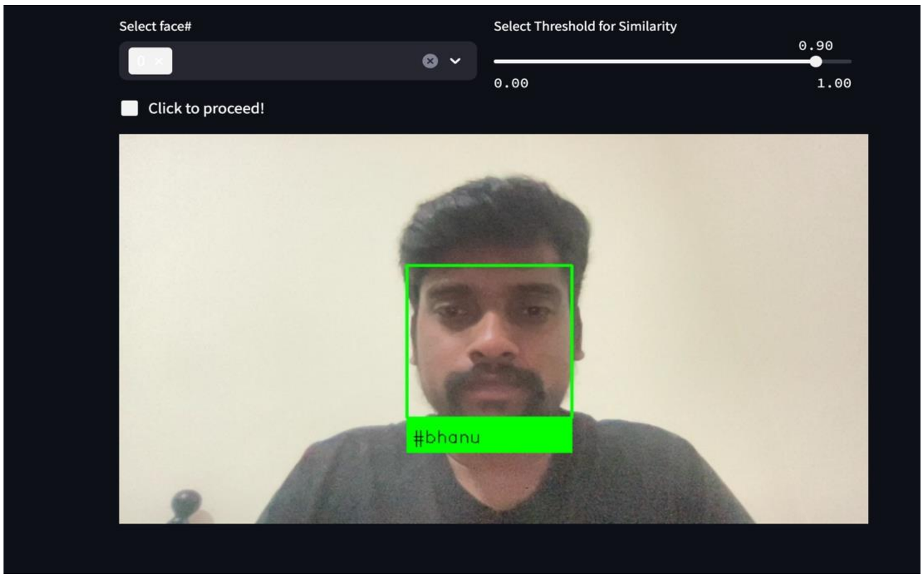


Figure 24: Database file

Test Case	Test Data	Expected Result
Validating the visitor.	Capturing the image through live camera.	Image should be identified and show the name of the person.


Select face#

Select Threshold for Similarity

0.90

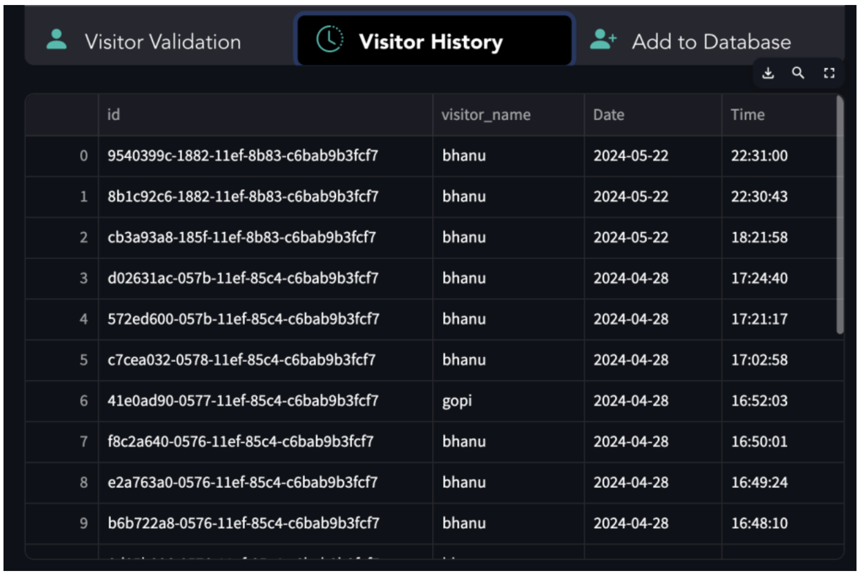
0.00

1.00

Click to proceed!

Figure 25: Visitor validation

Test Case	Test Data	Expected Result
Visitor history should be updated after successful detection.	Shows id, name, date and time.	Person data will be updated in the database table.



	id	visitor_name	Date	Time
0	9540399c-1882-11ef-8b83-c6bab9b3fcf7	bhanu	2024-05-22	22:31:00
1	8b1c92c6-1882-11ef-8b83-c6bab9b3fcf7	bhanu	2024-05-22	22:30:43
2	cb3a93a8-185f-11ef-8b83-c6bab9b3fcf7	bhanu	2024-05-22	18:21:58
3	d02631ac-057b-11ef-85c4-c6bab9b3fcf7	bhanu	2024-04-28	17:24:40
4	572ed600-057b-11ef-85c4-c6bab9b3fcf7	bhanu	2024-04-28	17:21:17
5	c7cea032-0578-11ef-85c4-c6bab9b3fcf7	bhanu	2024-04-28	17:02:58
6	41e0ad90-0577-11ef-85c4-c6bab9b3fcf7	gopi	2024-04-28	16:52:03
7	f8c2a640-0576-11ef-85c4-c6bab9b3fcf7	bhanu	2024-04-28	16:50:01
8	e2a763a0-0576-11ef-85c4-c6bab9b3fcf7	bhanu	2024-04-28	16:49:24
9	b6b722a8-0576-11ef-85c4-c6bab9b3fcf7	bhanu	2024-04-28	16:48:10

Figure 26: Visitor History

Test Case	Test Data	Expected Result
Searching the visitor with his id.	Selecting the id.	The visitor will be visible after giving valid id.

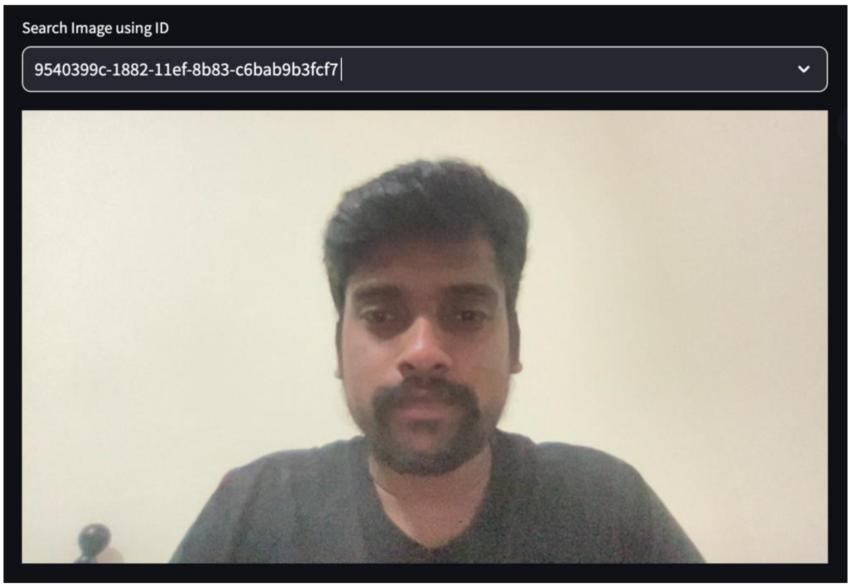


Figure 27: Searching with ID

Test Case	Test Data	Expected Result
Identifying multiple people at a time.	Capturing the images through live camera.	Two or more people should be detected with their name at a time.

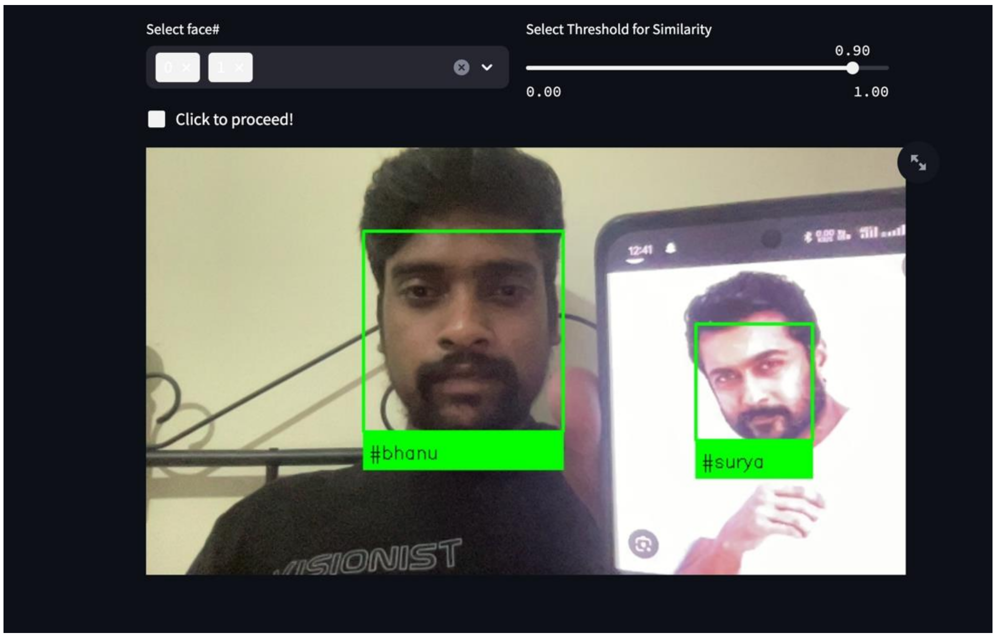


Figure 28: Multiple Visitor validations

Conclusion and Future works.:

In conclusion, developing and implementing the facial recognition system is a critical task that needs planning. As there are few methods and approaches to follow for better development, Agile will be the perfect approach for this facial recognition system, so we followed agile, where the task has been broken into sprints by getting feedback from the stakeholders after every sprint. This approach helps the system to improve and work efficiently.

Firstly, the design of the system was done by gathering the requirements, which are functional and non-functional, and then the designing of the system was done in a step-by-step manner. Then, the system was implemented using Python, ResNet, and Streamlit. ResNet is a type of CNN that is accurate in identifying the images and faces, which helped the system work well for the facial recognition system.

The main comeback of the resNet is , that this algorithm has been trained that it can identify the faces in some conditions like different angles, expressions and lighting conditions.

For more accuracy and future work, we can add voice recognition for facial recognition and also add more complex, advanced features like identifying emotions, gender, and age, which make the system more useful in different industries.

Overall, this report gave a good brief of facial recognition. For future enhancements, new technologies and meeting the user needs can be done.

Bibliography

Triantafyllidou, D. (2017, october). *A Fast Deep Convolutional Neural Network for Face Detection in Big Visual Data*. Retrieved from ResearchGate: https://www.researchgate.net/publication/308944615_A_Fast_Deep_Convolutional_Neural_Network_for_Face_Detection_in_Big_Visual_Data

Streamlit. (2024). *st.camera_input* . Retrieved from Streamlit: https://docs.streamlit.io/develop/api-reference/widgets/st.camera_input

Shi, H. (2020, July 14). *Principal Component Analysis (PCA)- Facial Recognition* □□□□. Retrieved from Medium: https://melaniesoek0120.medium.com/principal-component-analysis-pca-facial-recognition-5e1021f55151

Dwivedi, D. (2018, April 28). *Face Recognition for Beginners* . Retrieved from Medium: https://towardsdatascience.com/face-recognition-for-beginners-a7a9bd5eb5c2

Zhang, Z. (2023, feb). *A Multi-Dimensional Covert Transaction Recognition Scheme for Blockchain*. Retrieved from ResearchGate: https://www.researchgate.net/figure/Architecture-of-ResNet34-29_fig2_368590488

Moustafa, N. (2023, september 07). *Decoding the CNN Architecture: Unveiling the Power and Precision of Convolutional Neural Networks - Part Ⅰ*. Retrieved from Linkedin: https://www.linkedin.com/pulse/decoding-cnn-architecture-unveiling-power-precision-neural-moustafa/

Cantemir, E. (2024). *Use of artificial neural networks in architecture: determining the architectural style of a building with a convolutional neural networks*. Retrieved from researchgate: https://www.researchgate.net/publication/377532005_Use_of_artificial_neural_networks_in_architecture_determining_the_architectural_style_of_a_building_with_a_convolutional_neural_networks

Jones, J. (2023, 11 21). *Agile Implementation Methodology | Importance & Benefits* . Retrieved from study: https://study.com/academy/lesson/agile-implementation-methodology-strategy.html#:~:text=The%20implementation%20of%20the%20agile,(deliver%20to%20the%20customer).

Dhandapani, S. (2016, november). *Integration of User Centered Design and Software Development Process*. Retrieved from researchgate: https://www.researchgate.net/figure/Comparison-of-UCD-and-SDLC-process_tbl1_310548027

Staff, C. (2023, november 29). *What Is Agile and Who Uses It?* . Retrieved from coursera: https://www.coursera.org/gb/articles/what-is-agile-a-beginners-guide

Mnkandla, E. (2008, jan). *A Selection Framework For Agile Methodology Practices: A Family of Methodologies Approach*. Retrieved from researchgate: https://www.researchgate.net/figure/7-Methodology-Description-Table-for-Agile-Modeling_tbl15_228819895

Jachak, S. G. (2024, march 03). *FACE RECOGNITION BASED ON REAL-TIME ATTENDANCE MANAGEMENT SYSTEM*. Retrieved from irjmets: https://www.irjmets.com/uploadedfiles/paper//issue_3_march_2024/50179/final/fin_irjmets1709817290.pdf

aws. (n.d.). *What is Facial Recognition?* . Retrieved from amazon: https://aws.amazon.com/what-is/facial-recognition/#:~:text=Facial%20recognition%20can%20identify%20human,large%20collection%20of%20existing%20images.

AI, T. (2023, november 21). *Top 10 Open Source Facial Recognition libraries and tools*. Retrieved from twine: https://www.twine.net/blog/top-10-open-source-facial-recognition-library-and-tools/#:~:text=Face%20Recognition%20is%20a%20simple,powered%20by%20deep%20metric%20learning.

Komlavi, A. A. (2024, feb 01). *Comparative study of machine learning algorithms for face recognition* . Retrieved from hal.science: https://hal.science/hal-03620410/document#:~:text=Authors%20achieved%20performances%20of%2094.82,is%20the%20best%20classification%20algorithm.

mobibob. (2010, june). *Are UI choices functional or non-functional requirements?* Retrieved from stackoverflow: https://stackoverflow.com/questions/2955610/are-ui-choices-functional-or-non-functional-requirements/2955623#2955623

uxtweak. (n.d.). *Cross-platform compatibility* . Retrieved from uxtweak: https://www.uxtweak.com/ux-glossary/cross-platform-compatibility/#:~:text=Cross%2Dplatform%20compatibility%20refers%20to,tablets%2C%20and%20mobile%20devices

Begum, M. (2023, october). *Comparision of PCA and LDA for Face Recognition*. Retrieved from researchgate: https://www.researchgate.net/publication/374589799_Comparision_of_PCA_and_LDA_for_Face_Recognition

Paul, L. C. (2019, november). *Face recognition using principal component analysis method*. Retrieved from researchgate: https://www.researchgate.net/publication/318362885_Face_recognition_using _principal_component_analysis_method#:~:text=PCA%20is%20a%20statistical%20approach,of%20a%20training%20image%20set

Sunardi. (2022, june). *Face Recognition Using Machine Learning Algorithm Based on Raspberry Pi 4b*. Retrieved from mail.ijair: https://mail.ijair.id/index.php/ijair/article/download/321/pdf#:~:text=To%20improve%20the%20accuracy%20of,neural%20network%20algorithms%20%5B3%5D.

betsol. (2019, march 21). *7 Stages Of SDLC: How To Keep Development Teams Running*. Retrieved from betsol: https://www.betsol.com/blog/7-stages-of-sdlc-how-to-keep-development-teams-running/

Solutions, G. (2023, november 17). *Functional and Non-Functional Requirements: The Ultimate Checklist with Examples*. Retrieved from medium: https://medium.com/@growsolutions/functional-and-non-functional-requirements-the-ultimate-checklist-with-examples-cde16aba33d7

Idrizi, F. (n.d.). *FACE RECOGNITION – ATTENDANCE SYSTEM*. Retrieved from eprints: https://eprints.unite.edu.mk/1102/1/JAS%20SUT%2015-16%20Final-102-111.pdf

Ashritha, K. (2022, july). *Automated Attendance System Using Face Recognition*. Retrieved from ijraset: https://www.ijraset.com/research-paper/automated-attendance-system-using-face-recognition