N. S. S. POLYTECHNIC COLLEGE

Mannam Nagar, Pandalam

(Under the Department of Technical Education Govt.of kerala)

A

PROJECT ON

"COLLEGE E-VOTING SYSTEM"

DONE BY

1	AMALJITH S.	2101131534
2	AMARNATH AJAY.	2101131535
3	ATHUL M.	2101131546
4	SHYAM MOHAN.	2101131565
5	VIMAL VINOD V.	2101133069

DEPARTMENT OF COMPUTER ENGINEERING

2023-2024

N. S. S. POLYTECHNIC COLLEGE

PANDALAM

DEPARTMENT OF COMPUTER ENGINEERING

CERTIFICATE

This is to certified that this report of the project work on "College E-Voting System" done by Amaljith S. Reg.no 2101131534, Amarnath Ajay Reg.no 2101131535, Athul M Reg.no 2101131546, Shyam Mohan Reg.no 2101131565, Vimal Vinod V. Reg.no 2101133069 of the final year Computer Engineering in partial fulfillment of the requirement for the award of diploma in Computer Engineering under the Directorate of Technical Education, Kerala State during the academic year 2023-2024.

Smt. Parvathy G S	Smt. Sandhya S.
(Head of Section)	(Project Guide)
Internal Examiner	External Examiner

ACKNOWLEDGEMENT

First I thank God for his immense grace at every stage of the Project.

I extend my heartfelt acknowledgment to our esteemed Principal, Dr. Preetha R, whose vision and commitment to academic excellence have provided the foundation for this endeavor. Your constant encouragement and belief in innovative ideas have been a driving force behind the successful completion of this report.

I am also immensely thankful to our Head of Department (HOD), Smt. Parvathy G. S. , for her guidance and insights that have enriched my understanding of the subject matter. Your valuable suggestions and willingness to share your expertise have played a significant role in shaping the content of this report.

I would like to extend my appreciation to our dedicated and supportive Class Tutors, Smt. Arya J Nair and Smt. Sreeja Kumari S, for their continuous assistance and mentorship. Your willingness to address my queries and provide constructive feedback has been instrumental in refining the quality of this report

Lastly, I would like to express my gratitude to all the faculty members and fellow students who have contributed to my learning experience and helped me gain insights into the field of College E- Voting System.

Yours Sincerely

Amaljith S.
Amarnath Ajay.
Athul M.
Shyam Mohan.
Vimal Vinod V.

ABSTRACT

The College E-Voting System (CEVS) is a transformative solution for student elections, offering a seamless and secure digital platform for casting votes. With its user-friendly interface accessible via web or mobile devices, CEVS ensures convenience and accessibility for all students, regardless of their location or schedule constraints. This accessibility enhances voter turnout and engagement, fostering a more representative and inclusive electoral process within educational institutions.

CEVS prioritizes the security and integrity of the electoral process through stringent authentication measures and advanced encryption techniques. By verifying the identity of voters and safeguarding the confidentiality of their ballots, CEVS mitigates the risk of fraud and manipulation, instilling trust and confidence in the outcome of elections. Administrators can customize ballots according to specific election requirements and monitor voting progress in real-time, enabling swift action and ensuring compliance with established protocols.

Furthermore, CEVS promotes sustainability by eliminating the need for paper ballots and reducing environmental impact. By embracing digital innovation, educational institutions can not only modernize their election procedures but also contribute to conservation efforts. CEVS represents a significant advancement in the democratization of student governance, empowering students to participate actively in shaping the future of their institutions while promoting efficiency, transparency, and environmental responsibility.

CONTENTS

1.INTRODUCTION

In the dynamic realm of digital technology, the notion of democratizing processes through online platforms has garnered substantial attention. Nowhere is this more evident than in educational institutions such as colleges, where student governance profoundly influences campus dynamics. Yet, the conventional methods of conducting elections within these settings often prove cumbersome, burdened by logistical hurdles and low participation rates.

Recognizing these challenges, there arises a pressing need for a more efficient and accessible voting system. This is where the proposed project steps in: to develop an Online Voting System tailored explicitly for college elections. By leveraging the capabilities of technology, this system seeks to revolutionize the voting process, fostering transparency and inclusivity while stimulating greater engagement among students.

The envisioned Online Voting System represents a significant departure from traditional approaches. It promises to simplify the voting process, rendering it more intuitive and user-friendly. Through a user-friendly interface, students can seamlessly cast their votes from any location with internet access, eliminating the constraints imposed by physical polling stations.

One of the paramount objectives of this system is to enhance transparency in the electoral process. By digitizing the entire voting procedure, it minimizes the scope for errors and malpractices, ensuring the integrity of the elections. Moreover, the system incorporates robust security measures to safeguard the confidentiality and authenticity of each vote cast.

Central to the project's vision is the aspiration to foster broader participation among students. By offering a convenient and accessible voting mechanism, the Online Voting System aims to empower every student to exercise their democratic right effectively. Furthermore, the system facilitates real-time tracking of voter turnout, enabling administrators to implement targeted strategies to boost engagement.

In essence, the development of an Online Voting System for college elections represents a paradigm shift in student governance. By harnessing the potential of digital technology, the project seeks to redefine the democratic process within educational institutions, promoting transparency, efficiency, and inclusivity.

1.1 SYNPOSIS

The proposed project seeks to develop an Online Voting System tailored specifically for college elections, integrating advanced face detection technology to enhance the security and accessibility of the voting process. Recognizing the evolving landscape of digital democracy, the project aims to address the limitations of traditional paper-based voting methods and foster broader student engagement in campus governance.

The Online Voting System for College Elections will leverage modern web development frameworks to create an intuitive and user-friendly platform, facilitating seamless and secure voting procedures for various student elections. Central to the system's functionality is the integration of face detection technology, which will enable voters to authenticate their identities securely and prevent fraudulent activities such as multiple voting or impersonation.

The project will prioritize security and accessibility, employing robust encryption techniques to safeguard the integrity of the voting process and protect voter privacy. Additionally, the system will feature multi-factor authentication mechanisms to ensure that only eligible students can participate in the elections.

Furthermore, the Online Voting System will be designed with scalability in mind, allowing it to accommodate varying election sizes and voting requirements. Administrators will have access to comprehensive management tools, enabling them to configure elections, manage candidate nominations, and monitor voting activity in real-time.

To promote transparency and accountability, the system will generate detailed audit trails and provide voters with receipts confirming their participation in the elections. Moreover, it will support post-election analysis, allowing stakeholders to analyze voting patterns and trends to inform future decision-making processes.

1.2 ABOUT THE PROJECT

The project aims to revolutionize the electoral process within college campuses by developing an Online Voting System tailored specifically for college elections. Embracing modern web development frameworks and advanced face detection technology, the system will offer an intuitive and secure platform for students to participate in various student elections, including student council elections, club elections, and referendum votes. By leveraging digital technologies, the system seeks to overcome the limitations of traditional paper-based voting methods, enhancing accessibility, efficiency, transparency, and security. Through its innovative approach, the project aspires to empower students to actively engage in campus governance, fostering a culture of democratic participation and inclusivity.

In addition to its core functionalities, the Online Voting System will prioritize user experience, employing responsive design principles to ensure compatibility across various devices and screen sizes. This will enable students to conveniently access the voting platform from their smartphones, tablets, or computers, thereby increasing participation rates among tech-savvy student populations

Moreover, the system will feature personalized voting experiences, allowing students to easily navigate through the ballot and make informed decisions. Information about candidates, their platforms, and campaign materials will be readily accessible within the voting interface, empowering voters to make educated choices

To further enhance accessibility, the Online Voting System will accommodate diverse needs, including support for multiple languages and accessibility features for users with disabilities. By removing barriers to participation, the system will strive to ensure that all students have an equal opportunity to exercise their voting rights.

The project will also prioritize data security and privacy, implementing stringent measures to protect voter information and prevent unauthorized access or tampering. This includes encryption protocols, secure authentication mechanisms, and regular security audits to identify and address potential vulnerabilities.

In summary, the Online Voting System for College Elections represents a transformative step towards modernizing the electoral process within educational institutions, harnessing the power of technology to democratize student governance and promote active citizenship among college students.

2.DEVELOPMENT TOOLS

The development of the Online Voting System for College Elections will require the utilization of various development tools and technologies to ensure its successful implementation.

- Programming Languages:
  - Python: Python will serve as the primary programming language for backend development due to its versatility, ease of use, and robust libraries such as Django.
  - HTML/CSS/JavaScript: These web development languages will be used for frontend development to create the user interface and enhance interactivity.
- Web Framework:
  - Django: Django is a high-level Python web framework that will be utilized for rapid development of the voting system's backend. It provides built-in features for authentication, database management, and security, streamlining the development process.
- Database Management System:
  - SQLite/PostgreSQL/MySQL: These relational database management systems (RDBMS) will be considered for storing and managing the system's data securely. The choice of database will depend on factors such as scalability, performance, and deployment requirements.
- Face Detection Library:
  - OpenCV: OpenCV (Open Source Computer Vision Library) is a popular open-source library that provides extensive support for image and video analysis, including face detection. It will be utilized to implement the face detection feature for user authentication during the voting process
- Integrated Development Environment (IDE):
  - Visual Studio Code: These IDEs offer powerful tools for Python development, including syntax highlighting, code completion, and debugging capabilities. They will be utilized to write, test, and debug the system's codebase.

3.EXISTING SYSTEM

Currently, the electoral process in college campuses is predominantly conducted through manual methods, relying on paper-based ballots and in-person voting procedures. In this system, students are required to physically visit designated polling stations on election days to cast their votes. The process entails several steps:

- Student Enrollment: Students interested in voting must enroll in advance, typically at designated college administrative offices or election centers. They provide necessary identification documents and personal details to verify their eligibility.

- Manual Verification: College election officials manually verify the submitted documents to confirm the eligibility of enrolled students. This manual verification process is prone to errors and consumes significant time.

- Voting Day: On the scheduled election day, enrolled students visit their designated polling stations within the college premises. They present their identification, have their names checked against the voter list, and receive a paper ballot.

- Paper Ballots: Students mark their preferred candidates on the paper ballot provided, following which they fold the ballot and deposit it into a sealed ballot box.

- Manual Counting: Following the conclusion of the voting period, college election officials manually count the paper ballots to determine the election results. This manual counting process is time-consuming and susceptible to errors or disputes.

- Announcement of Results: Once the counting process is completed, the election results are publicly announced, typically through college notice boards or official announcements.

The existing electoral system within college campuses relies heavily on manual procedures, leading to inefficiencies and potential inaccuracies in the voting process. Moreover, the reliance on physical presence limits the accessibility and participation of students, particularly those with scheduling conflicts or remote learning arrangements. As such, there is a pressing need to modernize the college electoral process by transitioning towards a more efficient, transparent, and accessible online voting system.

3.1 PROBLEM IN EXISTING SYSTEM

The existing manual system for conducting elections on college campuses faces several challenges:

- Inefficiency: The manual processes of enrollment, verification, voting, and counting are time-consuming and labor-intensive. This inefficiency can lead to delays in election results and consume valuable resources.
- Error-Prone Verification: Manual verification of student documents for eligibility is prone to human errors, leading to potential inaccuracies in the voter list.
- Limited Accessibility: The requirement for physical presence on election day may deter participation, especially for students with scheduling conflicts or those engaged in remote learning arrangements.
- Potential Disputes: Manual counting of paper ballots can introduce discrepancies and raise concerns about the integrity of the election results. Disputes may arise due to errors in counting or challenges to the validity of ballots.
- Environmental Impact: Reliance on paper-based ballots contributes to environmental waste, as well as additional costs associated with printing and disposal.
- Security Risks: Paper-based systems are vulnerable to tampering or unauthorized access, posing security risks to the integrity of the electoral process.

Overall, the existing system lacks efficiency, transparency, and accessibility, highlighting the need for modernization through the implementation of an online voting system.

3.2 PROPOSED SYSTEM

The proposed Online Voting System aims to address the shortcomings of the existing system and modernize the election process. The key features of the proposed system include:

- Online Voter Registration: Voters can register online by providing their identification details and personal information. The system ensures data accuracy and reduces the need for physical visits to registration centers.

- Automated Verification: The system automates the verification process, ensuring accurate and efficient eligibility checks.

- Secure Online Voting: Eligible voters can cast their votes securely online using a user-friendly interface. The system ensures that each voter can vote only once and maintains the privacy of their votes.

- Real-time Results: The proposed system tallies votes in real-time, allowing for faster and more efficient result announcements.

- Enhanced Security: The system incorporates encryption, authentication mechanisms, and other security measures to prevent unauthorized access and tampering of data.

- Accessible and Convenient: The online platform makes voting accessible to a wider range of voters, including those who may have difficulty reaching polling stations.

- Party and Candidate Registration: Party representatives can register their parties and candidates online, with the Election Commission overseeing the approval process.

- Transparency and Efficiency: By digitizing the entire process, the proposed system enhances transparency, reduces errors, and minimizes the scope for fraudulent activities.

3.3ADVANTAGES

- Efficiency: Online voting streamlines the entire process, reducing time and resources required for traditional methods.
- Accuracy: Automation reduces the chances of human errors and improves data accuracy.
- Transparency: The system provides transparency at every stage, ensuring the integrity of the election process.
- Reduced Fraud: Online systems are more secure and less prone to fraudulent activities compared to manual methods.
- Convenience: Voters can participate from the comfort of their homes, increasing voter participation.
- Timely Results: Real-time counting leads to quicker announcement of results, avoiding delays.

3.4 AIM AND GOALS THE PROJECT

1. Develop an online voting system tailored for college elections.
2. Implement efficient voter registration procedures.
3. Ensure secure authentication and verification processes.
4. Enable seamless and accessible online voting for eligible students.
5. Streamline the process of candidate nomination and management.
6. Facilitate real-time vote counting and result tabulation.
7. Enhance transparency and integrity in the electoral process.
8. Improve overall efficiency and convenience for students participating in college elections

4. RELATED WORKS

A comprehensive body of research exists surrounding the design, implementation, and impact of online voting systems tailored for educational institutions such as colleges and universities. Research papers have delved into the intricacies of creating secure and efficient online voting platforms, addressing the unique needs and challenges of educational settings.

Case studies have analyzed the adoption and consequences of employing online voting systems within college environments, shedding light on their effectiveness and impact on student participation. Academic articles have explored the various challenges and opportunities associated with transitioning to electronic voting systems in educational contexts, highlighting both the benefits and potential pitfalls.

Additionally, technical reports have provided detailed insights into the architecture, features, and security measures of existing online voting platforms utilized in college elections, offering valuable guidance for system development and improvement. Surveys and studies have been conducted to assess student perceptions and satisfaction regarding online voting systems compared to traditional paper-based methods, informing ongoing efforts to enhance user experience and engagement.

Furthermore, white papers and policy documents have outlined best practices and guidelines for the development and deployment of online voting systems in educational contexts, helping to ensure compliance with legal and regulatory frameworks. Presentations and workshops at academic conferences or seminars have also contributed to the discourse on the use of technology in student governance, with a particular focus on online voting as a means of promoting democratic participation.

Moreover, publications from government agencies or electoral commissions have addressed the legal and regulatory frameworks governing online voting in educational institutions, providing valuable insights for policymakers and stakeholders involved in electoral processes. Together, these diverse sources of research and information form a rich tapestry of knowledge surrounding online voting in educational settings, informing ongoing efforts to enhance the integrity, accessibility, and effectiveness of electoral processes within colleges and universities.

5. SYSTEM ARCHITECTURE AND DATA FLOW

5.1 SYSTEM ARCHITECTURE

Encryption System:

- Data Encryption: Encryption techniques are used to secure sensitive data, such as voter information and ballot choices, both during transmission and storage.

Voting Module:

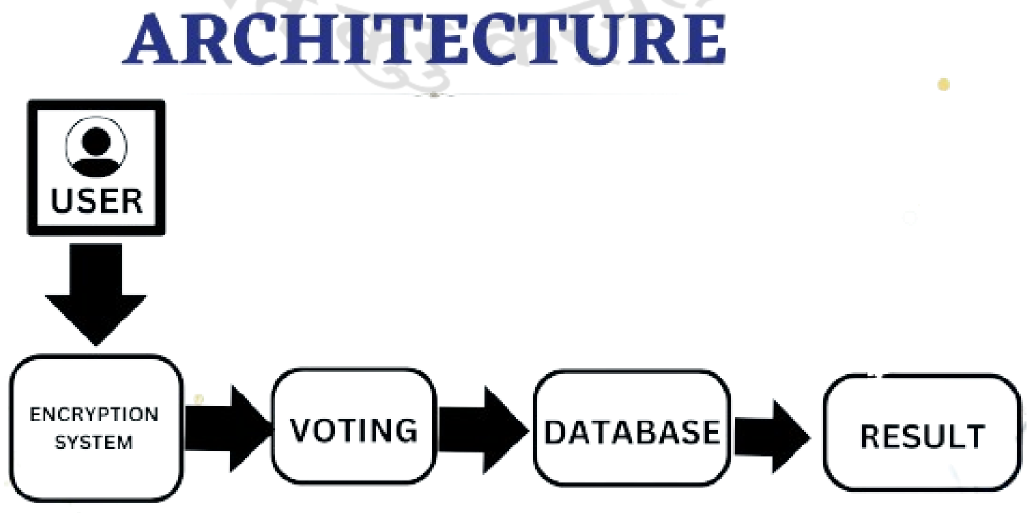
- Online Voter Registration: Allows voters to register online by providing necessary identification details and personal information. The system verifies and stores this information securely in the database.

- Ballot Generation: Generates digital ballots based on the candidates or issues relevant to the voter's constituency or jurisdiction.

- Vote Casting: Provides a secure platform for voters to cast their votes online, ensuring that each voter can only cast one vote and maintaining the secrecy of their choices

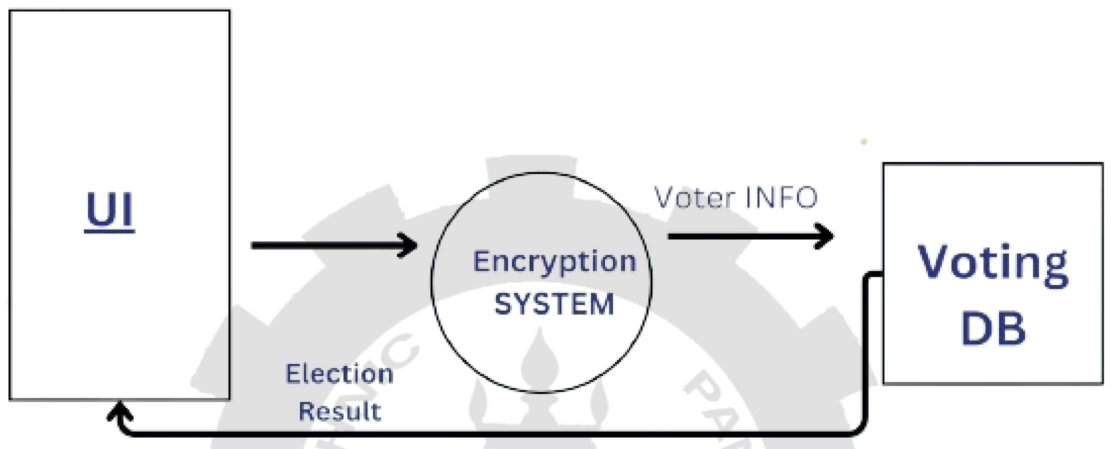
- Result Tallying Module::Real-time Tallying: Tallies votes in real-time as they are cast, allowing for quicker result announcements.

- DATABASE:The database component stores all relevant data, including voter registration information, encrypted votes, and election results. It ensures data integrity, reliability, and efficient retrieval of information.

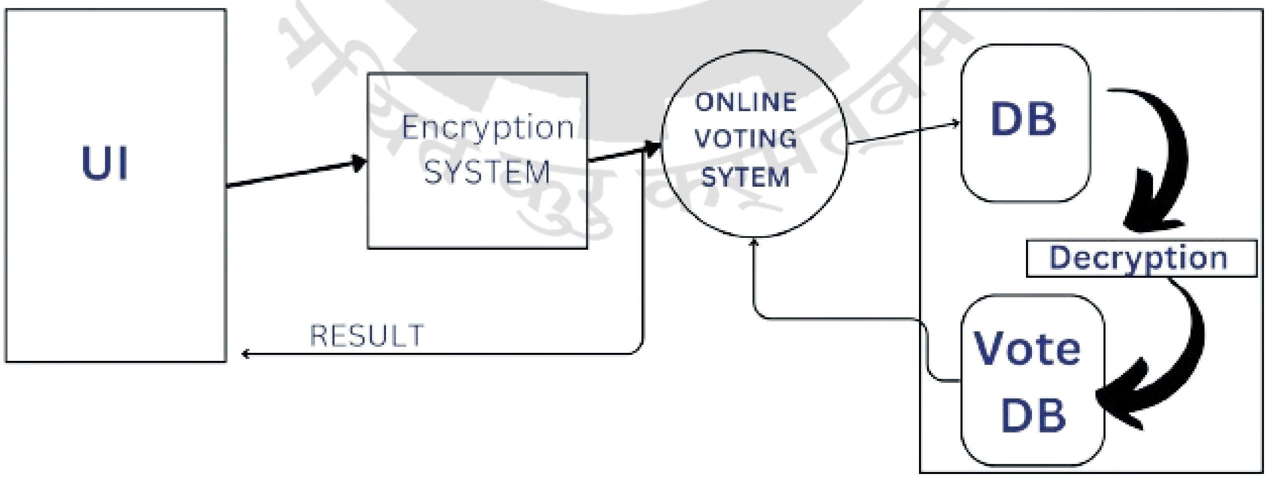


5.2 DATA FLOW DIAGRAM

Level 0



Level 1



6. IMPLMENTATION

Implementation of the Online Voting System for College Elections involves several key components, including the user interface (UI), backend logic, and authentication using face recognition technology.

1.User Interface (UI):

- Design and develop the user interface using HTML, CSS, and JavaScript to create a visually appealing and intuitive voting platform.
- The UI should include pages for user registration, login, candidate selection, and ballot submission.
- Implement responsive design principles to ensure the UI is accessible and functional across various devices and screen sizes.

2.Backend Logic:

- Develop the backend logic using the Django web framework in Python to handle user requests, process voting transactions, and manage system functionality.
- Define models for user accounts, candidate profiles, and voting records using Django's Object-Relational Mapping (ORM) capabilities.
- Implement views and controllers to handle user interactions, validate input data, and execute business logic operations.
- Create APIs to facilitate communication between the client-side interface and server-side application, enabling seamless data exchange.

3.Authentication Using Face Recognition:

- Integrate face recognition technology into the authentication process to verify the identity of voters during login or ballot submission.
- Utilize OpenCV, a popular computer vision library, to implement face detection and recognition algorithms.
- Upon user login, prompt the voter to capture their facial image using their device's camera.
- Compare the captured image with the pre-uploaded images stored in the system database (each image should correspond to a registered voter's profile).
- Use facial recognition algorithms to match the captured image with the stored images and authenticate the voter's identity.
- Grant access to the voting interface upon successful authentication, allowing the voter to cast their ballot securely.

4.Other Steps:

- Implement user registration functionality, allowing eligible students to create accounts by providing necessary personal information and identification documents.

- Develop candidate nomination features, enabling aspiring candidates to submit their profiles and election manifestos for review and approval by election officials.
- Implement ballot generation logic to dynamically generate digital ballots based on the available candidates and voting options.
- Enable voters to select their preferred candidates and submit their ballots securely, ensuring each voter can cast only one vote.
- Implement real-time result tabulation mechanisms to calculate and display election results as soon as all votes are cast and counted.

Throughout the implementation process, prioritize security measures, such as encryption of sensitive data, secure communication protocols, and robust access controls, to protect the integrity and confidentiality of the voting system. Additionally, conduct thorough testing at each stage to identify and address any bugs, vulnerabilities, or usability issues before deploying the system for actual use in college elections.

6.1 FACE DETECTION

```
import face_recognition
import cv2
import os
import glob
import numpy as np

class SimpleFacerec:
    def __init__(self):
        self.known_face_encodings = []
        self.known_face_names = []
        self.frame_resizing = 0.25

    def load_encoding_images(self, images_path):
        images_path = glob.glob(os.path.join(images_path, "*.*"))
        print("{} encoding images found.".format(len(images_path)))

        for img_path in images_path:
            img = cv2.imread(img_path)
            rgb_img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
            filename = os.path.splitext(os.path.basename(img_path))[0]
            img_encoding = face_recognition.face_encodings(rgb_img)[0]
            self.known_face_encodings.append(img_encoding)
            self.known_face_names.append(filename)
        print("Encoding images loaded")

    def detect_known_faces(self, frame):
        small_frame = cv2.resize(frame, (0, 0), fx=self.frame_resizing, fy=self.frame_resizing)
        rgb_small_frame = cv2.cvtColor(small_frame, cv2.COLOR_BGR2RGB)
        face_locations = face_recognition.face_locations(rgb_small_frame)
        face_encodings = face_recognition.face_encodings(rgb_small_frame, face_locations)

        face_names = []
        for face_encoding in face_encodings:
            matches = face_recognition.compare_faces(self.known_face_encodings,
face_encoding)
            name = "Unknown"

            face_distances = face_recognition.face_distance(self.known_face_encodings,
face_encoding)
            best_match_index = np.argmin(face_distances)
            if matches[best_match_index]:
                name = self.known_face_names[best_match_index]
            face_names.append(name)

        face_locations = np.array(face_locations)
        face_locations = face_locations / self.frame_resizing
        return face_locations.astype(int), face_names
```

6.2 WORKING

1. Initialization:

- The SimpleFacerec class is initialized with empty lists known_face_encodings and known_face_names to store the face encodings and corresponding names of known individuals.

2. Loading Encoding Images:

- The load_encoding_images method is used to load encoding images from a specified directory path (images_path).
- It iterates through all the image files in the directory using the glob module.
- For each image, it reads the image using OpenCV (cv2.imread), converts it to RGB format, and extracts the face encoding using the face_recognition.face_encodings function.
- The extracted face encoding and the file name (without extension) are stored in the known_face_encodings and known_face_names lists, respectively.

3. Detect Known Faces:

- The detect_known_faces method takes a frame from a video feed (or an image) as input.
- It resizes the frame to a smaller size (frame_resizing) to improve processing speed.
- Using the face_recognition library, it detects face locations (face_locations) and encodings (face_encodings) in the resized frame.
- For each detected face encoding, it compares it with the known face encodings stored in known_face_encodings using the face_recognition.compare_faces function.
- If a match is found, it retrieves the corresponding name from known_face_names and assigns it to the detected face. If no match is found, it labels the face as "Unknown".
- The face locations are adjusted based on the resizing factor, and both the adjusted face locations and corresponding names are returned

7.SYSTEM AND HARDWARE REQUIREMENTS

7.1 System Requirements

1. Operating System: Windows, macOS, or Linux
2. Python: Python 3.x
3. Django: Latest version (installable via pip)
4. OpenCV: Latest version (installable via pip)
5. Database: SQLite (for development), PostgreSQL or MySQL (for production)
6. Text Editor or IDE: Sublime Text, VS Code, PyCharm, etc.
7. Web Browser: Chrome, Firefox, Safari, etc.
8. Git (optional): Version control system for managing codebase
9. Image Processing Libraries: NumPy, Pillow (Python Imaging Library)
10. Web Server: Apache or Nginx (for production deployment)
11. Development Environment: Virtualenv or Conda for managing dependencies

7.2 Hardware Requirements

1. Processor: Intel Core i3 or equivalent
2. RAM: Minimum 4GB (8GB recommended for smoother performance)
3. Storage: At least 10GB of free disk space for development
4. Webcam: For capturing images or videos for OpenCV-based functionalities
5. Internet Connectivity: Required for online features and updates
6. Display: Monitor with at least 1280x720 resolution for development and testing

8. ALGORITHM

1. System Setup:
    - Configure servers and databases to handle user authentication, vote storage, and data processing, ensuring scalability for a large number of users.
    - Implement security measures like encryption and firewalls to protect sensitive voting data.

2. User Registration and Authentication:
    - Students register with the system using their college credentials or student ID.
    - Utilize college email verification or other methods to verify student authenticity.
    - Employ strong hashing algorithms to securely store passwords associated with student accounts.

3. Voter Eligibility Verification:
    - Verify voter eligibility based on enrollment status and other criteria set by the college (e.g., active student status, absence of disciplinary actions).
    - Authenticate voters securely while maintaining their privacy and confidentiality.

4. Ballot Generation:
    - Create digital ballots specific to the college election, including positions like student body president, vice president, treasurer, etc.
    - Ensure that ballots accurately list all eligible candidates running for each position.

5. Voting Process:
    - Students log in to the system using their college credentials.
    - Present students with the available ballot options for each position in the college election.
    - Allow students to securely select their choices for each position.

6. Vote Submission:
    - Encrypt and securely transmit students' votes to the server to prevent tampering or interception.
    - Store votes securely in the database, associating each vote with the respective student and position.

7. Vote Counting:
    - Implement algorithms(Django view function) to accurately count the votes cast for each candidate in the college election.
    - Ensure that the counting process maintains anonymity and integrity, adhering to college election regulations.

8. Result Presentation:
    - Once voting is closed, tally the votes for each position and determine the winners.
    - Present the election results to students in a clear and understandable format, possibly through announcements.

9. Security Measures:
    - Regularly update the system to address security vulnerabilities and protect against potential threats.

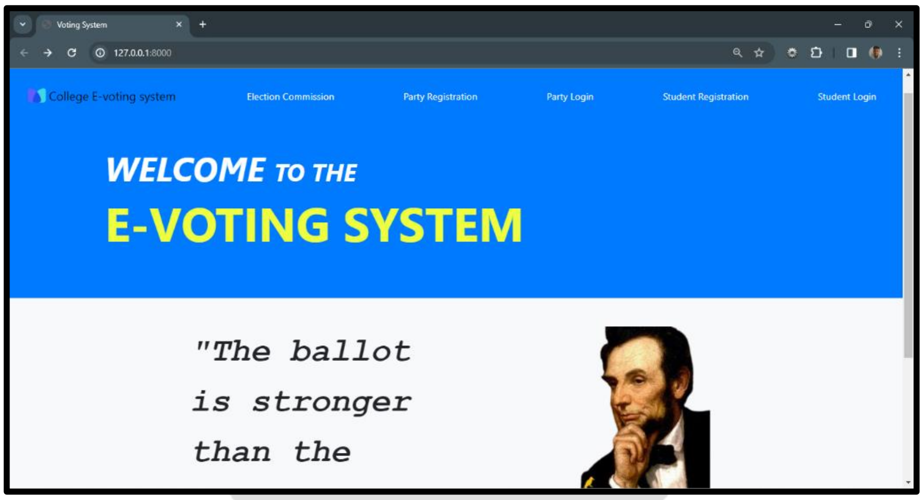
- Implement measures to prevent unauthorized access and ensure the integrity of the voting process.

10. Audit Trail:
    - Maintain an audit trail of all activities within the system, including user interactions and vote submissions, to ensure accountability and transparency.

11. Post-Election Procedures:
    - Archive voting data securely for future reference and auditing purposes.
    - Address any disputes or challenges to the election results according to predefined procedures established by the college.

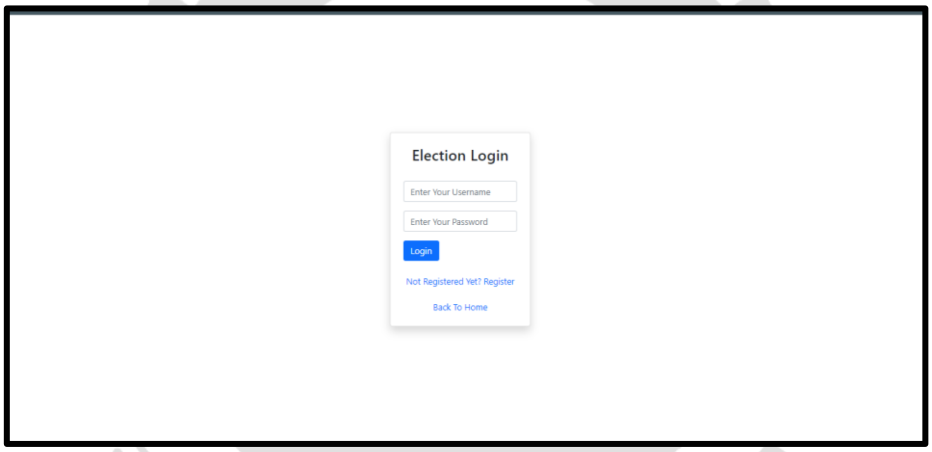
12. User Feedback and Improvement:
    - Collect feedback from students to enhance the usability and efficiency of the voting system for future college elections.
    - Continuously monitor and evaluate the system to identify areas for improvement and implement necessary enhancements.

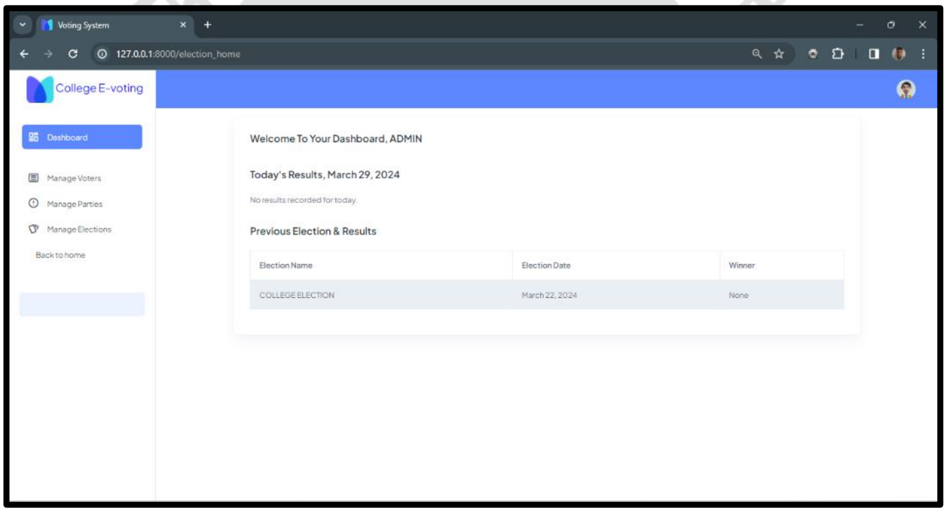
9. RESULTS AND DISCUSSIONS



WELCOME TO THE
E-VOTING SYSTEM

"The ballot
is stronger
than the





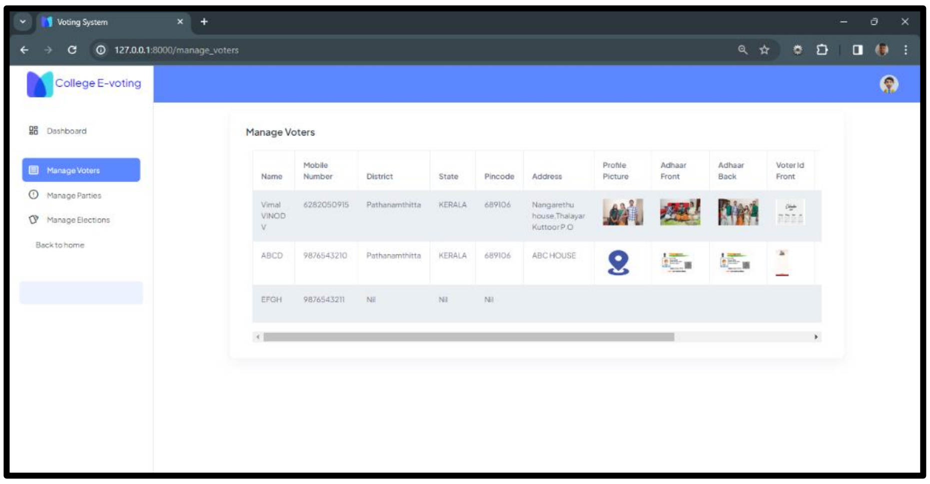
Welcome To Your Dashboard, ADMIN

Today's Results, March 29, 2024

No results recorded for today.

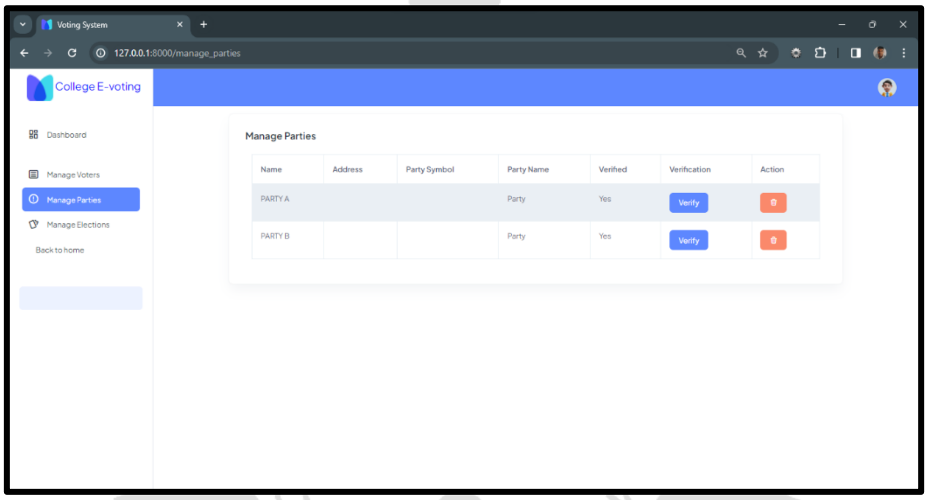
Previous Election & Results

Election Name	Election Date	Winner
COLLEGE ELECTION	March 22, 2024	None



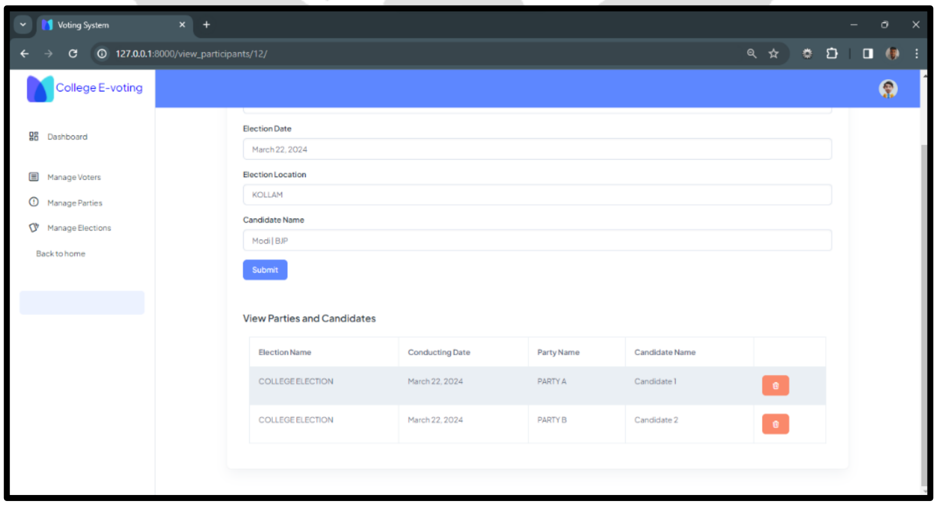
Manage Voters

Name	Mobile Number	District	State	Pincode	Address	Profile Picture	Adhaar Front	Adhaar Back	Voter Id Front
Vimal VINOD V	6282050915	Pathanamthitta	KERALA	689106	Nangarethu house,Thalayar Kuttoor P.O				
ABCD	9876543210	Pathanamthitta	KERALA	689106	ABC HOUSE				
EFGH	9876543211	Nil	Nil	Nil					



Manage Parties

Name	Address	Party Symbol	Party Name	Verified	Verification	Action
PARTY A			Party	Yes	Verify	
PARTY B			Party	Yes	Verify	



Election Date: March 22, 2024

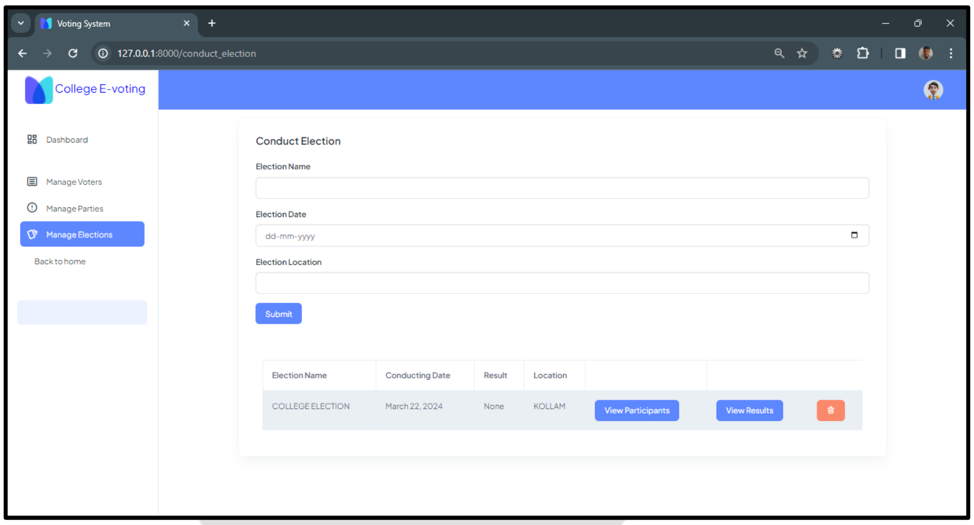
Election Location: KOLLAM

Candidate Name: Modi | BJP

Submit

View Parties and Candidates

Election Name	Conducting Date	Party Name	Candidate Name	
COLLEGE ELECTION	March 22, 2024	PARTY A	Candidate 1	
COLLEGE ELECTION	March 22, 2024	PARTY B	Candidate 2	


Voting System

127.0.0.1:8000/conduct_election

College E-voting

Dashboard

Manage Voters

Manage Parties

Manage Elections

Back to home

Conduct Election

Election Name

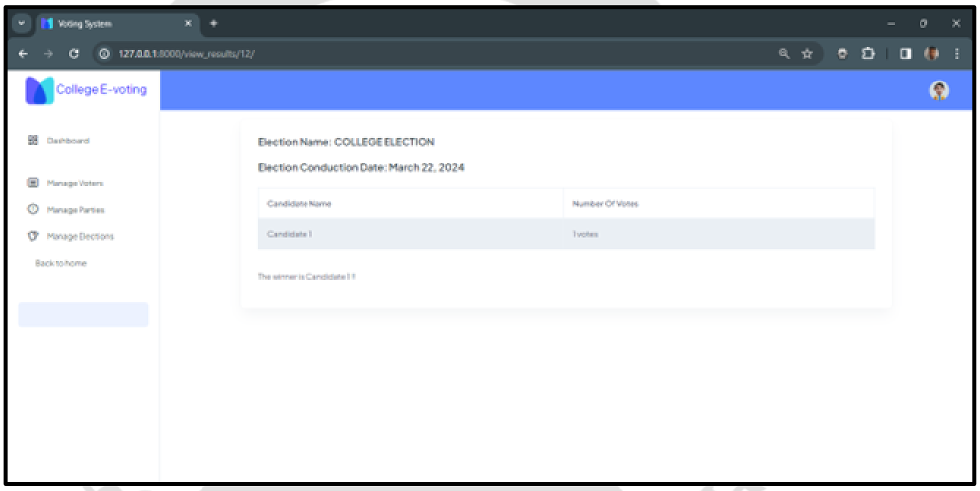
Election Date

dd-mm-yyyy

Election Location

Submit

Election Name	Conducting Date	Result	Location		
COLLEGE ELECTION	March 22, 2024	None	KOLLAM	View Participants	View Results


Voting System

127.0.0.1:8000/view_results/12/

College E-voting

Dashboard

Manage Voters

Manage Parties

Manage Elections

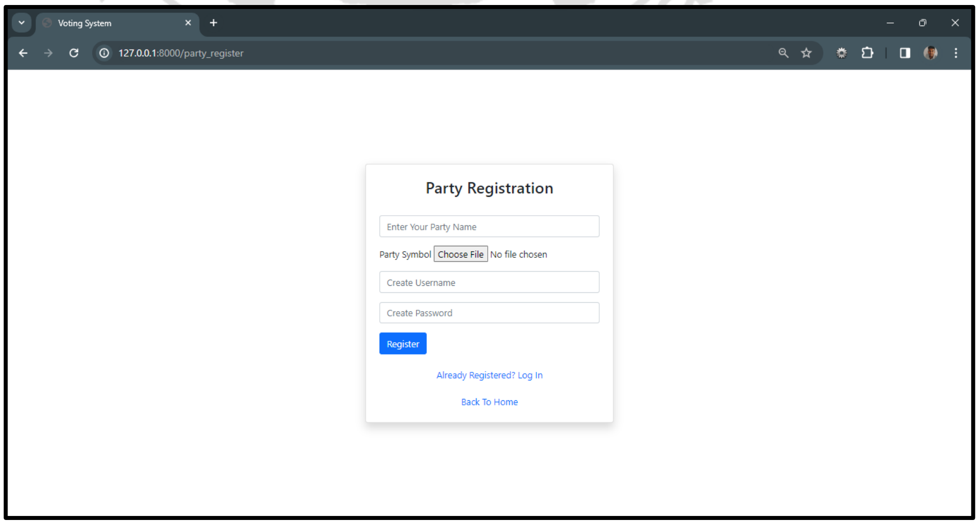
Back to home

Election Name: COLLEGE ELECTION

Election Conduction Date: March 22, 2024

Candidate Name	Number Of Votes
Candidate 1	1 votes

The winner is Candidate 1 !!


Voting System

127.0.0.1:8000/party_register

Party Registration

Enter Your Party Name

Party Symbol Choose File No file chosen

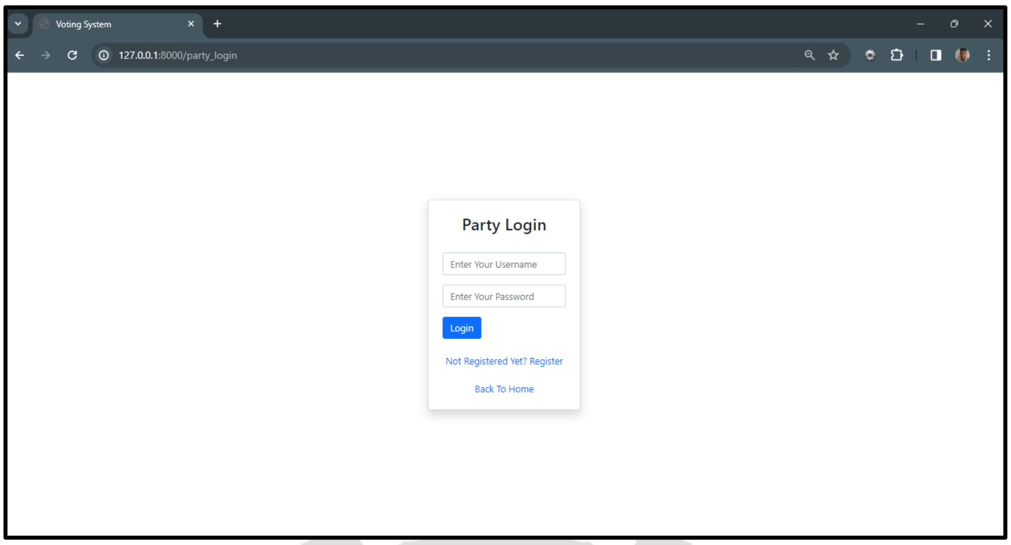
Create Username

Create Password

Register

Already Registered? Log In

Back To Home



Voting System

127.0.0.1:8000/party_login

Party Login

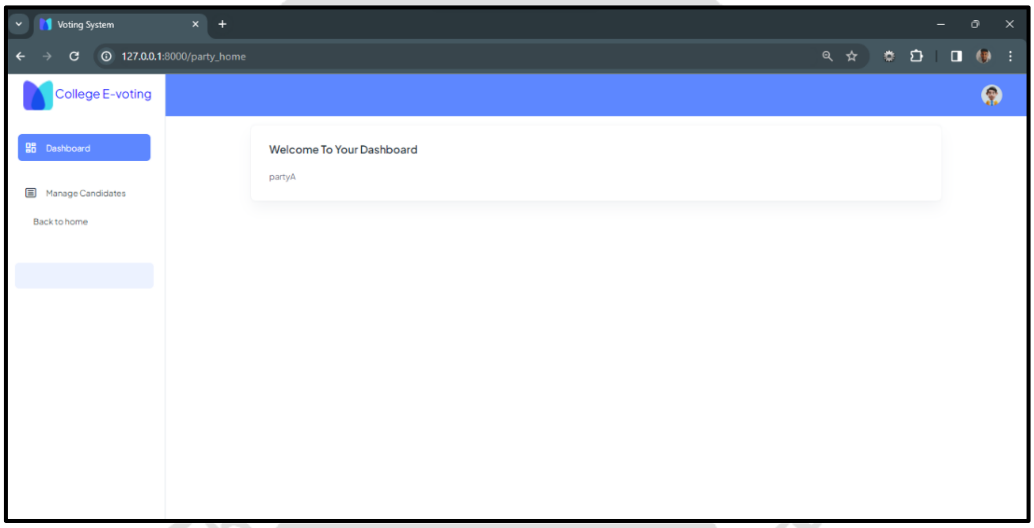
Enter Your Username

Enter Your Password

Login

Not Registered Yet? Register

Back To Home



Voting System

127.0.0.1:8000/party_home

College E-voting

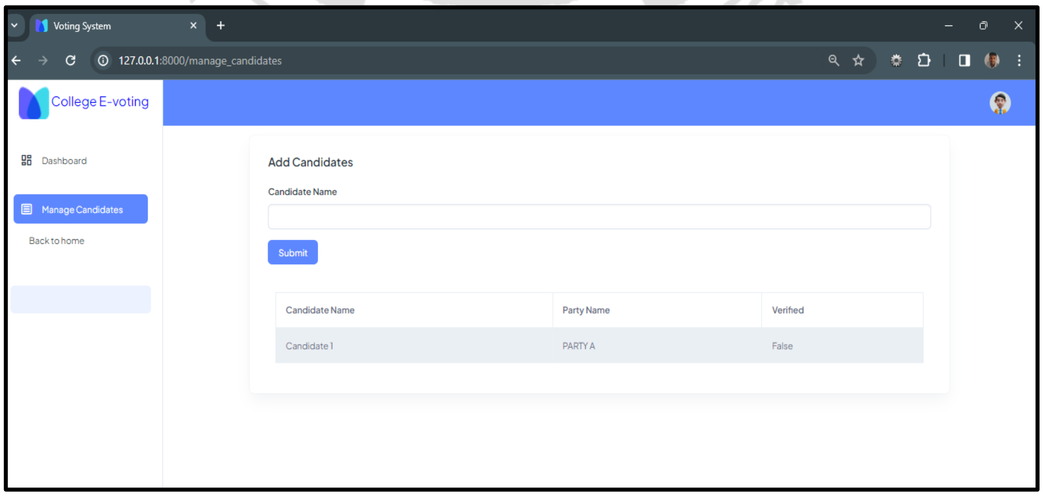
Dashboard

Manage Candidates

Back to home

Welcome To Your Dashboard

partyA



Voting System

127.0.0.1:8000/manage_candidates

College E-voting

Dashboard

Manage Candidates

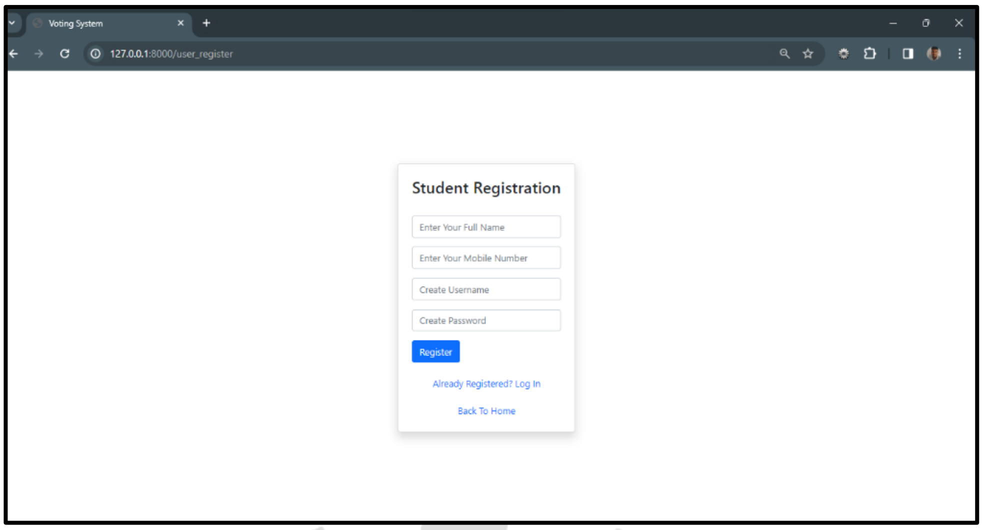
Back to home

Add Candidates

Candidate Name

Submit

Candidate Name	Party Name	Verified
Candidate 1	PARTY A	False


Voting System

127.0.0.1:8000/user_register

Student Registration

Enter Your Full Name

Enter Your Mobile Number

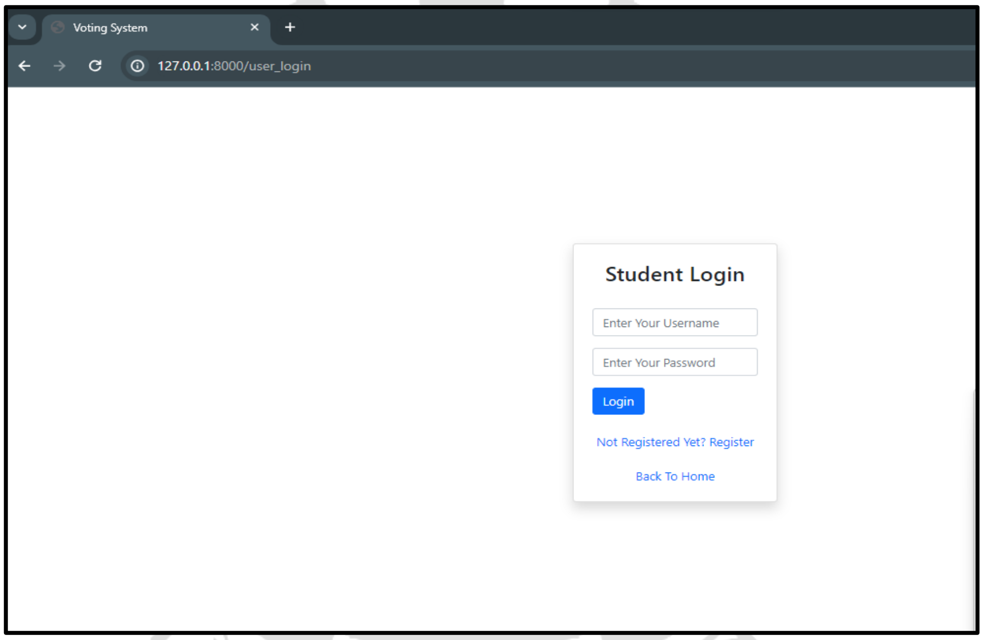
Create Username

Create Password

Register

Already Registered? Log In

Back To Home


Voting System

127.0.0.1:8000/user_login

Student Login

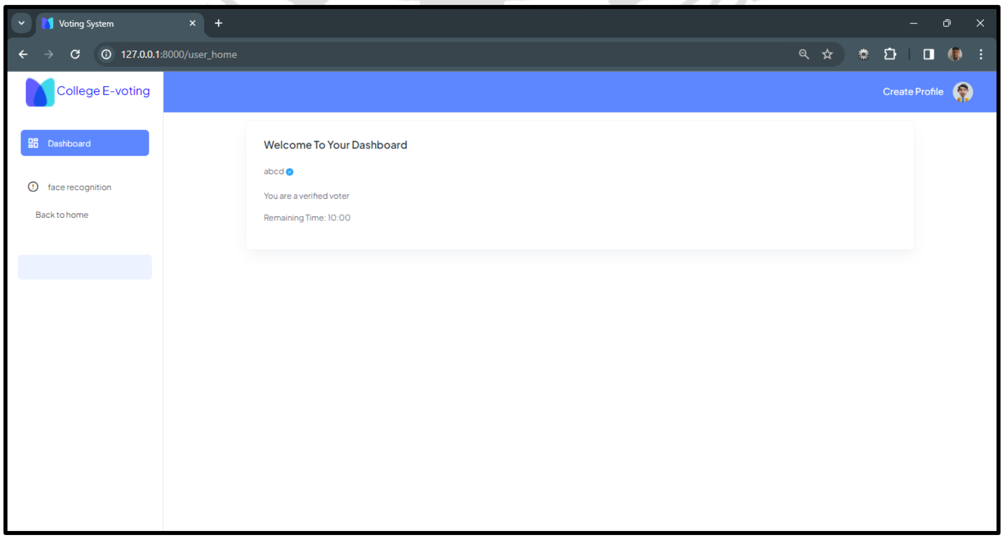
Enter Your Username

Enter Your Password

Login

Not Registered Yet? Register

Back To Home


Voting System

127.0.0.1:8000/user_home

College E-voting

Create Profile

Dashboard

face recognition

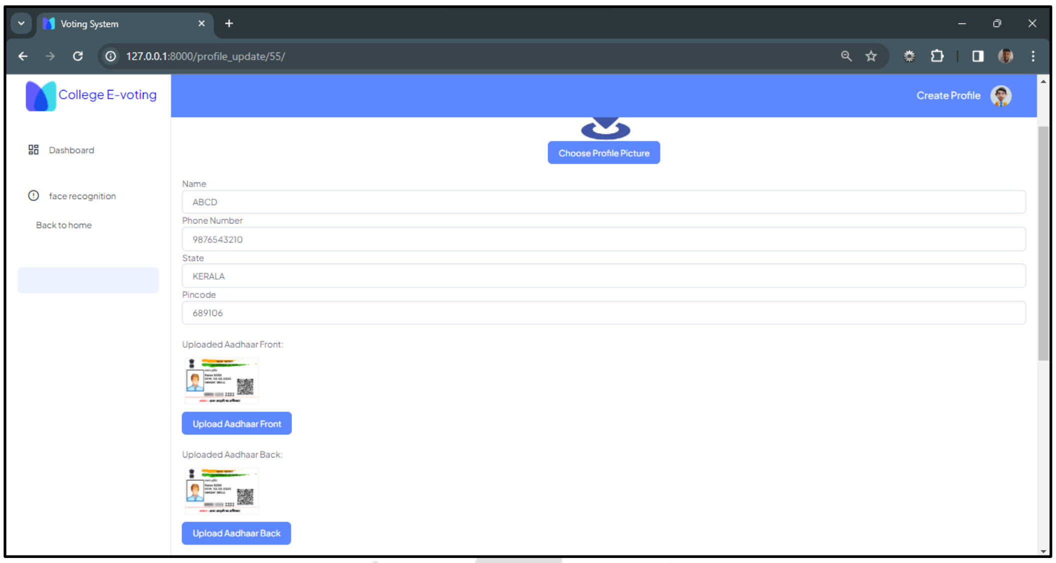
Back to home

Welcome To Your Dashboard

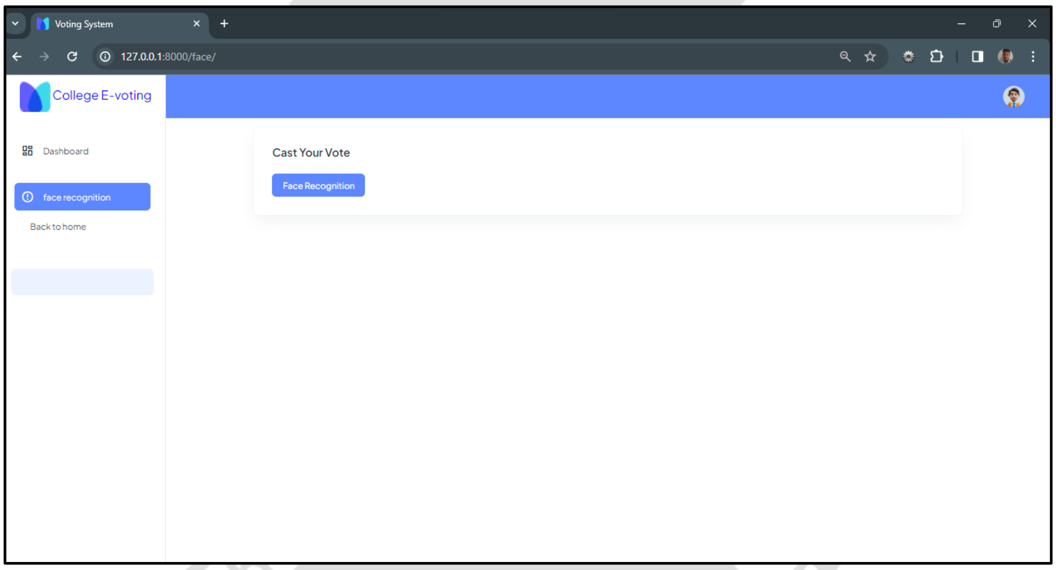
abcd

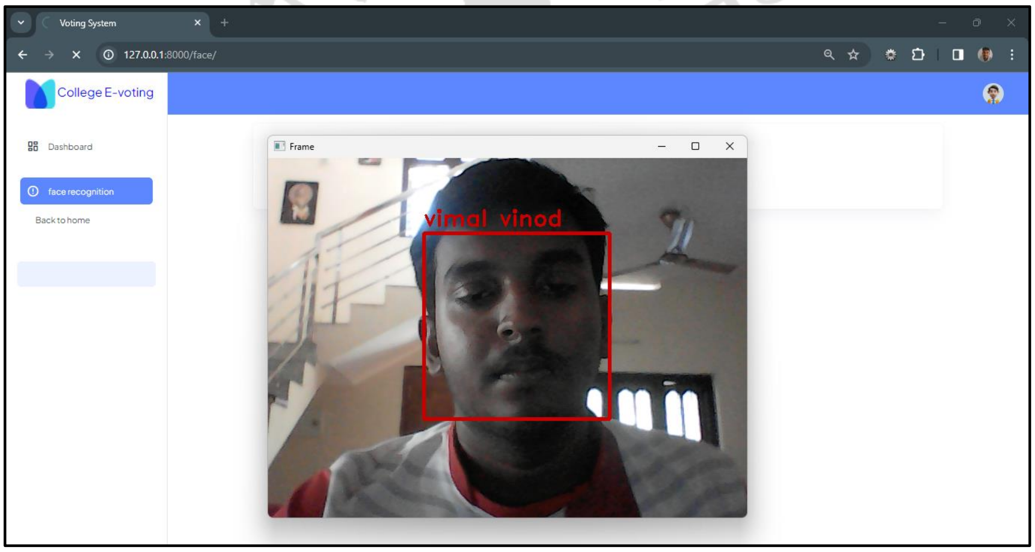
You are a verified voter

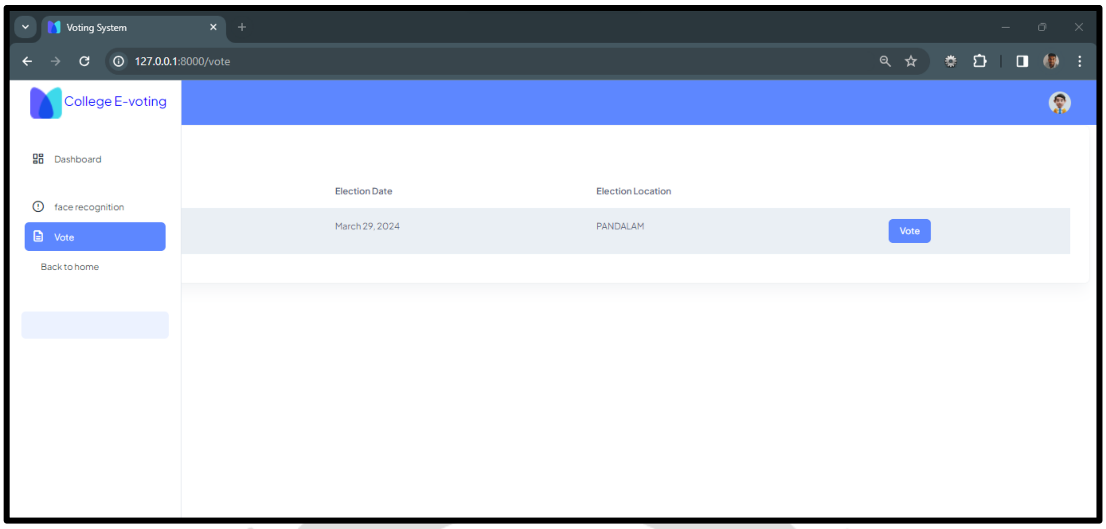
Remaining Time: 10:00



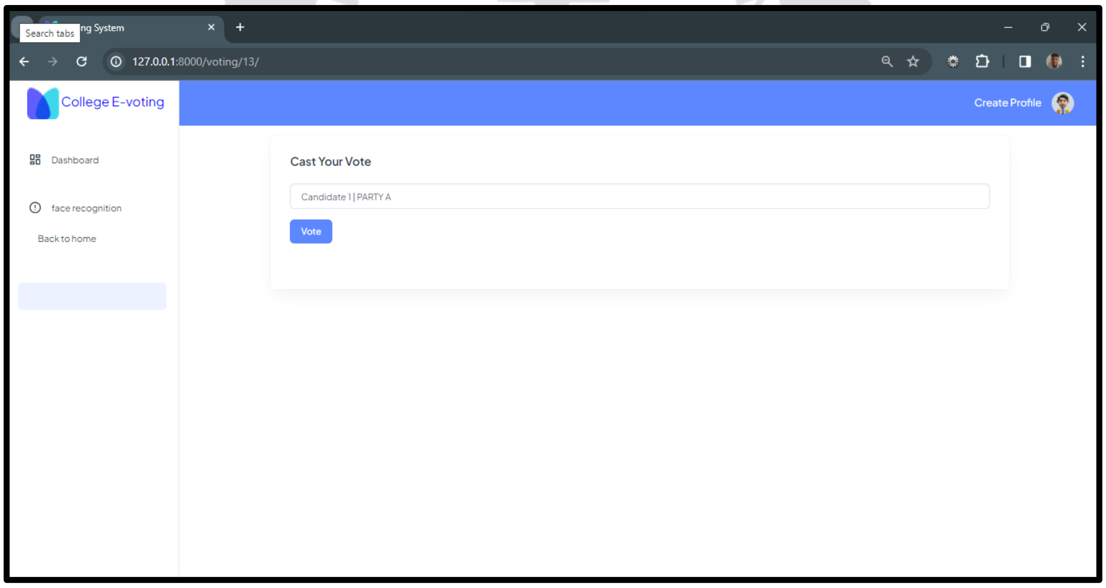
Name
ABCD
Phone Number
9876543210
State
KERALA
Pincode
689106
Uploaded Aadhaar Front:
Uploaded Aadhaar Back:







Election Date	Election Location	
March 29, 2024	PANDALAM	Vote



Cast Your Vote

Candidate 1 | PARTY A

Vote

10. CODES

10.1 Fronted

10.1.1 GENERAL

Index.html

```
<!DOCTYPE html>
{% load static %}
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Voting System</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css"
    rel="stylesheet" integrity="sha384-
    T3c6CoIi6uLrA9TneNEoa7RxnatzjcDSCmG1MXxSR1GAsXEV/Dwwykc2MPK8M2HN"
    crossorigin="anonymous">
  <!-- Add Bootstrap CSS link -->
  <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
  <!-- Add custom styles -->
  <style>
    body {
      background-color: #f8f9fa;
    }

    .cont {
      max-width: 1500px;
      margin: auto;
      /* text-align: center; */
      padding-top: 25px;
      background-color: #007bff;
      padding-left: 10rem;
    }

    h1, h2 {
      color: #007bff;
    }

    a {
      display: block;
      margin-top: 20px;
      color: #007bff;
    }
```

```
        .
        e-container {
      display: flex;


       justify-content: space-between; /* To center content horizontally */
      align-items: center; /* To center content vertically */
  }

  .row {
      display: flex;
      flex-direction: row;

  }

  .content {
      flex: 1;
      padding: 0 20px; /* Adjust as needed */
      animation: slide 0.8s ease-in-out;
  }
  @keyframes slide{
      0%{
          transform: translateX(-20px);
          opacity: 0.6;
      }
      100%{
          transform: translateX(0px);
          opacity: 1;
      }
  }
  .image-container {
      flex: 1;
      padding: 0 40px; /* Adjust as needed */
      animation: slide 0.5s ease-in-out;
  }

  .img1 {
      max-width: 100%;
      height: auto;
  }
  nav{
      height: 9rem;
  }
      </style>
  </head>
  <body>
      <nav class="navbar bg-primary">
          <div class="container-fluid">
            <a class="navbar-brand" href="#">
              <img  src="{%static '/assets/images/logos/favicon.png' %}" alt="Logo" width="30"
                    height="24" class="d-inline-block align-text-top">
            College E-voting system
            </a>
            <a href="{% url 'election_login' %}" class="btn btn-primary">Election Commission</a>
          <a href="{% url 'party_register' %}" class="btn btn-primary">Party Registration</a>


          <a href="{% url 'party_login' %}" class="btn btn-primary">Party Login</a>
          <a href="{% url 'user_register' %}" class="btn btn-primary">Student Registration</a>
          <a href="{% url 'user_login' %}" class="btn btn-primary">Student Login</a>
      </div>
          </div>
        </nav>
      <div class="cont container">
          <h2 style="color: white;font-size: 3.5rem;font-weight: bolder;"><em>WELCOME <span
                  style="font-weight: bold; font-size: 2.5rem;">TO THE </span></em></h2>
          <h2 style="color: #eeff41;font-size: 5rem;font-weight: bold;">E-VOTING SYSTEM</h2>

          <br>
          <!-- <img src="{% static '/assets/images/vote1.jpg' %}" height="200" style="border-radius:
                  15px;"> -->
          <br>
          <br>

      </div>
      <div class="e-container col-md-8 mx-auto w-80 p-5">
          <div class="row">
              <div class="content">
                  <p style="font-size: 3.5rem;font-weight: bolder; font-family: 'Courier New', Courier,
                    monospace;"><strong><em> "The ballot is stronger than the bullet".
              </em></strong></p>
                  <p style="padding-right: 10px;">- Abraham Lincoln</p>
              </div>
              <div class="image-container">
                  <img class="img1" src="{% static '/assets/images\al.png' %}" alt="" height="300"
                    width="800" style="border-radius: 20px;">
              </div>
          </div>
      </div><br><br>
      <footer class="bg-body-tertiary text-center text-lg-start">
          <!-- Copyright -->
          <div class="text-center p-3" style="background-color: rgba(0, 0, 0, 0.05);">
            © 2024 Copyright:
            <a class="text-body" href="#">collegeelection.com</a>
          </div>
          <!-- Copyright -->
        </footer>

      <!-- Add Bootstrap JS and Popper.js scripts (required for Bootstrap) -->
      <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>
      <script
      src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.0.9/dist/umd/popper.min.js"></script>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js"
  integrity="sha384-
  C6RzsynM9kWDrMNeT87bh95OGNyZPhcTNXj1NW7RuBCsyN/o0jlpcV8Qyq46cDfL"
  crossorigin="anonymous"></script>


  </body>
  </html>
```

10.1.2 ELECTION

Base.html

```
<!doctype html>
{% load static %}
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>Voting System</title>
  <link rel="shortcut icon" type="image/png" href="{% static 'assets/images/logos/favicon.png'
         %}" />
  <link rel="stylesheet" href="{% static 'assets/css/styles.min.css' %}" />

</head>

<body>
  <!--  Body Wrapper -->
  <div class="page-wrapper" id="main-wrapper" data-layout="vertical" data-navbarbg="skin6"
       data-sidebartype="full"
    data-sidebar-position="fixed" data-header-position="fixed">
    <!-- Sidebar Start -->
    <aside class="left-sidebar">
      <!-- Sidebar scroll-->
      <div>
        <div class="brand-logo d-flex align-items-center justify-content-between">
          <a href="{% url 'election_home' %}" class="text-nowrap logo-img m-2" style="color: blue;
               font-size: 20px;">
            <img src="{% static '/assets/images/logos/favicon.png' %}" width="50" height="50" alt=""
              />
            College E-voting
          </a>
          <div class="close-btn d-xl-none d-block sidebartoggler cursor-pointer"
              id="sidebarCollapse">
            <i class="ti ti-x fs-8"></i>
          </div>
        </div>
      <!-- Sidebar navigation-->
          <nav class="sidebar-nav scroll-sidebar" data-simplebar="">
          <ul id="sidebarnav">
            <li class="nav-small-cap">
             <i class="ti ti-dots nav-small-cap-icon fs-4"></i>
             <span class="hide-menu"></span>
            </li>
            <li class="sidebar-item">


             <a class="sidebar-link" href="{% url 'election_home' %}" aria-expanded="false">
              <span>
               <i class="ti ti-layout-dashboard"></i>
              </span>
              <span class="hide-menu">Dashboard</span>
             </a>
            </li>
            <li class="nav-small-cap">
             <i class="ti ti-dots nav-small-cap-icon fs-4"></i>
             <span class="hide-menu"></span>
            </li>
            <li class="sidebar-item">
             <a class="sidebar-link" href="{% url 'manage_voters' %}" aria-expanded="false">
              <span>
               <i class="ti ti-article"></i>
              </span>
              <span class="hide-menu">Manage Voters</span>
             </a>
            </li>
            <li class="sidebar-item">
             <a class="sidebar-link" href="{% url 'manage_parties' %}" aria-expanded="false">
              <span>
               <i class="ti ti-alert-circle"></i>
              </span>
              <span class="hide-menu">Manage Parties</span>
             </a>
            </li>
            <li class="sidebar-item">
             <a class="sidebar-link" href="{% url 'conduct_election' %}" aria-expanded="false">
              <span>
               <i class="ti ti-cards"></i>
              </span>
              <span class="hide-menu">Manage Elections</span>
             </a>
            </li>

            <li class="sidebar-item">
             <a class="sidebar-link" href="{% url 'index' %}" aria-expanded="false">
              <span>
               <i class=""></i>
```

```
</span>
            <span class="hide-menu">Back to home</span>
          </a>
        </li>
```

Election_home.html

```
{% extends 'election/base.html' %}
{% load static %}
{% block content %}

<div class="container-fluid">
    <div class="card">
      <div class="card-body">
        <h5 class="card-title fw-semibold mb-4">Welcome To Your Dashboard,
            {{request.user}}</h5>
        <p class="mb-0"></p>
        <br>

        <h5 class="card-title fw-semibold mb-4">Today's Results, {{ today_date }}</h5>

        {% if today_results %}
            <table class="table table-bordered table-striped">
                <thead>
                    <tr>
                        <th scope="col">Candidate Name</th>
                        <th scope="col">Number of Votes</th>
                    </tr>
                </thead>
                <tbody>
                    {% for result in today_results %}
                        <tr>
                            <td>{{ result.encrypted_candidate_name }}</td>
                            <td>{{ result.votes_count }} votes</td>
                        </tr>



                {% endfor %}
                </tbody>
            </table>
        {% else %}


            <p>No results recorded for today.</p>
        {% endif %}
        <br>
        <h5 class="card-title fw-semibold mb-4">Previous Election & Results</h5>
          <table class="table table-bordered table-striped">
            <thead>
              <tr>
                <th scope="col">Election Name</th>
                <th scope="col">Election Date</th>
                <th scope="col">Winner</th>
              </tr>
            </thead>
            <tbody>
              {% for i in election %}
                <tr>
                  <td>{{ i.election_name }}</td>
                  <td>{{ i.election_date }}</td>
                  <td>{{ i.result }}</td>

                </tr>
              {% endfor %}
            </tbody>
          </table>
      </div>
    </div>
  </div>

{% endblock %}
```

Election_login.html

```
<!DOCTYPE html>
{% load static %}
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Voting System</title>

   <!-- Add Bootstrap CSS -->
   <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0/dist/css/bootstrap.min.css">


   <link rel="preconnect" href="https://fonts.googleapis.com">
  <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
   <link
href="https://fonts.googleapis.com/css2?family=Nunito:wght@700&family=Poppins:wght@100;30
0;400;600&family=Quicksand:wght@500&display=swap" rel="stylesheet">


 <script src="https://kit.fontawesome.com/6bf2dfa19e.js" crossorigin="anonymous"></script>
</head>
```

```
<body>
  <div class="container d-flex justify-content-center align-items-center min-vh-100">
    <div class="card p-4 shadow">
      <form method="POST">
        {% csrf_token %}
        <h3 style="text-align:center;">Election Login</h3>
        <br>

        <div class="mb-3">
          <input type="text" class="form-control" name="username" placeholder="Enter Your
                Username" required>
        </div>
        <div class="mb-3">
          <input type="password" class="form-control" name="password" placeholder="Enter
                Your Password" required>
        </div>
        <input type="submit" value="Login" class="btn btn-primary">
        {% if messages %}
          <ul class="messages">
            {% for message in messages %}
              <li{% if message.tags %} class="{{ message.tags }}"{% endif %} style="list-
              style-type: none; color:red;">{{ message }}</li>
            {% endfor %}
          </ul>
        {% endif %}
      </form>
      <br>
      <a href="{%url 'election_register' %}" style="text-align:center; text-decoration:none;">Not
                Registered Yet? Register</a>
      <br>
      <a href="{% url 'index' %}" style="text-align:center; text-decoration:none;">Back To
       Home</a>
    </div>
  </div>
  <!-- Add Bootstrap JS and Popper.js -->
       <script src="https://cdn.jsdelivr.net/npm/popper.js@2.11.8/dist/umd/popper.min.js"
     integrity="sha384-
     I7E8VVD/ismYTF4hNIPjVp/Zjvgyol6VFvRkX/vR+Vc4jQkC+hVqc2pM8ODewa9r"
     crossorigin="anonymous"></script>
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0/dist/js/bootstrap.min.js"></script>


       </body>
       </html>
```

Election_register.html

```
<!DOCTYPE html>
{% load static %}
<html lang="en">
```

```
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Voting System</title>

    <!-- Add Bootstrap CSS -->
    <link rel="stylesheet"
  href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0/dist/css/bootstrap.min.css">

    <link rel="preconnect" href="https://fonts.googleapis.com">
    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
    <link
  href="https://fonts.googleapis.com/css2?family=Nunito:wght@700&family=Poppins:wght@100;
  300;400;600&family=Quicksand:wght@500&display=swap" rel="stylesheet">
    <script src="https://kit.fontawesome.com/6bf2dfa19e.js" crossorigin="anonymous"></script>

  </head>
  <body>
    <div class="container d-flex justify-content-center align-items-center min-vh-100">
      <div class="card p-4 shadow">
        <form method="POST">
          {% csrf_token %}
          <h3 style="text-align:center;">Election Registration</h3>
          <br>
          <div class="mb-3">
            <input type="text" class="form-control" name="username" placeholder="Create
                  Username" required>
          </div>
          <div class="mb-3">
            <input type="password" class="form-control" name="password"
                  placeholder="Create Password" required>
          </div>
          <input type="submit" value="Register" class="btn btn-primary">
          {% if messages %}
            <ul class="messages">
              {% for message in messages %}

              <li{% if message.tags %} class="{{ message.tags }}"{% endif %} style="list-
                  style-type: none; color:red;">{{ message }}</li>
              {% endfor %}
            </ul>
          {% endif %}
        </form>
        <br>
```

```
            <a href="{% url 'index' %}" style="text-align:center; text-decoration:none;">Back To
                Home</a>
        </div>
    </div>
    <!-- Add Bootstrap JS and Popper.js -->
    <script src="https://cdn.jsdelivr.net/npm/popper.js@2.11.8/dist/umd/popper.min.js"
integrity="sha384-
I7E8VVD/ismYTF4hNIPjVp/Zjvgyol6VFvRkX/vR+Vc4jQkC+hVqc2pM8ODewa9r"
crossorigin="anonymous"></script>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0/dist/js/bootstrap.min.js"></script>
</body>
</html>
```

Manage_parties.html

```
{% extends 'election/base.html' %}
{% load static %}
{% block content %}

<div class="container-fluid">
    <div class="card">
        <div class="card-body">
            <h5 class="card-title fw-semibold mb-4">Manage Parties</h5>

            <div class="container-fluid">
                <div class="table-responsive">
                    <table class="table table-striped table-bordered">
                        <thead class="thead-dark">
                            <tr>
                                <th>Name</th>
                                <th>Address</th>
                                <th>Party Symbol</th>
                                <th>Party Name</th>
                                <th>Verified</th>
                                <th>Verification</th>
                                <th>Action</th>

                              </tr>
                        </thead>
                        <tbody>
                            {% for i in voters %}
                            <tr>
                                <td>{{ i.name }}</td>
                                <td>{{ i.address }}</td>
                                <td>
                                    <a href="{{ i.imageURL }}" target="_blank">
                                        <img src="{{ i.imageURL }}" height="50px">
                                    </a>
                                </td>

                                <td>{{ i.party_name }}</td>

                                <td>
                                    {% if i.is_verified %}
                                        Yes
                                    {% else %}
                                        No
                                    {% endif %}
                                </td>
                                <td><a href="{% url 'verify_party' i.pk %}" class="btn btn-
                                        primary">Verify</a></td>

                                <td><a href="{% url 'remove_party' i.pk %}" class="btn btn-danger"><i
                                        class="ti ti-trash" style="font-size:15px;"></i></a></td>

                            </tr>
                            {% endfor %}
                        </tbody>
                    </table>
                </div>
            </div>
        </div>
    </div>
</div>

{% endblock %}
```

<u>Manage_voters.html</u>

```
{% extends 'election/base.html' %}
{% load static %}
{% block content %}
<div class="container-fluid">
    <div class="card">
        <div class="card-body">
            <h5 class="card-title fw-semibold mb-4">Manage Voters</h5>
```

```
<div class="container-fluid">
            <div class="table-responsive">
               <table class="table table-striped table-bordered">
                  <thead class="thead-dark">
                     <tr>
                        <th>Name</th>
                        <th>Mobile Number</th>
                        <th>District</th>
                        <th>State</th>
                        <th>Pincode</th>
                        <th>Address</th>
                        <th>Profile Picture</th>
                        <th>Adhaar Front</th>
                        <th>Adhaar Back</th>
                        <th>Voter Id Front</th>
                        <th>Voter Id Back</th>
                        <th>Verified</th>
                        <th>Verification</th>
                        <th>Action</th>


                        </tr>
                  </thead>
                  <tbody>
                     {% for i in voters %}
                     <tr>
                        <td>{{ i.name }}</td>
                        <td>{{ i.mobile_number }}</td>
                        <td>{{ i.district }}</td>
                        <td>{{ i.state }}</td>
                        <td>{{ i.pincode }}</td>
                        <td>{{ i.address }}</td>
                        <td>
                           <a href="{{ i.imageURL }}" target="_blank">
                              <img src="{{ i.imageURL }}" height="50px">
                           </a>
                        </td>
                        <td>
                           <a href="{{ i.imageURL2 }}" target="_blank">
                              <img src="{{ i.imageURL2 }}" height="50px">
                           </a>
                        </td>
                        <td>
```

```
    <a href="{{ i.imageURL3 }}" target="_blank">
                         <img src="{{ i.imageURL3 }}" height="50px">
                       </a>
                    </td>
                    <td>
                       <a href="{{ i.imageURL4 }}" target="_blank">
                         <img src="{{ i.imageURL4 }}" height="50px">
                       </a>
                    </td>
                    <td>
                       <a href="{{ i.imageURL5 }}" target="_blank">
                         <img src="{{ i.imageURL5 }}" height="50px">
                       </a>
                    </td>

                    <td>
                       {% if i.is_verified %}
                         Yes
                       {% else %}
                         No
                       {% endif %}


             </td>
                    <td><a href="{% url 'verify_voter' i.pk %}" class="btn btn-
                    primary">Verify</a></td>
                    <td><a href="{% url 'remove_voter' i.pk %}" class="btn btn-danger"><i
                           class="ti ti-trash" style="font-size:15px;"></i></a></td>
                  </tr>
                  {% endfor %}
                </tbody>
              </table>
            </div>
          </div>
        </div>
      </div>
    </div>

  {% endblock %}
```

11.1.3 PARTY

Party_login.html

```
<!DOCTYPE html>
{% load static %}
<html lang="en">
<head>
   <meta charset="UTF-8">


 <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Voting System</title>

   <!-- Add Bootstrap CSS -->
   <link rel="stylesheet"


href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0/dist/css/bootstrap.min.css">
   <link rel="preconnect" href="https://fonts.googleapis.com">
   <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
   <link
href="https://fonts.googleapis.com/css2?family=Nunito:wght@700&family=Poppins:wght@100;
300;400;600&family=Quicksand:wght@500&display=swap" rel="stylesheet">
   <script src="https://kit.fontawesome.com/6bf2dfa19e.js" crossorigin="anonymous"></script>

</head>
<body>
   <div class="container d-flex justify-content-center align-items-center min-vh-100">
      <div class="card p-4 shadow">
         <form method="POST">
            {% csrf_token %}
            <h3 style="text-align:center;">Party Login</h3>
            <br>

            <div class="mb-3">
               <input type="text" class="form-control" name="username" placeholder="Enter Your
                     Username" required>
            </div>
            <div class="mb-3">
               <input type="password" class="form-control" name="password" placeholder="Enter
                     Your Password" required>
```

```
</div>
            <input type="submit" value="Login" class="btn btn-primary">
            {% if messages %}
               <ul class="messages">
                  {% for message in messages %}
                     <li{% if message.tags %} class="{{ message.tags }}"{% endif %} style="list-
                        style-type: none; color:red;">{{ message }}</li>
                  {% endfor %}
               </ul>
            {% endif %}
         </form>
         <br>
         <a href="{%url 'party_register' %}" style="text-align:center; text-decoration:none;">Not
              Registered Yet? Register</a>
         <br>
         <a href="{% url 'index' %}" style="text-align:center; text-decoration:none;">Back To
              Home</a>
      </div>
   </div>


      <!-- Add Bootstrap JS and Popper.js -->
   <script src="https://cdn.jsdelivr.net/npm/popper.js@2.11.8/dist/umd/popper.min.js"
integrity="sha384-
I7E8VVD/ismYTF4hNIPjVp/Zjvgyol6VFvRkX/vR+Vc4jQkC+hVqc2pM8ODewa9r"
crossorigin="anonymous"></script>
   <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0/dist/js/bootstrap.min.js"></script>
</body>
</html>
```

Party_register.html

```
<!DOCTYPE html>
{% load static %}
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Voting System</title>

   <!-- Add Bootstrap CSS -->
   <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0/dist/css/bootstrap.min.css">
```

```
    <link rel="preconnect" href="https://fonts.googleapis.com">
    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
    <link
href="https://fonts.googleapis.com/css2?family=Nunito:wght@700&family=Poppins:wght@100;
300;400;600&family=Quicksand:wght@500&display=swap" rel="stylesheet">
    <script src="https://kit.fontawesome.com/6bf2dfa19e.js" crossorigin="anonymous"></script>

</head>
<body>
    <div class="container d-flex justify-content-center align-items-center min-vh-100">
        <div class="card p-4 shadow">
            <form method="POST">
                {% csrf_token %}


 <h3 style="text-align:center;">Party Registration</h3>
                <br>
                <div class="mb-3">
                    <input type="text" class="form-control" name="name" placeholder="Enter Your
                            Party Name" required>
                </div>
                <div class="mb-3">

  <label for="profile_pic">Party Symbol</label>
                    <input type="file" class="form-control-file" name="profile_pic" accept="image/*">
                </div>
                <div class="mb-3">
                    <input type="text" class="form-control" name="username" placeholder="Create
                  Username" required>
                </div>
                <div class="mb-3">
                    <input type="password" class="form-control" name="password"
                  placeholder="Create Password" required>
                </div>

                <input type="submit" value="Register" class="btn btn-primary">
                {% if messages %}
                    <ul class="messages">
                        {% for message in messages %}
                            <li{% if message.tags %} class="{{ message.tags }}"{% endif %} style="list-
                                style-type: none; color:red;">{{ message }}</li>
                        {% endfor %}
                    </ul>
    {% endif %}
            </form>
            <br>
            <a href="{%url 'party_login' %}" style="text-align:center; text-decoration:none;">Already
                Registered? Log In</a>
            <br>
            <a href="{% url 'index' %}" style="text-align:center; text-decoration:none;">Back To
                Home</a>

        </div>
    </div>
    <!-- Add Bootstrap JS and Popper.js -->
    <script src="https://cdn.jsdelivr.net/npm/popper.js@2.11.8/dist/umd/popper.min.js"
integrity="sha384-
I7E8VVD/ismYTF4hNIPjVp/Zjvgyol6VFvRkX/vR+Vc4jQkC+hVqc2pM8ODewa9r"


crossorigin="anonymous"></script>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0/dist/js/bootstrap.min.js"></script>
</body>
</html>
```

10.1.4 User

User_home.html

```
{% extends 'user/base.html' %}
{% load static %}
{% block content %}

<div class="container-fluid">
  <div class="card">
    <div class="card-body">

<h5 class="card-title fw-semibold mb-4">Welcome To Your Dashboard</h5>
      <p class="mb-0">{{ user }}
        {% if user.is_verified %}
          <img src="{% static 'verify.png' %}" alt="" height="15px">
        {% endif %}
      </p>
```

```
 <br>
      {% if user.is_verified %}
        <p>You are a verified voter</p>
      {% else %}
        <p>You are not a verified voter</p>
      {% endif %}



        {% if remaining_time_display %}
        <p id="remaining-time">Remaining Time: {{ remaining_time_display }}</p>
      {% endif %}
    </div>
  </div>
</div>

{% endblock %}
```

User_login.html

```
<!DOCTYPE html>
{% load static %}
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Voting System</title>

   <!-- Add Bootstrap CSS -->
   <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0/dist/css/bootstrap.min.css">

   <link rel="preconnect" href="https://fonts.googleapis.com">
   <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
   <link
href="https://fonts.googleapis.com/css2?family=Nunito:wght@700&family=Poppins:wght@100;
300;400;600&family=Quicksand:wght@500&display=swap" rel="stylesheet">


  <script src="https://kit.fontawesome.com/6bf2dfa19e.js" crossorigin="anonymous"></script>

</head>
```

```
body>
  <div class="container d-flex justify-content-center align-items-center min-vh-100">
    <div class="card p-4 shadow">
      <form method="POST">
        {% csrf_token %}
        <h3 style="text-align:center;">Student Login</h3>
        <br>

        <div class="mb-3">
          <input type="text" class="form-control" name="username" placeholder="Enter Your
          Username" required>
        </div>
        <div class="mb-3">

    <input type="password" class="form-control" name="password" placeholder="Enter
          Your Password" required>
        </div>
        <input type="submit" value="Login" class="btn btn-primary">
        {% if messages %}
          <ul class="messages">

  {% for message in messages %}
              <li{% if message.tags %} class="{{ message.tags }}"{% endif %} style="list-
                style-type: none; color:red;">{{ message }}</li>
            {% endfor %}
          </ul>
        {% endif %}
      </form>
      <br>
      <a href="{% url 'user_registration' %}" style="text-align:center; text-
          decoration:none;">Not Registered Yet? Register</a>
      <br>
      <a href="{% url 'index' %}" style="text-align:center; text-decoration:none;">Back To
          Home</a>

    </div>
  </div>
  <!-- Add Bootstrap JS and Popper.js -->
  <script src="https://cdn.jsdelivr.net/npm/popper.js@2.11.8/dist/umd/popper.min.js"
integrity="sha384-
I7E8VVD/ismYTF4hNIPjVp/Zjvgyol6VFvRkX/vR+Vc4jQkC+hVqc2pM8ODewa9r"
crossorigin="anonymous"></script>
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0/dist/js/bootstrap.min.js"></script>
```

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

User_register.html

```
<!DOCTYPE html>
{% load static %}
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Voting System</title>

  <!-- Add Bootstrap CSS -->

 <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0/dist/css/bootstrap.min.css">

  <link rel="preconnect" href="https://fonts.googleapis.com">
  <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>

  <link
href="https://fonts.googleapis.com/css2?family=Nunito:wght@700&family=Poppins:wght@100;
300;400;600&family=Quicksand:wght@500&display=swap" rel="stylesheet">
  <script src="https://kit.fontawesome.com/6bf2dfa19e.js" crossorigin="anonymous"></script>

</head>
<body>
  <div class="container d-flex justify-content-center align-items-center min-vh-100">
    <div class="card p-4 shadow">
      <form action="{% url 'user_registration' %}" method="POST">
        {% csrf_token %}
        <h3 style="text-align:center;">Student Registration</h3>
        <br>
        <div class="mb-3">
          <input type="text" class="form-control" name="name" placeholder="Enter Your Full
                Name" required>
        </div>
        <div class="mb-3">
          <input type="text" class="form-control" name="mobile_number" placeholder="Enter
                Your Mobile Number" required>
        </div>
        <div class="mb-3">
```

```
<input type="text" class="form-control" name="username" placeholder="Create
                    Username" required>
            </div>
            <div class="mb-3">
                <input type="password" class="form-control" name="password"
                    placeholder="Create Password" required>
            </div>
            <input type="submit" value="Register" class="btn btn-primary">
            {% if messages %}
                <ul class="messages">
                    {% for message in messages %}
                        <li{% if message.tags %} class="{{ message.tags }}"{% endif %} style="list-
                            style-type: none; color:red;">{{ message }}</li>
                    {% endfor %}
                </ul>
            {% endif %}
        </form>

      <br>
        <a href="{% url 'user_login' %}" style="text-align:center; text-decoration:none;">Already
            Registered? Log In</a>
        <br>
        <a href="{% url 'index' %}" style="text-align:center; text-decoration:none;">Back To
                Home</a>

    </div>
  </div>

  <!-- Add Bootstrap JS and Popper.js -->
  <script src="https://cdn.jsdelivr.net/npm/popper.js@2.11.8/dist/umd/popper.min.js"
integrity="sha384-
I7E8VVD/ismYTF4hNIPjVp/Zjvgyol6VFvRkX/vR+Vc4jQkC+hVqc2pM8ODewa9r"
crossorigin="anonymous"></script>
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0/dist/js/bootstrap.min.js"></script>
</body>
</html>
```

10.2 Backed

Admin.py

```
from django.contrib import admin
from . models import *

admin.site.register(User)
admin.site.register(Election)
admin.site.register(Participants)
# admin.py

from django.contrib import admin
from .models import Vote

class VoteAdmin(admin.ModelAdmin):
   list_display = ('election_id', 'election_name', 'election_date', 'encrypted_username',
'encrypted_party_name', 'encrypted_candidate_name')
   list_display_links = ('election_id', 'election_name')  # You can link certain fields to the change
  page if needed

   # Override the display value for the encrypted fields
   def encrypted_username(self, obj):
      return '********'  # Display whatever placeholder you want

   def encrypted_party_name(self, obj):
      return '********'

   def encrypted_candidate_name(self, obj):
      return '********'

   # You can also use this approach for other encrypted fields if needed

admin.site.register(Abc)
```

apps.py

```
from django.apps import AppConfig
class VotingappConfig(AppConfig):
   default_auto_field = 'django.db.models.BigAutoField'
   name = 'votingapp'
```

models.py

```
from django.db import models
from django.contrib.auth.models import AbstractUser
# Create your models here.
class User(AbstractUser):
   is_election = models.BooleanField('is election', default=False)
```

```
    is_party = models.BooleanField('is party', default=False)
    is_user = models.BooleanField('is user', default=False)
    is_candidate = models.BooleanField('is candidate', default=False)
    is_verified = models.BooleanField(default=False)

    name = models.CharField(max_length=100, default='User')
    mobile_number = models.CharField(max_length=20, default='Nil')
    district = models.CharField(max_length=100, default='Nil')
    state = models.CharField(max_length=100, default='Nil')
    pincode = models.CharField(max_length=20, default='Nil')
    address = models.TextField()

    party_name = models.CharField(max_length=100, default='Party')


    profile_pic = models.ImageField(upload_to='Photos', blank=True, null=True)
    aadhaar_front = models.ImageField(upload_to='Aadhaar Cards', blank=True, null=True)
    aadhaar_back = models.ImageField(upload_to='Aadhaar Cards', blank=True, null=True)
    voterid_front = models.ImageField(upload_to='VoterId Cards', blank=True, null=True)
    voterid_back = models.ImageField(upload_to='VoterId Cards', blank=True, null=True)

    @property
    def imageURL(self):
        try:
            url = self.profile_pic.url
        except:
            url = ''
        return  url

    @property
    def imageURL2(self):
        try:
            url = self.aadhaar_front.url
        except:
            url = ''
        return  url

    @property
    def imageURL3(self):
        try:
            url = self.aadhaar_back.url
        except:
            url = ''
        return  url
    @property
    def imageURL4(self):
        try:
            url = self.voterid_front.url
        except:
            url = ''
        return  url

    @property
    def imageURL5(self):
        try:
            url = self.voterid_back.url
        except:
            url = ''
        return  url


    def __str__(self):
        return self.username



class Election(models.Model):
    election_name = models.CharField(max_length=255, blank=True, null=True)
    election_date = models.DateField()

    party_name = models.CharField(max_length=255, blank=True, null=True)
    party_username = models.CharField(max_length=255, blank=True, null=True)
    party_id = models.IntegerField(blank=True, null=True)

    candidate_name = models.CharField(max_length=255, blank=True, null=True)
    candidate_username = models.CharField(max_length=255, blank=True, null=True)
    candidate_id = models.IntegerField(blank=True, null=True)
    location = models.CharField(max_length=100, default='Nil')

    number_of_votes = models.IntegerField(default=0, blank=True, null=True)
    result = models.CharField(max_length=255, blank=True, null=True)

    def __str__(self):
        return self.election_name


class Participants(models.Model):
    election_id = models.IntegerField(blank=True, null=True)
    election_name = models.CharField(max_length=255, blank=True, null=True)
    election_date = models.DateField()

    party_name = models.CharField(max_length=255, blank=True, null=True)
    party_username = models.CharField(max_length=255, blank=True, null=True)
    party_id = models.IntegerField(blank=True, null=True)

    candidate_name = models.CharField(max_length=255, blank=True, null=True)
    candidate_username = models.CharField(max_length=255, blank=True, null=True)
    candidate_id = models.IntegerField(blank=True, null=True)
    location = models.CharField(max_length=100, default='Nil')
    def __str__(self):
        return self.election_name

class Vote(models.Model):
    # Election information
    election_id = models.IntegerField(default=0, blank=True, null=True)
    election_name = models.CharField(max_length=255, blank=True, null=True)
    election_date = models.DateField()

    # User information
    encrypted_user_id = models.CharField(max_length=255, blank=True, null=True)
    encrypted_username = models.CharField(max_length=255, blank=True, null=True)

    # Party information
    encrypted_party_name = models.CharField(max_length=255, blank=True, null=True)
    encrypted_party_id =models.CharField(max_length=255, blank=True, null=True)

    # Candidate information
    encrypted_candidate_name = models.CharField(max_length=255, blank=True, null=True)
    encrypted_candidate_id = models.CharField(max_length=255, blank=True, null=True)

    def __str__(self):
        return self.election_name

class Abc(models.Model):
    # Election information
    election_id = models.IntegerField(default=0, blank=True, null=True)
    election_name = models.CharField(max_length=255, blank=True, null=True)
    election_date = models.DateField()

    # User information
    encrypted_user_id = models.IntegerField(default=0, blank=True, null=True)
    encrypted_username = models.CharField(max_length=255, blank=True, null=True)

    # Party information
    encrypted_party_name = models.CharField(max_length=255, blank=True, null=True)
    encrypted_party_id = models.IntegerField(default=0, blank=True, null=True)

    # Candidate information
    encrypted_candidate_name = models.CharField(max_length=255, blank=True, null=True)
    encrypted_candidate_id = models.IntegerField(default=0, blank=True, null=True)

    def __str__(self):
        return self.election_name
```

simple_facerec.py

```
import face_recognition
import cv2
import os
import glob
import numpy as np

class SimpleFacerec:
    def __init__(self):
        self.known_face_encodings = []
        self.known_face_names = []

        # Resize frame for a faster speed
        self.frame_resizing = 0.25

    def load_encoding_images(self, images_path):
        """
        Load encoding images from path
        :param images_path:
        :return:
        """
        # Load Images
        images_path = glob.glob(os.path.join(images_path, "*.*"))

        print("{} encoding images found.".format(len(images_path)))

        # Store image encoding and names
        for img_path in images_path:
            img = cv2.imread(img_path)
            rgb_img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

            # Get the filename only from the initial file path.
            basename = os.path.basename(img_path)
            (filename, ext) = os.path.splitext(basename)
            # Get encoding
            img_encoding = face_recognition.face_encodings(rgb_img)[0]

            # Store file name and file encoding
            self.known_face_encodings.append(img_encoding)
            self.known_face_names.append(filename)
        print("Encoding images loaded")

    def detect_known_faces(self, frame):
        small_frame = cv2.resize(frame, (0, 0), fx=self.frame_resizing, fy=self.frame_resizing)
        # Find all the faces and face encodings in the current frame of video
        # Convert the image from BGR color (which OpenCV uses) to RGB color (which
        face_recognition uses)
        rgb_small_frame = cv2.cvtColor(small_frame, cv2.COLOR_BGR2RGB)
        face_locations = face_recognition.face_locations(rgb_small_frame)
        face_encodings = face_recognition.face_encodings(rgb_small_frame, face_locations)

        face_names = []
        for face_encoding in face_encodings:
            # See if the face is a match for the known face(s)
            matches = face_recognition.compare_faces(self.known_face_encodings, face_encoding)
            name = "Unknown"

            # # If a match was found in known_face_encodings, just use the first one.
            # if True in matches:
            #     first_match_index = matches.index(True)
            #     name = known_face_names[first_match_index]

            # Or instead, use the known face with the smallest distance to the new face
            face_distances = face_recognition.face_distance(self.known_face_encodings,
                   face_encoding)
            best_match_index = np.argmin(face_distances)
            if matches[best_match_index]:
                name = self.known_face_names[best_match_index]
            face_names.append(name)

        # Convert to numpy array to adjust coordinates with frame resizing quickly
        face_locations = np.array(face_locations)
        face_locations = face_locations / self.frame_resizing
        return face_locations.astype(int), face_names
```

tests.py

```
from django.test import TestCase
# Create your tests here.
Urls.py
from django.urls import path
from . import views

urlpatterns = [
    path('', views.index, name='index'),
    path('user_register', views.user_register, name='user_register'),
    path('user_registration', views.user_registration, name='user_registration'),
    path('user_login', views.user_login, name='user_login'),
    path('user_home', views.user_home, name='user_home'),
    path('SignOut', views.SignOut, name='SignOut'),
    path('party_register', views.party_register, name='party_register'),
    path('party_login', views.party_login, name='party_login'),
    path('party_home', views.party_home, name='party_home'),
    path('view_upcoming_elections/', views.view_upcoming_elections,
            name='view_upcoming_elections'),
    path('view_election_rules/', views.view_election_rules, name='view_election_rules'),
    path('manage_candidates', views.manage_candidates, name='manage_candidates'),
    path('SignOut2', views.SignOut2, name='SignOut2'),
    path('election_login', views.election_login, name='election_login'),
    path('election_home', views.election_home, name='election_home'),
    path('SignOut3', views.SignOut3, name='SignOut3'),
    path('election_register', views.election_register, name='election_register'),
    path('conduct_election', views.conduct_election, name='conduct_election'),
    path('profile_update/<int:pk>/', views.profile_update, name='profile_update'),
    path('vote', views.vote, name='vote'),
    path('voting/<int:pk>/', views.voting, name='voting'),
    path('view_participants/<int:pk>/', views.view_participants, name='view_participants'),
    path('view_results/<int:pk>/', views.view_results, name='view_results'),
    path('delete_election/<int:pk>/', views.delete_election, name='delete_election'),

    path('remove_candidate/<int:pk>/', views.remove_candidate, name='remove_candidate'),
    path('manage_voters', views.manage_voters, name='manage_voters'),
    path('remove_voter/<int:pk>/', views.remove_voter, name='remove_voter'),
    path('verify_voter/<int:pk>/', views.verify_voter, name='verify_voter'),
    path('manage_parties', views.manage_parties, name='manage_parties'),

    path('verify_party/<int:pk>/', views.verify_party, name='verify_party'),
    path('remove_party/<int:pk>/', views.remove_party, name='remove_party'),
    path('face_recognition/', views.face_recognition, name='face_recognition'),
    path('face/', views.face, name='face'),
]
```

Utils.py

```
# from cryptography.fernet import Fernet
# # Generate a key for encryption and decryption
# key = 'jzNmyN5nS8Idxk3MDLdnbFME21dX8J3His842EsSkTw='
# cipher_suite = Fernet(key)
# # Encrypt data
# def encrypt_data(data):
#     encrypted_data = cipher_suite.encrypt(data.encode())
#     return encrypted_data
# # Decrypt data
# def decrypt_data(encrypted_data):
#     decrypted_data = cipher_suite.decrypt(encrypted_data).decode()
#     return decrypted_data
# from cryptography.fernet import Fernet

# # Generate a key for encryption and decryption
# key = b'jzNmyN5nS8Idxk3MDLdnbFME21dX8J3His842EsSkTw='  # Ensure the key is bytes
# cipher_suite = Fernet(key)

# # Encrypt data
# def encrypt_data(data):
#     encrypted_data = cipher_suite.encrypt(data.encode())
#     return encrypted_data

# # Decrypt data
# def decrypt_data(encrypted_data):
#     try:
#         decrypted_data = cipher_suite.decrypt(encrypted_data).decode()
#         return decrypted_data
#     except (Fernet.InvalidToken, ValueError) as e:
#         # Handle specific decryption errors
#         print(f"Decryption error: {str(e)}")
#         return None

from cryptography.fernet import Fernet, InvalidToken

# Generate a new key for decryption
decryption_key = 'jzNmyN5nS8Idxk3MDLdnbFME21dX8J3His842EsSkTw='
decryption_cipher_suite = Fernet(decryption_key.encode())  # Encode the key to bytes

# Encrypt data
def encrypt_data(data):
    encrypted_data = decryption_cipher_suite.encrypt(data.encode())
    return encrypted_data

# # Decrypt data
# def decrypt_data(encrypted_data):
#     try:
#         decrypted_data = decryption_cipher_suite.decrypt(encrypted_data).decode()
#         return decrypted_data
#     except (InvalidToken, ValueError) as e:  # Use InvalidToken directly
#         # Handle specific decryption errors
#         print(f"Decryption error: {str(e)}")
#         return None

def decrypt_data(encrypted_data):
    try:
        print(f"Attempting to decrypt: {encrypted_data}")
        decrypted_data = decryption_cipher_suite.decrypt(encrypted_data.encode()).decode()
        print(f"Decrypted data: {decrypted_data}")
        return decrypted_data
    except (InvalidToken, ValueError) as e:
        print(f"Decryption error: {str(e)}")
        return None
```

views.py

```
from django.shortcuts import render
from . models import *
from django.shortcuts import render, redirect, get_object_or_404
from django.contrib import messages
from django.contrib.auth import authenticate, login, logout
from cryptography.fernet import Fernet, InvalidToken
from .utils import decrypt_data
from django.utils.dateparse import parse_date
from django.http import HttpResponseBadRequest


from votingapp.utils import *

# Create your views here.
def index(request):
    return render(request, 'general/index.html')

def user_register(request):
    return render(request, 'user/user_register.html')

def user_registration(request):
    if request.method == 'POST':

        name = request.POST['name']
        mob = request.POST['mobile_number']
        username = request.POST['username']
        password = request.POST['password']

        if User.objects.filter(username=username).exists():
            messages.error(request, 'Username is already taken.')
            return render(request, 'user/user_register.html')

        if User.objects.filter(mobile_number=mob).exists():
            messages.error(request, 'Mobile number is already registered.')
            return render(request, 'user/user_register.html')

        user = User.objects.create_user(
            name=name,
            mobile_number=mob,
            username=username,
            password=password,
            is_user=True,
        )

        return redirect('user_login')

    return render(request, 'user/user_register.html')
def user_login(request):
    if request.method == 'POST':
        username_or_number = request.POST.get('username')
        password = request.POST.get('password')

        user = User.objects.filter(username=username_or_number).first()

        if user is not None and user.check_password(password) and user.is_user:
            login(request, user)
            return redirect('user_home')
        else:
            messages.error(request, 'Invalid login credentials.')

    return render(request, 'user/user_login.html')

from django.shortcuts import render
from cryptography.fernet import Fernet, InvalidToken

secret_key = 'jzNmyN5nS8Idxk3MDLdnbFME21dX8J3His842EsSkTw='
decryption_cipher_suite = Fernet(secret_key)

# Decrypt function
def decrypt_data(encrypted_data):
    try:
        # Print the encrypted data before decryption
        print(f"Encrypted data before decryption: {encrypted_data}")

        decrypted_data = decryption_cipher_suite.decrypt(encrypted_data.encode()).decode()

        # Print the decrypted data
        print(f"Decrypted data: {decrypted_data}")

        return decrypted_data
    except InvalidToken as e:
        # Handle InvalidToken exception
        print(f"InvalidToken exception: {str(e)}")
        return None
```

```
            except ValueError as e:
        # Handle other decryption errors
        print(f"Decryption error: {str(e)}")
        return None

# Your view function
from django.shortcuts import render, redirect
from django.contrib import messages
from django.contrib.auth import logout
from django.utils import timezone
from datetime import timedelta

def user_home(request):
    # Check if the user is authenticated
    if request.user.is_authenticated:
        # Start or resume the timer if it's not already running
        if 'login_time' not in request.session:
            request.session['login_time'] = timezone.now().timestamp()  # Store timestamp

        # Calculate remaining time
        login_timestamp = request.session['login_time']
        current_timestamp = timezone.now().timestamp()
        elapsed_time = current_timestamp - login_timestamp
        remaining_time = 10 * 60 - elapsed_time  # Remaining time in seconds

        # If remaining time is less than or equal to 0, logout the user
        if remaining_time <= 0:
            messages.error(request, 'Session timed out. Please log in again.')
            logout(request)
            return redirect('user_login')

        # Format remaining time for display
        remaining_minutes, remaining_seconds = divmod(int(remaining_time), 60)
        remaining_time_display = f"{remaining_minutes:02d}:{remaining_seconds:02d}"  # Convert
to string format

        # Pass remaining_time_display as a string to the template
        context = {'remaining_time_display': remaining_time_display}
        return render(request, 'user/user_home.html', context)
    else:
        # If the user is not authenticated, redirect to the login page
        return redirect('user_login')

def user_profile(request):
    return render(request, 'user/user_profile.html')
```

```
def profile_update(request,pk):
    get_profile = get_object_or_404(User, pk=pk)

    if request.method == "POST":
        new_name = request.POST['name']
        new_mob = request.POST['mob']
        new_profile_picture = request.FILES.get('profile_picture')
        aadhaar_front = request.FILES.get('aadhaar_front')
        aadhaar_back = request.FILES.get('aadhaar_back')
        voterid_front = request.FILES.get('voterid_front')
        voterid_back = request.FILES.get('voterid_back')

        new_state = request.POST['state']
        new_district = request.POST['district']
        new_address = request.POST['address']
        new_pincode = request.POST['pincode']

        if new_profile_picture:
            get_profile.profile_pic = new_profile_picture

        if aadhaar_front:
            get_profile.aadhaar_front = aadhaar_front

        if aadhaar_back:
            get_profile.aadhaar_back = aadhaar_back

        if voterid_front:
            get_profile.voterid_front = voterid_front

        if voterid_back:
            get_profile.voterid_back = voterid_back

        get_profile.name = new_name
        get_profile.pincode = new_pincode

        get_profile.mobile_number = new_mob
        get_profile.state = new_state

        get_profile.address = new_address
        get_profile.district = new_district

        get_profile.save()

    context={'get_profile':get_profile}

    return render(request, 'User/profile_update.html', context)
```

```
###face recognition
# views.py
from django.shortcuts import render, redirect
from django.contrib import messages
from .simple_facerec import SimpleFacerec
import cv2
import cv2

def face_recognition(request):
    # Initialize SimpleFacerec
    sfr = SimpleFacerec()
    sfr.load_encoding_images("images/")  # Replace "images/" with your image directory path

    # Open webcam and start face recognition
    cap = cv2.VideoCapture(0)

    recognized_face = False  # Initialize a variable to track if a face is recognized

    while True:
        ret, frame = cap.read()

        # Detect Faces
        face_locations, face_names = sfr.detect_known_faces(frame)
        for face_loc, name in zip(face_locations, face_names):
            y1, x2, y2, x1 = face_loc[0], face_loc[1], face_loc[2], face_loc[3]
            cv2.putText(frame, name, (x1, y1 - 10), cv2.FONT_HERSHEY_DUPLEX, 1, (0, 0, 200), 2)
            cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 0, 200), 4)

            # Set recognized_face to True if a face is recognized
            if name != "Unknown":
                recognized_face = True

        cv2.imshow("Frame", frame)

        key = cv2.waitKey(1)
        if key == ord('q'):  # Exit the loop if 'q' is pressed
            break

    cap.release()
    cv2.destroyAllWindows()
    if recognized_face:
        messages.success(request, 'Face recognized successfully!')
        return redirect('vote')  # Redirect to the voting page
    else:
        messages.error(request, 'Face not recognized. Unable to vote.')
        return render(request, 'user/vote.html')  # Stay on the voting page with a message
```

```
def face(request):
    return render(request,"user/facerec.html")

from django.utils import timezone

def vote(request):
    today = timezone.now().date()
    elections_today = Election.objects.filter(election_date=today)

    context = {'elections_today': elections_today}
    return render(request, 'user/vote.html', context)

from django.utils import timezone

def voting(request, pk):
    election = get_object_or_404(Election, pk=pk)
    user = request.user
    userid = user.pk
    username = user.username

    election_pk = election.pk
    election_name = election.election_name
    election_date = election.election_date

    participants = Participants.objects.filter(election_name=election_name)
    already_voted_message = None  # Initialize the message variable

    if request.method == "POST":
        candidate_name = request.POST.get("vote")


        # Check if the user has already voted on the specified date
        existing_vote = Abc.objects.filter(encrypted_user_id=userid, election_date=election_date)

        if not existing_vote.exists():
            # Use filter instead of get_object_or_404 to handle multiple results
            parties = Participants.objects.filter(candidate_name=candidate_name)

            if parties.exists():
                party = parties.first()  # Choose the first result, you can adjust this based on your logic
                party_name = party.party_name

                # Store encrypted data in the database
                Abc.objects.create(
                    election_id=election_pk,
                    election_name=election_name,
                    election_date=election_date,
                    encrypted_user_id=userid,
                    encrypted_username=username,

                    encrypted_party_name=party_name,
                    encrypted_candidate_name=candidate_name,
                )
            else:
                pass
        else:
            already_voted_message = "You have already voted in this election on the specified date."

    context = {'participants': participants, 'already_voted_message': already_voted_message}
    return render(request, 'user/voting.html', context)



def SignOut(request):
    logout(request)
    return redirect('user_login')

##################################################################################
##################################################################################

from django.shortcuts import render, redirect
from django.contrib import messages
from .models import User

def party_register(request):
    if request.method == 'POST':
        name = request.POST['name']
        username = request.POST['username']
        password = request.POST['password']
        profile_pic = request.FILES.get('profile_pic', None)

        if User.objects.filter(username=username).exists():
            messages.error(request, 'Username is already taken.')
            return render(request, 'party/party_register.html')

        user = User.objects.create_user(
            name=name,
            username=username,
            password=password,
            is_party=True,
            profile_pic=profile_pic,
        )
```

```
            return redirect('party_login')

        return render(request, 'party/party_register.html')

    def party_login(request):
        if request.method == 'POST':
            username_or_number = request.POST.get('username')
            password = request.POST.get('password')

            user = User.objects.filter(username=username_or_number).first()

            if user is not None and user.check_password(password) and user.is_party:
                login(request, user)
                return redirect('party_home')
            else:
                messages.error(request, 'Invalid login credentials.')

        return render(request, 'party/party_login.html')


    def party_home(request):
        return render(request, 'party/party_home.html')


    def manage_candidates(request):
        user = request.user
        partyname = user.name
        candidates = User.objects.filter(is_candidate = True, party_name=partyname)

        if request.method=="POST":
            candidate_name = request.POST.get("candidate_name")

            User.objects.create(
                name = candidate_name,
                username = candidate_name,
                party_name = partyname,
                is_candidate = True,
            )

        context = {'candidates':candidates}
        return render(request, 'party/manage_candidates.html', context)

    #party
    from django.shortcuts import render
    from .models import Election
    from django.utils import timezone
```

```
def view_upcoming_elections(request):
    # Query upcoming elections
    upcoming_elections = Election.objects.filter(election_date__gte=timezone.now().date())

    # Pass upcoming elections data to the template
    context = {'upcoming_elections': upcoming_elections}
    return render(request, 'view_upcoming_elections.html', context)
def view_election_rules(request):
    # Add logic here to fetch election rules data if needed
    return render(request, 'view_election_rules.html')

def SignOut2(request):
     logout(request)
     return redirect('party_login')

##########################################################################################
##########################################################################################

def election_register(request):

    if request.method == 'POST':

        username = request.POST['username']
        password = request.POST['password']

        if User.objects.filter(username=username).exists():
            messages.error(request, 'Username is already taken.')
            return render(request, 'election/election_register.html')


        user = User.objects.create_user(
            username=username,
            password=password,
            is_election=True,
        )

        return redirect('election_login')

    return render(request, 'election/election_register.html')

def election_login(request):

    if request.method == 'POST':
        username_or_number = request.POST.get('username')
        password = request.POST.get('password')
        user = User.objects.filter(username=username_or_number).first()

        if user is not None and user.check_password(password) and user.is_election:
            login(request, user)
            return redirect('election_home')
        else:
            messages.error(request, 'Invalid login credentials.')

    return render(request, 'election/election_login.html')

from django.db.models import Count

def election_home(request):
    # Get today's date
    election = Election.objects.all()
    today_date = timezone.now().date()

    # Filter Abc objects for today's date and annotate with the count of votes
    today_results = Abc.objects.filter(election_date=today_date).values(
        'encrypted_candidate_name'
    ).annotate(votes_count=Count('encrypted_candidate_name'))

    context = {
        'today_results': today_results,
        'today_date': today_date,
        'election':election
    }

    return render(request, 'election/election_home.html', context)



def SignOut3(request):
     logout(request)
     return redirect('election_login')

from datetime import datetime

def conduct_election(request):
    if request.method == "POST":
        election_name = request.POST.get('electionName')
        election_date_str = request.POST.get('electionDate')  # Assuming the date is in the format
YYYY-MM-DD
        election_location = request.POST.get('election_location')

        # Convert the date string to a datetime object
```

```
         election_date = datetime.strptime(election_date_str, '%Y-%m-%d').date()

            Election.objects.create (
                election_name=election_name,
                election_date=election_date,
                location=election_location,
            )

            return redirect('conduct_election')

        context = {'election': Election.objects.all()}
        return render(request, 'election/conduct_election.html', context)


    from datetime import datetime

    from datetime import datetime

    def view_participants(request, pk):
        election_details = Election.objects.all()
        parties = User.objects.filter(is_party=True)
        candidates = User.objects.filter(is_candidate=True)
        election = get_object_or_404(Election, pk=pk)
        election_name = election.election_name
        participants = Participants.objects.filter(election_name=election_name)

        if request.method == "POST":
            candidate_name = request.POST.get('candidate_name')
            election_name = request.POST.get('electionName')
            election_date_str = request.POST.get('election_date')
            election_location = request.POST.get('election_location')

            # Attempt to parse the date string
            try:
                election_date = datetime.strptime(election_date_str, '%B %d, %Y').date()
            except ValueError:
                return HttpResponseBadRequest("Invalid date format. Please use the format 'Month Day,
                        Year', e.g., 'March 7, 2024'.")

            party_name = get_object_or_404(User, name=candidate_name)
            party = party_name.party_name

            Participants.objects.create(
                election_name=election_name,
                election_date=election_date,
                location=election_location,
                party_name=party,
                candidate_name=candidate_name
            )

        context = {
            'election': election,
            'election_details': election_details,
            'parties': parties,
            'candidates': candidates,
            'participants': participants,
        }

        return render(request, 'election/view_participants.html', context)

    from django.shortcuts import render, get_object_or_404
    from django.utils import timezone

    def view_results(request, pk):
        election = get_object_or_404(Election, pk=pk)
        votes = Abc.objects.filter(election_id=election.pk)

        # Create a dictionary to store the count of votes for each candidate
        candidate_votes_count = {}

        for vote in votes:
            candidate_name = vote.encrypted_candidate_name

            # If the candidate_name is already in the dictionary, increment the count; otherwise, set it to 1
            candidate_votes_count[candidate_name] = candidate_votes_count.get(candidate_name, 0) + 1

        # Determine the winner only if the current date is greater than the election date
        today_date = timezone.now().date()

        winner = None  # Default value for winner
        if today_date > election.election_date:
            winner = max(candidate_votes_count, key=candidate_votes_count.get) if
    candidate_votes_count else None

        election.result = winner
        election.save()

        context = {
            'election': election,
            'candidate_votes_count': candidate_votes_count.items(),  # Convert dictionary items to a list
    of tuples
            'winner': winner,
        }
        return render(request, 'election/view_results.html', context)
    def delete_election(request,pk):
        eleciton = get_object_or_404(Election, pk=pk)
        eleciton.delete()
        return redirect('conduct_election')

    def remove_candidate(request,pk):
        eleciton = get_object_or_404(Participants, pk=pk)
        eleciton.delete()
        return redirect('conduct_election')

    def manage_voters(request):
        voters = User.objects.filter(is_user=True)
        context = {'voters':voters}
        return render(request, 'election/manage_voters.html', context)

    def remove_voter(request,pk):
        eleciton = get_object_or_404(User, pk=pk)
        eleciton.delete()
        return redirect('manage_voters')

    from django.shortcuts import get_object_or_404, redirect

    def verify_voter(request, pk):
        election = get_object_or_404(User, pk=pk)
        election.is_verified = not election.is_verified  # Toggle the value
        election.save()
        return redirect('manage_voters')

    def manage_parties(request):
        voters = User.objects.filter(is_party=True)
        context = {'voters':voters}
        return render(request, 'election/manage_parties.html', context)

    def verify_party(request, pk):
        election = get_object_or_404(User, pk=pk)
        election.is_verified = not election.is_verified  # Toggle the value
        election.save()
        return redirect('manage_parties')

    def remove_party(request, pk):
        party = get_object_or_404(User, pk=pk)
        party.delete()
        return redirect('manage_parties')
```

manage.py

```
#!/usr/bin/env python
"""Django's command-line utility for administrative tasks."""
import os
import sys


def main():
    """Run administrative tasks."""
    os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'voting.settings')
    try:
        from django.core.management import execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed and "
            "available on your PYTHONPATH environment variable? Did you "
            "forget to activate a virtual environment?"
        ) from exc
    execute_from_command_line(sys.argv)


if __name__ == '__main__':
    main()
```

11. FUTURE WORKS

In the future, the development and enhancement of a college e-voting system with face recognition technology hold significant potential for fostering a more inclusive and efficient democratic process on campus. Firstly, expanding the system's capabilities to accommodate a larger student population is crucial. This entails refining the face recognition algorithm to improve accuracy and scalability, allowing for seamless authentication of a greater number of students during peak voting periods. Moreover, ensuring robust security measures to safeguard against unauthorized access or tampering is paramount in maintaining the integrity of the voting process.

Secondly, prioritizing improvements to the system's user interface and accessibility features can greatly enhance the voting experience for all students, including those with disabilities. By developing a more intuitive and user-friendly interface, incorporating options for alternative voting methods such as voice commands or text-to-speech technology, and ensuring compatibility with assistive devices, the system can effectively cater to diverse needs and promote equal participation among students.

Additionally, integrating the e-voting system with other campus systems and platforms can significantly streamline administrative processes and enhance overall efficiency. This integration could involve connecting with student information systems to automatically verify student eligibility and update voter rolls, as well as linking with student organizations' platforms to facilitate candidate nominations and campaign activities. Furthermore, implementing mechanisms for real-time monitoring and reporting of voting metrics can provide valuable insights for assessing voter turnout and engagement, enabling stakeholders to make informed decisions for future enhancements.

By addressing these areas of improvement, the college e-voting system can evolve into a robust, inclusive, and user-friendly tool for facilitating democratic participation among students on campus. Embracing technological advancements and adopting a user-centric approach will be key in harnessing the full potential of e-voting technology to empower the student body and strengthen the democratic process within educational institutions.

12. LITERATURE SURVEY

Electronic voting systems have revolutionized democratic processes, offering unparalleled advantages over traditional paper-based methods. These systems, facilitated through online portals, mobile apps, and electronic kiosks, significantly enhance accessibility and convenience for voters, thereby fostering greater inclusivity and participation in elections. However, challenges such as security vulnerabilities, privacy concerns, and the imperative for robust authentication mechanisms remain significant barriers to widespread adoption.

The integration of face recognition technology into e-voting systems has emerged as a promising solution to enhance security and authentication processes. Leveraging techniques ranging from traditional Eigenfaces and Fisherfaces to modern deep learning-based Convolutional Neural Networks (CNNs), researchers have strived to achieve high accuracy in facial recognition tasks. Nevertheless, the successful deployment of such systems requires careful consideration of design aspects, implementation challenges, and adherence to legal and ethical standards.

In navigating the future of e-voting systems with face recognition technology, researchers must prioritize improvements in algorithm accuracy, address privacy concerns through advanced encryption methods, and develop user-friendly interfaces. Continuous monitoring of system vulnerabilities, regular audits of facial recognition algorithms, and collaboration with stakeholders are essential steps towards advancing the security, transparency, and accessibility of e-voting platforms. By addressing these challenges and opportunities, researchers can contribute significantly to the development of robust and trustworthy e-voting solutions, not only for educational institutions but also for broader democratic processes.

13. CONCLUSION

In conclusion, the development and implementation of a college e-voting system represent a significant step towards modernizing and streamlining the electoral process within educational institutions. Throughout this project, we have explored the various aspects of electronic voting systems, including their advantages, challenges, and integration with advanced technologies such as face recognition.The college e-voting system offers numerous benefits, including increased accessibility, convenience, and transparency. By enabling students to cast their votes remotely through online portals or mobile applications, the system enhances participation and inclusivity in campus elections. Additionally, it reduces the administrative burden associated with traditional paper-based voting methods, leading to more efficient and cost-effective elections.However, the deployment of an e-voting system comes with its own set of challenges, particularly in terms of security, privacy, and voter authentication. Addressing these challenges requires robust security measures, stringent data protection protocols, and reliable authentication mechanisms to safeguard the integrity of the electoral process.Integration of face recognition technology into the e-voting system adds an extra layer of security and authentication, enhancing the overall reliability and trustworthiness of the platform. By leveraging facial biometrics for voter identification, the system can mitigate the risk of fraudulent activities and ensure the integrity of election results.In conclusion, the college e-voting system represents a significant advancement in campus governance, providing students with a convenient and secure platform to exercise their democratic rights. Moving forward, continuous monitoring, evaluation, and improvement of the system are essential to address emerging challenges and ensure its effectiveness in facilitating fair and transparent elections within the college community. Through ongoing collaboration between stakeholders and adherence to best practices in electoral management, the college e-voting system can serve as a model for democratic engagement and governance in educational institutions

14.REFERENCE

Abdul, H. M., Fadzil, M. H. M., & Zakaria, Z. (2017). "Development of a Secure Online Voting System (SOVS) Using Fingerprint Biometric Authentication." In 2017 7th IEEE International Conference on Control System, Computing and Engineering (ICCSCE) (pp. 107-111). IEEE.

Behrang, A., & Rahmani, A. M. (2020). "Blockchain-Based E-Voting System: A Systematic Review." In 2020 IEEE International Conference on Blockchain and Cryptocurrency (ICBC) (pp. 1-6). IEEE.

Healy, J. M., & Shamir, M. (2019). "Blockchain-Based E-Voting System: A Systematic Review." In Proceedings of the 9th International Conference on Cloud Computing and Services Science (CLOSER 2019) (Vol. 1, pp. 269-276).

Kavitha, M., & Kumar, S. M. (2021). "A Survey on E-Voting System Using Blockchain Technology." In 2021 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT) (pp. 1-6). IEEE.

Malik, S. S., & Abbasi, A. (2020). "Design and Development of a Secure E-Voting System." In 2020 International Conference on Computing, Electronics & Communications Engineering (iCCECE) (pp. 1-5). IEEE.

Mishra, P., & Sharma, A. (2021). "Design and Implementation of Secure E-Voting System Using Blockchain." In 2021 International Conference on Data Science and Business Analytics (ICDSBA) (pp. 1-5). IEEE.

Shaikh, F., & Gadge, P. (2019). "Secure E-Voting System Using Biometric Authentication." In 2019 International Conference on Intelligent Computing and Control Systems (ICICCS) (pp. 1489-1493). IEEE.

Sundar, A., & Priyadarshini, K. (2021). "A Review on Electronic Voting System." In 2021 International Conference on Artificial Intelligence and Sustainable Computing (ICAISC) (pp. 1-5). IEEE.

Yadav, D., & Kaur, S. (2018). "Design and Implementation of Online Voting System Using Blockchain Technology." In 2018 4th International Conference on Computing Sciences (ICCS) (pp. 65-70). IEEE.

Zhang, X., & Feng, W. (2019). "Blockchain Based E-Voting System: A Survey." In 2019 2nd International Conference on Advances in Electronics, Computers and Communications (ICAECC) (pp. 1-7). IEEE.