Biometric Voter Identification System Using Face Recognition

A PROJECT REPORT

submitted

in the partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

By

Polishetty Pranay	20B81A0586
Mohammed Rahimuddin Amaan	20B81A0587
Perumbudur Sai Deepak	20B81A0596

Under the guidance of

Mr P Shankar

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CVR COLLEGE OF ENGINEERING

(*An Autonomous institution, NBA, NAAC Accredited and Affiliated to JNTUH, Hyderabad*)

Vastunagar, Mangalpalli (V), Ibrahimpatnam (M),

Rangareddy (D), Telangana- 501 510

May 2023

CVR COLLEGE OF ENGINEERING

(A UGC Autonomous Institution, Affiliated to JNTUH, Accredited by NBA, and NAAC)

Vastunagar, Mangalpalli (V), Ibrahimpatnam (M),

Ranga Reddy (Dist.) - 501510, Telangana State.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project entitled **"Biometric Voter Identification System Using Face Recognition"** being submitted by **Polishetty Pranay (20B81A0586), Mohammed Rahimuddin Amaan (20B81A0587), Perumbudur Sai Deepak (20B81A0596)** in partial fulfilment for theaward of Bachelor of Technology in **Computer Science and Engineering** to the CVR College of Engineering, during the academic year 2022-2023.

Signature of Project guide,	**Professor-in-charge projects**
Mr P Shankar	**Dr. G. Balakrishna**
Assistant Professor	Professor, Department of CSE
Department of CSE	
External Examiner	**Signature of the HOD**
	Dr. A. Vani Vathsala
	Head of Department, CSE

ACKNOWLEDGEMENT

The satisfaction of that accompanies the successful completion of any task would be incomplete without the mention of the people who made it possible and whose encouragement and guidance has been a source of inspiration throughout the course of the project.

It is a great pleasure to convey out profound sense of gratitude to our principal **Dr. Rama Mohan Reddy, Vice-Principal Prof. L. C. Siva Reddy, Dr. A. Vani Vathsala**, Head of CSE Department, CVR College of Engineering, for having been kind enough to arrange necessary facilities for executing the project in the college.

We sincerely thank our project guide **Mr P Shankar** underwhom we have carried out the project work. Their incisive and objective guidance and timely encouraged us with constant flow of energy to continue the work.

We wish a deep sense of gratitude and heartful thanks to management for providing excellent lab facilities and tools. Finally, we thank all those guidance helpful to us in this regard.

ABSTRACT

The Biometric Voter Identification System using face recognition is a mini project that aims to improve the accuracy and security of the voting process. The system uses facial recognition technology to authenticate voters and ensure that they are eligible to cast their vote. The project involves the development of a user-friendly software application that captures and stores biometric data, including the facial features of registered voters. On election day, voters are required to present themselves at the polling station, and their facial features are matched against the database to verify their identity. The Biometric Voter Identification System using face recognition provides several benefits over traditional voting systems. It helps to prevent voter impersonation and multiple voting, thereby ensuring the fairness and integrity of the election process. The system also enables faster and more accurate voter identification, reducing wait times and enhancing the overall voting experience.

Overall, the Biometric Voter Identification System using face recognition is an innovative solution that leverages advanced biometric technology to enhance the security and accuracy of the electoral process. With its ability to prevent electoral fraud and streamline voter identification, this mini project has the potential to revolutionize the way we conduct elections.

TABLE OF CONTENTS

Table of Contents			
			Page No.
		List of Figures	V
1		**Introduction**	1
	1.1	Motivation	2
	1.2	Objectives	3
	1.3	Problem Statement	4
	1.4	Report Organization	5
2		**Literature Survey**	6
	2.1	Existing Work and its limitations	6
3		**Hardware & Software specifications**	8
	3.1	Hardware requirements	8
	3.2	Software requirements	8
4		**Design**	9
	4.1	UML Diagram	9
	4.2	Activity Diagram	9
	4.3	Class Diagram	10
	4.4	Sequence Diagram	11
	4.5	Algorithms	13
5		**Implementation**	16
6		**Conclusion**	28
7		**References**	30

LIST OF FIGURES

Figure	Title	Page
4.1	UML Diagram	9
4.2	Activity Diagram	9
4.3	Class Diagram	10
4.4	Sequence Diagram	11

1. INTRODUCTION

The introduction section of the documentation provides an overview of the biometric voter identification system using face recognition. It sets the context for the project, outlines its objectives, and presents a brief overview of the subsequent chapters.

Elections are a cornerstone of democratic societies, representing the collective voice of the citizens in shaping the future of their nations. Ensuring the integrity and transparency of elections is of paramount importance to maintain public trust and uphold the democratic principles of fairness and equity. One critical aspect of a robust electoral process is the accurate identification and verification of voters.

The traditional methods of voter identification, such as identity cards and signatures, have limitations in terms of accuracy and vulnerability to fraudulent practices. To overcome these challenges, the adoption of biometric technologies has gained prominence. Biometrics leverage unique physiological or behavioral characteristics to establish an individual's identity, offering higher accuracy and enhanced security.

The focus of this project is to develop a biometric voter identification system using face recognition technology. Face recognition is a biometric modality that analyzes facial features and patterns to identify and verify individuals. By capturing and comparing facial images against a database of registered voters, this system aims to provide a reliable and efficient method for voter authentication.

1.1 MOTIVATION

The motivation behind developing a biometric voter identification system using face recognition stems from the critical need to ensure the integrity and transparency of elections. Voting is the cornerstone of democratic societies, and it is imperative to establish robust mechanisms to prevent fraud, impersonation, and other malpractices that may compromise the electoral process.

Traditional methods of voter identification, such as identity cards and signatures, have limitations in terms of accuracy and security. Biometric technologies, particularly face recognition, offer a promising solution by leveraging unique physiological features to verify a person's identity. By utilizing facial characteristics that are distinct and difficult to forge, a biometric voter identification system enhances the accuracy and reliability of the identification process.The widespread availability and advancements in facial recognition algorithms and hardware components have made it feasible to implement such a system. Facial recognition technology can efficiently analyze facial patterns and match them against a database of registered voters, allowing for quick and accurate identification.

The adoption of a biometric voter identification system addresses several key concerns. It helps eliminate the possibility of multiple voting, impersonation, and fraudulent practices, thereby reinforcing the fairness and credibility of elections. Additionally, it enhances the convenience and efficiency of the voting process by reducing queues and ensuring a streamlined check-in procedure.

Moreover, the biometric voter identification system serves as a deterrent to those intending to manipulate the electoral process, as the high accuracy and non-repudiation provided by face recognition technology act as strong deterrents against fraudulent activities.

By implementing a biometric voter identification system using face recognition, we aim to contribute to the advancement of secure and reliable voting systems. This technology-driven approach offers a robust and scalable solution to uphold the fundamental principles of democracy, safeguarding the rights of citizens and maintaining public trust in the electoral process.

1.2OBJECTIVES

- The primary objectives of this project are as follows:
- Designing and implementing a biometric voter identification system using face recognition technology.
- Enhancing the accuracy and reliability of the identification process to prevent fraudulent practices.
- Streamlining the voter check-in process, reducing queues, and improving overall voting efficiency.
- Contributing to the integrity and transparency of elections by ensuring a secure and tamper-resistant identification system.

1.3 PROBLEM STATEMENT

The problem this project aims to address is the need for a robust and secure method of voter identification in elections. The existing methods of identification have limitations in terms of accuracy, vulnerability to manipulation, and time-consuming processes. By developing a biometric voter identification system using face recognition, we aim to overcome these challenges and provide a more reliable and efficient solution.

1.4 REPORT ORGANIZATION

This report is structured into several chapters to provide a comprehensive understanding of the project. Following this introduction, Chapter 2 presents a literature survey that explores the existing work and its limitations in the field of biometric voter identification systems. Chapter 3 outlines the hardware and software specifications required for implementing the system. Chapter 4 delves into the design aspects of the project, including flowchart diagrams, sequence diagrams, activity diagrams, and algorithms. Chapter 5 focuses on the implementation details, followed by Chapter 6, which presents the conclusion of the project. Lastly, Chapter 7 provides a list of references used throughout the documentation.

By adhering to the guidelines and objectives outlined in this project, we aim to contribute to the development of a reliable, secure, and efficient biometric voter identification system using face recognition technology.

2. LITERATURE SURVEY

2.1 EXISTING WORK AND ITS LIMITATIONS

Numerous studies and research papers have focused on biometric-based voter identification systems, with face recognition emerging as a prominent modality. These studies have highlighted the potential of face recognition technology in enhancing the accuracy and security of voter identification processes. Some key findings from the existing literature are as follows:

Accuracy and Performance: Face recognition algorithms have shown significant advancements in terms of accuracy and performance. Studies have reported high verification and identification rates, demonstrating the potential of face recognition in reliably identifying individuals in large-scale databases.

Robustness to Variations: Face recognition systems have exhibited robustness to variations in pose, illumination, facial expressions, and aging. Advanced algorithms have been developed to handle these challenges, ensuring accurate recognition across different conditions.

Hardware and Computational Requirements: With the increasing availability of powerful computing resources and advancements in hardware, the computational requirements for face recognition systems have become more manageable. Efficient algorithms and hardware optimization techniques have contributed to real-time processing capabilities, making face recognition viable for practical applications.

Despite these advancements, there are several limitations and challenges identified in the existing work:

Data Collection and Quality: The performance of face recognition systems heavily relies on the quality and diversity of the training data. Insufficient data or biased datasets may lead to reduced accuracy and potential biases in the identification process.

Privacy and Ethical Considerations: Biometric systems, including face recognition, raise concerns regarding privacy and the storage of sensitive personal information. Ensuring proper data protection measures and addressing ethical considerations are crucial to maintaining public trust and compliance with legal frameworks.

Vulnerability to Attacks: Face recognition systems can be susceptible to attacks such as spoofing, where adversaries attempt to deceive the system using counterfeit facial images or masks. Robust anti-spoofing techniques and liveness detection methods are essential to mitigate such vulnerabilities.

Scalability and Deployment Challenges: Deploying a biometric voter identification system on a large scale poses logistical challenges, including hardware deployment, database management, and integration with existing electoral processes. Ensuring seamless integration and scalability are vital considerations for practical implementation.

Based on the insights gained from the existing literature, this project aims to address these limitations and build upon the advancements in biometric voter identification systems using face recognition. By incorporating robust algorithms, addressing privacy concerns, and considering practical deployment aspects, the goal is to develop a system that enhances the accuracy, security, and efficiency of voter identification processes in electoral settings.

The subsequent chapters of this documentation will delve into the hardware and software specifications, design, implementation details, and conclusions, providing a comprehensive understanding of the developed biometric voter identification system using face recognition technology.

3. SOFTWARE AND HARDWARE SPECIFICATIONS

This chapter outlines the hardware and software requirements necessary for the implementation of the biometric voter identification system using face recognition technology. Ensuring the availability of suitable hardware components and compatible software is essential for the successful deployment of the system.

3.1 HARDWARE REQUIREMENTS

The hardware requirements for the biometric voter identification system are as follows:

CPU: Intel i5 or equivalent processors

GPU: NVIDIA GeForce GTX 1060 or higher

RAM: 6 GB or higher

Storage: 256 GB or higher

Internet Connection: High Speed Broadband Internet Connection

3.2 SOFTWARE REQUIREMENTS

The software requirements for the biometric voter identification system are as follows:

Programming Languages Used : Python,HTML,CSS

Integrated Development Environment: PyCharm, Visual Studio Code.

Libraries used :

a) **OpenCV:** OpenCV is a library that is widely used in computer vision and machine learning applications.

b) **Face Recognition:** Face Recognition is a Python library used for face detection, recognition, and identification.

c) **Flask:** Flask is a lightweight and versatile web framework for building web applications in Python.

d) **OS:** "os" module in Python provides a way to interact with the operating system, allowing you to perform various tasks like as file operations, directory management, and process handling.

4. DESIGN

4.1 UML DIAGRAMS

The UML (Unified Modeling Language) diagram provides an overview of the system components and their relationships. It illustrates the interactions between actors and system elements. The UML diagram for the "Biometric Voter Identification System" is represented follows:

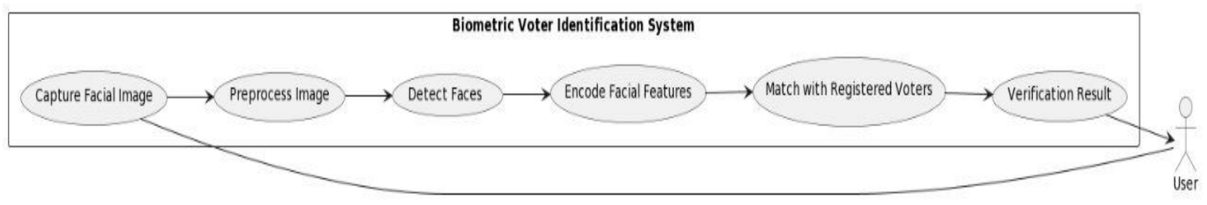

Fig 4.1.1: UML Diagram

4.2 ACTIVITY DIAGRAM

The activity diagram depicts the flow of activities and actions within the system. It shows the sequence of steps involved in the voter identification process. The activity diagram for the "Biometric Voter Identification System" is represented as follows:

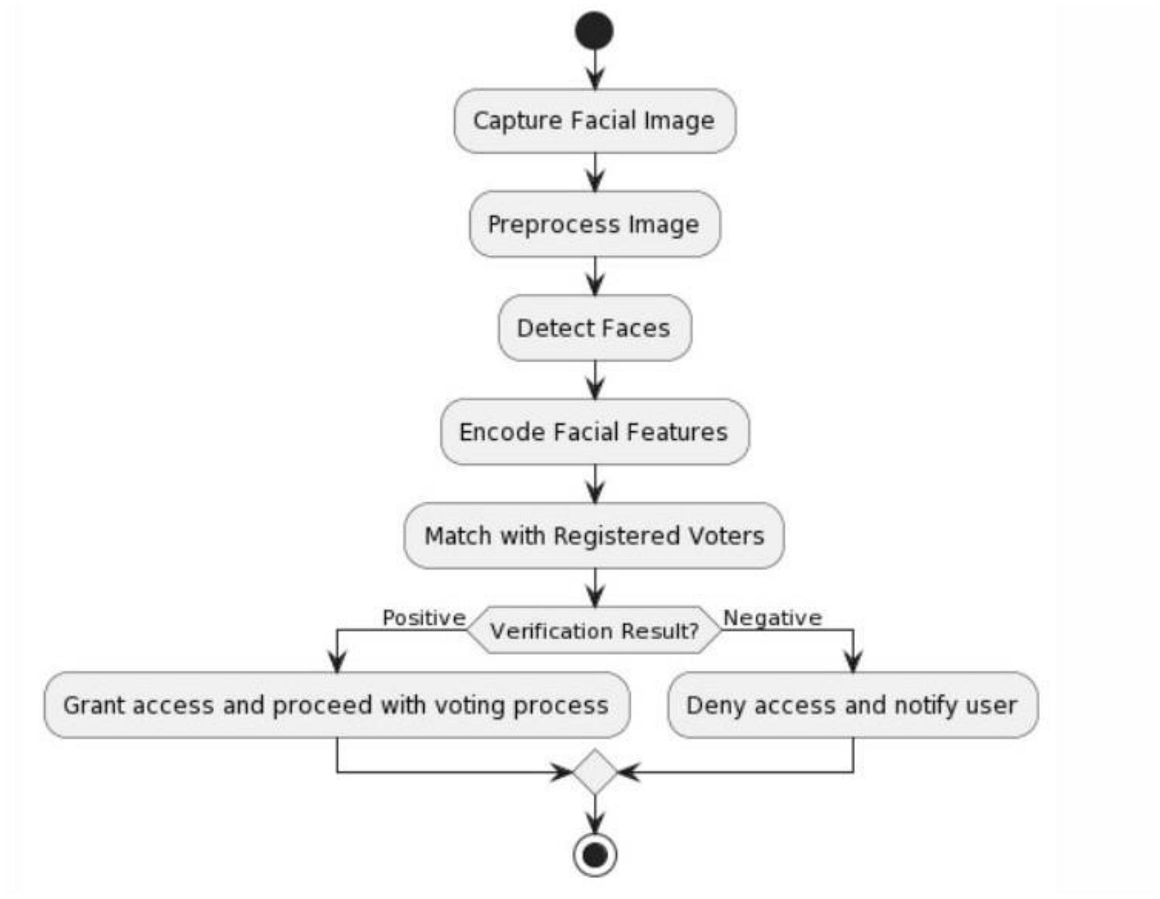

Fig 4.2.1: Activity Diagrams

4.3 CLASS DIAGRAMS

The class diagram represents the static structure of the system by illustrating the classes, their attributes, and relationships. It provides an overview of the system's object-oriented design. The class diagram for the "Biometric Voter Identification System" is represented as follows:

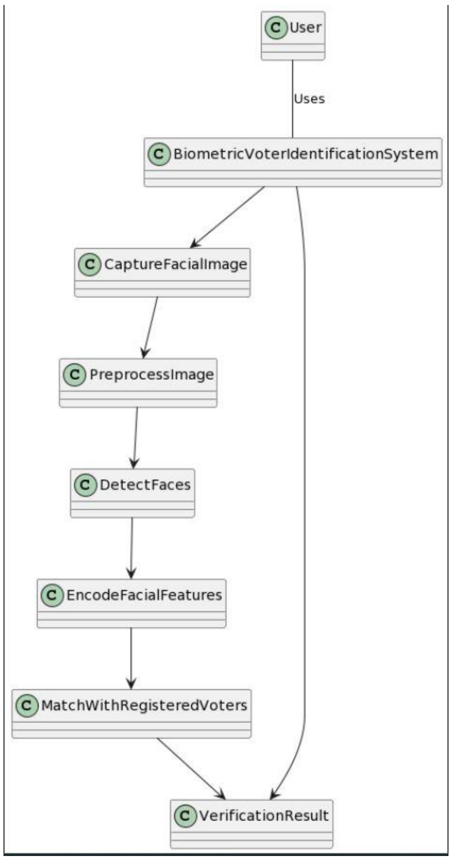

Fig 4.3.1: Class Diagram

4.3 SEQUENCE DIAGRAMS

Visualizes the interaction between different objects or components in the system. it shows the Sequence of messages exchanged between the web application, camera, face recognition module, and voter objects during the voter identification process.

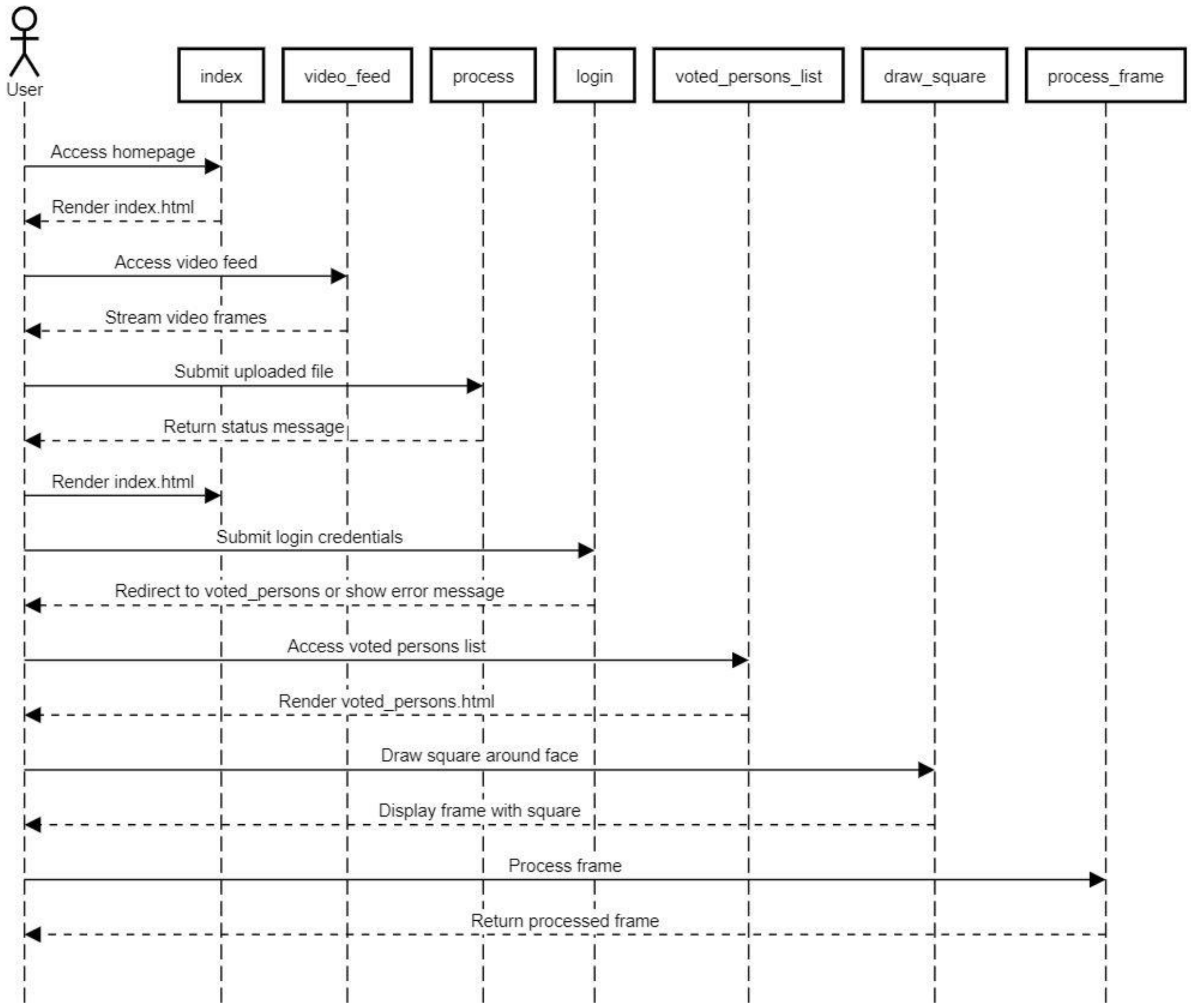

Fig 4.3.1: Sequence Diagram

4.4 ALGORITHMS

4.4.1 VOTER INITIALIZATION ALGORITHM:

The algorithm begins by initializing an empty list to store the voter objects.It iterates through the list of voterIDs, which contains the filenames of the training images for each voter.For each voter ID, a Voter object is created with the name set to the uppercase version of the voter ID.The Voter object is appended to the voters list.This algorithm ensures that a Voter object is created for each voter and added to the list for further processing.

4.4.2 Face Encoding Extraction Algorithm:

The algorithm defines a function called findFaceEncodings, which takes a list of images as input.It initializes an empty list called encodeList to store the face encodings.For each image in the input list, the algorithm performs the following steps:

- The image is converted from the default BGR format to RGB format using the cv2.cvtColor function.
- The face_recognition library's face_encodings function is used to extract the facial features from the image. It returns a list of face encodings.
- The first (and only) face encoding in the list is appended to the encodeList.After processing all the images, the algorithm returns the encodeList, which contains the face encodings for each image.
- This algorithm ensures that the face encodings are extracted from the training images and Stored in a list for later comparison with the captured faces.

4.4.3 Real-time Face Recognition Algorithm:

The algorithm defines a generator function called gen_frames, which continuously captures frames from the webcam using the cv2.VideoCapture object.Inside the loop, it reads a frame from the webcam using the cap.read() function and checks if the reading was successful.The captured frame is then resized and converted from BGR to RGB format.The face_recognition library's face_locations and face_encodings functions are used to detect and encode faces in the resized frame.

For each detected face, the algorithm performs the following steps:

- It compares the face encodings with the known Voter Face Encodings using the face_recognition compare_faces function.
- The distance between the face encodings and known Voter Face Encodings is computed using the face recognition.face_distance function.
- The index of the closest match is obtained using np.argmin function.If the match is positive (i.e., the face belongs to a registered voter), the corresponding Voter object is retrieved from the voters list.
- The bounding box and label are drawn on the frame, indicating the recognized voter's name.
- The mark As Voted method of the Voter object is called to mark the voter as voted.
- The processed frame is converted to JPEG format and yielded as a response.This algorithm continuously captures frames from the webcam, performs real-time face recognition, and updates the frame with bounding boxes and labels for recognized voters.

4.4.4 Image Upload and Processing Algorithm:

- The algorithm defines a route named '/process', which handles the image upload and processing.
- It first checks if a file was uploaded and if it is a valid image file.If the image is valid, it is saved to the specified path.
- The face_recognition library's load_image_file and face_encodings functions are used to load and encode the uploaded image.
- The algorithm then compares the face encoding with the knownVoterFaceEncodings to identify any matches.

- If a match is found, the voter's name is flashed as a message, indicating successful identification.If no match is found, an 'Unknown Voter' message is flashed.
- Additionally, if no face is detected in the uploaded image, a 'No face detected' message is flashed.
- This algorithm allows users to upload an image for identification and performs face matching against the known voter face encodings.

These algorithms collectively enable the biometric voter identification system to perform real-time face recognition using webcam input and process uploaded images for identification purposes.

5. IMPLEMENTATION

I. app.py

```
 from flask import Flask, render_template, request, flash, redirect, Response
import cv2
import numpy as np
import face_recognition
import os
import csv

app = Flask(__name__)

# Specify the directory path where the training images are stored
path = 'training-images'
# Initialize an empty list to store the training images
images = []
# Initialize an empty list to store the voter IDs
voterIDs = []
# Get the list of files in the specified directory
voterList = os.listdir(path)

current_user = None

for vid in voterList:
    curImg = cv2.imread(f'{path}/{vid}')
    images.append(curImg)
    voterIDs.append(os.path.splitext(vid)[0])


class Voter:
    def __init__(self, name):
        self.name = name
        self.voted = False

    def markAsVoted(self):
        if self.voted:
            return "Already Voted"
        else:
            with open('Voters-List.csv', 'a+', newline='') as f:
                writer = csv.writer(f)
                f.seek(0)
                voterDataList = [line for line in csv.reader(f)]
                voter_names = [row[1] for row in voterDataList]
                if self.name in voter_names:
                    self.voted = True
                    return "Already Voted"
                else:
                    writer.writerow([len(voterDataList) + 1, self.name, "Voted"])
                    self.voted = True
                    return "Voted"


def findFaceEncodings(images):
    encodeList = []

    for img in images:
        # Convert the image from BGR to RGB color space
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        # Encode the face in the image using face_recognition library
        faceEncode = face_recognition.face_encodings(img)[0]
        # Append the face encoding
        encodeList.append(faceEncode)
    return encodeList


# Get the face encodings for the known voters' images

knownVoterFaceEncodings = findFaceEncodings(images)
cap = cv2.VideoCapture(0)

voters = []
for name in voterIDs:
    voters.append(Voter(name.upper()))


# def gen_frames():
#     while True:
#         success, img = cap.read()
#         imgS = cv2.resize(img, (0, 0), None, 0.25, 0.25)
#         imgS = cv2.cvtColor(imgS, cv2.COLOR_BGR2RGB)
#
#         facesCurFrame = face_recognition.face_locations(imgS)
#         faceEncodingsCurFrame = face_recognition.face_encodings(imgS, facesCurFrame)
#
#         for faceEncode, faceLoc in zip(faceEncodingsCurFrame, facesCurFrame):
#             matches = face_recognition.compare_faces(knownVoterFaceEncodings,
faceEncode)
#             faceDis = face_recognition.face_distance(knownVoterFaceEncodings,
faceEncode)
#
#             matchIndex = np.argmin(faceDis)
```

```
#
#               if matches[matchIndex]:
#                   voter = voters[matchIndex]
#                   y1, x2, y2, x1 = faceLoc
#                   y1, x2, y2, x1 = y1 * 4, x2 * 4, y2 * 4, x1 * 4
#                   color = (0, 255, 0)
#
#                   if voter.voted:
#                       color = (0, 0, 255)
#
#                   cv2.rectangle(img, (x1, y1), (x2, y2), color, 2)
#                   cv2.rectangle(img, (x1, y2 - 35), (x2, y2), color, cv2.FILLED)
#                   cv2.putText(img, voter.name, (x1 + 6, y2 - 6),
cv2.FONT_HERSHEY_COMPLEX, 1, (255, 255, 255), 2)
#                   voter.markAsVoted()
#
#           ret, buffer = cv2.imencode('.jpg', img)
#           frame = buffer.tobytes()
#           yield b'--frame\r\n'b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n\r\n'

def gen_frames():
    global current_user

    while True:
        success, img = cap.read()
        imgS = cv2.resize(img, (0, 0), None, 0.25, 0.25)
        imgS = cv2.cvtColor(imgS, cv2.COLOR_BGR2RGB)

        facesCurFrame = face_recognition.face_locations(imgS)
        faceEncodingsCurFrame = face_recognition.face_encodings(imgS, facesCurFrame)

        for faceEncode, faceLoc in zip(faceEncodingsCurFrame, facesCurFrame):
            matches = face_recognition.compare_faces(knownVoterFaceEncodings, faceEncode)
            faceDis = face_recognition.face_distance(knownVoterFaceEncodings, faceEncode)

            matchIndex = np.argmin(faceDis)

            if matches[matchIndex]:
                voter = voters[matchIndex]
                current_user = voter.name  # Set the current recognized user

                y1, x2, y2, x1 = faceLoc
                y1, x2, y2, x1 = y1 * 4, x2 * 4, y2 * 4, x1 * 4
                color = (0, 255, 0)

                if voter.voted:
                    color = (0, 0, 255)
                cv2.rectangle(img, (x1, y1), (x2, y2), color, 2)
                cv2.rectangle(img, (x1, y2 - 35), (x2, y2), color, cv2.FILLED)
                cv2.putText(img, voter.name, (x1 + 6, y2 - 6), cv2.FONT_HERSHEY_COMPLEX,
1, (255, 255, 255), 2)
                voter.markAsVoted()

        ret, buffer = cv2.imencode('.jpg', img)
        frame = buffer.tobytes()
        yield b'--frame\r\n'b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n\r\n'


# Render the index.html template
@app.route('/')
def index():
    return render_template('index.html')


# Stream video frames as a response
@app.route('/video_feed')
def video_feed():
    return Response(gen_frames(), mimetype='multipart/x-mixed-replace; boundary=frame')


# Process the uploaded file
@app.route('/process', methods=['POST'])
def process():
    if 'file' not in request.files:
        flash('No file uploaded!')
        return render_template('index.html')

    file = request.files['file']

    if file.filename == '':
        flash('No file selected!')
        return render_template('index.html')

    if file:
        file_path = os.path.join(path, file.filename)
        file.save(file_path)
        # Load the uploaded image
        unknown_image = face_recognition.load_image_file(file_path)
        # Encode the face in the image
        unknown_encoding = face_recognition.face_encodings(unknown_image)

        if len(unknown_encoding) > 0:
            unknown_encoding = unknown_encoding[0]

            for voter in voterList:
                match = face_recognition.compare_faces([voter['encoding']],
unknown_encoding)
                if match[0]:
                    flash('Voter Identified: {}'.format(voter['name']))
                    return render_template('index.html')

            flash('Unknown Voter!')
            return render_template('index.html')

        else:
            flash('No face detected in the uploaded image!')
            return render_template('index.html')

    flash('Error processing the file!')
    return render_template('index.html')


# Handle the login page
@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']

        # Check if the login credentials are correct (e.g., admin:admin)
        if username == 'admin' and password == 'admin':
            return redirect('/voted_persons')
        else:
            flash('Invalid login credentials!', 'error')

    return render_template('login.html')


# Render the list of voted persons
# @app.route('/voted_persons')
# def voted_persons_list():
#     return render_template('voted_persons.html', voted_persons=voterIDs)
@app.route('/voted_persons')
def voted_persons_list():
    voted_persons = [voter.name for voter in voters if voter.voted]
    return render_template('voted_persons.html', voted_persons=voted_persons,
current_user=current_user)



# Function to draw a square around a face in a frame
def draw_square(frame, coordinates):
    top, right, bottom, left = coordinates
    cv2.rectangle(frame, (left, top), (right, bottom), (0, 255, 0), 2)


# Function to process a frame
def process_frame(frame):
    # Convert BGR frame to RGB
    rgb_frame = frame[:, :, ::-1]

    # Find faces in the frame
    face_locations = face_recognition.face_locations(rgb_frame)
    face_encodings = face_recognition.face_encodings(rgb_frame, face_locations)

    for (top, right, bottom, left), face_encoding in zip(face_locations, face_encodings):
        # Check if the detected face matches with any known voter
        matches = face_recognition.compare_faces([voter[0] for voter in voterList],
face_encoding)

        if True in matches:
            matched_voter = voterList[matches.index(True)]

            # Check if the voter has already voted
            if matched_voter[1] in voterList:
                flash('You have already voted!', 'warning')
                continue

            # Check if the voter is a registered voter
            if matched_voter[1] not in voterIDs:
                flash('You are not a registered voter!', 'danger')
                continue

            # Add the voted person to the list
            voterList.append(matched_voter[1])

            # Save the voter's details in a CSV file
            with open('voted_persons.csv', 'a') as file:
                csv_writer = csv.writer(file)
                csv_writer.writerow([matched_voter[1]])

            flash('You May Go For Voting!', 'success')

        draw_square(frame, (top, right, bottom, left))

    return frame


# Run Flask application on server
if __name__ == '__main__':
    app.run(host="0.0.0.0", debug=True)
```

- The implementation of the biometric voter identification system utilizes the Flask framework to create a web-based interface for users to interact with the system. The Flask app is initialized, and routes are defined to handle different functionalities.
- The system uses the OpenCV library to capture video frames from the webcam. The frames are processed using the face_recognition library, which provides face detection and recognition capabilities. The system first loads the training images of registered voters and extracts their facial features, known as face encodings. These encodings serve as the reference for identifying voters later.
- The Flask app has routes that handle various tasks. The index route renders the main web page, where users can see the real-time video feed from the webcam. The video feed is processed by the gen_frames generator function, which detects faces in each frame, compares them with the known face encodings, and identifies the voters if a match is found. The identified voters are displayed on the video feed with a bounding box and their names.
- Additionally, the system allows users to upload an image for identification through the /process route. The uploaded image is compared with the known face encodings to determine if the person is a registered voter. If a match is found, the voter is identified, and a corresponding message is displayed.
- The implementation includes features to handle voting status. The Voter class keeps track of voters and their voting status. Once a voter is identified, the system checks if the voter has already voted. If not, the voter's status is marked as voted, and the information is recorded in a CSV file.
- The system also includes a login functionality, where an admin can log in to view the list of voters who have already voted. The admin can access this functionality through the /login route.

Overall, the implementation combines webcam video streaming, face detection and recognition, and Flask web development to create a user-friendly interface for biometric voter identification. It provides real-time identification of voters through the webcam feed and allows users to upload images for identification. The system ensures that voters can only vote once by keeping track of their voting status.

II. Index.html :

```html
<!DOCTYPE html>
<html>
  <head>
    <title>Biometric Voter Validation System</title>
    <link
      rel="stylesheet"
      href="{{ url_for('static', filename='styles.css') }}"
    />
    <link
      rel="stylesheet"
      href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css"
    />
    <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
  </head>
  <body>
    <h1>Biometric Voter Validation System</h1>
    {% with messages = get_flashed_messages() %} {% if messages %}
    <div class="container">
      <div class="row">
        <div class="col-md-6 offset-md-3">
          {% for message in messages %}
          <div
            class="alert alert-{{ message.category }} text-center"
            role="alert"
          >
            {{ message }}
          </div>
          {% endfor %}
        </div>
      </div>
    </div>
    {% endif %} {% endwith %}
    <div class="video-container">
      <img src="{{ url_for('video_feed') }}" />
    </div>
  </body>
</html>
```

- The provided HTML code represents the index.html template for the Biometric Voter Validation System web application. Here's a breakdown of its structure and functionality:

- The HTML document begins with the usual doctype declaration and the opening <html> tag.
- The <head> section includes the title of the web page, which is set to "Biometric Voter Validation System". It also includes links to external stylesheets for custom styling and the Bootstrap CSS framework for responsive design. Additionally, it includes a script tag that imports the Bootstrap JavaScript library for enhanced functionality.
- A block of Jinja template code is used to retrieve and display flashed messages. The code uses the get_flashed_messages() function to get any flashed messages from the Flask application and iterates over them to display each message in an alert box. The messages are displayed inside a Bootstrap container and centered on the page.

- Next, there is a <div> element with a class of "video-container". Inside this container, an <img> tag is used to display the video feed from the Flask route "/video_feed". The url_for('video_feed') function generates the URL for the video feed route.

III. Login.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>Login</title>
    <link
      rel="stylesheet"
      href="{{ url_for('static', filename='styles.css') }}"
    />
    <link
      rel="stylesheet"
      href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css"
    />
    <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
  </head>
  <body>
    <h1>Login</h1>
    {% with messages = get_flashed_messages() %} {% if messages %}
    <div class="container">
      <div class="row">
        <div class="col-md-6 offset-md-3">
          {% for message in messages %}
          <div
            class="alert alert-{{ message.category }} text-center"
            role="alert"
          >
            {{ message }}
          </div>
          {% endfor %}
        </div>
      </div>
    </div>
    {% endif %} {% endwith %}
    <form action="/login" method="post">
      <input
        type="text"
        name="username"
        placeholder="Username"
        required
      /><br />
      <input
        type="password"
        name="password"
        placeholder="Password"
        required
      /><br />
      <input type="submit" value="Login" />
      <input type="reset" value="Reset" />
    </form>
  </body>
</html>
```

- The above page is login.html , it is specifically designed for administrative personnel to check the persons who have voted and who is voting.
- It shows any errors by using the get_flashed_messages() function to get any flashed messages from the Flask application and iterates over them to display each message in an alert box. The messages are displayed inside a Bootstrap container and centered on the page.

IV. Voted_persons.html :

```
<!DOCTYPE html>
<html>
<head>
    <title>Voted Persons</title>
    <link
            rel="stylesheet"
            href="{{ url_for('static', filename='styles.css') }}"/>
    <link
            rel="stylesheet"
            href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.cs
s"/>
    <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
</head>
<body>
<h1>Voted Persons</h1>
<div class="container">
    <div class="row">
        <div class="col-md-6 offset-md-3">
            <table class="table">
                <thead>
                <tr>
                    <th>Serial No.</th>
                    <th>Name</th>
                </tr>
                </thead>
                <tbody>
                {% for person in voted_persons %}
                <tr>
                    <td>{{ loop.index }}</td>
                    <td>{{ person }}</td>
                </tr>
                {% endfor %}
                </tbody>
            </table>

            {% if current_user %}
            <div class="alert alert-success" role="alert">
                Current User: {{ current_user }}
            </div>
            {% endif %}

        </div>
    </div>
</div>
</body>
</html>
```

- The above file is a html file which shows the no of persons who have voted and the person who is currently voting.

6. CONCLUSION

Based on the results obtained following conclusions can be drawn:

In conclusion, the implementation of the biometric voter identification system using face recognition provides an efficient and reliable solution for voter authentication. The system leverages the power of computer vision techniques and web development to create a seamless and secure voting experience.

Through the use of face detection and recognition algorithms, the system can accurately identify registered voters by comparing their facial features with the known encodings. This eliminates the need for traditional identification methods such as ID cards or manual verification, making the voting process more convenient and efficient.

The implementation also incorporates real-time video streaming from the webcam, allowing voters to be identified in real-time as they appear in the camera feed. This feature enhances the system's responsiveness and ensures a smooth user experience.

Furthermore, the system incorporates measures to prevent multiple voting by maintaining a record of voters who have already cast their votes. This ensures the integrity of the voting process and helps maintain fairness and transparency.

Overall, the biometric voter identification system using face recognition offers several benefits. It improves the accuracy and efficiency of voter authentication, reduces the risk of fraudulent voting, and enhances the overall security of the electoral process. By leveraging modern technologies, the system contributes to the advancement of democratic practices and helps build trust and confidence in the voting system.

While the implemented system showcases the potential of biometric voter identification, it is important to consider additional factors such as scalability, data privacy, and security measures when deploying such systems in real-world scenarios. Further research and development are necessary to address these challenges and ensure the widespread adoption of biometric-based voting systems.

In conclusion, the implemented system provides a solid foundation for further advancements in biometric voter identification, paving the way for more secure, efficient, and inclusive voting processes in the future.

7. REFERENCES

[1] Face Recognition Library (face_recognition): https://github.com/ageitgey/face_recognition

[2] OpenCV: Open Source Computer Vision Library: https://opencv.org/

[3] Flask: A Python Microframework: https://flask.palletsprojects.com/

[4] Python Programming Language: https://www.python.org/

[5] Processing and recognition of face images and its applications:
https://books.google.com/books?hl=en&lr=&id=kkNRcZAWmuQC&oi=fnd&pg=PA162&dq=Cognitec.+http://www.cognitec-systems.de/index.html.&ots=FFzED1Zpo1&sig=qC6SICEMpIyToT2nQKqYUZjS3gc