WEB FACE LOCK

A PROJECT REPORT

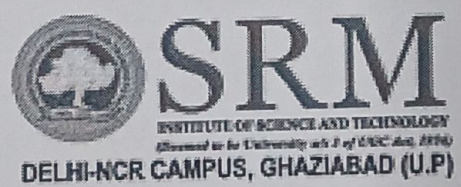
Submitted by

ADITYA SRIVASTAVA [RA2311003030583]
SIDDHARTH NARELA [RA2311003030591]

Under the guidance of

Ms. Neetu Bansla

(Assistant Professor, Department of Computer Science & Engineering)



SRM INSTITUTE OF SCIENCE & TECHNOLOGY,

NCR CAMPUS, **OCT 2024**

SRM INSTITUTE OF SCIENCE & TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified to be the bonafide record of work done by ***ADITYA SRIVASTAVA (RA2311003030583)*** *,* ***SIDDHARTH NARELA (RA2311003030591)*** *of 3rd semester 2nd year* ***B.TECH*** *degree course in* ***SRM INSTITUTE OF SCIENCE AND TECHNOLOGY, NCR Campus*** *of Department of Computer Science & Engineering, during the academic year 2024-2025.*

SIGNATURE

Ms. Neetu Bansla

Assistant Professor

Computer Science & Engineering

SIGNATURE

Dr. Avneesh Vashistha

Head of the Department

Computer Science & Engineering

Table of Contents

INTRODUCTION

In an era where digital security is paramount, the **Web Face Lock** project stands as an innovative solution that harnesses the power of facial recognition technology to provide a secure access method for web applications. As traditional password-based authentication systems face vulnerabilities, including phishing attacks and password theft, facial recognition emerges as a more robust alternative, enabling users to authenticate themselves effortlessly and securely.

The primary aim of Web Face Lock is to create a seamless and secure user experience. By utilizing OpenCV, a leading library in computer vision, this application captures real-time video input from users' webcams to detect and analyze facial features. Upon initiating the login process, the application identifies the user's face and compares it against a pre-defined database of authorized users. If a match is confirmed, the user is granted access to the protected areas of the web application, such as the admin dashboard. This instant recognition process not only enhances security but also streamlines access, eliminating the need for users to remember and input complex passwords.

Built on Flask, a lightweight yet powerful web framework for Python, Web Face Lock efficiently handles web requests and responses. Flask's simplicity allows developers to create a robust backend while focusing on delivering a smooth user experience. The application employs HTML, CSS, and JavaScript to craft a responsive frontend, ensuring users can interact with the system seamlessly across various devices, whether on a desktop or a mobile browser.

Security is a core consideration in the development of Web Face Lock. The application not only protects against unauthorized access through facial recognition but also emphasizes secure data handling practices. User data is stored securely, and the application can be configured to run over HTTPS, protecting user information during transmission. These measures are crucial, as they help build trust and credibility among users in an age where data breaches and privacy concerns are prevalent.

Web Face Lock also provides an opportunity for future enhancements. The foundational architecture is designed to be extensible, allowing for the integration of features such as multi-user support, improved facial recognition algorithms, and additional authentication methods like two-factor authentication (2FA). These enhancements can further increase the system's reliability and adaptability in various use cases, from corporate environments to personal applications.

As users become more aware of the importance of digital security, projects like Web Face Lock illustrate the shift towards more advanced and secure authentication methods. The application not only addresses current security challenges but also paves the way for future innovations in the field of biometric authentication. Developers, security professionals, and users alike will find value in exploring the capabilities and implications of facial recognition technology within web applications.

INSTALLATION & SETUP

To get started with the Web Face Lock application, you'll first need to ensure that your development environment is properly set up. Begin by installing Python 3.x, as it serves as the backbone of the application. You can download it from the official Python website. Once Python is installed, you'll need to set up a virtual environment to manage your project dependencies effectively. You can do this by running the following commands in your terminal or command prompt:

```
python -m venv venv
```

```
source venv/bin/activate
```

With the virtual environment activated, proceed to install the necessary libraries using pip. The main dependencies for Web Face Lock include Flask for web framework functionality, OpenCV for facial recognition capabilities, and NumPy for numerical operations. To install these libraries, run the following command:

```
pip install Flask opencv-python numpy
```

After the libraries are successfully installed, you can clone the project repository (if applicable) using Git:

```
git clone https://github.com/aditya-ig10/web-face-lock.git
```

Navigate to the project directory:

```
cd web-face-lock
```

Next, ensure you have the Haar Cascade XML file for face detection, which is a part of OpenCV. This file is essential for the application's facial recognition functionality. It can typically be found in the OpenCV installation directory or downloaded from the OpenCV GitHub repository.

Finally, you can run the application by executing the following command in your terminal:

```
python app.py #initialising flask
```

This will start the Flask development server, and you can access the application by navigating to

http://127.0.0.1:5000 in your web browser. With these steps completed, you are ready to explore and utilize the Web Face Lock application, enhancing your web security with facial recognition technology.

FRONTEND DEVEOPMENT

The frontend development of the Web Face Lock application is vital for delivering a user-friendly and visually appealing interface. It encompasses the use of HTML, CSS, and JavaScript to create an interactive experience that facilitates secure access through facial recognition.

1. HTML Structure

HTML (HyperText Markup Language) forms the foundation of the frontend, providing the basic structure and semantics of the application. Key components include:

Login Page: This page is the first point of interaction for users, featuring areas for webcam previews and buttons to initiate the facial recognition process. It also includes input fields for optional credentials like username and password.

Admin Dashboard: Once users successfully log in, they are redirected to an admin dashboard, which displays user information, access controls, and management options. This layout allows for easy navigation and efficient access to features.

2. CSS Styling

CSS (Cascading Style Sheets) is used to enhance the visual presentation of the application. Key considerations include:

Layout and Responsiveness: CSS techniques, such as Flexbox and Grid, are employed to ensure that the application is responsive and adapts to different screen sizes. This is essential for accessibility across various devices, including desktops, tablets, and smartphones.

Visual Consistency: A cohesive color scheme and typography improve the overall aesthetic and usability. Consistent styles for buttons, inputs, and backgrounds make the application more intuitive and user-friendly.

3. JavaScript Functionality

JavaScript adds interactivity to the frontend, making the application dynamic. Key functionalities

include:

Webcam Access: JavaScript interacts with the browser's APIs to access the user's webcam, allowing for real-time video streaming. This feature is critical for the facial recognition process.

Interactive Elements: Elements such as buttons and forms are enhanced with JavaScript to respond to user actions, like clicking a login button to initiate facial recognition. This creates a fluid experience without the need for page reloads.

Asynchronous Communication: JavaScript enables the frontend to communicate with the backend asynchronously. This means that when a user captures their face, the image can be sent to the server for processing without disrupting the user interface, allowing for a smooth user experience.

4. Integration with Backend

The frontend is designed to work seamlessly with the backend, allowing for effective data exchange. When a user attempts to log in, the captured facial data is sent to the backend for recognition processing. The frontend receives feedback from the backend to determine whether to grant access or display an error message.

Overall, the frontend development of the Web Face Lock application focuses on creating a responsive, visually appealing, and interactive user experience. By leveraging HTML, CSS, and JavaScript, the application not only meets security requirements through facial recognition but also ensures that users can navigate and interact with the system effortlessly. This thoughtful approach is essential for promoting user engagement and trust in the application's capabilities.

BACKEND DEVELOPMENT

The backend development of the Web Face Lock application is critical for handling data processing, authentication, and overall application logic. It serves as the bridge between the frontend user interface and the underlying data storage and processing systems. Built primarily using Flask, a lightweight Python web framework, the backend is responsible for managing user interactions, executing facial recognition algorithms, and providing secure access control.

1. Framework and Architecture

Flask is chosen for its simplicity and flexibility, allowing developers to create robust applications with minimal overhead. The backend architecture typically consists of several components:

Application Logic: This includes the core functionalities of the Web Face Lock, such as user authentication, session management, and handling requests from the frontend. The application logic processes incoming data, such as facial images captured from the webcam, and orchestrates the responses sent back to the frontend.

Routing: Flask's routing capabilities enable the application to define various endpoints that correspond to different functionalities. For example, routes can be created for login requests, user registration, and data retrieval. Each route is associated with specific functions that execute when a user interacts with the frontend.

2. User Authentication and Facial Recognition

A key feature of the backend is its ability to manage user authentication through facial recognition:

Image Processing: When a user attempts to log in, the backend receives the facial image from the frontend. The application processes this image using OpenCV, applying techniques to detect and recognize faces. This step often involves comparing the captured image against a database of authorized users.

Database Management: The backend interacts with a database to store user information, including facial data, usernames, and access permissions. This database serves as a repository for authorized users, allowing for efficient lookups and updates.

Security Measures: Implementing security best practices is essential. The backend ensures that sensitive data, such as facial recognition models and user credentials, are stored securely. Techniques like hashing passwords and using secure connections (HTTPS) help protect user data from unauthorized access.

3. API Integration

The backend serves as an API (Application Programming Interface) that the frontend interacts with. This integration enables the application to operate smoothly:

Request Handling: When the frontend sends requests (e.g., to log in or retrieve user data), the backend processes these requests and sends appropriate responses. This includes sending success or error messages based on the outcome of the facial recognition process.

Data Exchange: The backend formats data in a way that the frontend can easily understand, often using JSON (JavaScript Object Notation) for data interchange. This allows for efficient communication between the frontend and backend, ensuring that user interactions are handled promptly.

4. Error Handling and Logging

Robust error handling is a key aspect of backend development. The application should be capable of gracefully managing unexpected situations:

Error Responses: The backend is designed to send informative error messages back to the frontend when something goes wrong, such as a failed login attempt or an issue with data processing.

Logging: Implementing logging mechanisms helps track application performance and user activity. This information is valuable for debugging and improving the application over time.

The backend development of the Web Face Lock application is fundamental for ensuring secure, efficient, and effective user authentication through facial recognition. By leveraging Flask and integrating with OpenCV for image processing, the backend manages application logic, handles user requests, and interacts with the database, all while prioritizing security and performance. This robust backend architecture enables the frontend to provide a smooth user experience, ultimately making the application a reliable solution for secure web access.

CODE SNIPPET (FOR FACE RECOGNITION)

```
import face_recognition

import cv2

import os

import time

import webbrowser

import sys

import pygetwindow as gw


known_people_dir = "known_people"

known_face_encodings = []

known_face_names = []


for file_name in os.listdir(known_people_dir):
```

```
    if file_name.endswith(".jpg") or file_name.endswith(".png"):
        name = os.path.splitext(file_name)[0]
        image_path = os.path.join(known_people_dir, file_name)
        image = face_recognition.load_image_file(image_path)
        encoding = face_recognition.face_encodings(image)[0]
        known_face_encodings.append(encoding)
        known_face_names.append(name)


video_capture = cv2.VideoCapture(0)
video_width = 640
video_height = 480
video_capture.set(cv2.CAP_PROP_FRAME_WIDTH, video_width)
video_capture.set(cv2.CAP_PROP_FRAME_HEIGHT, video_height)
video_capture.set(cv2.CAP_PROP_FPS, 1)


start_time = time.time()
known_face_detected = False
frame_number = 0


while True:
```

```
ret, frame = video_capture.read()

frame_number += 1


if frame_number % 2 != 0:

    continue


frame = cv2.flip(frame, 1)

small_frame = cv2.resize(frame, (0, 0), fx=0.25, fy=0.25)


    face_locations = face_recognition.face_locations(small_frame)

    face_encodings    =    face_recognition.face_encodings(small_frame,
face_locations)



    for    (top,    right,    bottom,    left),    face_encoding    in
zip(face_locations, face_encodings):

        matches = face_recognition.compare_faces(known_face_encodings,
face_encoding)

        name = "Unknown"



        if True in matches:

            first_match_index = matches.index(True)
```

```
            name = known_face_names[first_match_index]


            if not known_face_detected:
                start_time = time.time()
                known_face_detected = True


            if time.time() - start_time > 3:
                print("Signing You IN")
                webbrowser.open("D:/LoginIntegration/login.html")
                window = gw.getWindowsWithTitle('Video')[0]
                window.activate()

                sys.exit()


            else:
                known_face_detected = False
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break


video_capture.release()
```

```
cv2.destroyAllWindows()
```

OVERVIEW

The Web Face Lock project is an innovative web application designed to enhance security by utilizing facial recognition technology for user authentication. As traditional password systems face increasing vulnerabilities, this project aims to provide a more secure and convenient alternative, ensuring that only authorized users can access sensitive areas of the application.

At the core of the Web Face Lock is OpenCV, a powerful library for computer vision. The application captures real-time video from the user's webcam, detects faces, and compares them against a database of authorized individuals. When a user attempts to log in, the application evaluates the captured image for a match. If successful, the user is granted access to the admin dashboard; otherwise, access is denied. This seamless process not only enhances security but also improves the user experience by eliminating the need for complex passwords.

The application is built using Flask, a lightweight Python web framework, which facilitates smooth handling of web requests and responses. The frontend is developed with HTML, CSS, and JavaScript, creating an intuitive and responsive user interface. Users can easily navigate through the application, whether logging in or managing settings.

Security is a paramount consideration in the Web Face Lock project. User data is handled with strict protocols, ensuring secure storage and transmission. The application can be configured to run over HTTPS, providing an additional layer of protection against data breaches.

Future enhancements for the Web Face Lock project include the possibility of multi-user support, improved accuracy in facial recognition, and integration with additional authentication methods, such as two-factor authentication. These features will further bolster security while expanding the application's usability.

In summary, the Web Face Lock project represents a significant step forward in secure web authentication, combining advanced technology with user-friendly design to create a reliable and effective solution for digital security challenges.

REFERENCES

1. OpenCV

- Official Documentation: [OpenCV Documentation](https://docs.opencv.org/)

- GitHub Repository: [OpenCV GitHub](https://github.com/opencv/opencv)

2. Flask

- Official Documentation: [Flask Documentation](https://flask.palletsprojects.com/)

- Flask Quickstart Guide: [Flask Quickstart](https://flask.palletsprojects.com/en/2.0.x/quickstart/)

3. Python

- Official Python Website: [Python.org](https://www.python.org/)

- Python Package Index (PyPI): [PyPI](https://pypi.org/)

4. Facial Recognition

- A Beginner's Guide to Facial Recognition: [Towards Data Science Article](https://towardsdatascience.com/facial-recognition-using-opencv-and-python-4b12b4c2f5f7)

- Face Detection using OpenCV: [OpenCV Face Detection Guide](https://docs.opencv.org/master/d7/d8b/tutorial_py_face_detection.html)

5. Community and Forums

- Stack Overflow: [Stack Overflow](https://stackoverflow.com/)

- Reddit - r/learnpython: [r/learnpython](https://www.reddit.com/r/learnpython/)