FACIAL AUTHENTICATION SYSTEM FOR WEB

DEPARTMENT OF COMPUTER SCIENCE

NATIONAL INSTITUTE OF TECHNOLOGY PATNA

Present by :

Vineet Raj (2106019)

Mayank kumar(2106068)

Rupesh(2106005)

Vijay (2106059)

Under the Guidance of :

Dr. Kakali Chatterjee

OUTLINE :

1. INTRODUCTION
2. LITERATURE REVIEW
3. OBJECTIVE
4. METHODOLOGY
5. IMPLEMENTATION
6. CODE AND EVALUATION
7. OUTCOME
8. REFERENCES

PROJECT REPORT

INTRODUCTION :

In the digital era, ensuring secure access to online services is crucial. Traditional methods like passwords are prone to security threats. Facial authentication, using facial recognition technology, offers a convenient and secure alternative. This project aims to develop a facial authentication system tailored for web applications. By integrating facial recognition into web interfaces, users can securely authenticate themselves without passwords. This system promises improved security, user convenience, and privacy, making online interactions smoother and safer.

LITERATURE REVIEW :

- **Advantages**: Facial authentication offers increased security, convenience, and user experience compared to traditional methods like passwords.
- **Accuracy**: Modern facial recognition algorithms achieve high levels of accuracy and reliability, thanks to advances in deep learning and computer vision techniques.
- **Privacy**: While facial authentication enhances security, it also raises privacy concerns regarding the collection and storage of biometric data. Compliance with privacy regulations is crucial.
- **Integration**: Integrating facial authentication into web applications requires user-friendly interfaces compatible with different web browsers and devices.
- **Security**: Facial authentication systems must address security vulnerabilities like spoofing attacks through multi-factor authentication and robust encryption.

OBJECTIVE :

- **Develop a Secure Authentication Method**: Create a facial authentication system that offers a secure and reliable alternative to traditional authentication methods.
- **Enhance User Experience**: Improve user experience by implementing seamless and convenient facial authentication for web applications.
- **Ensure Accuracy and Reliability**: Implement facial recognition algorithms that achieve high accuracy and reliability while minimizing false positives and negatives.
- **Integrate with Web Applications**: Design the system to seamlessly integrate with various web platforms, providing consistent authentication across devices.
- **Address Privacy Concerns**: Develop privacy-conscious practices for collecting, storing, and processing facial biometric data, ensuring compliance with privacy regulations.
- **Mitigate Security Risks**: Identify and address security risks such as spoofing attacks through multi-factor authentication and encryption techniques.
- **Promote User Adoption**: Educate users about the benefits of facial authentication and provide clear and intuitive interfaces to encourage adoption.
- **Evaluate Performance**: Assess system performance through rigorous testing, measuring accuracy, speed, and user satisfaction metrics.

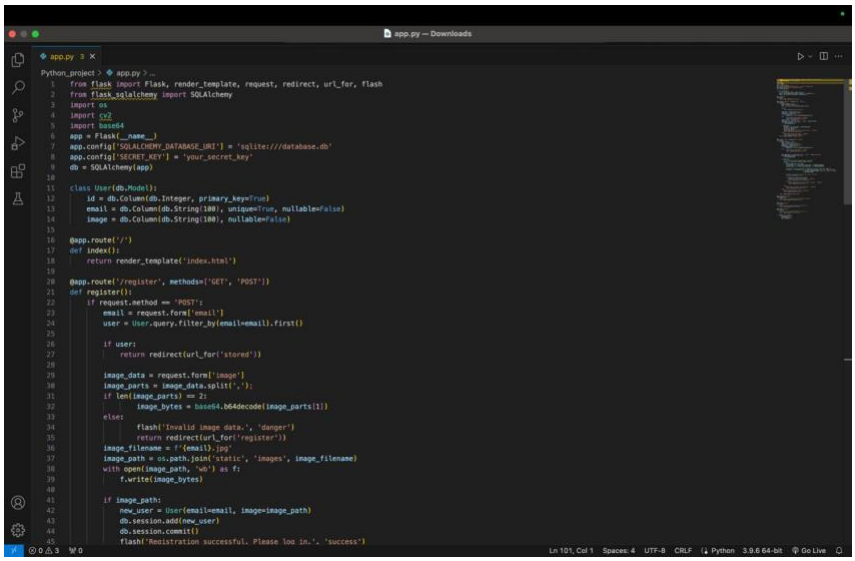
TECHNOLOGIES USED (METHODOLOGY) :

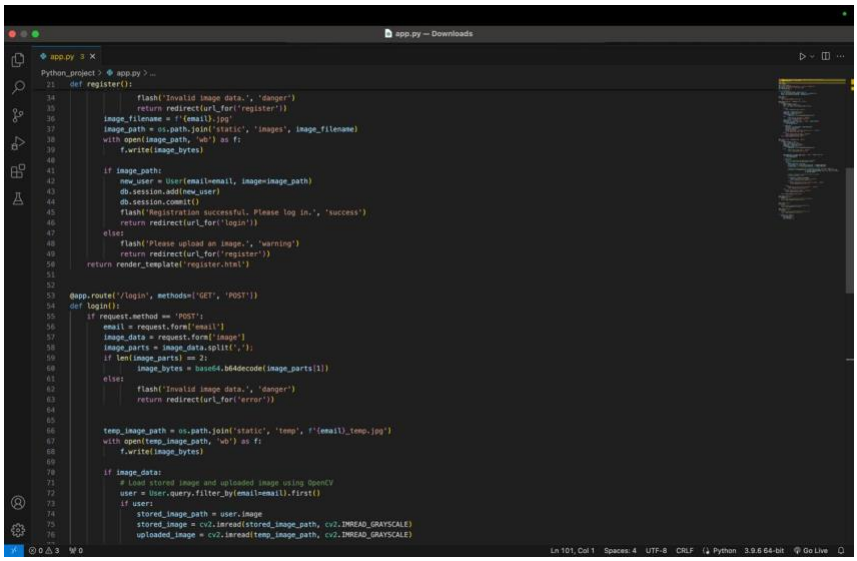
1. **Python**
2. **Flask**
3. **opencv library**
4. **sqlite database**
5. **Javascript**

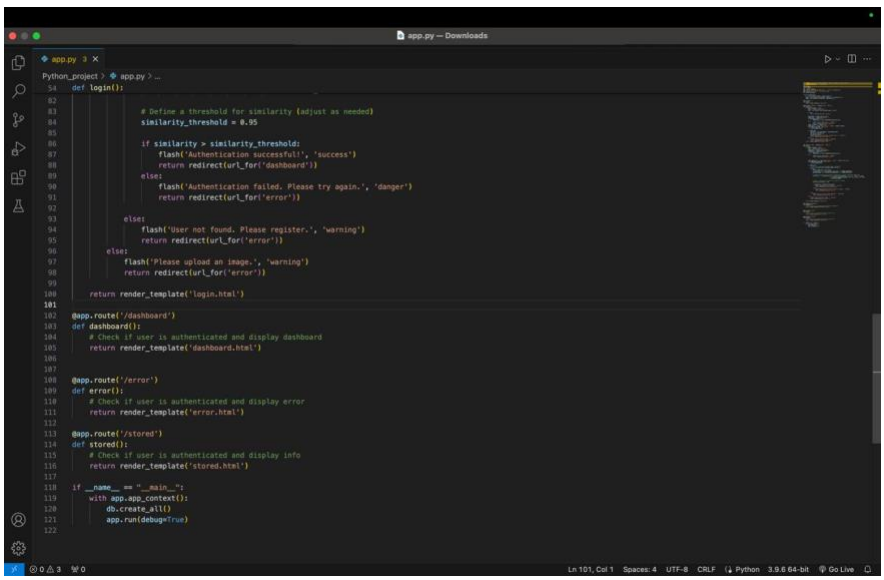
IMPLEMENTATION :

- **Setup Development Environment:** Install Python, Flask, SQLite, and OpenCV.
- **Front-End Design:** Create user interface using HTML, CSS, and JavaScript.
- **Back-End Development:** Write Flask routes for registration, login, and authentication.
- **Facial Recognition Model Training:** Prepare and train a facial recognition model using OpenCV.
- **Integration:** Integrate the facial recognition model with Flask for real-time authentication.
- **User Registration:** Implement a registration form to capture user's username, password, and facial image.
- **User Authentication:** Implement a login form for users to authenticate using facial recognition.
- **Error Handling and Security:** Handle errors gracefully and ensure secure data transmission.
- **Testing:** Test the system thoroughly for performance, accuracy, and reliability.
- **Deployment:** Deploy the system to a web server and configure security measures.

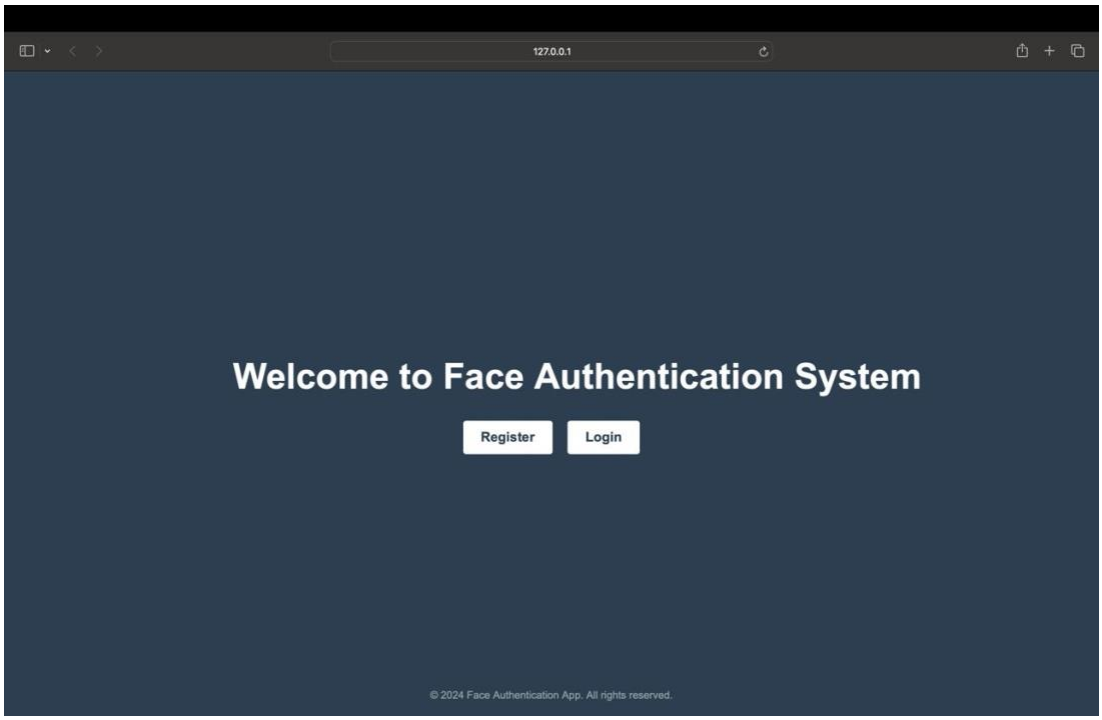
CODE AND EVALUATION :

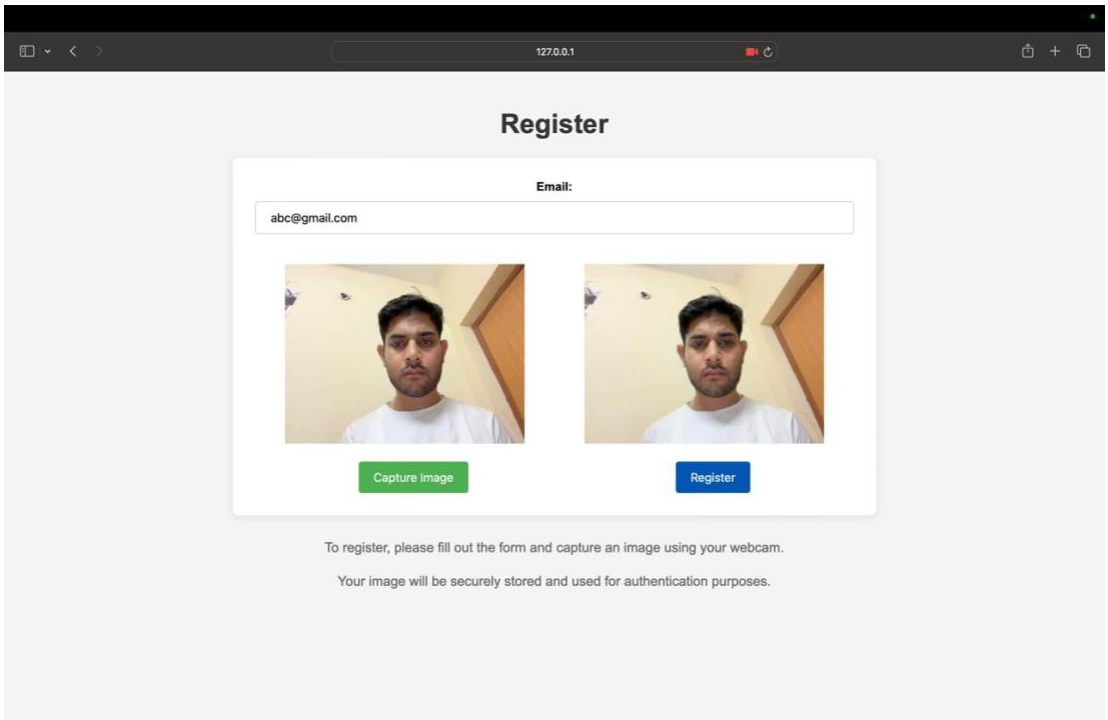

app.py — Downloads
app.py 3
Python_project > app.py > ...
from flask import Flask, render_template, request, redirect, url_for, flash
from flask_sqlalchemy import SQLAlchemy
import os
import cv2
import base64
app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///database.db'
app.config['SECRET_KEY'] = 'your_secret_key'
db = SQLAlchemy(app)
class User(db.Model):
id = db.Column(db.Integer, primary_key=True)
email = db.Column(db.String(100), unique=True, nullable=False)
image = db.Column(db.String(100), nullable=False)
@app.route('/')
def index():
return render_template('index.html')
@app.route('/register', methods=['GET', 'POST'])
def register():
if request.method == 'POST':
email = request.form['email']
user = User.query.filter_by(email=email).first()
if user:
return redirect(url_for('stored'))
image_data = request.form['image']
image_parts = image_data.split(',');
if len(image_parts) == 2:
image_bytes = base64.b64decode(image_parts[1])
else:
flash('Invalid image data.', 'danger')
return redirect(url_for('register'))
image_filename = f'{email}.jpg'
image_path = os.path.join('static', 'images', image_filename)
with open(image_path, 'wb') as f:
f.write(image_bytes)
if image_path:
new_user = User(email=email, image=image_path)
db.session.add(new_user)
db.session.commit()
flash('Registration successful. Please log in.', 'success')
Ln 101, Col 1 Spaces: 4 UTF-8 CRLF Python 3.9.6 64-bit Go Live

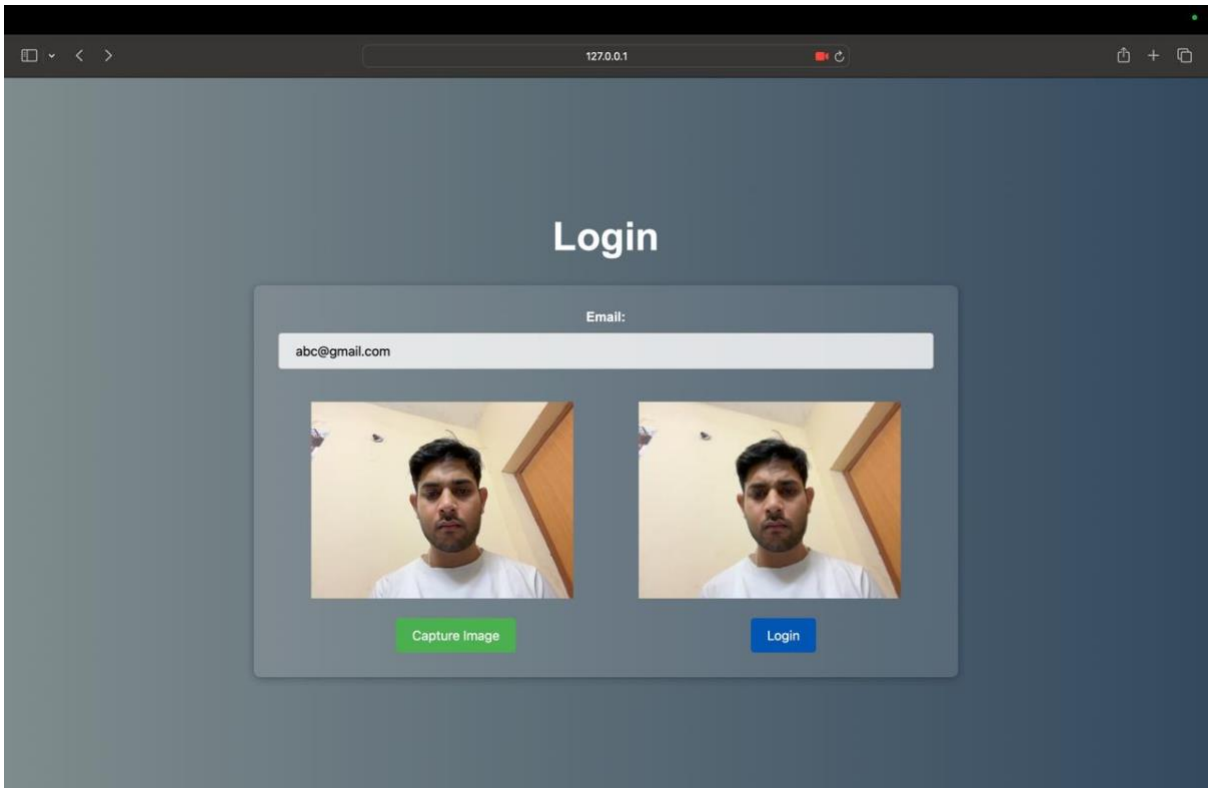

app.py — Downloads
app.py 3
Python_project > app.py > ...
def register():
flash('Invalid image data.', 'danger')
return redirect(url_for('register'))
image_filename = f'{email}.jpg'
image_path = os.path.join('static', 'images', image_filename)
with open(image_path, 'wb') as f:
f.write(image_bytes)
if image_path:
new_user = User(email=email, image=image_path)
db.session.add(new_user)
db.session.commit()
flash('Registration successful. Please log in.', 'success')
return redirect(url_for('login'))
else:
flash('Please upload an image.', 'warning')
return redirect(url_for('register'))
return render_template('register.html')
@app.route('/login', methods=['GET', 'POST'])
def login():
if request.method == 'POST':
email = request.form['email']
image_data = request.form['image']
image_parts = image_data.split(',');
if len(image_parts) == 2:
image_bytes = base64.b64decode(image_parts[1])
else:
flash('Invalid image data.', 'danger')
return redirect(url_for('error'))
temp_image_path = os.path.join('static', 'temp', f'{email}_temp.jpg')
with open(temp_image_path, 'wb') as f:
f.write(image_bytes)
if image_data:
# Load stored image and uploaded image using OpenCV
user = User.query.filter_by(email=email).first()
if user:
stored_image_path = user.image
stored_image = cv2.imread(stored_image_path, cv2.IMREAD_GRAYSCALE)
uploaded_image = cv2.imread(temp_image_path, cv2.IMREAD_GRAYSCALE)
Ln 101, Col 1 Spaces: 4 UTF-8 CRLF Python 3.9.6 64-bit Go Live

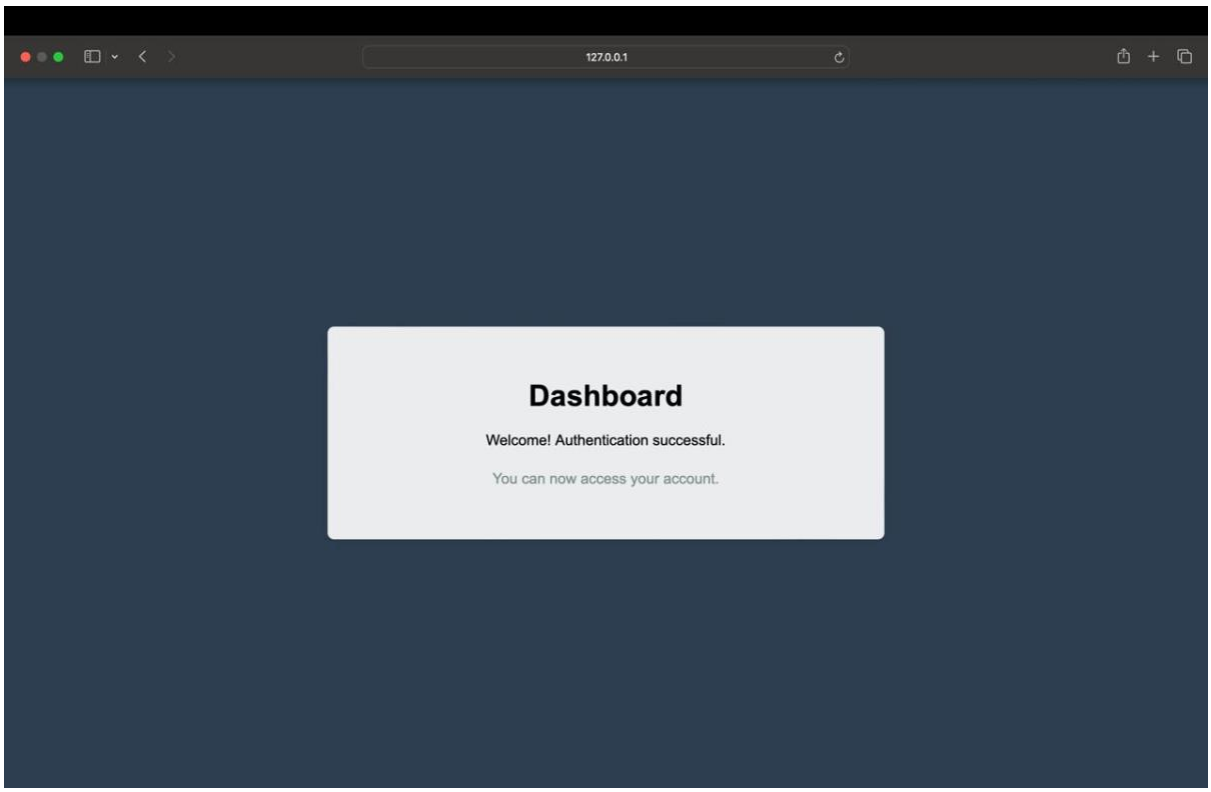

app.py — Downloads
app.py 3
Python_project > app.py > ...
def login():
# Define a threshold for similarity (adjust as needed)
similarity_threshold = 0.95
if similarity > similarity_threshold:
flash('Authentication successful!', 'success')
return redirect(url_for('dashboard'))
else:
flash('Authentication failed. Please try again.', 'danger')
return redirect(url_for('error'))
else:
flash('User not found. Please register.', 'warning')
return redirect(url_for('error'))
else:
flash('Please upload an image.', 'warning')
return redirect(url_for('error'))
return render_template('login.html')
@app.route('/dashboard')
def dashboard():
# Check if user is authenticated and display dashboard
return render_template('dashboard.html')
@app.route('/error')
def error():
# Check if user is authenticated and display error
return render_template('error.html')
@app.route('/stored')
def stored():
# Check if user is authenticated and display info
return render_template('stored.html')
if __name__ == "__main__":
with app.app_context():
db.create_all()
app.run(debug=True)
Ln 101, Col 1 Spaces: 4 UTF-8 CRLF Python 3.9.6 64-bit Go Live

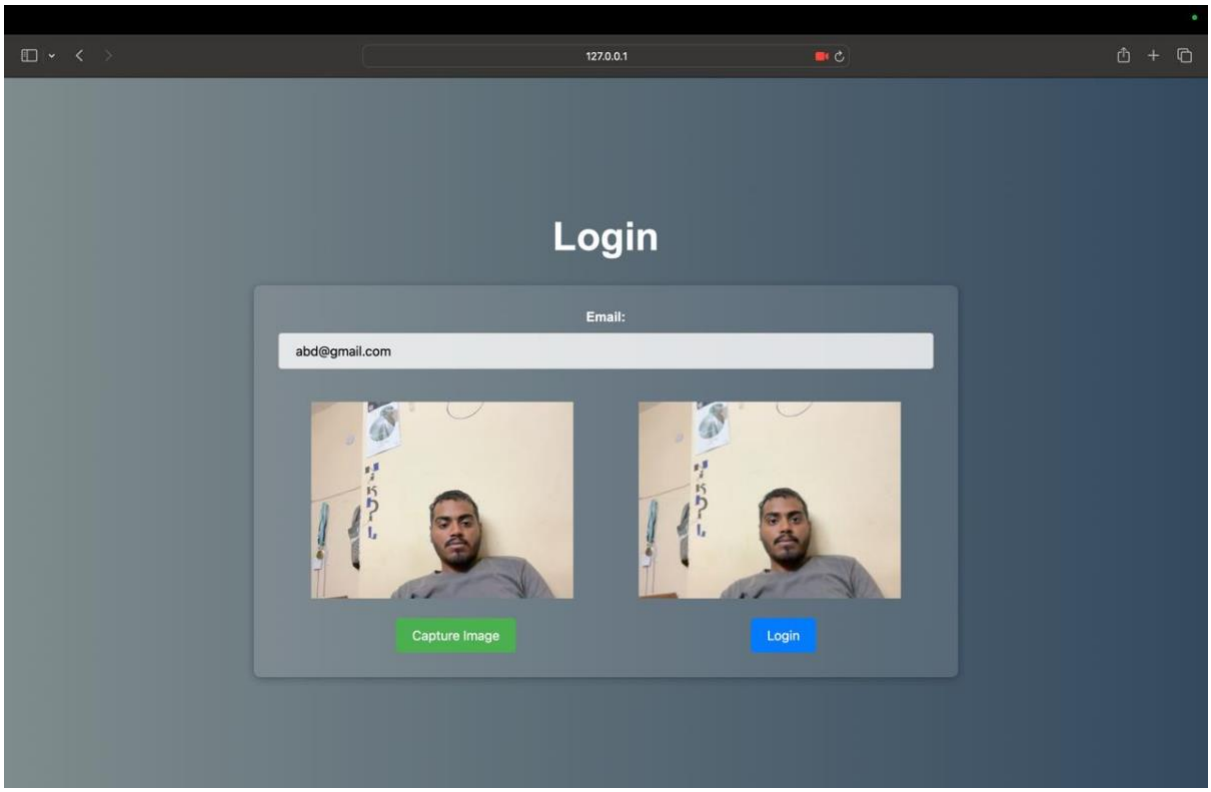
OUTCOME :

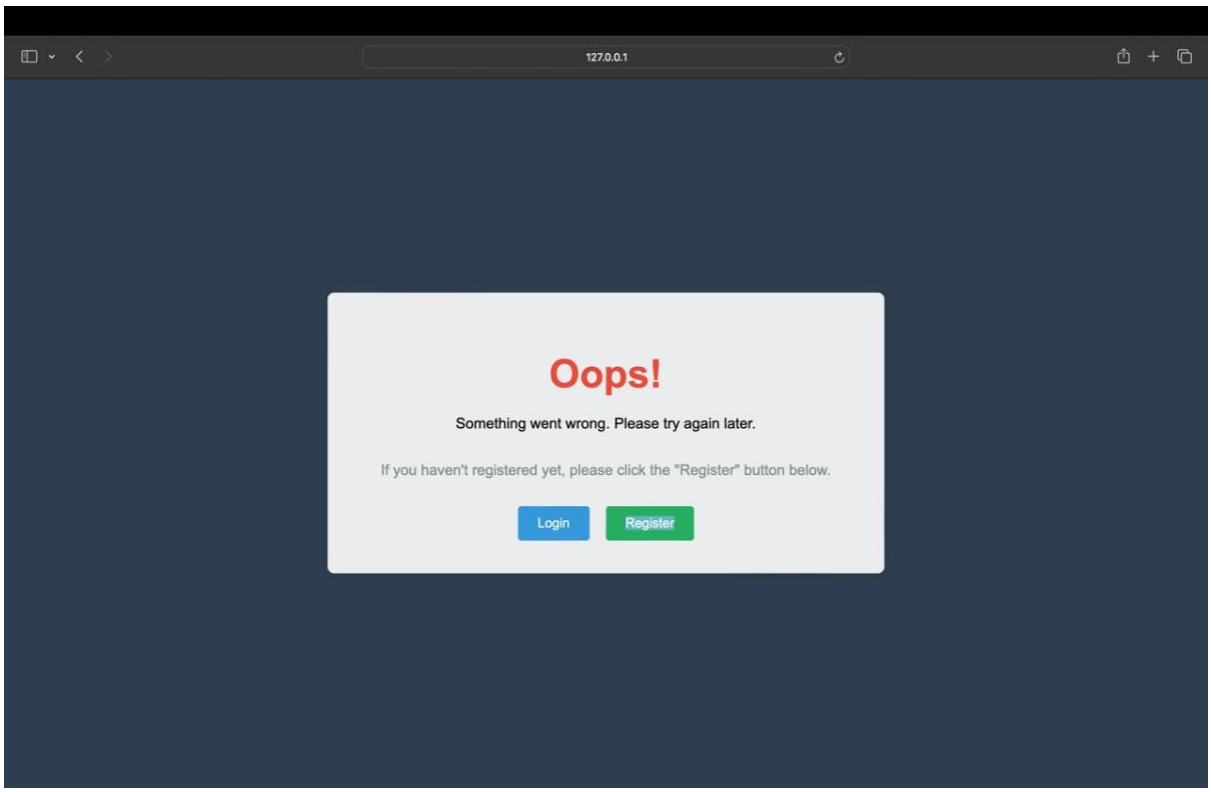

127.0.0.1
Welcome to Face Authentication System
Register
Login
© 2024 Face Authentication App. All rights reserved.

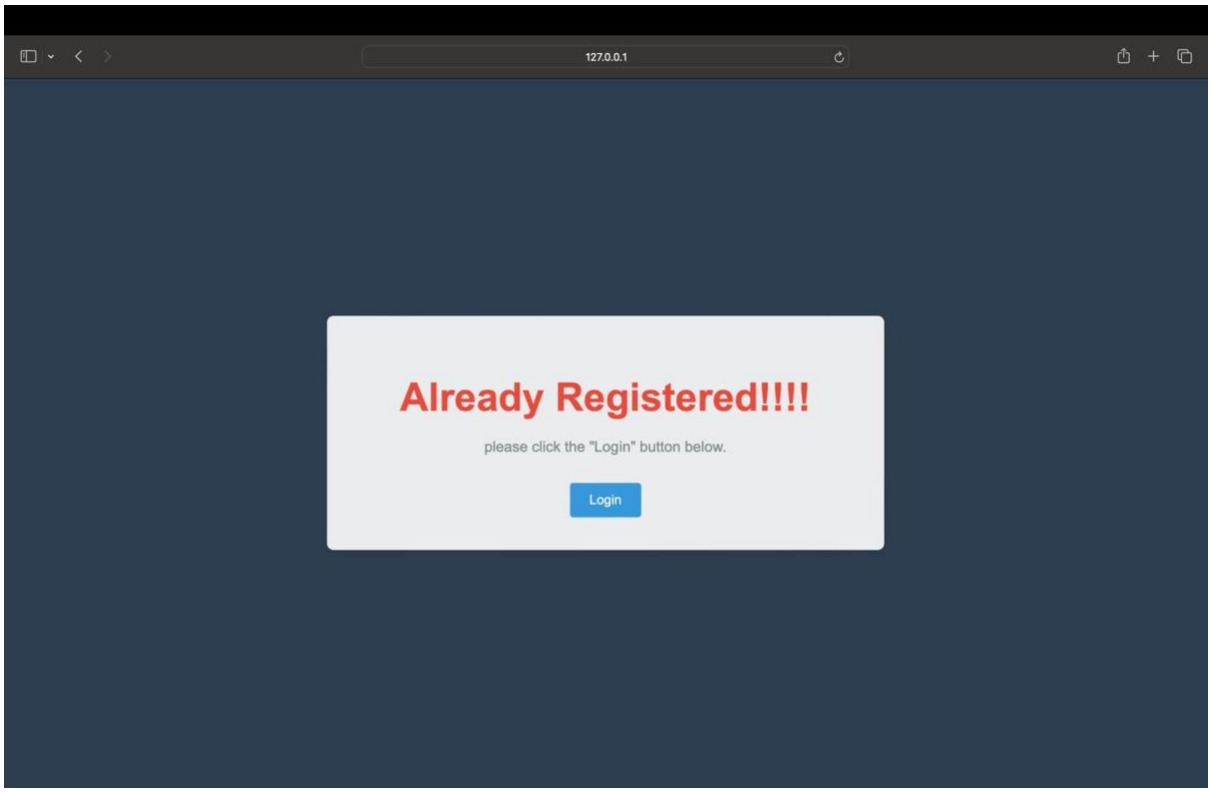

127.0.0.1
Register
Email:
abc@gmail.com
Capture Image
Register
To register, please fill out the form and capture an image using your webcam.
Your image will be securely stored and used for authentication purposes.


127.0.0.1
Login
Email:
abc@gmail.com
Capture Image
Login


127.0.0.1
Dashboard
Welcome! Authentication successful.
You can now access your account.


127.0.0.1
Login
Email:
abd@gmail.com
Capture Image
Login


127.0.0.1
Oops!
Something went wrong. Please try again later.
If you haven't registered yet, please click the "Register" button below.
Login
Register



Accuracy = 95%

REFERENCES :

1. https://www.kaspersky.com/resource-center/definitions/what-is-facial-recognition.
2. https://www.geeksforgeeks.org/face-recognition-using-artificial-intelligence/
3. https://www.geeksforgeeks.org/opencv-python-program-face-detection/
4. https://medium.com/@draj0718/opencv-face-detection-deployment-in-flask-web-framework-1a9b9772d9fd