FACE RECOGNITION VOTING SYSTEM

A PROJECT REPORT

Submitted by

DEVADHARSHINI.G

DHARANYA.T

MALINI.S

In partial fulfilment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

P. A. COLLEGE OF ENGINEERING AND TECHNOLOGY

(An Autonomous Institution)

Pollachi, Coimbatore Dt. - 642 002

NOVEMBER 2024

P.A. COLLEGE OF ENGINEERING AND TECHNOLOGY

BONAFIDE CERTIFICATE

Certified that this project report **"FACE RECOGNITION VOTING SYSTEM"** is the bonafide work of **"DEVADHARSHINI.G(721721104015), DHARANYA.T(721721104018), MALINI.S(721721104057)"** who carried out the project work under my supervision.

---	---
SIGNATURE	**SIGNATURE**
Dr. D. CHITRA M.E, Ph.D.,	**Mrs. E. JANANANDHINI M.E.,**
Professor	Assistant Professor
HEAD OF THE DEPARTMENT	**SUPERVISOR**
Department of Computer Science and Engineering	Department of Computer Science and Engineering,
P. A. College of Engineering and Technology	P. A. College of Engineering and Technology
Pollachi-642 002.	Pollachi-642 002.

Submitted to the Viva-Voce Examination held on ……...

Internal Examiner **External Examiner**

ACKNOWLEDGEMENT

First and foremost, we thank the **GOD ALMIGHTY** for blessing us with the mental strength that was needed to carry out the project work. We thank our Chairman **Dr. P. APPUKUTTY M.E, FIE, FIV.,** for his extensive support to successfully carry out this project.

We take privilege in expressing our sincere and heartfelt thanks and gratitude to our beloved Principal **Dr. T. MANIGANDAN M.E., Ph.D.,** for providing us an opportunity to carry out this project work.

We express our sincere thanks to **Dr. D. CHITRA M.E., Ph.D.,** Professor and Head, Department of Computer Science and Engineering for her technical guidance and support.

We express our gratitude and sincere thanks to our Project Guide **Mrs. E. JANANANDHINI M.E.,** Assistant Professor, Department of Computer Science and Engineering, for her technical guidance, constructive criticism and many valuable suggestions provided throughout the project work.

We also express our gratitude and sincere thanks to our Project Coordinator **Mr. T. DINESH KUMAR M.TECH.,** Assistant Professor, Department of Computer Science and Engineering and all teaching and non-teaching staffs of CSE department, P. A. College of Engineering and Technology, Pollachi, for their encouragement and valuable suggestions.

We take this opportunity to express our indebted gratitude to our parents, friends, family and other members whose belongings and love has always been with us.

ABSTRACT

The Face Recognition Voting System revolutionizes the voting process by integrating advanced facial recognition technology with a digital platform to ensure secure, transparent, and efficient elections. By using machine learning algorithms, the system captures and verifies voters' facial features in real-time, comparing them against a pre-registered database to authenticate their identities. The system eliminates risks such as fraud, identity theft, and multiple voting, allowing only eligible voters to participate. At polling stations, voters' facial images are taken, matched with stored biometric data, and upon successful verification, they access an electronic voting interface to cast their vote, receiving instant confirmation.

The innovative system enhances voter privacy, reduces fraud, and minimizes administrative expenses while speeding up the voting process. It is user-friendly, scalable, and compatible with existing election infrastructures, promoting greater participation and trust in the electoral system. By modernizing voting procedures, the face recognition voting system represents a significant advancement in securing democratic elections globally, ensuring fair and efficient electoral processes.

TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGE NO
	ABSTRACT	IV
	LIST OF FIGURES	IX
	LIST OF ABBREVIATIONS	X
1	**INTRODUCTION**	1
	1.1 INTRODUCTION TO FACE RECOGNITION	1
	1.2 PROBLEM STATEMENT	3
	1.3 BACKGROUND OF VOTING SYSTEM	4
	1.4 CHALLENGES IN TRADITIONAL VOTING SYSTEM	5
	1.5 IMPORTANCE OF SECURITY IN VOTING	6
	1.6 OBJECTIVES OF THE FACE RECOGNITION VOTING SYSTEM	7
	1.7 INTRODUCTION TO DATABASE IN FACE RECOGNITION VOTING SYSTEM	8
2	**LITERATURE SURVEY**	10
3	**SYSTEM REQUIREMENTS**	15
	3.1 HARDWARE REQUIREMENTS	15
	3.2 SOFTWARE REQUIREMENTS	15
	3.2.1 PYTHON	16
	3.2.2 OPEN CV	16
4	**SMART VOTING SYSTEM USING FACE DETECTION AND RECOGNITION ALGORITHMS**	18

	4.1 TECHNOLOGIES USED	18
	4.2 PROGRAMMING LANGUAGES	19
	4.3 PROCESS FLOW	19
	4.4 DRAWBACKS	20
5	**FACE RECOGINITION VOTING SYSTEM**	22
	5.1 VOTER REGISTERATION	22
	5.2 AUTHENTHICATION	22
	5.3 VOTE CASTING	23
	5.4 VOTE STORAGE	23
	5.5 LIVENESS DETECTION	23
	5.6 ALGORITHMS USED	24
	5.6.1 HAAR CASCADE ALGORITHM	24
	5.6.2 DEEPFACE ALGORITHM	24
	5.7 CROSS PLATFORM ACCESSBILITY	25
	5.8 PROCESS FLOW	25
	5.8.1 INITIALIZATION	25
	5.8.2 APPLICATION STARTUP	26
	5.8.3 CANDIDATE PROFILES	26
	5.8.4 FACE AUTHENTICATION	27
	5.8.5 VOTING PROCESS	27
	5.8.6 VOTE COUNTING AND RESULTS DISPLAY	28
	5.8.7 APPLICATION TERMINATION	28
	5.9 MITIGATING THE DRAWBACKS OF SMART VOTING SYSTEM	29
	5.9.1 OVERCOMING INITIAL COST	29
	5.9.2 OVERCOMING TECHNOLOGY DEPEDENCE	29
	5.9.3 OVERCOMING ACCURACY CHALLENGES	30
	5.9.4 OVERCOMING PRIVACY CONCERNS	31

6	**IMPLEMENTATION AND RESULT**	33
	6.1 SOURCE CODE	33
	6.1.1 IMPORTS	33
	6.1.2 INITIALIZING THE DATABASE	34
	6.1.3 ADDING A VOTER	35
	6.1.4 RECOGNIZING A FACE	36
	6.1.5 CASTING A VOTE	37
	6.1.6 KIVY APPLICATION (USER INTERFACE)	38
	6.1.7 TITLE AND BACKGROUND	38
	6.1.8 CANDIDATE PROFILES AND VOTING UI	39
	6.1.9 VOTING ACTION PANEL	39
	6.1.10 CAMERA FEED UPDATE	39
	6.1.11 CAPTURING AND RECOGNIZING A FACE	40
	6.1.12 VOTING FOR A CANDIDATE	40
	6.1.13 CLOSING THE APPLICATION	41
	6.1.14 RUNNING THE APLLICATION	41
	6.2 RESULTS	42
7	**CONCLUSION AND FUTURE ENHANCEMENT**	45
	7.1 CONCLUSION	45
	7.2 FUTURE ENHANCEMENT	45
	REFERENCES	47

LIST OF FIGURES

FIGURE NO	TITLE	PAGE NO
1.1	SYSTEM ARCHITECTURE DIAGRAM	22
1.2	READY TO CAPTURE IMAGE	42
1.3	AUTHENTICATION SUCCESSFUL	42
1.4	VOTING SUCCESSFUL	43
1.5	DUPLICATE VOTE ATTEMPTED	43
1.6	ELECTION RESULT	44

LIST OF ABBREVIATIONS

CV2	Open Source Computer Vision Version 2
DB	DataBase
GUI	Graphical User Interface
ID	Identification
JSON	JavaScript Object Notation
OPENCV	Open Source Computer Vision Library
RAM	Random Access Memory
UI	User Interface
CNN	Convolutional Neural Networks
KNN	K-Nearest Neighbors

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION TO FACE RECOGNITION

A Face Recognition System is an advanced biometric technology used to identify or verify an individual's identity by analyzing facial features. This process begins with capturing a digital image of a person's face, which is then analyzed to extract key features such as the distance between the eyes, the shape of the nose, the contour of the cheeks and the jawline. These facial data points are converted into a unique numerical representation known as a face embedding. The system compares these embedding against a stored database of faces to either verify the identity (1:1 matching) or identify the individual among a group (1: N matching).

The recognition process involves several stages including face detection, preprocessing, feature extraction and matching. In face detection, the system identifies the face within an image using algorithms like Haar Cascades or Convolutional Neural Networks (CNNs). Once the face is detected, the system preprocesses the image, often by converting it into grayscale, adjusting lighting, or aligning the face to standardize the angle and size for better recognition. In the feature extraction stage, deep learning models like DeepFace, FaceNet and VGG-Face are used to extract distinctive facial features, creating a unique identifier for the face. These models automatically detect and extract features without requiring manual input, ensuring a high level of accuracy.

The system compares them against a database of known faces using algorithms like Euclidean Distance or K-Nearest Neighbors (KNN) to measure similarity and make a decision about whether a match exists. Face recognition systems can be applied in various fields, including security and surveillance, where they help law enforcement identify suspects in crowds or match faces from surveillance footage to criminal databases. The technology is also widely used in access control systems, enabling individuals to unlock devices, doors, or secure areas using only their face for authentication. Identity verification in sectors such as banking, airports and online services has become increasingly common, providing a secure and convenient method of confirming identity without the need for passwords or ID cards.

Retailers use face recognition to personalize shopping experiences by identifying repeat customers and tracking behavior for targeted marketing. Despite its advantages, there are significant challenges and ethical concerns related to face recognition technology. Privacy is a major issue, as the collection and storage of biometric data raise concerns about surveillance and unauthorized access to personal information. Bias is another concern, as many face recognition systems have been found to perform less accurately when identifying individuals from certain demographics, such as people of color or women. These biases can lead to unfair outcomes, such as misidentification or exclusion. Additionally, the widespread use of face recognition has sparked debates about consent, as individuals may not always be aware that their faces are being scanned or used for identification purposes. In response to these challenges, researchers and developers are working to improve the accuracy, fairness and privacy of face recognition systems. Advances in AI and deep learning continue to enhance the capabilities of these systems, allowing them to perform more accurately in diverse real-world conditions. At the same time, ethical guidelines and regulations are being developed to govern the use of face recognition technology, ensuring that it is used responsibly and does not

infringe upon individual rights. Despite these concerns, face recognition remains a transformative technology that continues to shape the future of security, convenience and user experience across various industries.

1.2 PROBLEM STATEMENT

The voting process is a cornerstone of democracy, allowing citizens to express their choices freely and fairly. However, traditional voting systems face several issues that compromise their integrity, efficiency, and reliability. Voter impersonation is a significant problem, where individuals cast votes on behalf of others, undermining the fairness of elections and leading to disputed results. Duplicate voting, where the same individual casts multiple votes, further distorts election outcomes, making it difficult to trust the results. Additionally, manual processes in traditional voting systems are prone to errors. For example, counting paper ballots manually can result in human errors, delays, and mismanagement, all of which can lead to contested elections. Electronic voting machines (EVMs) have been introduced to address some of these issues, but they are not immune to tampering or malfunctioning, raising concerns about their transparency and reliability. The absence of real-time monitoring in many systems makes it difficult for authorities to identify and address issues as they occur, such as technical failures or unauthorized access. Data security is another pressing concern, as traditional systems often lack robust mechanisms to prevent vote tampering or unauthorized access to sensitive information. To address these challenges, there is a pressing need for a more secure, transparent, and tamper-resistant voting system. This system should authenticate voters accurately, prevent fraudulent activities, and ensure the integrity of every vote cast.

1.3 BACKGROUND OF VOTING SYSTEMS

Voting has evolved over centuries, beginning with simple verbal declarations or hand-raising in small communities to the complex systems used today. Paper ballots**,** introduced in the 19th century, allowed voters to express their choices privately and anonymously. While effective, paper ballots introduced logistical challenges, including storage, transportation, and manual counting errors.

In the 20th century, electronic voting machines (EVMs) were introduced to address these inefficiencies. EVMs enabled faster counting and reduced the possibility of human error. However, these machines brought new challenges, such as potential hacking and a lack of transparency. Without a physical trail, verifying the results of electronic votes remains difficult.

With advancements in technology, there has been growing interest in digital voting systems**,** including internet-based voting and biometric voting solutions. Biometric systems, such as fingerprint or facial recognition, aim to address the issues of voter impersonation and multiple voting. These systems offer a higher level of security by ensuring that only verified individuals can cast their votes.

The evolution of voting systems reflects society’s efforts to balance efficiency, accessibility, and security. As challenges persist, the integration of modern technologies like face recognition and real-time databases marks a significant step toward achieving these goals.

1.4 CHALLENGES IN TRADITIONAL VOTING SYSTEMS

Traditional voting systems, whether paper-based or electronic, are faced with numerous challenges that significantly undermine their effectiveness and reliability. Voter fraud is one of the most pressing issues, with impersonation and duplicate voting being particularly common, especially in large-scale elections where identity verification mechanisms are often limited or inefficient. These fraudulent practices severely compromise the integrity of the election process, leading to skewed results and eroded public trust. In addition to fraud, manual counting errors remain a persistent challenge. When votes are counted by hand, human errors are inevitable, potentially resulting in incorrect totals that could require costly and time-consuming recounts to rectify. The logistical complexity of traditional voting systems further exacerbates these problems, especially in the case of paper ballots. Transporting and securing these ballots from polling stations to counting centers introduces a significant administrative overhead, and with it comes the risk of tampering or loss, further compromising the integrity of the election.

Moreover, a lack of transparency is another significant concern, particularly with Electronic Voting Machines (EVMs). Many EVMs do not leave an audit trail, making it difficult to verify election results or detect discrepancies. This opacity can fuel suspicions of tampering or malfunction, undermining public confidence in the election outcome. Limited accessibility is yet another problem, as individuals with disabilities or those living in remote areas often face significant barriers to participation in traditional voting systems. These barriers can prevent eligible voters from casting their ballots, thus limiting the inclusivity of the electoral process. The issue of delayed results is also a notable challenge in traditional systems. Manual processes, whether for counting paper ballots or verifying voter identities, can significantly slow down the release of results, causing frustration and uncertainty among voters.

Furthermore, data security is a critical vulnerability for both paper-based and electronic voting systems. Paper ballots, while seemingly secure, can still be subject to tampering, loss, or misplacement. On the other hand, electronic systems are particularly vulnerable to cyber-attacks, where hackers may exploit weaknesses to alter or steal sensitive voting data. Given these multiple challenges, it is clear that traditional voting systems need significant improvements. Addressing these issues requires innovative solutions that prioritize security, scalability and efficiency. The integration of advanced technologies, such as biometric verification and real-time databases, holds the potential to revolutionize the voting process, making it more secure, transparent and accessible to all eligible voters.

1.5 IMPORTANCE OF SECURITY IN VOTING

Security is the cornerstone of any voting system, as it ensures the integrity and fairness of the electoral process. Without robust security measures, elections are at risk of being compromised by fraud, tampering or cyber-attacks. A secure voting system guarantees that each vote cast is recorded accurately, attributed to the correct candidate and counted as intended. This ensures that every voter’s choice is respected and that election outcomes are a true reflection of the electorate's will.

Preventing fraud is essential, as a secure system can block unauthorized individuals from casting votes, eliminating the risk of impersonation or other fraudulent activities. Moreover, the system must ensure that each voter votes only once, thus preventing multiple voting attempts, which could distort results. Data integrity is another key concern; security measures must protect the system from tampering, ensuring that vote data remains accurate and unaltered from the time it is cast until it is counted. This guarantees that the election results are reliable and trustworthy.

Confidentiality is also critical, as voters must be able to cast their votes without fear of retaliation or coercion. A secure system safeguards voter anonymity and ensures that personal information is kept confidential, protecting voters' privacy throughout the voting process. Additionally, transparency plays an essential role in building trust. A secure voting system provides mechanisms for verifying results, allowing voters, administrators and observers to have confidence that the election process is fair and free of manipulation.

The integration of technologies like facial recognition, real-time databases, and encryption adds multiple layers of security to the voting process. By addressing traditional vulnerabilities, these advanced technologies help create a more resilient and reliable voting system, ultimately enhancing voter confidence and participation.

1.6 OBJECTIVES OF THE FACE RECOGNITION VOTING SYSTEM

The primary objective of the Face Recognition Voting System is to ensure secure, accurate, and efficient voter authentication. Using facial recognition technology, the system can verify the identity of voters with high accuracy, ensuring that only eligible voters are allowed to participate. This biometric verification minimizes the risk of voter impersonation and other fraudulent activities, such as duplicate voting.

The system will implement real-time database checks to track voter activity, preventing multiple voting attempts and ensuring that each individual casts only one vote. The voting process will be simplified through the use of a user-friendly interface developed with Kivy, making the system accessible and intuitive for users of all technical backgrounds. This design will ensure that the system is easy to navigate, regardless of a voter's familiarity with technology.

Votes will be securely stored in real time using Firebase, allowing administrators to monitor the election as it progresses and ensuring the vote data is safe and up-to-date. The system will be scalable, capable of handling elections of varying sizes, from small institutional votes to larger municipal elections, making it adaptable to different voting environments.

Fraud prevention will be a key feature, with technologies like OpenCV and DeepFace being used to prevent spoofing attempts and other forms of voter impersonation. These technologies will ensure that only legitimate voters can participate, maintaining the integrity of the election. Finally, transparency will be built into the system, enabling administrators and authorized users to audit and verify votes, providing additional assurance that the election process is both fair and transparent.

1.7 INTRODUCTION TO DATABASE IN THE FACE RECOGNITION VOTING SYSTEM

In the Face Recognition Voting System, the database plays an essential role in ensuring secure and efficient storage of voter and voting data as well as supporting the smooth operation of the system throughout the election process. The database stores vital information, including voter details, voting records, and results, making it a cornerstone of the system's functionality. By integrating a database into the system, all voter interactions—such as voter authentication, vote casting, and activity tracking—are accurately recorded and managed.

The system uses an SQLite database, which is an efficient, lightweight relational database management system that is well-suited for small to medium-sized applications. SQLite is used here to store essential data such as voter information, voting status, and authentication records. Voter details include a unique identifier for each voter, their facial recognition data (in the form of feature embeddings or other representations) and their voting history. This

ensures that the system can effectively match each voter’s facial features with their registration record, preventing impersonation or fraud during voting.

Each vote is linked to a specific voter, ensuring that only registered voters can cast their votes. As part of the system's real-time voting feature, the database checks for duplicate voting by tracking voter participation and status. Once a vote is cast, the system immediately updates the voter’s status in the database to reflect that they have already voted. This helps prevent multiple votes from the same individual, enhancing the integrity of the election results.

The database also stores voting records, such as vote counts and results for each candidate or option in the election. These records are updated continuously as votes are cast, ensuring that the most current information is available for monitoring by election administrators. Moreover, the system logs relevant actions—such as successful authentication attempts, votes cast, and failed authentication attempts—in a separate table, providing an audit trail for transparency and accountability.

Given that the database is integral to ensuring the accuracy, security, and efficiency of the voting system, it is designed to handle large amounts of data while maintaining performance and reliability. The system is built to scale to accommodate elections with varying voter populations, from small institutional votes to larger municipal elections.

Overall, the use of an SQLite database in the Face Recognition Voting System ensures that the entire election process is well-documented, secure, and tamper-resistant. It supports real-time voter verification and voting history tracking, providing administrators with full control and transparency over the election process.

CHAPTER 2

LITERATURE SURVEY

SCHEME ON EFFICIENT MULTI-FACTOR AUTHENTICATION FOR E-VOTING

A. K. Roy et al (2019) proposes a multi-factor authentication (MFA) system for e-voting that combines face recognition with one-time password (OTP) verification. The authors argue that using only one authentication method, like facial recognition, may not provide enough security, especially in systems that are vulnerable to spoofing. Thus, integrating OTP with facial recognition ensures that even if an attacker gains access to a person's face image, they would still need the OTP to cast a vote. The paper also focuses on optimizing the time complexity of the authentication process, ensuring that the voting system is both secure and efficient. It provides an in-depth analysis of how combining these two methods enhances the security of e-voting systems while maintaining a user-friendly experience.

SCHEME ON AI-POWERED FACE RECOGNITION FOR SECURE E-VOTING

J. Patel et al (2020) delves into the use of artificial intelligence (AI), specifically convolutional neural networks (CNNs), for improving face recognition in e-voting systems. The authors argue that AI can significantly enhance the accuracy and efficiency of facial recognition by training models with large datasets of facial images. CNNs are particularly well-suited for face

recognition tasks due to their ability to learn from data and recognize patterns. The paper discusses the importance of selecting diverse datasets to ensure that of AI models in real-time applications, addressing challenges such as lighting, pose, and occlusion. The use of AI for face recognition is critical for increasing the reliability and security of e-voting systems.

SCHEME ON VERIFIABLE RANKED CHOICE ONLINE VOTING SYSTEM

Xuechao Yang et al (2020) proposed a secure, verifiable ranked-choice online voting system using biometric authentication, including facial recognition and homomorphic encryption to ensure privacy. The ranked-choice voting system is designed to allow voters to rank their preferences instead of selecting just one candidate, which provides a more nuanced and democratic approach. The biometric authentication ensures that only legitimate voters can cast a vote, while homomorphic encryption protects the votes during transmission and storage, making them unreadable by anyone, including administrators. The authors propose using distributed cryptographic systems to eliminate the possibility of vote manipulation, even if the system is compromised. This paper emphasizes the integration of facial recognition within the cryptographic framework to guarantee voter identity while maintaining privacy.

SCHEME ON SECURE E-VOTING USING ETHEREUM BLOCKCHAIN

Ali Kaan Koç et al (2020) proposed a hybrid model that integrates blockchain technology (specifically Ethereum) with facial recognition for secure and transparent e-voting. Ethereum's blockchain is leveraged to store votes in a tamper-proof way, ensuring both transparency and security. The use

of facial recognition is introduced as a method for voter authentication to prevent identity theft or fraud. The blockchain guarantees that once a vote is cast, it cannot be altered or deleted, ensuring the integrity of the election process. The authors also explore the scalability and privacy issues that arise when integrating facial recognition and blockchain, making it a robust solution for future elections. This system also eliminates the possibility of vote manipulation, as each vote is recorded on the blockchain and is securely linked to the voter's biometric data.

SURVEY ON BIOMETRIC-BASED SECURE ELECTRONIC VOTING SYSTEM

G. Uma Maheswari et al (2021) provides an overview of biometric methods, including facial recognition, for enhancing the security of electronic voting systems. The paper examines various biometric modalities like fingerprints, face, iris and voice recognition, focusing on their applicability in electronic voting systems. It discusses the scalability challenges in integrating biometric authentication with cryptographic methods, such as public-key infrastructure (PKI) and encryption, to ensure both security and privacy. The paper identifies the issues in maintaining the accuracy of biometric systems, especially under different environmental conditions (e.g., lighting for facial recognition), and emphasizes the need for secure databases to store biometric information. The paper is significant for understanding the foundational principles and challenges in using facial recognition for secure voting systems.

SCHEME ON BLOCKCHAIN-BASED E-VOTING SYSTEM USING FACIAL BIOMETRICS

Rohit Sharma et al (2021) introduced a secure e-voting system that integrates blockchain and facial biometrics to ensure the integrity and transparency of the voting process. The paper discusses how blockchain can be used to store vote data in a decentralized, tamper-proof manner. Each voter's identity is verified using facial recognition, ensuring that the vote is cast by an authorized person and reducing the chances of impersonation. The decentralized nature of blockchain means that no central authority controls the votes, making it impossible to manipulate or tamper with the election result. The authors also explore how combining facial recognition with blockchain can streamline the voting process and enhance voter trust in the election system.

SCHEME ON DESIGN OF REAL-TIME VOTING SYSTEMS USING FIREBASE

P. Gupta et al (2021) presents the design and implementation of a real-time voting system using Firebase, a cloud-based backend solution. The authors focus on using Firebase to ensure fast synchronization of votes in real-time across multiple devices. Firebase's real-time database allows for immediate updates, ensuring that the voting system remains current and accurate. The paper discusses the database schema design for storing vote data and voter information, as well as techniques for preventing issues such as duplicate votes. While facial recognition is not the primary focus of the paper, it provides useful insights into how such a system could be integrated with real-time voting platforms to ensure quick authentication and voting. The paper also touches on the scalability and security aspects of using cloud-based platforms like Firebase for large-scale elections.

SCHEME ON SMART VOTING SYSTEM USING FACE DETECTION AND RECOGNITION ALGORITHMS

M.Kandan Mariyadoss(2022) designed the system to enhance the electoral process by integrating biometric technology. It leverages advanced face detection and recognition algorithms to ensure secure, efficient and tamper-proof voting. The system addresses issues such as voter impersonation and manual verification by automating identity validation. It employs high-precision image processing to authenticate registered voters in real time. By minimizing human intervention, it aims to streamline operations, reduce fraud and ensure transparency. The architecture includes a voter database, a user-friendly interface and robust security mechanisms. This innovation holds potential for revolutionizing democratic processes globally.

DRAWBACKS

1. Reliance on Technology
2. High Cost
3. Privacy Concerns
4. Accuracy Challenges

CHAPTER 3

SYSTEM REQUIREMENTS

3.1 HARDWARE REQUIREMENTS

COMPONENTS		SPECIFICATIONS
Processor	:	Multi-core (Intel Core i3 or equivalent)
Memory	:	4 GB RAM (8 GB recommended)
Storage	:	10 GB free (SSD preferred)
Camera	:	720p (1080p ideal)

3.2 SOFTWARE REQUIREMENTS

SOFTWARE		VERSION/SPECIFICATION
Operating System	:	Windows 10 or higher, or Linux (Ubuntu/Kali)
Python	:	Python 3.8 or higher
Libraries/Packages	:	OpenCV (for image processing), Dlib (for facial landmark detection), Face Recognition (for face embedding and comparison), Kivy (for GUI development), SQLite (for local database management)

Database Management	:	SQLite (for local database storage)
IDE/Editor	:	Visual Studio Code, PyCharm, or any preferred Python IDE

3.2.1 PYTHON

Python is a high-level, interpreted programming language known for its simplicity, readability and versatility. It was created by Guido van Rossum and released in 1991. Python emphasizes code readability, which allows developers to write clear and logical code for both small and large-scale projects. Its syntax is clean and easy to understand, making it an ideal choice for beginners as well as experienced programmers. Python supports multiple programming paradigms, including procedural, object-oriented and functional programming, offering flexibility for different types of software development.

Due to its extensive standard library and a large number of third-party packages, Python is widely used in various fields, including web development, data science, artificial intelligence (AI), machine learning, automation and more. The language's vast ecosystem of libraries and frameworks makes it the go-to solution for developing a wide array of applications, from simple scripts to complex systems.

3.2.2 OPENCV

OpenCV is an open-source, cross-platform computer vision library designed for real-time image processing. It is widely used for tasks such as facial recognition, object detection, image filtering and video stream processing. OpenCV offers a variety of tools and functions to manipulate and

analyze visual data, making it an essential library for building facial recognition systems, particularly in applications like automated voting systems.

In this system, OpenCV is primarily used for face detection, where it identifies and locates faces within an image or video stream. This is achieved using built-in algorithms such as Haar Cascade Classifiers or deep learning-based methods. The library also provides methods for image preprocessing, which involves tasks like converting images to grayscale for easier analysis, resizing images to a standard size and adjusting lighting conditions to enhance recognition accuracy.

Additionally, OpenCV handles video stream operations, allowing the system to capture real-time facial data from a camera, which is crucial for dynamic and live voter verification. Its ability to process and manipulate visual data in real-time ensures smooth operation, making it a powerful tool for applications that require fast and accurate face recognition.

Overall, OpenCV's wide range of features and support for real-time processing makes it an indispensable part of any facial recognition-based project, ensuring that tasks like face detection, image processing and video handling are efficient and reliable.

CHAPTER 4

SMART VOTING SYSTEM USING FACE DETECTION AND RECOGNITION ALGORITHMS

The Smart Voting System is an innovative approach to modernize traditional voting methods using advanced biometric technologies. It employs face detection and recognition algorithms to authenticate voters, ensuring security, efficiency and transparency. By leveraging machine learning and computer vision, the system addresses the challenges of manual voting systems, such as identity fraud and inefficiency.

4.1 TECHNOLOGIES USED

The system integrates cutting-edge technologies, including advanced face recognition algorithms like Haar Cascade and deep learning models. These technologies enable accurate real-time detection and recognition of voter faces. The implementation relies on OpenCV for image processing and TensorFlow/Keras for model development and deployment. MySQL or equivalent databases are used to securely store and manage voter information, ensuring scalability and reliability. The use of Python, with its robust ecosystem of libraries like NumPy, Flask and Pandas, supports the system's core functionality and facilitates seamless integration of machine learning models and database operations. Furthermore, encryption techniques are implemented to secure sensitive data during storage and transmission.

4.2 PROGRAMMING LANGUAGES

The project is primarily developed using Python, a versatile language well-suited for machine learning, database management and backend development. SQL is used for efficient database management, enabling the storage and retrieval of face recognition data and voting records. Frontend components, if web-based, are developed using HTML, CSS and JavaScript, ensuring a responsive and user-friendly interface.

4.3 PROCESS FLOW

The Smart Voting System operates through several streamlined phases:

1. **Registration:** Voters register their details, including a face image, which is processed and securely stored in the database.
2. **Authentication:** During the voting process, the system captures a real-time image of the voter. This image is analyzed using face recognition algorithms and compared with the database to authenticate the voter.
3. **Voting:** Once authenticated, the voter can cast their vote. The vote is securely recorded in the database while ensuring anonymity.
4. **Verification:** Real-time checks prevent duplicate voting, and audit logs maintain the integrity of the voting process.

The system's control flow is designed to handle multiple operations simultaneously, ensuring a seamless experience for voters. It begins with input acquisition (face capture), processes the data through recognition algorithms, and allows or denies access based on the results.

4.4 DRAWBACKS

HIGH INITIAL COST

- Implementing the system involves significant expenses for acquiring advanced hardware such as high-resolution cameras and GPUs for real-time processing. Additionally, developing custom software solutions for face recognition and database management adds to the overall cost.

DEPENDENCY ON TECHNOLOGY

- The system relies heavily on technology infrastructure, including internet connectivity, servers and facial recognition algorithms. Any malfunction, such as hardware failures or software bugs, could lead to system outages or delays during the voting process.

ENVIRONMENTAL SENSITIVITY

- Face recognition algorithms can be affected by environmental factors such as poor lighting, background noise or occlusions like glasses, masks or headgear. These factors might lead to errors in voter authentication or delays.

PRIVACY CONCERNS

- Storing and processing sensitive biometric data, like facial images, raises privacy concerns. Without proper encryption and compliance with data protection regulations, there is a risk of data breaches, misuse or unauthorized access to voter information.

LEGAL AND ETHICAL CHALLENGES

- Deploying biometric systems in voting processes may face legal and ethical scrutiny. Governments and organizations must address concerns related to data ownership, consent and

compliance with laws like GDPR (General Data Protection Regulation) to ensure public trust in the system.

LIMITED INCLUSIVITY

- Certain populations, such as older individuals, persons with disabilities, or those unfamiliar with technology, might find it challenging to use the system. Additionally, face recognition algorithms may have biases or inaccuracies based on age, gender or ethnicity.

CHAPTER 5

FACE RECOGNITION VOTING SYSTEM

The Face Recognition Voting System is designed to overcome the limitations of existing voting methods, including impersonation, multiple voting attempts, and data security challenges. This system leverages facial recognition technology, real-time databases to create a secure, scalable and efficient platform for elections.

5.1 VOTER REGISTRATION

Voters must first register by providing their personal identity details along with their facial data. The system captures the facial features of the voter using a real-time camera feed and stores this data in a SQLite database. During registration, a Haar Cascade Classifier is used for face detection to identify the face in real-time. The system uses DeepFace, a deep learning model, to analyze the facial features, ensuring that the facial data is unique and accurate. The combination of these technologies guarantees a reliable registration process, preventing unauthorized individuals from registering.

5.2 AUTHENTICATION

During the voting process, the system uses Haar Cascade to detect the voter's face in real-time and then uses DeepFace to compare the captured face with the registered facial data. If the system detects a match, the voter is authenticated. This multi-layered authentication prevents impersonation and

ensures that only legitimate voters are allowed to cast their votes. The DeepFace algorithm, based on deep learning, performs an extensive analysis of facial features to ensure high accuracy even under different lighting conditions and orientations.The Haar Cascade algorithm is fast and efficient, allowing real-time face detection even on low-resolution images, making it ideal for use in this system.

5.3 VOTE CASTING

After the voter has been authenticated, they can proceed to cast their vote. The system provides an intuitive interface created using Kivy, a Python framework for building multi-platform applications. The user interface (UI) displays the voting options, and the voter can select their candidate of choice.

5.4 VOTE STORAGE

Votes are stored in a SQLite database, which is updated in real-time to ensure no duplication or tampering of data. The system also generates an anonymized vote receipt that is stored for transparency and auditing purposes. The use of SQLite guarantees that votes are recorded securely and synchronized with the server for immediate updates. Each voter is assigned a unique ID based on their facial data to prevent multiple voting attempts.

5.5 LIVENESS DETECTION

To prevent fraud attempts, the system incorporates liveness detection, which ensures that the person trying to authenticate is present in front of the camera and not using a photo or video.

5.6 ALGORITHMS USED

5.6.1 HAAR CASCADE ALGORITHM

The Haar Cascade algorithm is a classical machine learning-based method for face detection. The algorithm works by detecting simple rectangular features that represent differences in intensity in the image, such as the contrast between the eyes, nose, and other facial features. The Haar-like features are computed for different regions of the image and passed through a cascade of classifiers to quickly and efficiently detect faces. This is a key component of the voter authentication process, as it enables the system to detect faces in real-time, even in low-resolution images.

Working Flow:

1. The system captures the live video feed.
2. The Haar Cascade classifier scans the image for potential face-like structures.
3. The classifier marks the face region and passes it for further analysis.

5.6.2 DEEPFACE ALGORITHM

The DeepFace algorithm is a deep learning-based face recognition framework developed by Facebook. In this system, DeepFace is used for verifying the identity of the voter after a face is detected by the Haar Cascade algorithm. DeepFace generates a numerical embedding for a face image by passing it through a deep convolutional neural network (CNN) trained on millions of face images. This embedding is a compact representation of the face's unique features, allowing for comparison between faces. When a captured face is compared to a database of registered faces, DeepFace calculates the Euclidean distance between their embeddings. If the distance

is below a predefined threshold, the system verifies the identity of the person, granting access to vote. DeepFace's deep learning model is robust and highly accurate, making it highly reliable for face verification, reducing the chances of false positives and ensuring a high level of security for the voting process.

Working Flow:

1. The system compares the real-time face with the stored face using the DeepFace model.
2. The model processes various facial features, such as the distance between the eyes, nose, and mouth.
3. The system calculates a similarity score and authenticates the voter if the score exceeds a threshold.

5.7 CROSS PLATFORM ACCESSBILITY

The system is designed using the Kivy framework, which allows it to be deployed on multiple platforms including Windows, macOS, Linux, and Android. This cross-platform capability ensures that the system can be accessed by a wide range of devices, making it easy for voters to participate in elections from their preferred devices.

5.8 PROCESS FLOW

5.8.1. INITIALIZATION

Database Initialization (init_db):

The SQLite database is created with two tables:

1. **voters:** Stores registered voters' names and their respective face image paths.

2. **votes:** Stores the votes cast by authenticated voters.

Predefined voter data is added using the add_voter function.

5.8.2 APPLICATION STARTUP

Kivy Application Setup:

The Kivy app is built with a user-friendly interface:

1. Title bar displaying the name of the system.
2. Scrollable view listing candidates with their profiles, symbols, and descriptions.
3. A camera feed panel for real-time face capture.
4. Buttons for face capture and voting, initially disabled until the user is authenticated**.**

Camera Initialization:

1. OpenCV (cv2) captures live video feed from the webcam.
2. The feed is displayed on the app and updated at 30 frames per second using Kivy's Clock.schedule_interval.

5.8.3 CANDIDATE PROFILES

Dynamic Candidate List:

1. Candidates' profiles are dynamically added using add_candidate_profile.
2. Each profile includes:
   - Name and symbol (e.g., "Book" or "Car").
   - An image representing the candidate's symbol.
   - A brief description of their qualifications.

5.8.4 FACE AUTHENTICATION

Capture Face:

When the user clicks the "Capture Face" button

- The current frame from the webcam is saved as captured_face.jpg**.**

Face Recognition (recognize_face):

1. The system uses DeepFace to compare the captured face with the pre-stored voter images.
2. If a match is found:
    - The authenticated voter's name is returned.
    - Voting buttons are enabled, allowing the user to cast their vote.
3. If no match is found:
    - The system notifies the user with a message indicating unrecognized face data.

5.8.5 VOTING PROCESS

Vote Casting (cast_vote):

1. After authentication, the user selects a candidate by clicking the corresponding button.
2. The system checks the votes table to ensure the user hasn't already voted.

If not:

1. The vote is recorded in the database.
2. A success message is displayed**.**

If already voted:

1. A message informs duplicate voting.

5.8.6 VOTE COUNTING AND RESULTS DISPLAY

Vote Tally (count_votes):

1. The system aggregates votes from the database.
2. Counts the total votes for each candidate.
3. Calculates the percentage of votes received by each candidate.

Result Compilation:

1. The system identifies the winning candidate based on the maximum vote count.
2. Results, including vote counts, percentages, and the winner, are formatted as a string.

5.8.7 APPLICATION TERMINATION

Resource Cleanup:

1. The OpenCV camera feed is released to free up system resources.

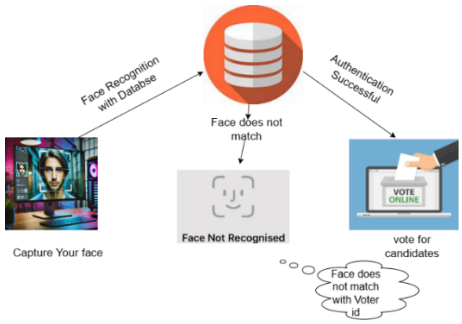

FIGURE 1.1: SYSTEM ARCHITECTURE DESIGN

The figure 1.1 shows the steps involved in the processing of the system.

5.9 MITIGATING THE DRAWBACKS OF SMART VOTING SYSTEM USING FACE DETECTION AND RECOGNITION ALGORITHMS

5.9.1 OVERCOMING HIGH INITIAL COST

1. **Use of Open-Source Libraries:** The system leverages open-source libraries like DeepFace (for face recognition) and OpenCV (for camera functionality), which significantly reduces software development costs. These libraries are free to use and well-documented, which reduces the need for costly commercial software.
2. **Scalable Hardware:** The system is designed to work with basic hardware like webcams, which are relatively inexpensive. Since the system works with a standard webcam for face capture, it eliminates the need for specialized biometric equipment. This approach makes the system scalable and affordable for deployment in different environments.
3. **Software Flexibility:** The database management (SQLite) is simple and lightweight, meaning that the system can be run on smaller, less expensive servers for small-scale elections. For larger elections, the system can be migrated to more robust servers and databases, thus supporting growth without significant initial investments.

5.9.2 OVERCOMING TECHNOLOGY DEPENDENCE

1. **Error Handling and Recovery**: The code includes mechanisms for error handling in the recognize_face function, where it catches any errors that may occur during face recognition. This ensures that the system can notify users if something goes wrong, such as

a failure in recognizing a face, without causing the entire system to crash.

2. **Camera Failures**: If the camera feed fails or is interrupted, the system would ideally provide user feedback, allowing the user to try again. While the code doesn't have a detailed backup plan, this can be further enhanced by implementing fallback authentication methods (like a PIN or a verification code via SMS) in case the camera feed is interrupted or the recognition fails.
3. **Resilience to Network or Server Failures**: The system is designed to operate locally on a single machine with a simple SQLite database, so it's not reliant on an external server or network for its core functionality. This reduces dependency on external infrastructure and minimizes the risk of disruption from server crashes or network issues.

5.9.3 OVERCOMING ACCURACY CHALLENGES

1. **DeepFace Algorithm**: The DeepFace library uses a deep learning-based face recognition algorithm that is relatively resilient to small variations in lighting and camera angles. While extreme variations might still affect performance, the algorithm is robust and trained on diverse datasets to handle many common challenges in real-world scenarios.
2. **Handling Occlusions**: While the code doesn't directly address handling extreme occlusions (like masks or glasses), face recognition algorithms like DeepFace typically perform well with minor occlusions. To further improve accuracy in such scenarios, additional techniques like liveness detection (to ensure the face is from a live person) can be integrated. This would help mitigate issues like using photos or videos of a person.

3. **Multiple Attempts and Feedback**: The system allows multiple attempts at face recognition. If the system fails to authenticate a user, it provides immediate feedback ("Face not recognized. Please try again."), allowing users to retry without disrupting the voting process. This is crucial for ensuring that minor errors or misalignments don't prevent a user from voting.

5.9.4 OVERCOMING PRIVACY CONCERNS

1. **Encrypted Data Storage:** While the code does store voter data (including face images) in a local SQLite database, it is possible to enhance this by encrypting the database or applying encryption to sensitive fields such as voter images or names. Implementing such encryption would protect sensitive voter information from unauthorized access.
2. **Minimal Data Exposure:** The system only stores paths to the facial images rather than the biometric data itself. This reduces the risk of data being compromised or misused, as the stored information is less sensitive. However, additional measures such as data anonymization (e.g., storing encrypted identifiers rather than names) can further enhance privacy.
3. **Face Image Security:** The captured face image (e.g., captured_face.jpg) can be stored in a more secure location (e.g., in an encrypted directory) to further reduce privacy risks. Additionally, facial recognition libraries like DeepFace use the image for comparison purposes rather than storing detailed biometric data, reducing the amount of personally identifiable information (PII) at risk.
4. **Audit Trails:** The system's database tracks voter actions (e.g., voting events) and ensures that only one vote is cast per voter. This

provides an audit trail, which can be important for verifying the integrity of the voting process. However, further improvements could include additional logging of who accessed the system and when, enhancing transparency and traceability.

CHAPTER 6

IMPLEMENTATION AND RESULT

6.1 SOURCE CODE

6.1.1 IMPORTS

```
import sqlite3
import cv2
from deepface import DeepFace
from kivy.app import App
from kivy.uix.image import Image
from kivy.uix.boxlayout import BoxLayout
from kivy.uix.button import Button
from kivy.uix.label import Label
from kivy.uix.scrollview import ScrollView
from kivy.uix.gridlayout import GridLayout
from kivy.uix.image import Image as KivyImage
from kivy.graphics import Color, Rectangle
from kivy.clock import Clock
from kivy.graphics.texture import Texture
```

Explanation:

1. **sqlite3**: This is a built-in Python library for interacting with SQLite databases. It allows you to create, read, update, and delete data from a local database.
2. **cv2**: OpenCV (Open Source Computer Vision Library) is used for image and video processing. Here it is used for capturing video from the camera and processing it for face detection.
3. **DeepFace**: A deep learning library for facial recognition. It helps in verifying whether a face from the camera matches a previously stored face.
4. **Kivy**: A framework for building graphical user interfaces (GUIs) in Python. Various widgets (such as BoxLayout, Label, Button) are imported to build the user interface of the app.

6.1.2 INITIALIZING THE DATABASE

```
def init_db():
    conn = sqlite3.connect('voting_system.db')
    cursor = conn.cursor()
    cursor.execute("""
    CREATE TABLE IF NOT EXISTS voters (
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        name TEXT NOT NULL,
        face_path TEXT NOT NULL)""")
    cursor.execute("""
    CREATE TABLE IF NOT EXISTS votes (
```

```
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        user_name TEXT NOT NULL,
        candidate_name TEXT NOT NULL)""")
    conn.commit()
    conn.close()
```

Explanation:

- sqlite3.connect('voting_system.db'): Connects to a SQLite database file named voting_system.db. If it doesn't exist, a new file is created.
- cursor.execute(): Executes SQL commands to create the tables.
- conn.commit(): Saves the changes to the database.
- conn.close(): Closes the database connection.

6.1.3 ADDING A VOTER

```
def add_voter(name, face_path):
    conn = sqlite3.connect('voting_system.db')
    cursor = conn.cursor()
    cursor.execute("INSERT INTO voters (name, face_path) VALUES (?, ?)"
(name, face_path))
    conn.commit()
    conn.close()
```

Explanation:

- INSERT INTO voters (name, face_path): Inserts a new record into the voters table, storing the name and the path to the voter's face image.

- conn.commit(): Commits the changes to the database.
- conn.close(): Closes the database connection.

6.1.4 RECOGNIZING A FACE

```
def recognize_face(captured_face_path):
    conn = sqlite3.connect('voting_system.db')
    cursor = conn.cursor()
    cursor.execute("SELECT name, face_path FROM voters")
    voters = cursor.fetchall()
    for voter in voters:
        name, face_path = voter
        try:
            result = DeepFace.verify(captured_face_path, face_path)
            if result["verified"]:
                return name  # Return recognized user's name
        except Exception as e:
            print(f"Error during face recognition: {e}")
    return None  # No match found
```

Explanation:

- DeepFace.verify(): This function from the DeepFace library compares the captured face with each stored face to determine if they match. If they do, it returns True in the "verified" key of the result.

- The loop iterates over all stored voters in the database, and if a match is found, the user's name is returned.
- If no match is found, the function returns None.

6.1.5 CASTING A VOTE

```
def cast_vote(user_name, candidate_name):
    conn = sqlite3.connect('voting_system.db')
    cursor = conn.cursor()
    cursor.execute("SELECT * FROM votes WHERE user_name = ?", (user_name,))
    existing_vote = cursor.fetchone()
    if existing_vote:
        return False  # Duplicate vote detected
    cursor.execute("INSERT INTO votes (user_name, candidate_name) VALUES (?, ?)", (user_name, candidate_name))
    conn.commit()
    conn.close()
    print(f"Vote for {candidate_name} by {user_name} registered successfully.")
    return True
```

Explanation:

- cursor.execute("SELECT * FROM votes WHERE user_name = ?", (user_name,)): Queries the votes table to check if the user has already voted.

- If a record is found (existing_vote), the function returns False to indicate that a duplicate vote is detected.
- If no record is found, the vote is inserted into the database, and True is returned to indicate that the vote was successfully cast.

6.1.6 KIVY APPLICATION (USER INTERFACE)

```
class VotingApp(App):
    def build(self):
        self.layout = BoxLayout(orientation='vertical', padding=10, spacing=10)
```

Explanation:

- BoxLayout: This layout arranges widgets vertically (orientation='vertical'). Padding and spacing are applied for layout aesthetics.
- self.layout: The main container for all the widgets in the application.

6.1.7 TITLE AND BACKGROUND

```
with self.layout.canvas.before:
    Color(1, 1, 1, 1)  # White background
    self.rect = Rectangle(size=self.layout.size, pos=self.layout.pos)
self.layout.bind(size=self.update_rect, pos=self.update_rect)
title = Label(text="Face Recognition Voting System", font_size='24sp', bold=True, color=(0, 0, 0, 1), size_hint=(1, 0.1))
self.layout.add_widget(title)
```

Explanation:

- The background is set to white (Color(1, 1, 1, 1)).
- Rectangle: A graphical rectangle representing the background.
- Label: Adds a title to the top of the screen.
- self.layout.add_widget(): Adds the title label to the layout.

6.1.8 CANDIDATE PROFILES AND VOTING UI

self.profile_scroll = ScrollView(size_hint=(1, 0.4), do_scroll_x=False, do_scroll_y=True)

self.profile_layout = GridLayout(cols=1, spacing=10, size_hint_y=None, padding=10)

self.profile_layout.bind(minimum_height=self.profile_layout.setter('height')

Explanation:

- ScrollView: A scrollable container for the candidate profiles.
- GridLayout: A layout where widgets are arranged in a grid. The profiles will be added as rows in a single-column grid.

6.1.9 VOTING ACTION PANEL

self.action_panel = BoxLayout(orientation='vertical', size_hint=(1, 0.5), padding=10, spacing=10)

Explanation:

- The action_panel will hold the widgets related to voting: a camera feed, a button for capturing faces, and buttons to vote for candidates.

6.1.10 CAMERA FEED UPDATE

self.capture = cv2.VideoCapture(0)

Clock.schedule_interval(self.update, 1.0 / 30.0)

Explanation:

- cv2.VideoCapture(0): Captures video from the default camera.
- Clock.schedule_interval: Schedules the update function to run at the given interval (30 FPS), updating the camera feed.

6.1.11 CAPTURING AND RECOGNIZING A FACE

```
def capture_face(self, instance):
    ret, frame = self.capture.read()
    if ret:
        cv2.imwrite("captured_face.jpg", frame)
        recognized_name = recognize_face("captured_face.jpg")
        if recognized_name:
            self.vote_for_candidate(recognized_name)
```

Explanation:

- self.capture.read(): Captures a frame from the camera.
- cv2.imwrite(): Saves the captured image to disk.
- recognize_face(): Tries to identify the person from the captured face.

6.1.12 VOTING FOR A CANDIDATE

```
def vote_for_candidate(self, user_name):
    candidate_name = "Candidate A"  # Placeholder candidate
    success = cast_vote(user_name, candidate_name)
```

```
    if success:
        popup = Popup(title='Vote Successful', content=Label(text="Your vote has been registered."), size_hint=(None, None), size=(400, 200))
        popup.open()
```

Explanation:

- Placeholder candidate name is used for simplicity.
- cast_vote(user_name, candidate_name): Calls the previously defined function to cast a vote for the user.
- If successful, a popup is shown to inform the user.

6.1.13 CLOSING THE APPLICATION

```
def on_stop(self):
    self.capture.release()
```

Explanation:

- self.capture.release(): Releases the camera resource to free it up for other applications when the app stops.

6.1.14 RUNNING THE APPLICATION

```
if __name__ == '__main__':
    VotingApp().run()
```

Explanation:

- This line ensures that the app runs when the script is executed directly.

6.2 RESULT

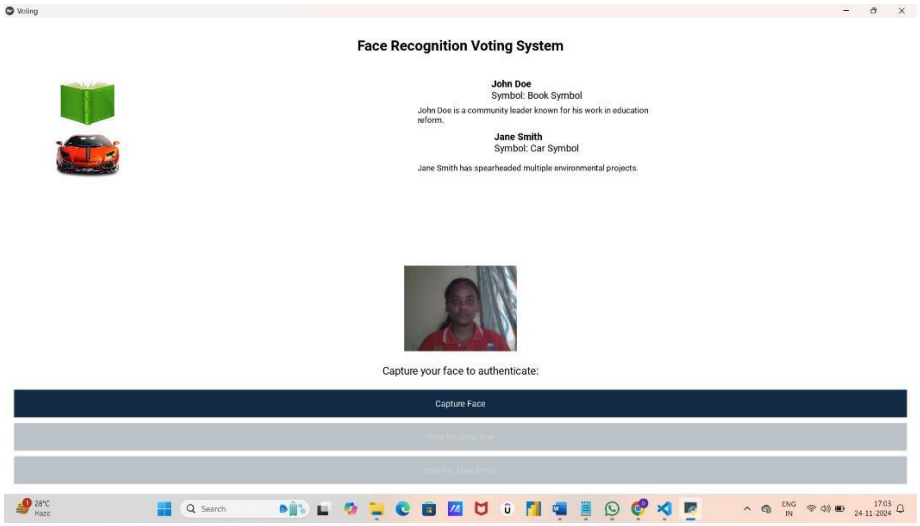

FIGURE 1.2 : READY TO CAPTURE IMAGE

The above figure 1.2 shows that the face recognition system is ready for capturing the voters face. Shows the candidate profiles with their respective symbols.

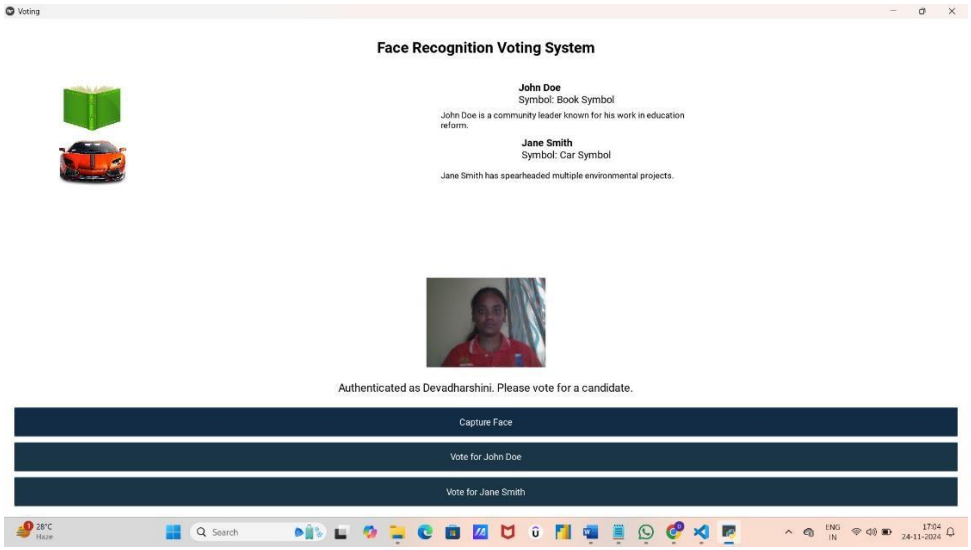

FIGURE 1.3 : AUTHENTICATION SUCCESSFUL

The above figure1.3 is the output of face authentication process, it shows that the voters face has been matched with the database.

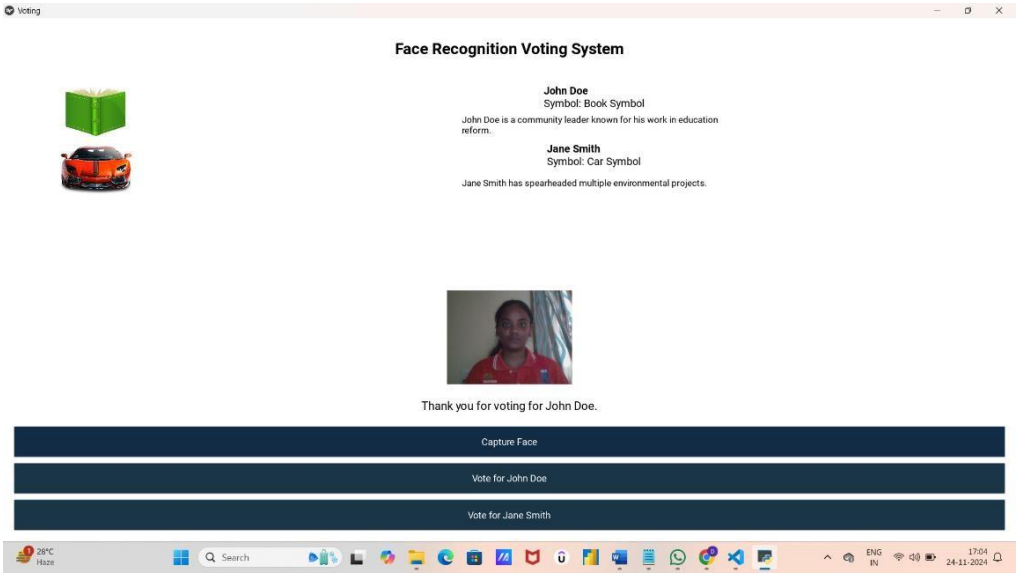

FIGURE 1.4 : VOTING SUCCESSFUL

The above figure1.4 shows that the vote has been successfully casted and the system greets the user for voting.

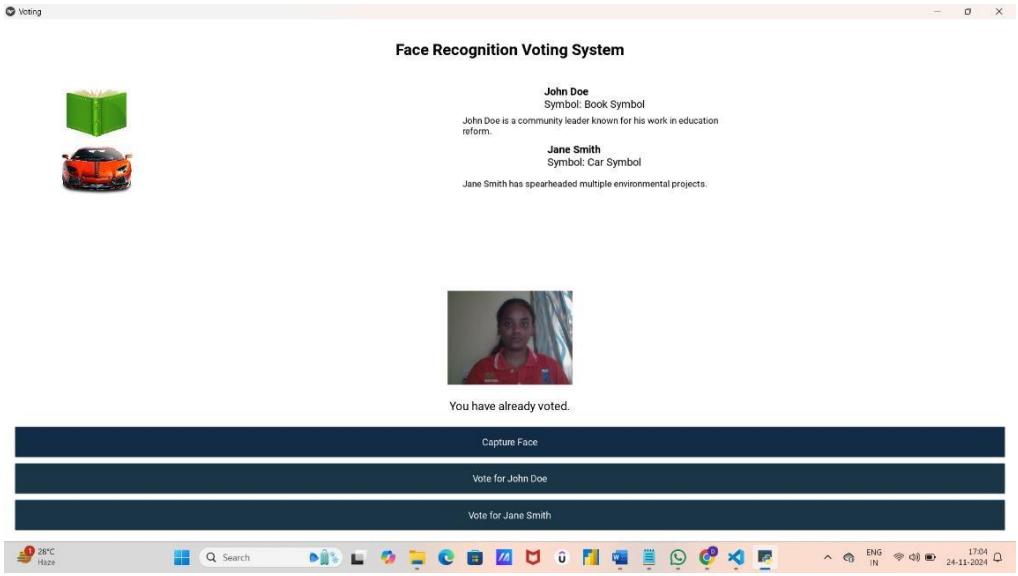

FIGURE 1.5 : DUPLICATE VOTE ATTEMPTED

The above figure1.4 shows “you have already voted” if the user tries to vote more than one time.

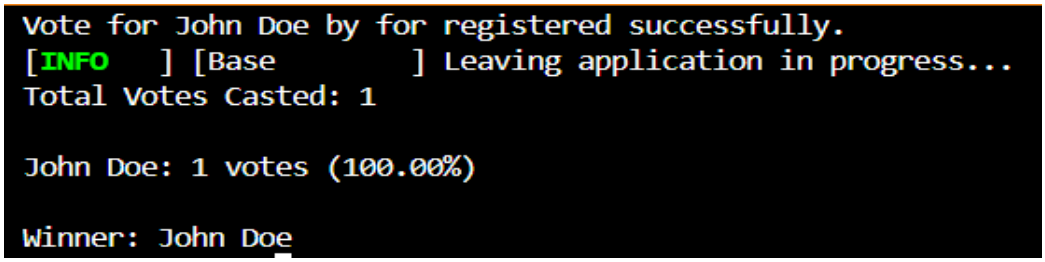

FIGURE 1.6 : ELECTION RESULT

The above figure1.6 shows the total votes casted and number of votes casted for each candidate and their percentage. It also shows the election winner.

CHAPTER 7

CONCLUSION AND FUTURE ENHANCEMENT

7.1 CONCLUSION

The Smart Voting System leveraging face detection and recognition provides a modern, efficient and secure way to conduct elections. By using DeepFace for biometric authentication, SQLite for database management and Kivy for the user interface, the system ensures a seamless voting experience. It addresses key challenges like identity fraud and inefficiency present in traditional voting systems while maintaining transparency and accountability through real-time vote tracking and result generation. The system's simplicity and scalability make it a promising solution for both small-scale and large-scale elections.

7.2 FUTURE ENHANCEMENT

For future improvements, the system could be enhanced by incorporating more advanced biometric authentication methods, such as iris scanning or voice recognition, to improve accuracy and security. Additionally, scaling the system for large-scale elections would require transitioning from SQLite to more robust database systems like MySQL or PostgreSQL. Improving user inclusivity by adding support for accessibility features, multilingual interfaces, and voice commands would make the system more accessible to diverse voter

populations. Finally, privacy and data protection enhancements, such as stronger encryption and compliance with global data protection laws, will help build trust and ensure the system’s adoption in various regions.

REFERENCES

1. Mayank Agarwal, Nikunj Jain, Mr. Manish Kumar, and Himanshu Agrawal (2010), "Face recognition using eigenfaces and artificial neural network," International Journal of Computer Theory and Engineering, vol. 2, no. 4, pp. 624-629.
2. Maitri Chokshi, Nikhil Shah, and Trisha Patel (2013), "Smart device-based election voting system endorsed through face recognition," International Journal of Advanced Research in Computer Science and Software Engineering, vol.3, Issue. 11, pp. 529-531.
3. A. R. Syafeeza, M. Khalil-Hani, S. Liew, and R. Bakhteri (2014), "Convolutional neural network for face recognition with pose and illumination variation," International Journal of Engineering and Technology, vol.6, no.1, pp.44-57.
4. Namala Naresh Kumar, and E. Venkat Reddy (2015), "Biometric system based electronic voting machine with security algorithm and password protection on ARM microcontroller and GSM," International Journal & Magazine of Engineering, Technology, Management and Research, vol.2, Issue. 1, pp. 283-285.
5. Mandavkar Ashwini Ashok, and Rohini Agawane (2016), "Mobile-base faced recognition using OTP verification for voting system," Proceeding in International Advance Computing Conference (IACC) Conference, IEEE, pp.644-649.
6. S. V. Tathe, A. S. Narote, and S. P. Narote (2016), "Face detection and recognition in videos," Proceeding in Annual India Conference (INDICON), IEEE, pp.1-6.

7. M. Santhosh, R. Keerthana, L. Suganya, S. Krishnakumar, and S.Kavitha (2016), "Electronic voting machine using internet," International Journal of Communication and Computer Technologies, vol.4, Issue. 2, pp.72-75.
8. Swati Gawhale, Vishal Mulik, Pooja Patil, and Nilisha Rau, (2017) "IoT based e-voting system," International Journal for Research in Applied Science & Engineering Technology (IJRASET), vol.5, Issue.5, pp.1064- 1067.
9. Amna Qureshi, David Megias, and Helena Rifa-Pous (2019), "SeVEP: verifiable, secure and privacy-preserving remote polling with untrusted computing devices," IEEE Access, vol.7, pp.19266-19290.
10. Ishani Mondal, and Sombuddha Chatterjee (2019), "Secure and Hassle Free EVM through deep learning face recognition," Proceeding in International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon), IEEE, pp. 109-113.
11. A. K. Roy, R. Singh, (2019) 'Efficient Multi-Factor Authentication for E Voting', Elsevier Science Direct.
12. Ali Kaan Koç, Emre Yavuz, Umut Can Çabuk, Gökhan Dalkılç (2020),'Towards Secure E-Voting Using Ethereum Blockchain', ResearchGate.
13. Xuechao Yang, Xun Yi, Surya Nepal, Andrei Kelarev, Fengling Han (2020),'A Secure Verifiable Ranked Choice Online Voting System', peer- reviewed journal.
14. J. Patel, N. Desai, R. Mishra (2020).'AI-Powered Face Recognition for Secure E-Voting', International Journal of AI Research.
15. G. Uma Maheswari, K. Kalaivani, K. B. Somasundaram (2021), 'A Survey on Biometric-based Secure Electronic Voting System', Springer Link.

16. Rohit Sharma, Priya Sharma, Tarun K. Sharma (2021),'Blockchain-Based E-Voting System Using Facial Biometrics', Wiley OnlineLibrary.
17. P. Gupta, S. Tiwari (2021), 'Design of Real-Time Voting Systems Using Firebase', IJCSIT.
18. S. Ahmed, M. Rafique, A. Aslam (2022),'Design and Development of Facial Recognition Voting System', IEEE Xplore.