E-Voting Web Application with Django

Table of Contents

[1. Introduction 4](#_Toc123149408)

[1.1 General Information 4](#_Toc123149409)

[1.2 Purpose 4](#_Toc123149410)

[1.3 Document Conventions 4](#_Toc123149411)

[1.4 Project Objective 5](#_Toc123149412)

[1.5 Intended Audience and Reading Suggestions 5](#_Toc123149413)

[1.6 Project Scope 6](#_Toc123149414)

[2. Overall Description 7](#_Toc123149415)

[2.1 Product Perspective 7](#_Toc123149416)

[2.2 Product Features 7](#_Toc123149417)

[2.3 User Classes and Characteristics 9](#_Toc123149449)

[2.4 Operating Environment 10](#_Toc123149450)

[2.5 Design and Implementation Constraints 10](#_Toc123149451)

[2.6 User Documentation 10](#_Toc123149452)

[2.7 Assumptions and Dependencies 10](#_Toc123149453)

[3. System Features 10](#_Toc123149454)

[4. External Interface Requirements 13](#_Toc123149455)

[4.1 User Interfaces 13](#_Toc123149456)

[4.2 Hardware Interfaces 13](#_Toc123149457)

[4.3 Software Interfaces 13](#_Toc123149458)

[4.4 Communications Interfaces 13](#_Toc123149459)

[5. Other Nonfunctional Requirements 14](#_Toc123149460)

[5.1 Performance Requirements 14](#_Toc123149461)

[5.2 Safety Requirements 14](#_Toc123149462)

[5.3 Security Requirements 14](#_Toc123149463)

[5.4 Software Quality Attributes 14](#_Toc123149464)

[6. Analysis model 15](#_Toc123149465)

[***6.1*** *Use Case Diagram* 15](#_Toc123149466)

[**6.2** Activity Diagram 16](#_Toc123149467)

[**6.3** Sequence Diagram 17](#_Toc123149468)

[**6.4** Database Diagram 18](#_Toc123149469)

[**6.5** Object Diagram 19](#_Toc123149470)

[**6.6** Class Diagram 20](#_Toc123149471)

[7. Tools & Technologies 21](#_Toc123149472)

[7.1 Programming Languages 21](#_Toc123149473)

[7.2 Databases/Data storages 21](#_Toc123149474)

[7.3 Operating System 21](#_Toc123149475)

[Appendix A: Glossary 22](#_Toc123149476)

Code:............................................................................................................................................................................

Django Example manage.py.........................................................................................................................................

# Introduction

## General Information

***Project Name:*** *E-Voting* Web Application with Django

***Starting Date:*** *28/12/2022*

***Controlling Agency:***

***Final Date:***

***Prepared By:***

*xxxxx*

***Authorized by:***

## Purpose

The motivation behind this document is to give a detailed portrayal of the project. This document will assist us with making sense of the extent of the item, framework highlights, and connection points of the framework to users and clients. Besides it will make sense of how will the framework respond and under what imperatives or conditions the framework will work appropriately.

## Document Conventions

|  |  |
| --- | --- |
| **Document alignment** | • Entire document is justified. |
| **Convention for heading 1** | • Font face: Times New Roman  • Font style: Bold  • Font size: 16 |
| **Convention for heading 2** | • Font face: Times New Roman  • Font style: Bold  • Font size: 14 |
| **Convention for heading 3** | • Font face: Times New Roman  • Font style: Bold, Italics  • Font size: 13 |
| **Convention for body** | • Font face: Times New Roman  • Font size: 12 |

## Project Objective

The principal objective of the undertaking is to configure, create, and carry out a framework that will computerize everyday tasks and exchanges of a game store and that will be able to achieve the following objectives:

* To implement robust user authentication and authorization mechanisms to ensure that only eligible voters can participate in the election process.
* To integrate OTP (One-Time Password) verification to authenticate voters' identities and enhance security.
* To manage detailed voter profiles, including personal information, phone numbers for OTP, and voting status.
* To create and administer multiple election positions with specific attributes such as name, maximum votes, and priority.
* To facilitate the easy addition and management of candidates, including their profiles, photos, and bios.
* To enable voters to cast their votes for candidates running for different positions in a user-friendly manner.
* To record and track votes accurately, ensuring data integrity and preventing tampering or duplication.
* To generate real-time election results, providing transparency and immediate feedback on election outcomes.
* To ensure data privacy and security, protecting sensitive voter and election information from unauthorized access.
* To incorporate administrative features that allow for the oversight and control of the election process by authorized personnel.
* To provide a scalable and flexible architecture that can handle varying numbers of voters and election positions.
* To develop an intuitive and user-friendly interface that simplifies the voting process for users.
* To reduce the risk of election fraud through comprehensive security measures and verification processes.
* To ensure compliance with relevant legal and regulatory requirements related to online voting and data protection.

## Intended Audience and Reading Suggestions

The target audiences for this document incorporate the development group, the employees of the university who will assess this item, the project supervisor and the Co-supervisor. This report will be surveyed regularly by the above crowds to check the various periods of the task are being finished by meeting the given necessities. Assuming that there are any progressions in the prerequisites of the venture they should be included in this report by rolling out the essential improvements. Readers ought to have a comprehension of the project so first, they need to go through the scope section and afterward continue to features.

## Project Scope

The scope of this project is to develop a comprehensive and secure E-voting management system that facilitates seamless online elections. The system will include robust user authentication and authorization mechanisms to ensure only eligible voters can participate. It will incorporate OTP-based verification for enhanced security, and manage detailed voter profiles, including personal information and voting status. The project will support the creation and administration of multiple election positions and candidates, enabling voters to cast their votes easily and accurately. Real-time election results will be generated to provide transparency and immediate feedback. Data privacy and security will be prioritized to protect sensitive information. The system will also include administrative features for election oversight and control, ensuring a scalable and flexible architecture that can handle varying numbers of voters and election positions. The user interface will be designed to be intuitive and user-friendly, reducing the risk of election fraud and ensuring compliance with relevant legal and regulatory requirements.

# Overall Description

## Product Perspective

The E-voting management system is designed as a modern, web-based application that leverages advanced technologies to provide a secure and efficient platform for conducting elections. It is intended to replace traditional paper-based voting systems, offering a more streamlined and accessible voting process. The system integrates seamlessly with existing IT infrastructure, ensuring compatibility and ease of deployment. It is built on a scalable architecture that can accommodate a growing number of users and election events. By incorporating user-friendly interfaces and robust security measures, the system aims to enhance voter confidence and participation. The E-voting management system also includes administrative tools for managing voter registration, candidate information, and election results, providing a comprehensive solution for election management. Overall, the product aims to modernize the voting process, making it more transparent, efficient, and secure.

## Features

## User Classes and Characteristics

The E-voting management system is designed to cater to various user classes, each with distinct characteristics and roles. The primary user classes include administrators, voters, and candidates. Administrators are responsible for managing the entire election process, including voter registration, candidate management, and result tabulation. They require a comprehensive set of tools and access to detailed system functionalities to ensure the smooth operation of elections. Voters are individuals who participate in the elections by casting their votes. They need a simple, intuitive interface to register, verify their identity through OTP, and vote for their preferred candidates. Candidates are individuals running for election positions, and they need the ability to manage their profiles, including uploading photos and bios. Each user class has specific needs and access levels, and the system is designed to provide an optimized experience tailored to their roles and responsibilities, ensuring efficiency, security, and ease of use for all participants.

## Development Methodology

### 1. Waterfall Model

For the development of the **E-Voting System**, we adopted the **Waterfall Model**, which follows a linear and sequential approach. This model was chosen due to the structured nature of the project and the importance of clearly defined requirements.

#### Phases in the Waterfall Model:

1. **Requirement Analysis**:
   * The first phase involved gathering detailed requirements for the E-Voting System, such as the functionalities for voters, admins, and candidates. Security and scalability requirements were also identified at this stage.
2. **System Design**:
   * The next phase involved designing the architecture of the system, including the database schema, user interface design, and overall flow of the application. Attention was given to security protocols like encryption and SSL for protecting sensitive data.
3. **Implementation**:
   * During the implementation phase, the team developed the application using the Django framework for the backend and JavaScript for frontend interactivity. Key functionalities such as user authentication, voting, and result publication were implemented.
4. **Testing**:
   * Rigorous testing was carried out to ensure the system functions correctly. This included unit tests for individual components, integration tests for system workflows, and stress testing to ensure the platform could handle high traffic during election periods.
5. **Deployment**:
   * After successful testing, the E-Voting System was deployed to a live server. It is now accessible to registered users and ready for elections.
6. **Maintenance**:
   * The system is monitored regularly for any issues or bugs. Maintenance includes applying software updates, adding new features, and ensuring security measures are up-to-date.

### 2. Justification for Waterfall Model

The Waterfall Model was chosen due to its structured nature and the need for well-defined milestones. Elections require strict deadlines, and the Waterfall approach allowed us to plan, execute, and deliver the project in a timely manner with minimal risk.

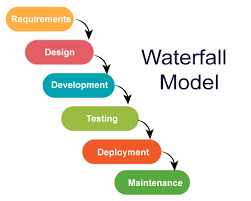
The **water fall diagram** is basically divided into following 5 models.

• Requirement

• Design

• Implementation

• Verification

• Maintenance

Water fall Model figure 3. 1

**Disadvantages:**

• It is difficult for the customers to state the requirements clearly at the beginning. There is always certain degree of natural uncertainty at beginning of each project.

• Difficult and costlier to change when the changes occur at later stages.

• Customer can see the working version only at the end. Thus, any changes suggested here are not only difficult to incorporate but also expensive. This may result in disaster if any undetected problems are precipitated at this stage

**Justification:**

• We have implemented Incremental model because of following features: • The Design phase goes much faster, as designs are only done on the items in the current release (Release 1.0 for example).

• Coding and Testing go much faster because there are less items to code and test. If major design flaws are found, re-work is much faster since the functional areas have been greatly reduced.

• The client gets into production in less than 3 months, allowing them to begin earning revenue or reducing expenses quicker with their product.

• If market conditions change for the client, changes can be incorporated in the next iterative release, allowing the software to be much more nimble.

• As the software is implemented, the client can make recommendations for the next iteration due to experiences learned in the past iteration.

• Deliverables are produced early in the software development lifecycle in each iteration.

• It is flexible and easy to manage

• Risk Management and Testing is easy.

**Milestones:**

Sometimes there are events externals to your project that you want to track. If you cannot link to them because they are not in a Project Plan, you can create a milestone to represent them in your own project. For example, you cannot begin a certain task until another company completes a software application you need to use. You can create a milestone in your project that represents the completion of that application and reminds you to track its progress. Some milestones may need duration. For example, your project has an approval milestone at the end of a phase and you know that the approval process will take a week.

## 3.4 When are You Doing

Project Plan Project planning establishes a plan for the software engineering work that follows. It describes the technical tasks to be conducted, the risks that are likely, the recourses that will be required, the work product to be produced, and a work schedule.

**Roles and Responsibility**

We members of project have performed all the activities related to the analysis, design, documentation, coding, and testing from start to end collectively by consensus.

**Analysis:**

Understanding the requirements is among the most difficult tasks that face a software engineer. As far as our project is concern, we had studied various projects related to administration. We also studied projects previously developed in this organization We have analyzed this system for One Twenty days.

**Designing**

Design engineering encompasses the set of principles concepts and practices that lead to the development of high-quality system or product design principles establishes an overriding philosophy that guides designer in the work that is performed. Software design sits at the technical kernel of software engineering and is applied regardless of the software process model that is used. Beginning once software requirements have been analyzed and modeled, software design is the last software engineering action within the modeling activity and sets the stage for construction. As far as our project is concerned, we have designed the process model, data model and also viewer model that will be going to be implemented in the coding phase. To make efficient development model we have developed various diagrams like use case diagram, to understand the user interactions, flow of data through the information system respectively

**Coding**

On completion of design phase, we started coding in .Net 2005 express edition. • In case of responsibility.

**Testing**

Software testing has a dual function; it is used to establish the presence of defects in the program and it is used to help judge whether or not the program is usable in practice. Thus, software testing is use for validation and verification, which ensure that software confirms to its specification and meets the need of the software customer. We have tested almost all interface components together and almost all tests have been succeeded. Testing has been applied for all the stages of process.

## Operating Environment

This application solely a web based application. All module are web based. **Windows** is used as Operating system. This project is implemented using **python with Django 5.0 framework.**

## Design and Implementation Constraints

Any browser can be used to run the modules. **Adobe Photoshop** and **Figma** is utilized. The Django system is utilized for the development of web applications. **Python** is utilized as the principal language for development as Programming lanaguage for the logic purposes or of run. This is purely a **web-based application.**

## User Documentation

The user documentation will not be provided.

## Assumptions and Dependencies

* **Users have knowledge of using web-based applications:**

1. It is assumed that the users are familiar with navigating and interacting with web-based applications.
2. Users should be comfortable with basic web functionalities such as logging in, browsing through different sections, and performing tasks online.

* **Users understand English:**

1. The application will be designed with English as the primary language.
2. Users are expected to have a basic understanding of English to interact with the application and comprehend instructions, notifications, and messages.

* **Users want a clear picture of the product:**

1. The system will be deployed in an environment with reliable internet connectivity.
2. Users will have access to devices capable of running modern web browsers.
3. Users are familiar with basic web navigation and online authentication processes.
4. Administrators will have the necessary training to manage the system effectively.
5. The system will comply with relevant legal and regulatory requirements for online voting.
6. Adequate security measures will be in place to protect user data and prevent fraud.

# System Features

| **Module** | **Features** | **Prerequisites** |
| --- | --- | --- |

|  |  |  |
| --- | --- | --- |
| **User Authentication** | - Secure login and  logout  - OTP-based verification  - Password recovery | - Email service for OTP  - Secure database for user credentials |

|  |  |  |
| --- | --- | --- |
| **User Management** | - Create and manage user profiles  - Role-based access control | - Admin access  - User registration form |

|  |  |  |
| --- | --- | --- |
| **Voter Management** | - Voter registration-  Profile management-  Voting status tracking | - Admin access  - User authentication |

|  |  |  |
| --- | --- | --- |
| **Candidate Management** | - Add and edit candidate profiles | - Admin access  - File storage system |

|  |  |  |
| --- | --- | --- |
| **Election Setup** | - Create and configure elections  - Define positions and candidates | - Admin access  - User and candidate management |

|  |  |  |
| --- | --- | --- |
| **Voting Process** | - Cast votes securely  - Real-time vote validation | - User authentication  - Election setup |

|  |  |  |
| --- | --- | --- |
| **Result Tabulation** | - Real-time result generation  - Display of election outcomes | - Voting process completed  - Secure database |

|  |  |  |
| --- | --- | --- |
| **Security** | - Data encryption  - Fraud detection mechanisms | - Secure server environment  - Regular security audits |

|  |  |  |
| --- | --- | --- |
| **Administrative Tools** | - Dashboard for system overview  - Election monitoring and control | - Admin access  - Comprehensive training |

|  |  |  |
| --- | --- | --- |
| **User Interface** | - Intuitive and user-friendly design  - Multi-device compatibility | - Modern web browsers  - Responsive design |

|  |  |  |
| --- | --- | --- |
| **Reporting** | - Generate detailed reports  - Export data in various formats | - Completed elections  - Admin access |

|  |  |  |
| --- | --- | --- |
| **Maintenance** | - Regular updates  - System backup and recovery | - Technical support team  - Backup storage solutions |

# External Interface Requirements

## User Interfaces

The system will feature an intuitive and user-friendly web interface accessible via modern web browsers. It will include:

1. **Login and Dashboard:** Secure login for administrators and voters, with a dashboard providing an overview of election activities.
2. **Voter Interface:** A straightforward interface for voters to register, verify their identity via OTP, view candidate profiles, and cast their votes securely.
3. **Administrative Tools:** Tools for administrators to manage users, candidates, elections, and view real-time election results.

## Software Interfaces

The system will integrate with:

1. **Database System:** Utilizing a secure database (e.g., MySQL, PostgreSQL) to store user credentials, election data, and voting records.
2. **OTP Service Provider:** Integration with an OTP service provider for user verification during the voting process.
3. **File Storage System:** Integration for storing candidate photos and other media assets securely.

## Communications Interfaces

The system will require:

1. **Internet Connectivity:** Reliable internet access for users to interact with the voting system and for real-time data synchronization.
2. **Email Service:** Integration with an email service provider for sending OTPs, notifications, and system alerts securely.

# Other Nonfunctional Requirements

## Performance Requirements

1. **Scalability:** The system should handle a large number of concurrent users during peak voting periods without performance degradation.
2. **Response Time:** Ensure quick response times for user interactions (e.g., login, vote casting) to provide a seamless voting experience.
3. **Reliability:** Minimize downtime with robust backup and recovery procedures to maintain system availability.

## Safety Requirements

1. **User Privacy:** Protect user data and voting information through encryption and secure data handling practices.
2. **System Integrity:** Implement measures to prevent tampering with election results and ensure the accuracy of vote counting.

## Security Requirements

1. **Access Control:** Role-based access control (RBAC) to restrict system access based on user roles (admin, voter, candidate).
2. **Authentication:** Strong authentication mechanisms, including OTP verification, to ensure only authorized users can access the system.
3. **Audit Trails:** Maintain comprehensive audit trails of user activities and system events for accountability and traceability.

## Software Quality Attributes

1. **Usability:** Design an intuitive and user-friendly interface to facilitate ease of use for all system users.
2. **Maintainability:** Structure codebase and system architecture to support easy maintenance, updates, and enhancements.
3. **Portability:** Ensure compatibility with different operating systems and devices to maximize accessibility for all users.

# SYSTEM IMPLEMENTATIONS

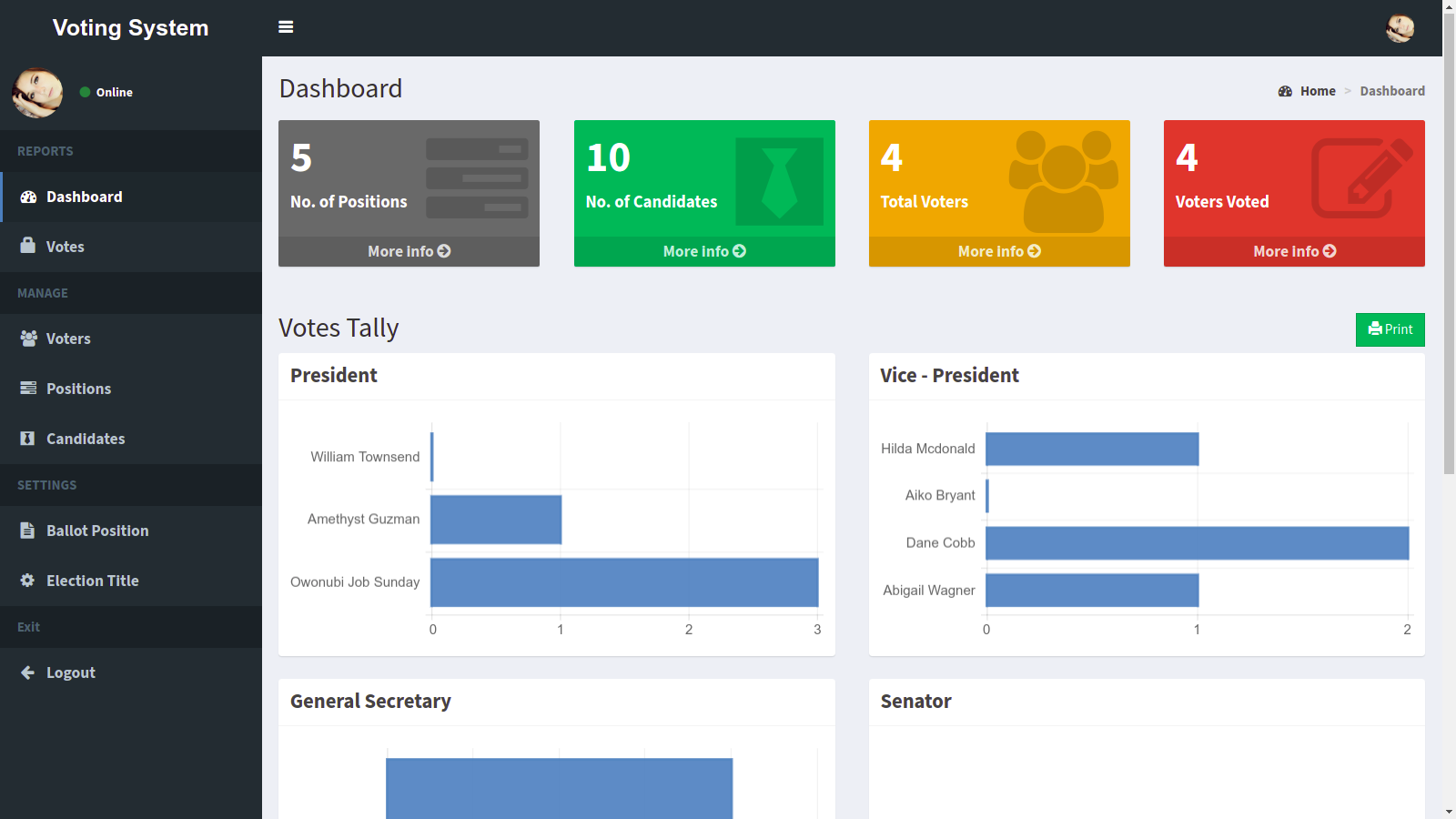
## 5.1. System Implementation

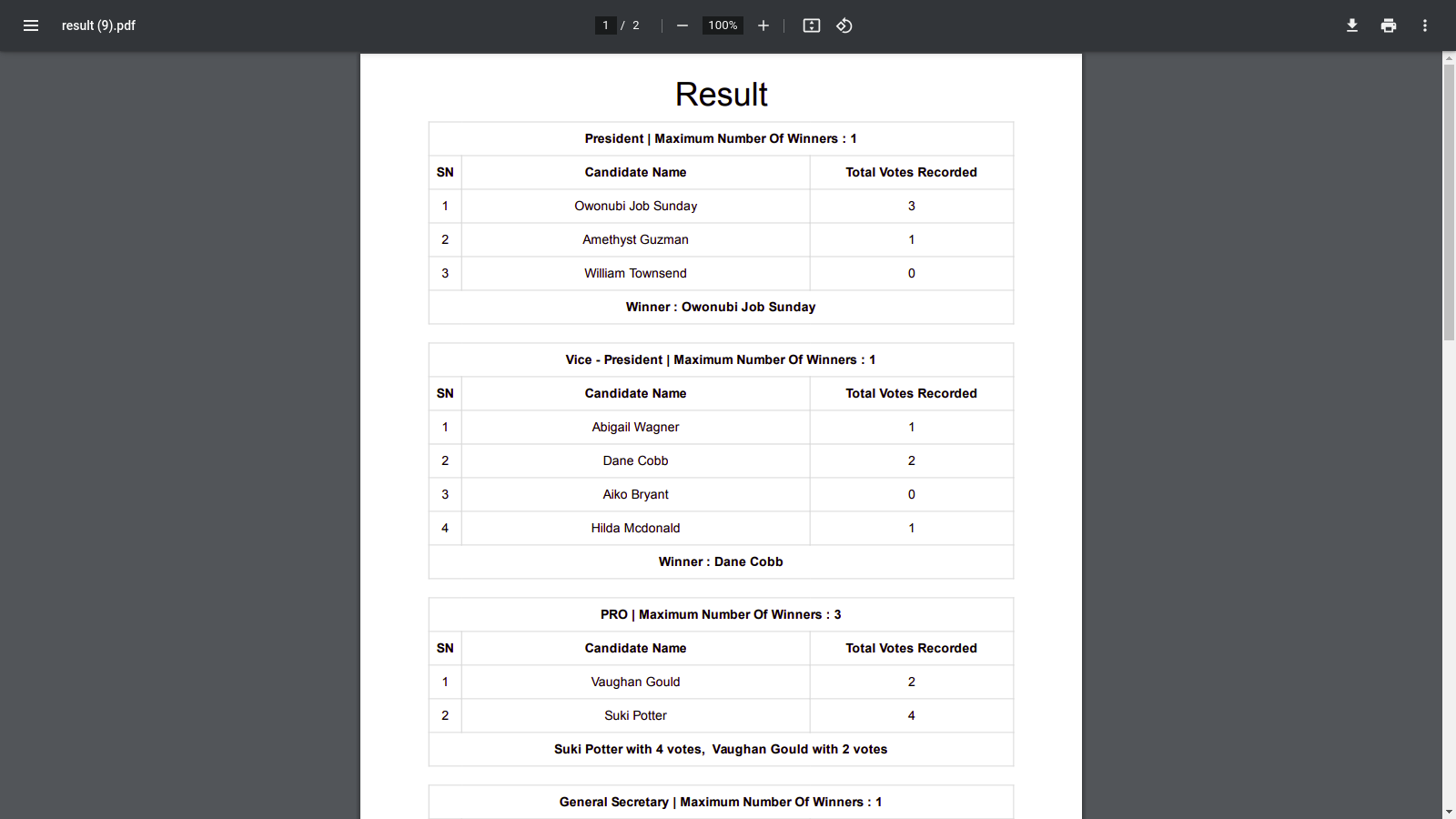
Finally, after all phases has been done, the website has been implemented to the local server and it has been used.

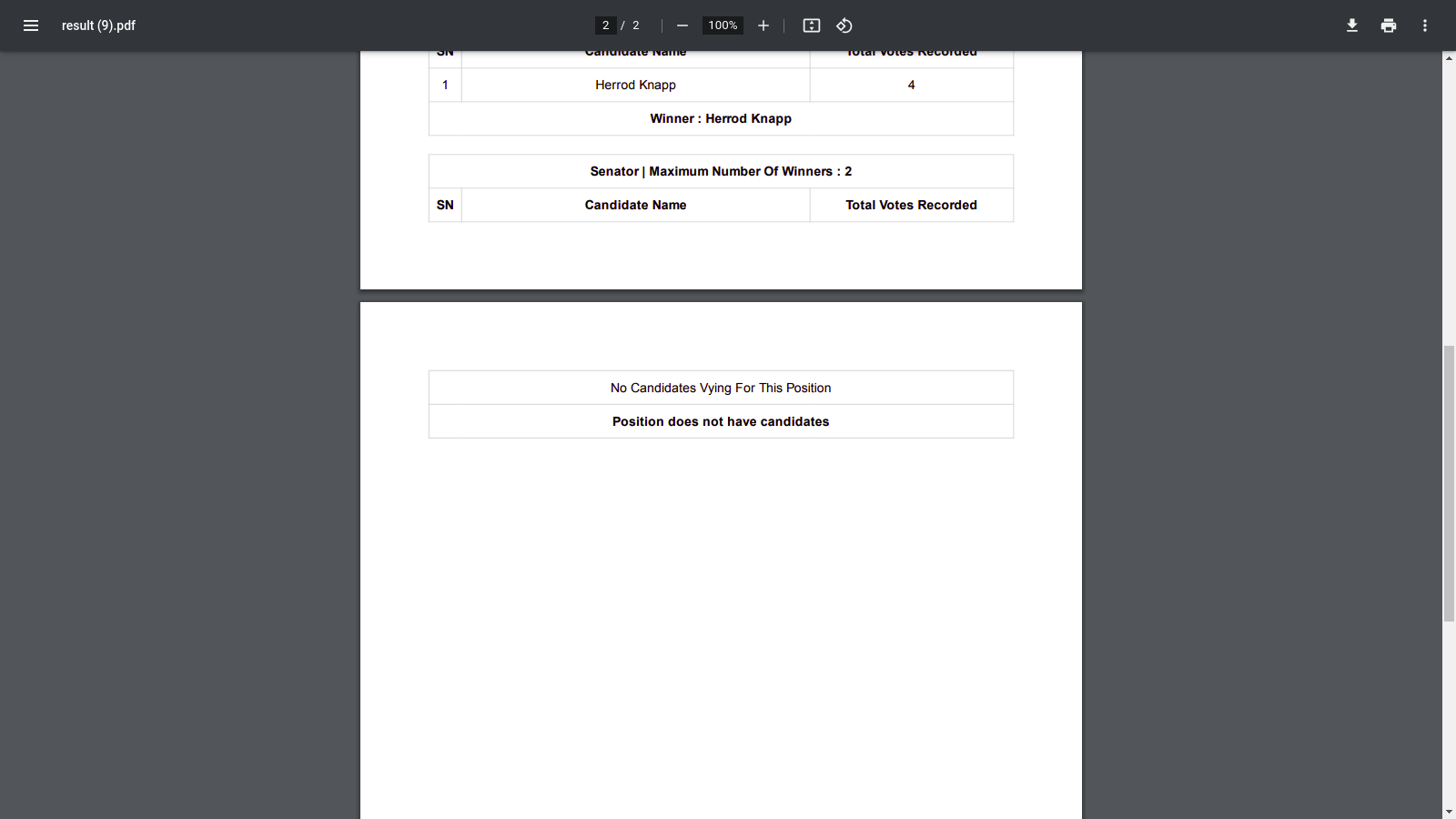
## 5.2. User Interface

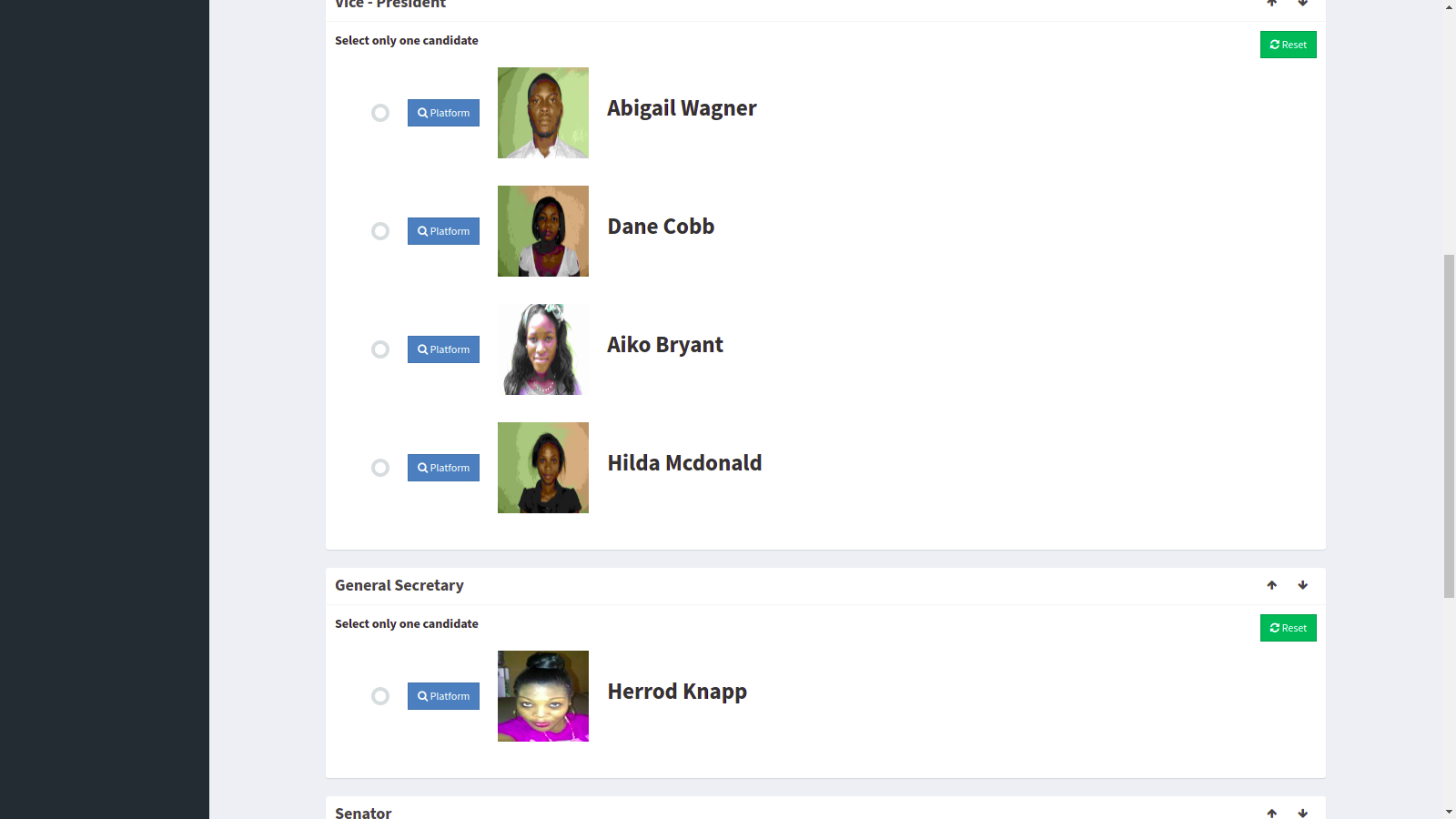
All screenshots have been which is related to my website.  
5.2.1 Homepage Admin and also their dashboards

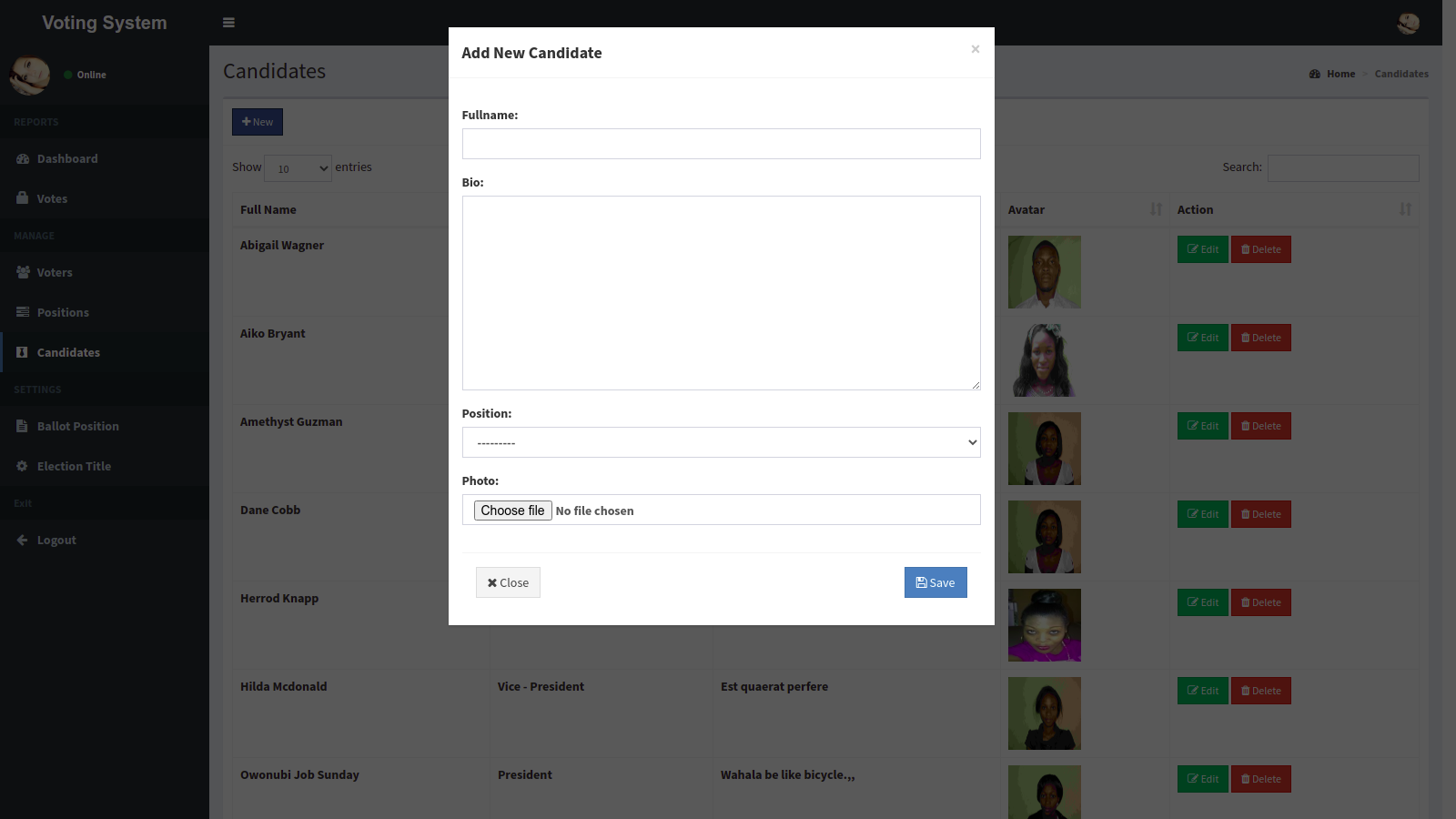
It’s a front side of project. This image has several 4 things home, Customer, Mechanic, Admin, About us while contact us these are helping tools of this page.

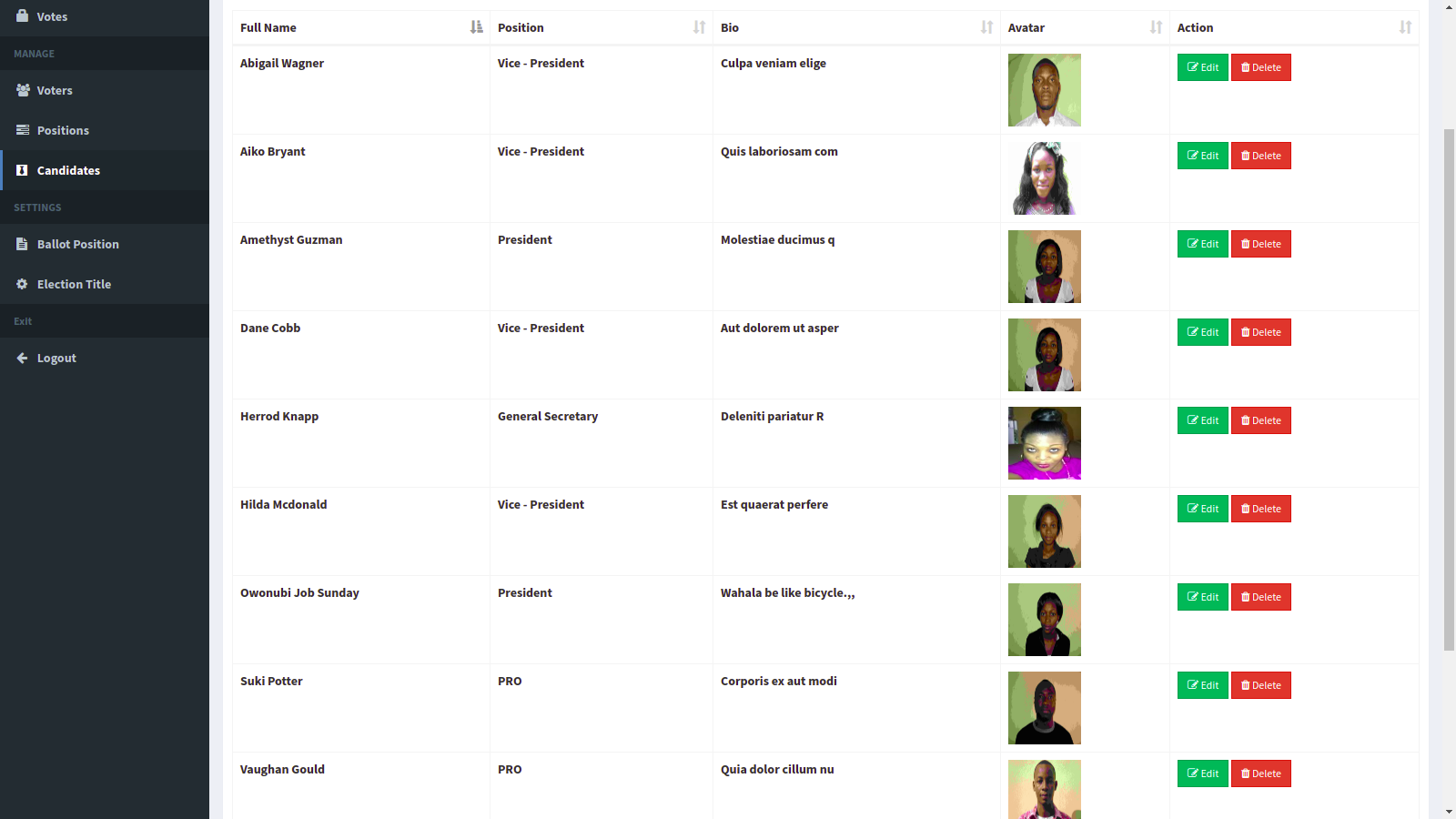


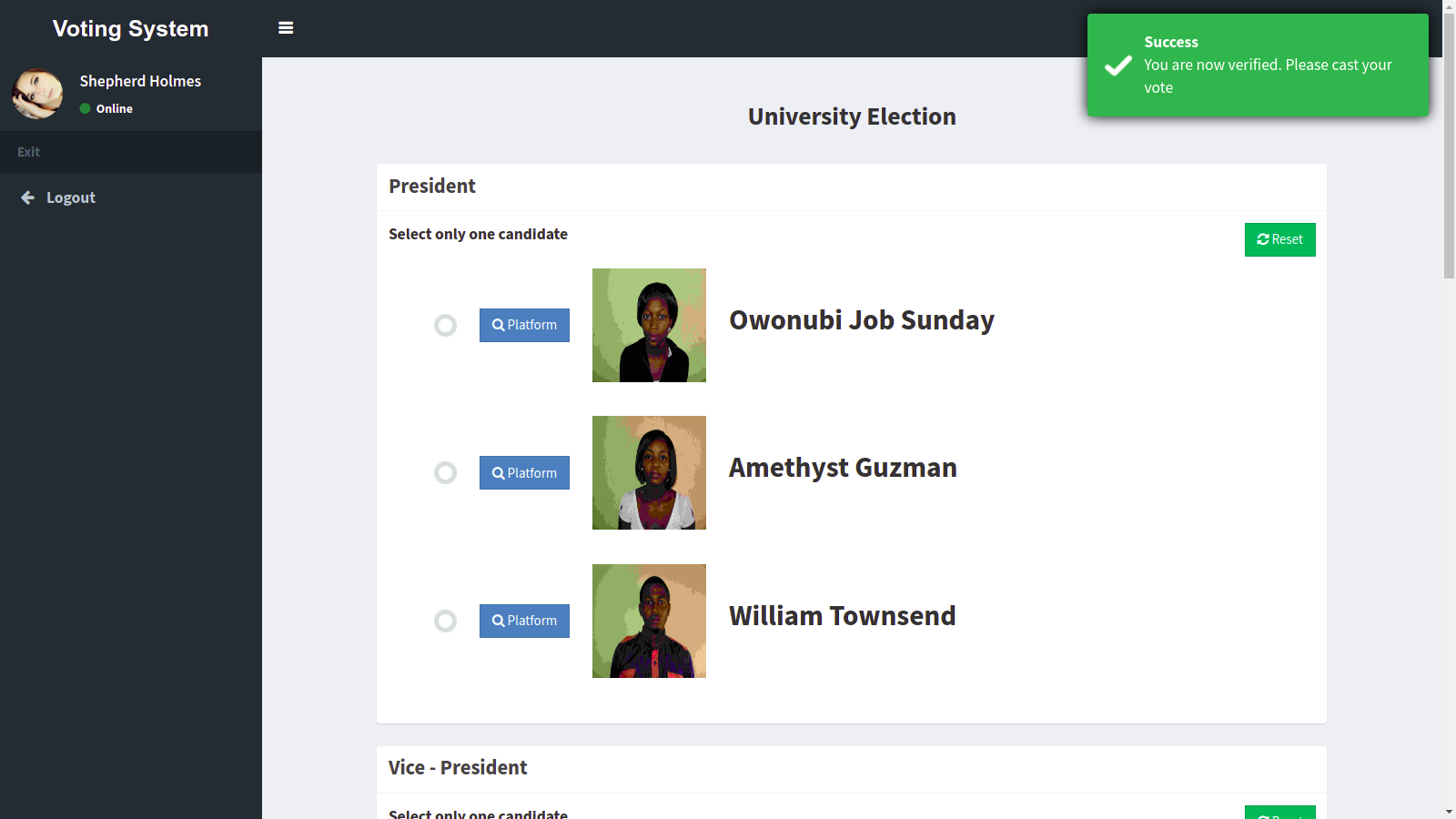


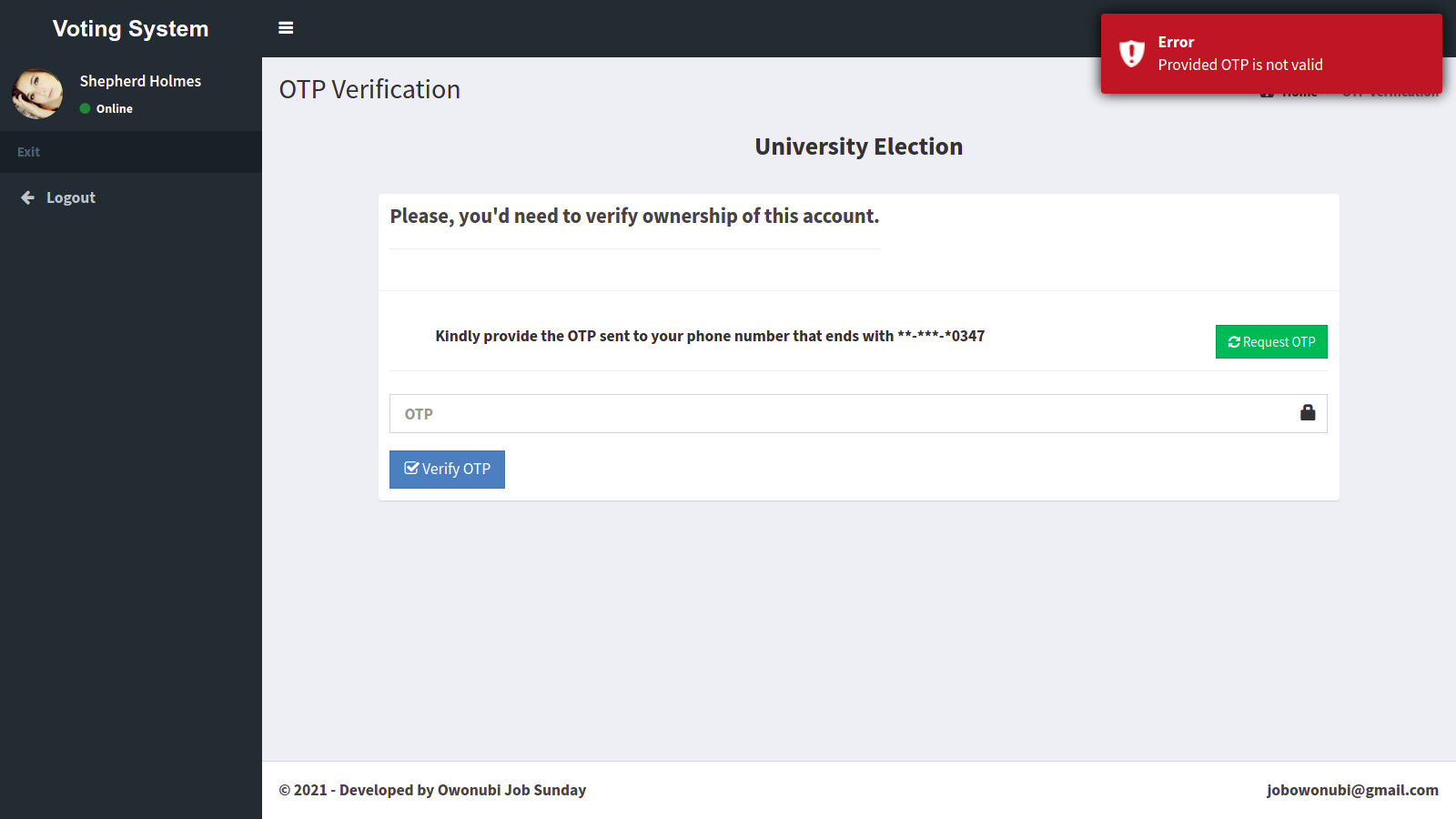


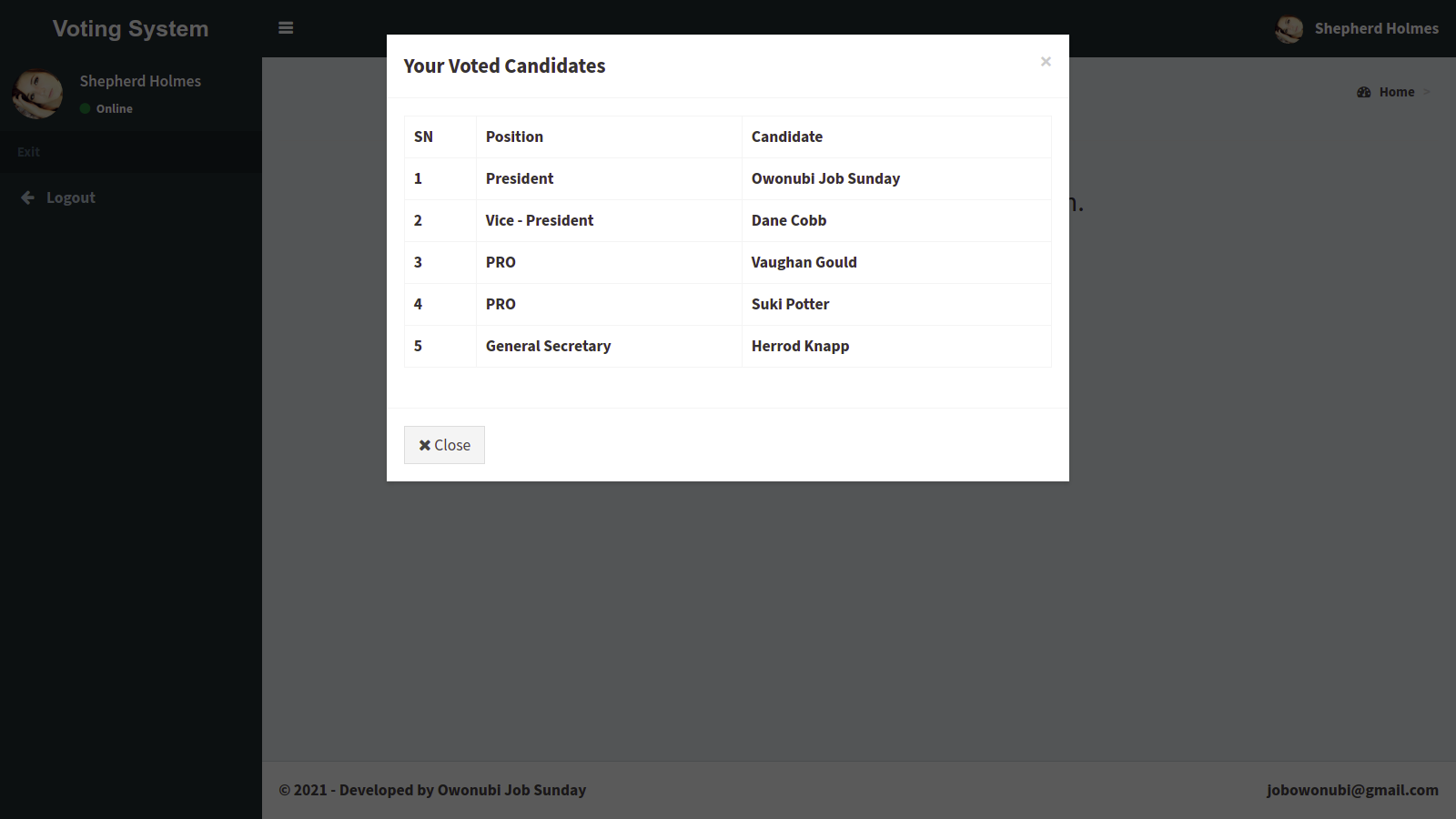
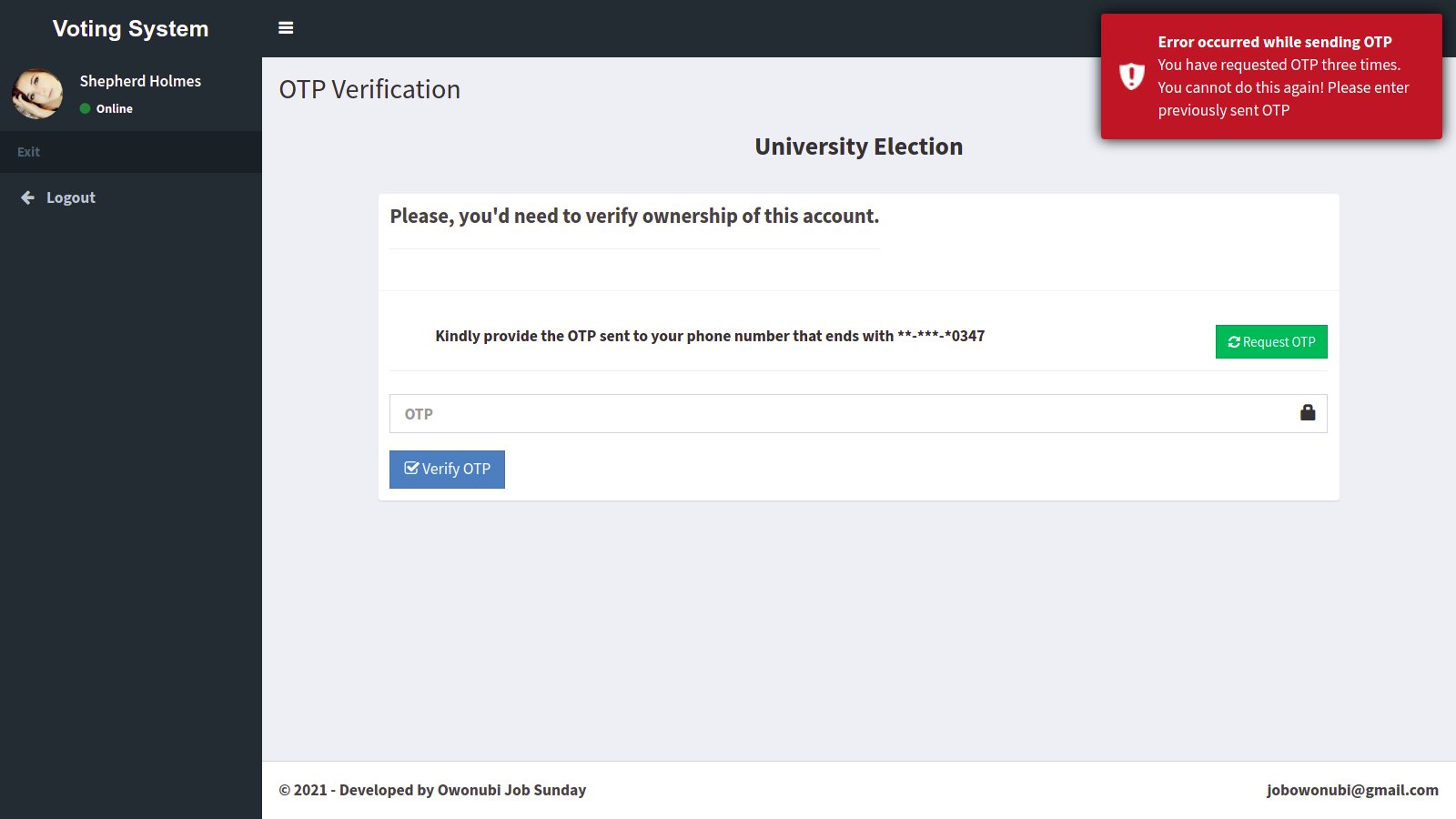


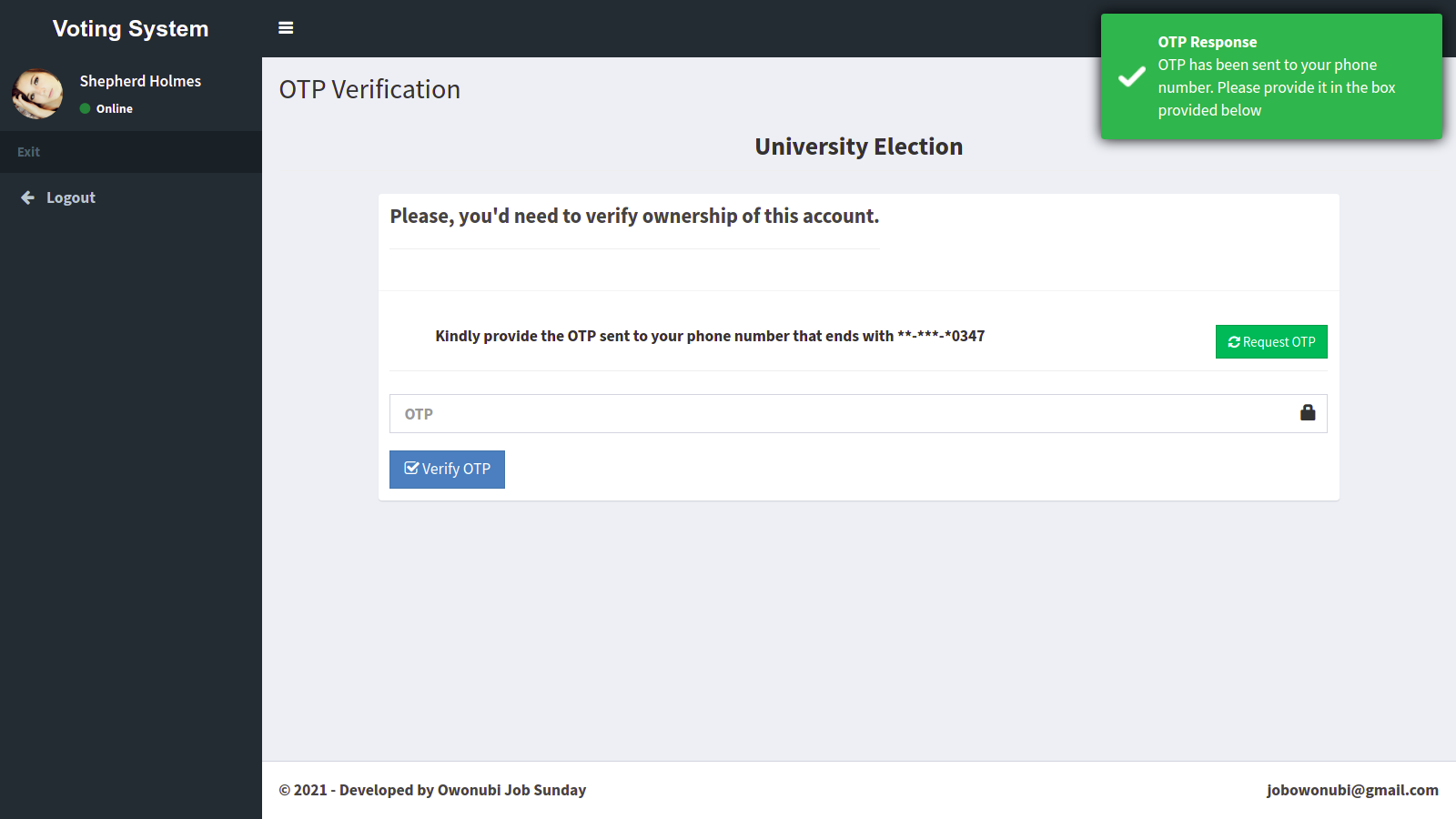


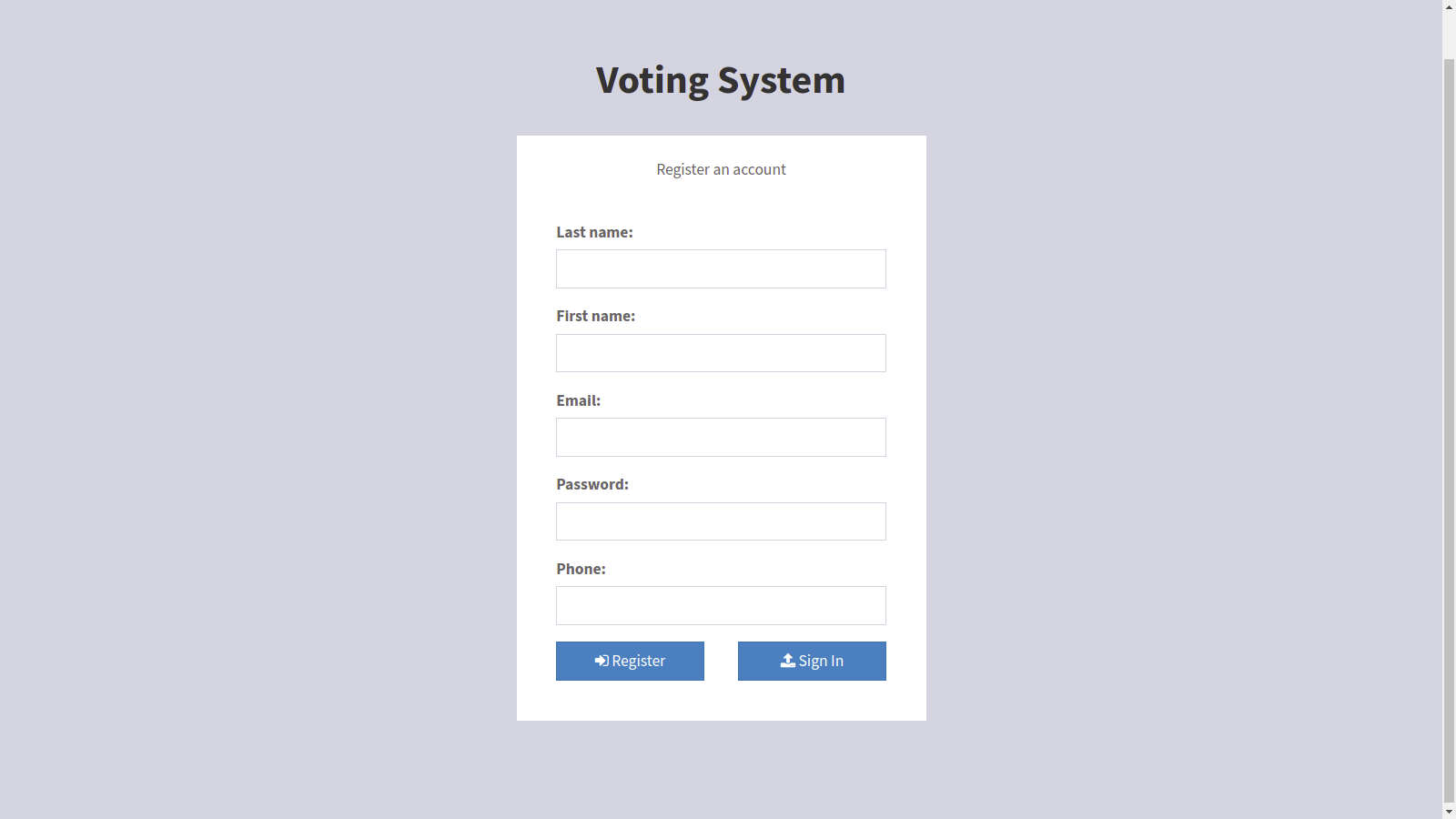
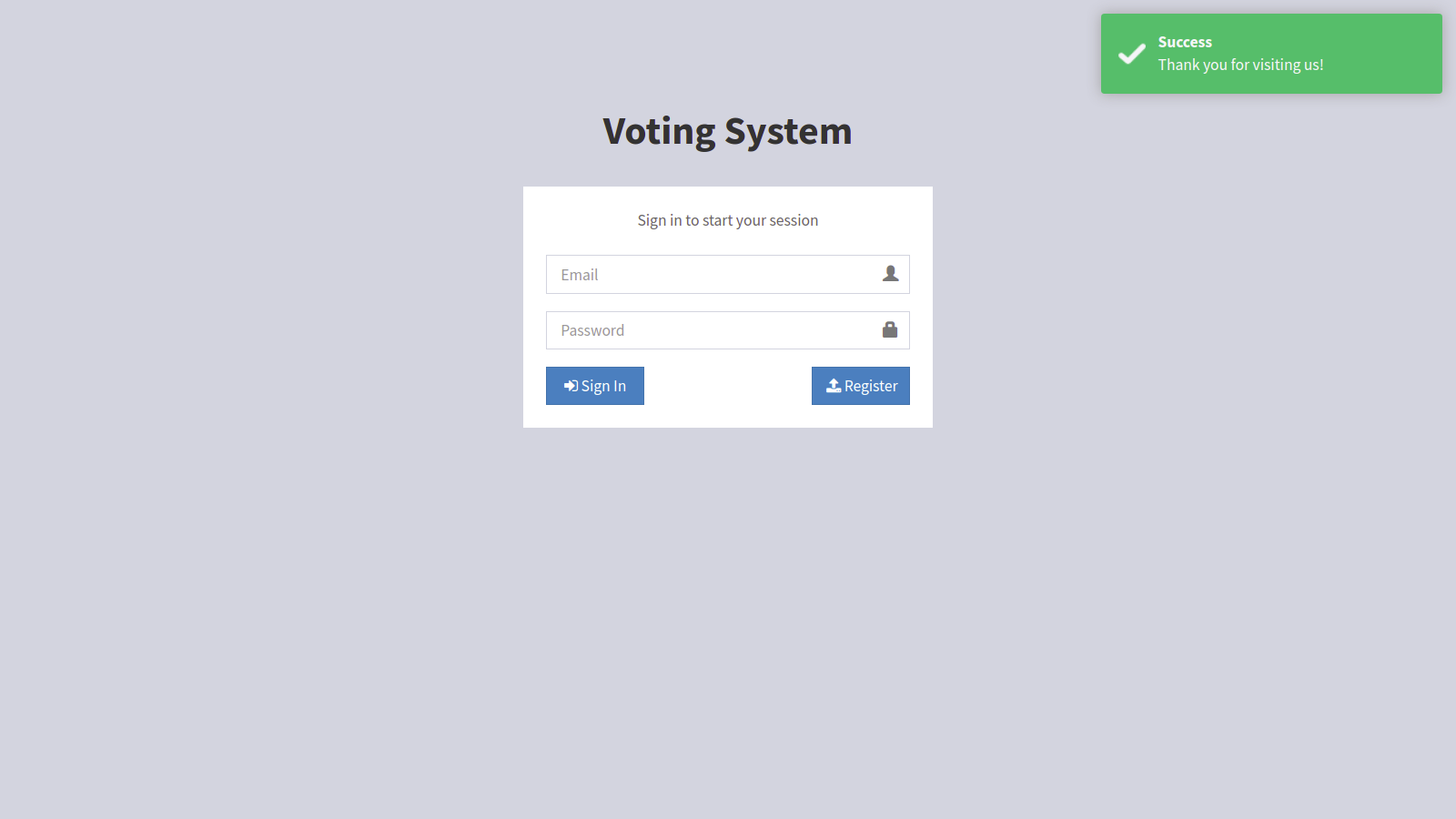
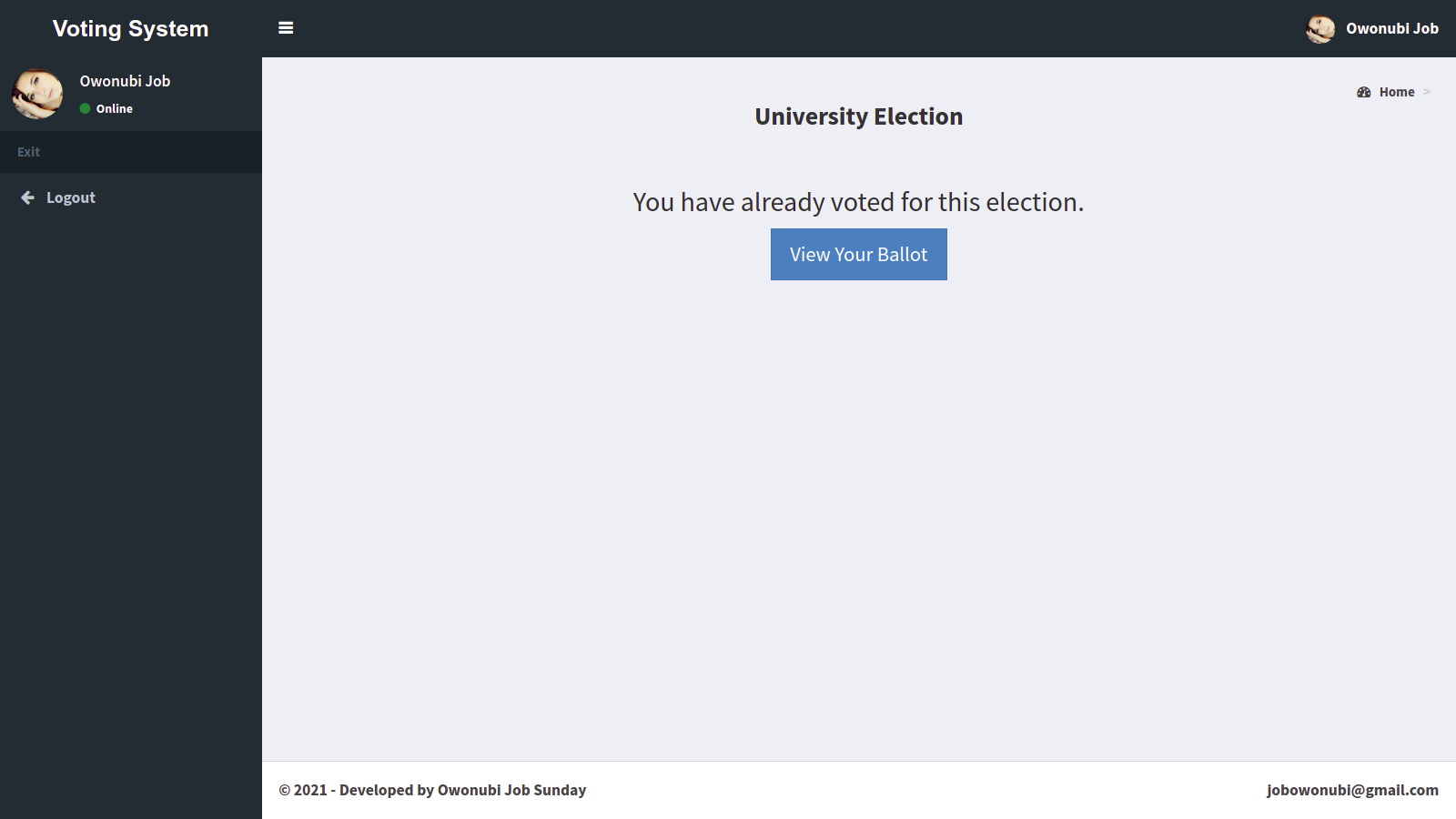
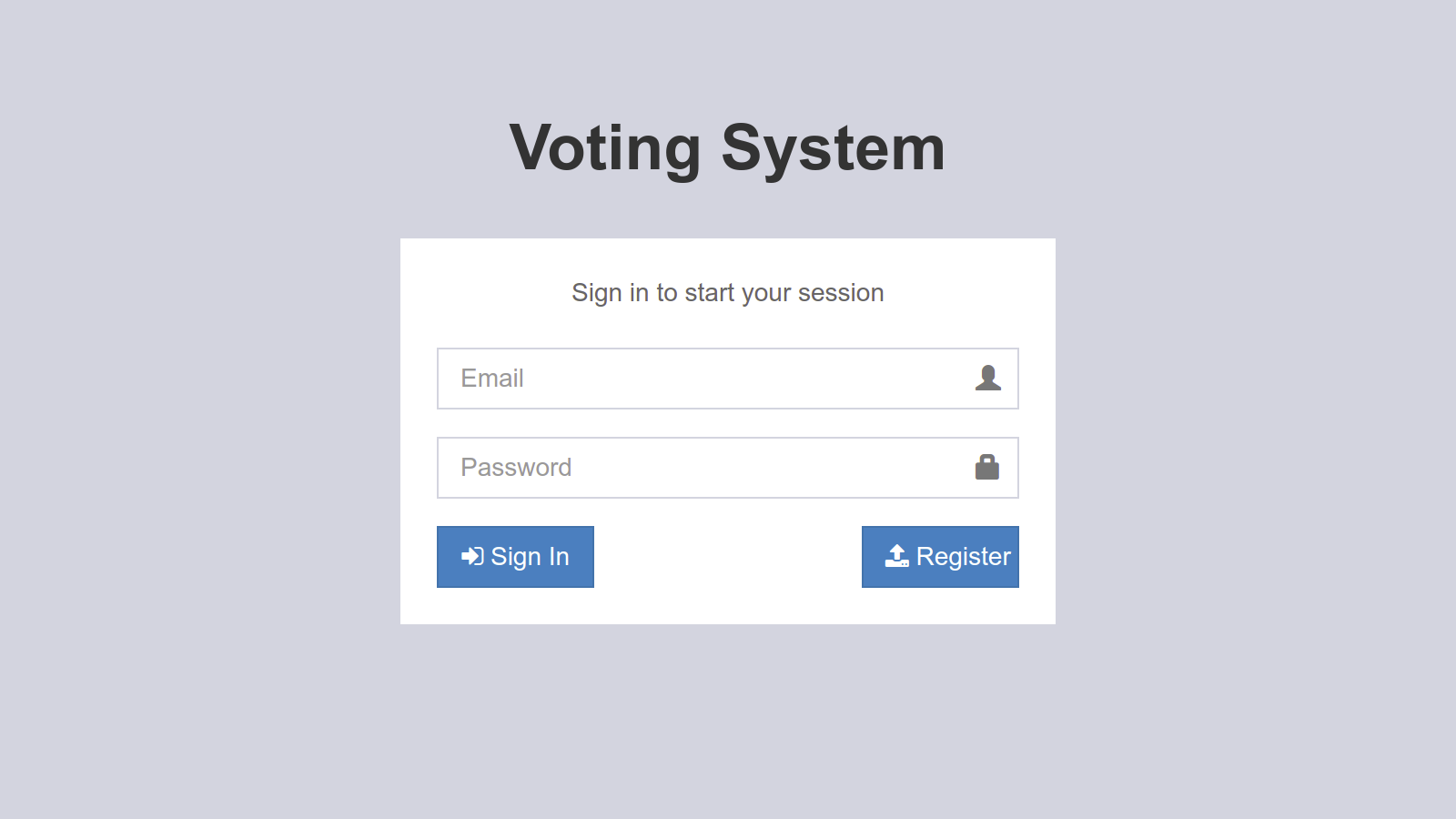


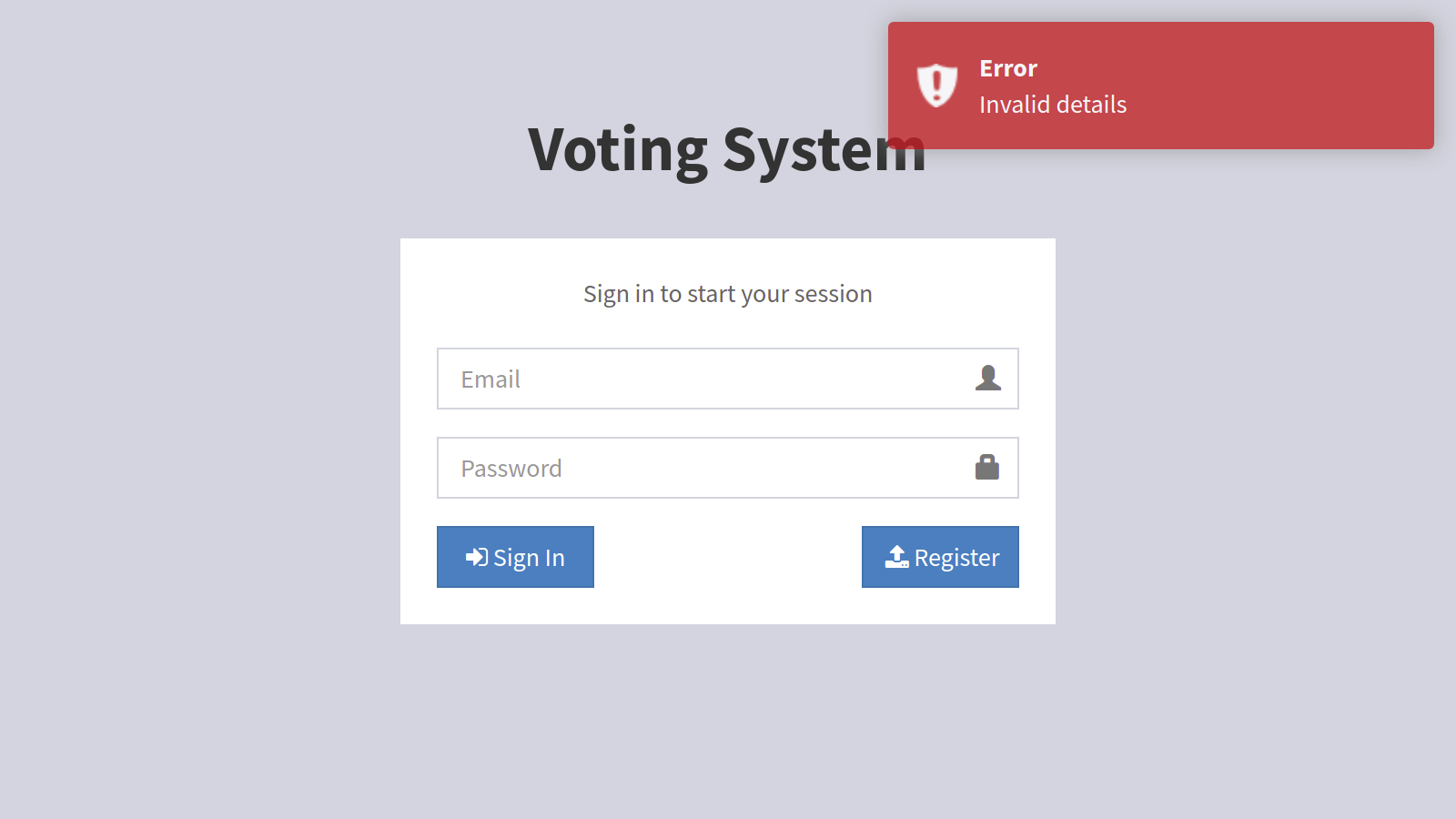
**Voter Board****:**

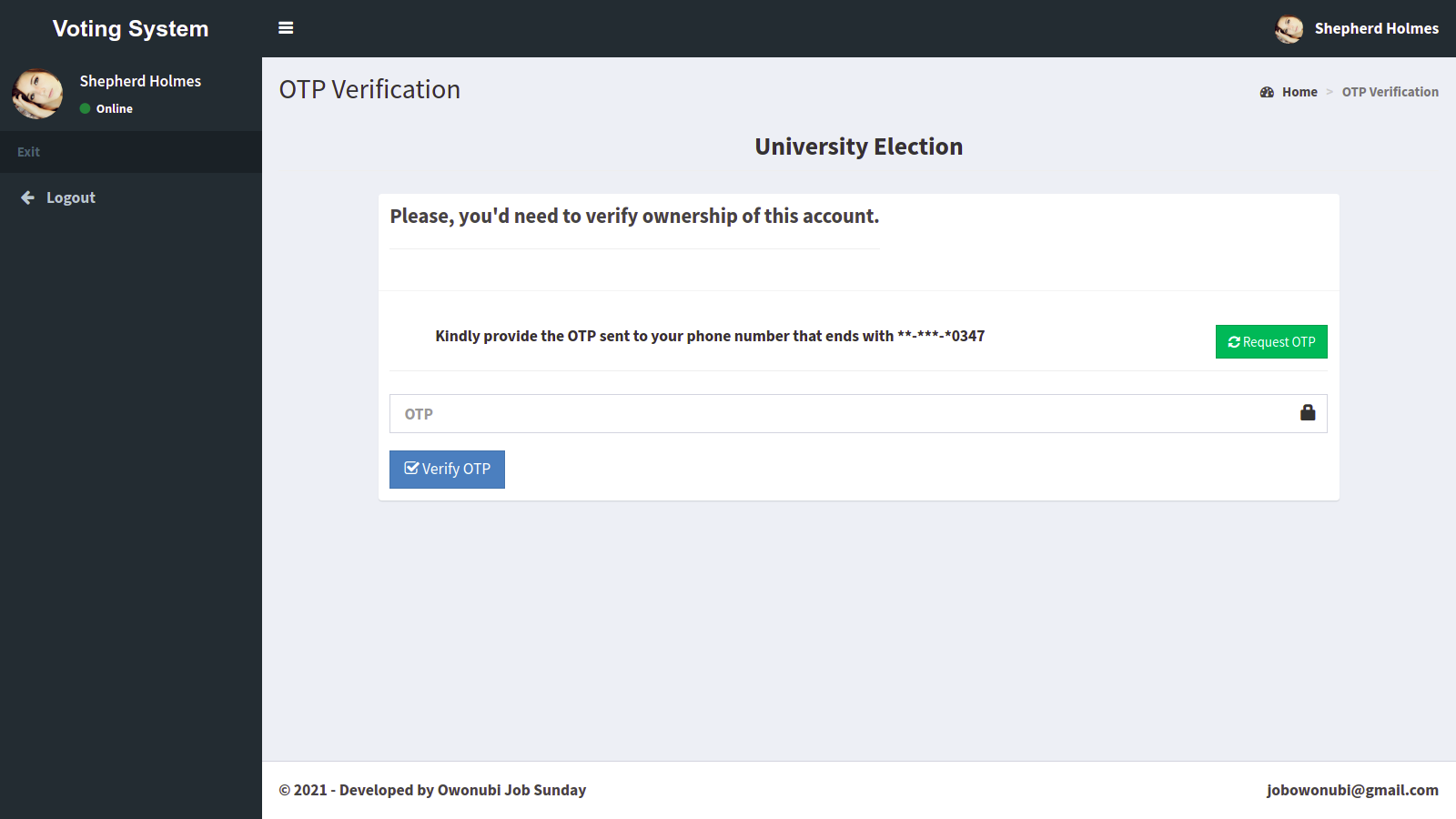
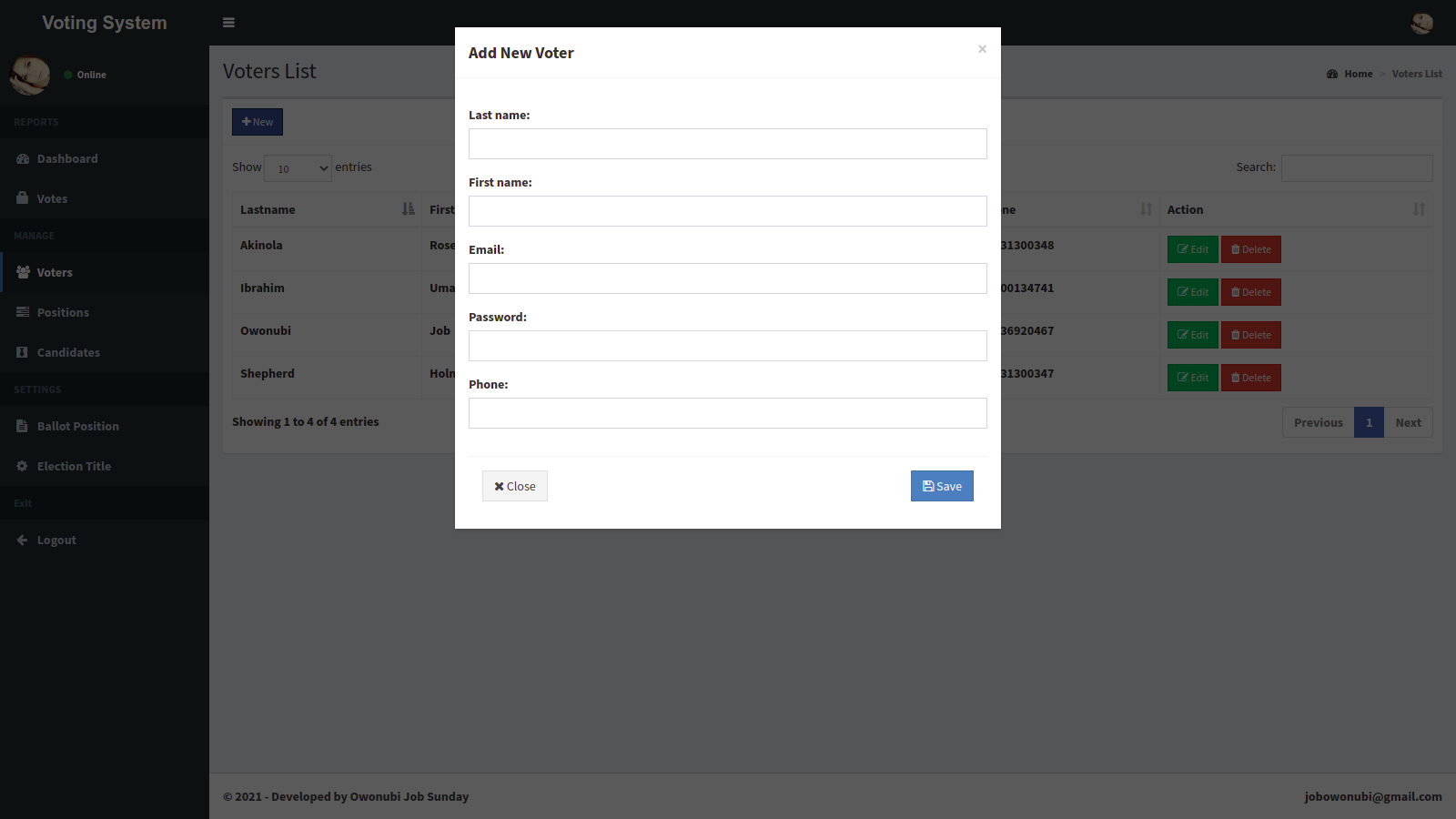
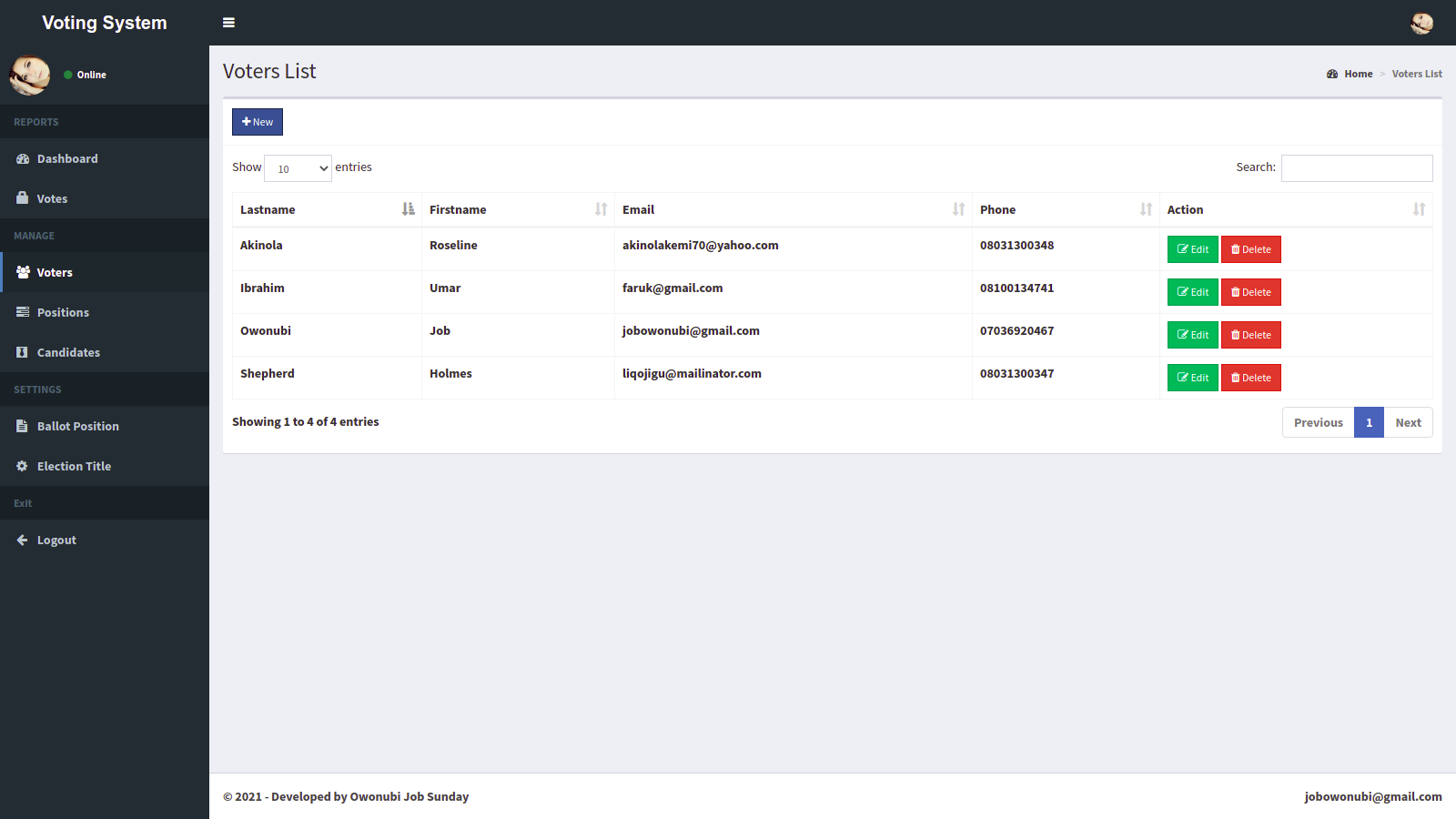
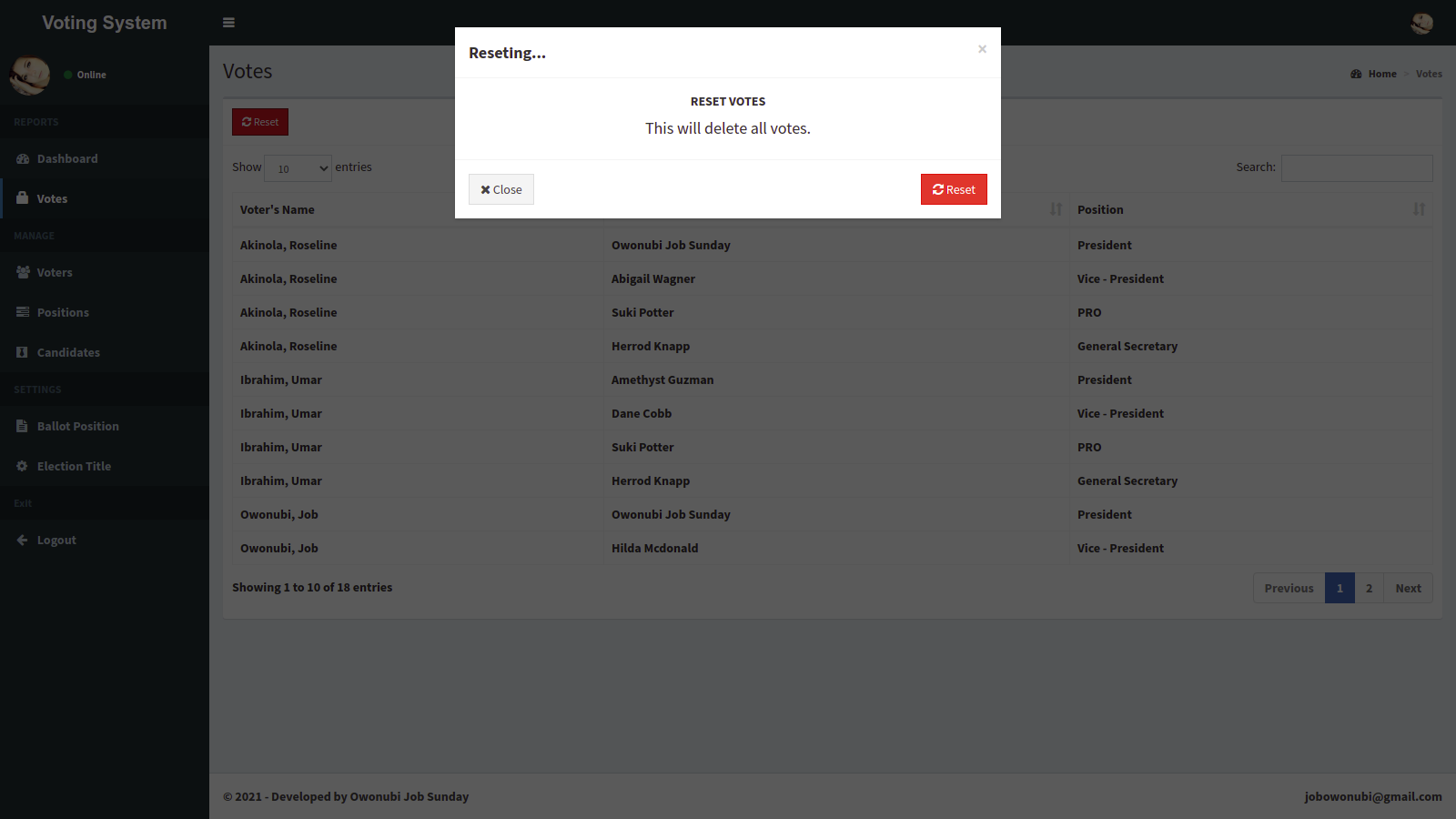
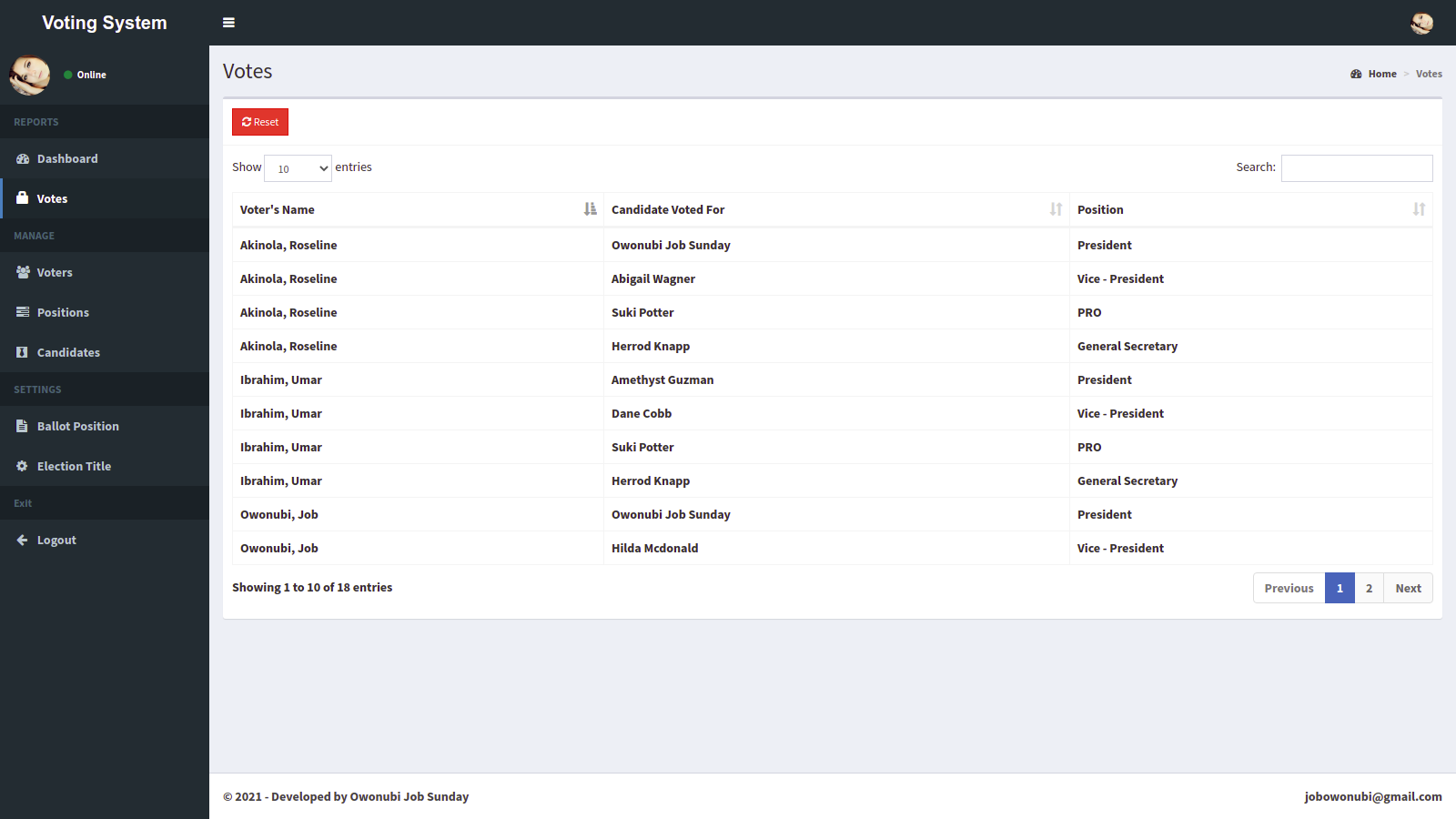
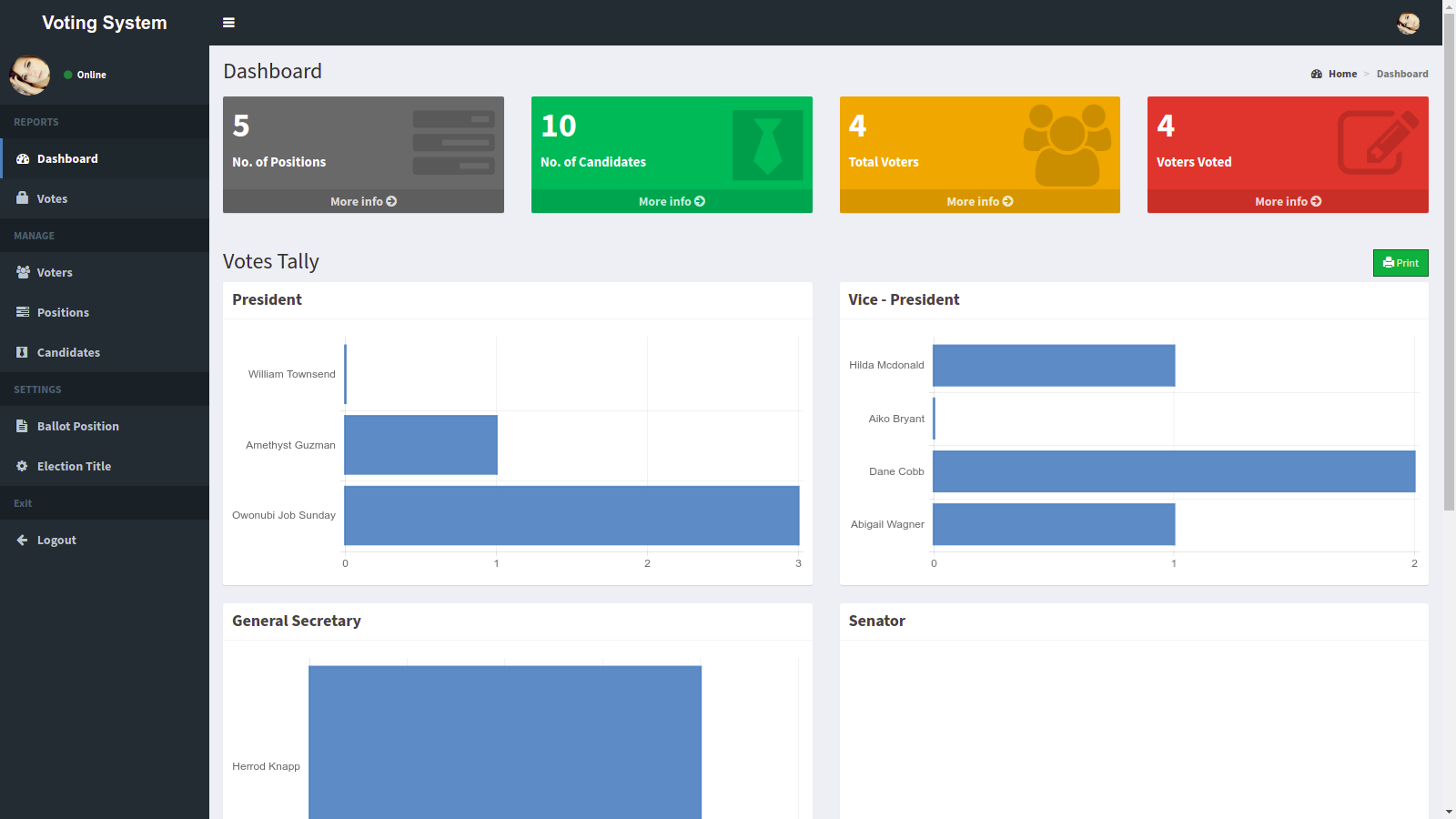
****

****

****

****





# Analysis model

### ***Use Case Diagram***

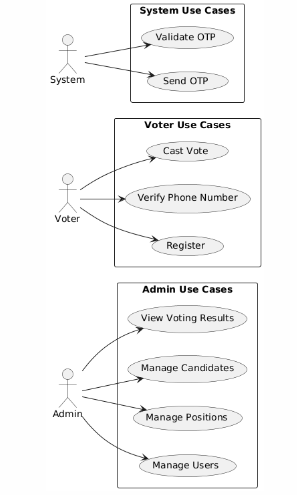
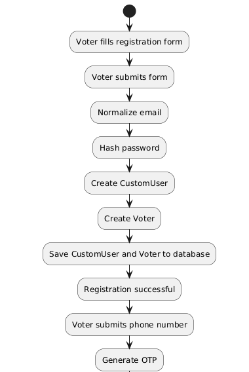


Figure 1 Use case diagram

### **Activity Diagram**



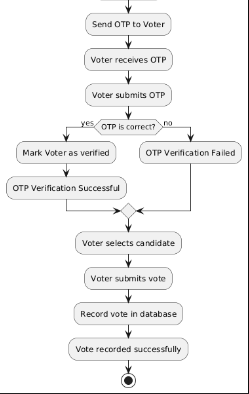
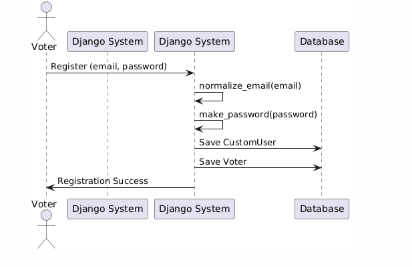
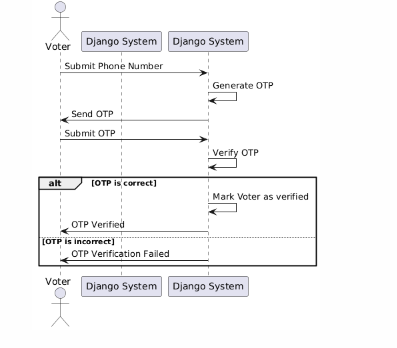


Figure 2 Activity diagram

### **Sequence Diagram**





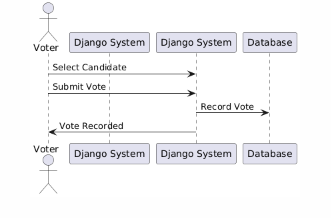


Figure 3: Sequence diagram

### **Database Diagram**

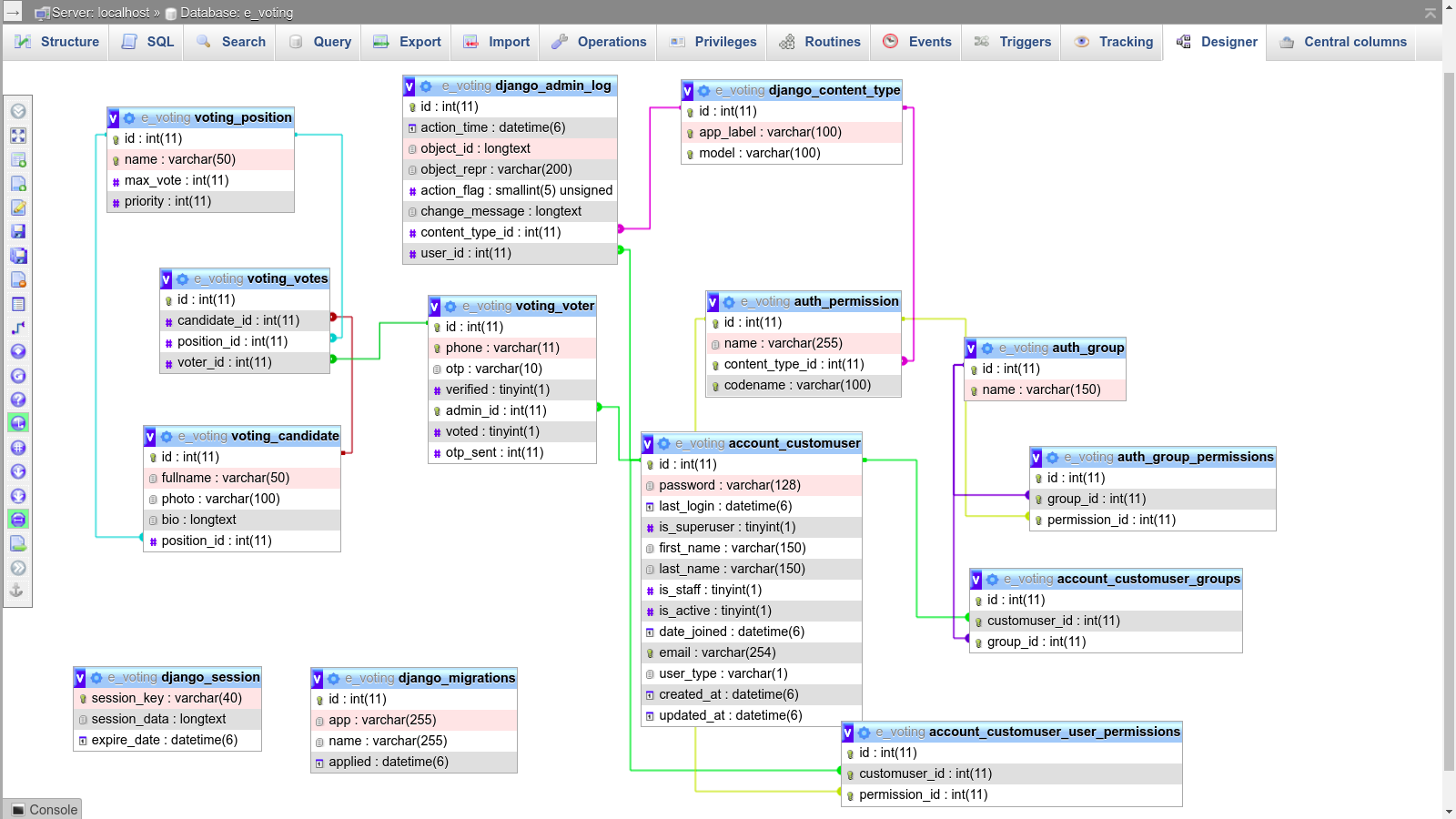


Figure 4: Database Diagram

### **Object Diagram**

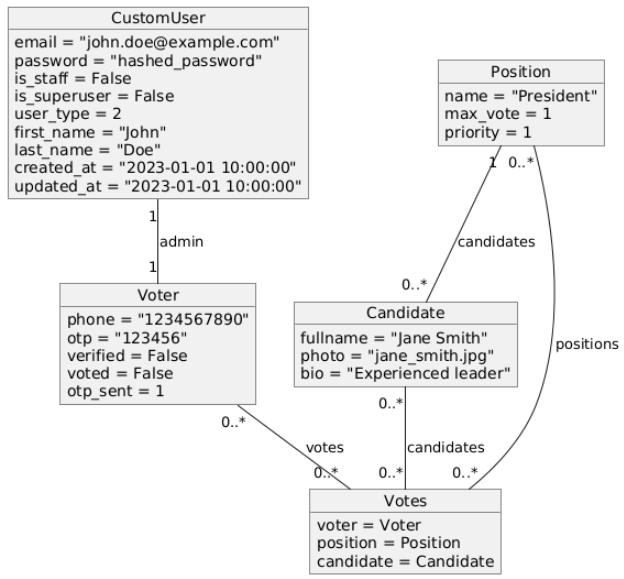


Figure 5: Object diagram

### Class Diagram

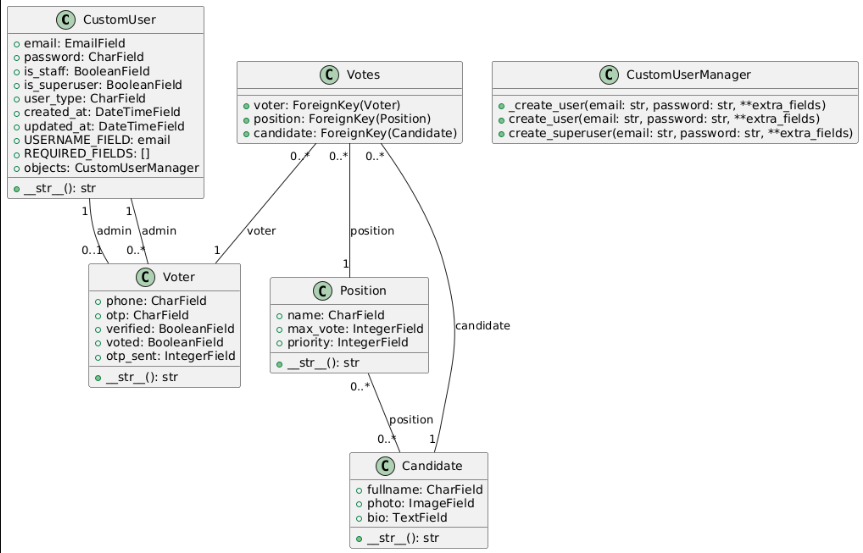


Figure 6: Class Diagram

# Tools & Technologies

## Programming Languages

#### Python with Django Framework

Python is a versatile, high-level programming language known for its readability and broad applicability in various domains, from web development to data science. For this project, we have chosen Python as the primary programming language due to its simplicity and the extensive ecosystem of libraries and frameworks it supports.

* **Django Framework**:

**Overview**: Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design. It is known for its "batteries-included" philosophy, offering a comprehensive suite of features out of the box.

**Key Features**:

1. **ORM (Object-Relational Mapping)**: Simplifies database interactions by allowing developers to interact with the database using Python code instead of SQL.
2. **Admin Interface**: Provides a powerful and customizable administrative interface automatically generated based on the models defined in the application.
3. **Security**: Includes built-in protection against common security threats like SQL injection, cross-site scripting (XSS), and cross-site request forgery (CSRF).
4. **Scalability**: Designed to handle high-traffic websites, making it suitable for both small projects and large-scale applications.
5. **Community and Documentation**: A robust community and extensive documentation make it easier for developers to find resources and support.

## Databases/Data storages

#### SQLite 3

SQLite is a C-language library that provides a lightweight, disk-based database. It does not require a separate server process and allows access to the database using a nonstandard variant of the SQL query language.

**Likely SQLite 3**:

**Overview**: SQLite is a self-contained, serverless, and zero-configuration database engine. It is widely used in embedded systems and applications where simplicity, reliability, and minimal setup are crucial.

**Key Features**:

1. **Serverless**: Unlike traditional database systems, SQLite operates as an embedded database, meaning it runs within the application itself.
2. **Zero Configuration**: No setup or administration is needed. SQLite databases are simply files on the disk, making them easy to manage and deploy.
3. **Transactional**: Implements ACID (Atomicity, Consistency, Isolation, Durability) properties fully, ensuring reliable and consistent data storage.
4. **Portable**: SQLite databases are platform-independent, meaning they can be copied between 32-bit and 64-bit systems or different operating systems without any changes.
5. **Lightweight**: With a small footprint, SQLite is ideal for use in situations where resource usage must be minimized.

## Operating System

#### Windows

For this project, Windows is used as the operating system. Windows is a series of operating systems developed by Microsoft. It provides a robust and user-friendly environment for development and deployment of applications.

**Likely Windows**:

**Overview**: Windows is one of the most widely used operating systems globally, known for its graphical user interface and support for a wide range of hardware and software.

**Key Features**:

1. **Ease of Use**: Known for its user-friendly interface, Windows allows developers to easily navigate and utilize various tools and applications.
2. **Compatibility**: Supports a vast array of hardware and software, making it a versatile choice for development environments.
3. **Development Tools**: Offers a variety of development tools such as Visual Studio, which can enhance the development process for applications built with Python and Django.
4. **Security**: Includes built-in security features like Windows Defender, BitLocker, and regular security updates to protect the development environment from threats.
5. **Support and Community**: Extensive documentation and a large community provide ample support for troubleshooting and optimizing the development setup.

# Appendix A: Glossary

|  |  |
| --- | --- |
| **Term** | **Definition** |
| **Authorized Users** | Individuals who have been granted permission to access and use the system based on their roles and privileges. |
| **Audit Trails** | Detailed records that track and log all user activities and system events within the E-voting management system. These trails provide accountability, traceability, and a means to investigate incidents. |

### **Appendix A: User Documentation**

#### What is a User Manual?

User documentation for the **E-Voting Management System** is a guide provided to users to help them effectively utilize the system. It is designed to assist voters, election officials, and administrators in understanding the features and functionalities of the e-voting system.

User documentation can come in various forms, such as printed manuals, video tutorials, or an online knowledge base. The format provided will depend on what is most useful for the user base of the e-voting system.

### **Benefits of User Documentation for the E-Voting Management System**

#### 1. Facilitate New User Onboarding

When voters or election administrators first use the **E-Voting Management System**, they may face a learning curve. No system is so intuitive that users can instantly grasp all of its features and workflows.

Providing new users with comprehensive user documentation will help them get familiar with the e-voting system, increasing the likelihood of successful integration and usage. Voters can refer to the manual to understand how to cast their votes, verify their choices, and ensure that their vote has been securely submitted.

Providing a smooth onboarding experience is crucial for building trust, and studies show that 86% of customers prefer businesses that offer well-structured onboarding materials.

#### 2. Lower the Cost of Customer Assistance

When voters and administrators have access to detailed user guides, they are less likely to reach out to support teams via phone or email. For instance, administrators might use the documentation to troubleshoot common system issues or questions about adding voters or configuring the system for an election.

Reducing the number of inquiries can lower costs and increase the efficiency of support teams. With fewer repetitive queries, support teams can focus on more complex issues, resulting in a better overall user experience. Self-service engagements via user documentation are cost-effective and reduce the burden on live support channels, which can be more expensive.

#### 3. Boost User Satisfaction

User documentation that simplifies the use of the **E-Voting Management System** results in happier users. When users can independently resolve common issues, such as how to log in or how to reset their credentials, they have a better experience.

A well-documented system prevents the frustration of needing to call support or being unable to cast a vote due to confusion. Well-structured documentation also allows election officials and administrators to efficiently manage elections, leading to a more streamlined voting process. Missing or poor-quality documentation can lead to dissatisfaction, so ensuring comprehensive user guides is critical.

#### 4. Reduce Liability for Incorrect Usage

For any system, especially one as critical as an e-voting platform, improper usage can lead to serious issues. Misusing the system could result in invalid votes, data loss, or system breaches. User documentation provides clear instructions and warnings about what to avoid and how to properly interact with the system.

By effectively communicating how to use the system and warning against improper actions (e.g., leaving sessions unattended, sharing login credentials), the documentation helps safeguard users and minimizes the risk of system misuse. This, in turn, reduces the risk of legal liabilities related to improper usage or security breaches.

**Refferences:**

### **Books**

1. **"Electronic Voting: The Shift from Paper Ballot to E-Voting"** by Robert Krimmer and Melanie Volkamer (2010).
   * This book discusses the development, challenges, and future prospects of electronic voting systems, particularly the move from paper-based systems to electronic methods.
2. **"Secure Electronic Voting"** edited by Dimitris Gritzalis (2003).
   * A collection of articles focused on the security aspects of e-voting systems. The book provides an in-depth analysis of cryptographic protocols, voter authentication, and data integrity in e-voting.
3. **"The Future of Election Systems: Organizing Electronic Democracy"** by John A. Stake and Michael O. Rousseau (2012).
   * This work covers a wide range of topics, including election system design, digital infrastructure, and the role of technology in ensuring the transparency and reliability of elections.

### Articles and Papers

1. **"A Survey of Electronic Voting Protocols"** by Rui Joaquim and Carlos Ribeiro, *ACM Computing Surveys* (2014).
   * This paper presents a comprehensive survey of the existing electronic voting protocols, highlighting key security features and vulnerabilities of each system.
2. **"Blockchain-Based Electronic Voting System"** by Zheng Zibin et al., *Future Generation Computer Systems* (2019).
   * Explores the use of blockchain technology to enhance the security and transparency of e-voting systems. This study offers a solution to many existing challenges in e-voting.
3. **"Analysis of E-Voting Security Protocols"** by A. Belenkiy and S. Goldberg, *Proceedings of the IEEE Symposium on Security and Privacy* (2007).
   * Discusses various e-voting protocols, focusing on their security features, such as anonymity, voter verifiability, and resistance to fraud.
4. **"Trust and Technology in Electronic Voting"** by Aviel D. Rubin, *Communications of the ACM* (2002).
   * Examines the public's trust in electronic voting systems, the technology behind them, and how issues such as usability and transparency can affect the adoption of e-voting.

### Government and Institutional Reports

1. **National Institute of Standards and Technology (NIST)**, *"Electronic Voting Security Considerations"* (2015).
   * This NIST report covers the security considerations in implementing e-voting systems in public elections, focusing on best practices for securing the voting infrastructure.
2. **"E-Voting: The Key Issues"** by the Council of Europe (2010).
   * This report highlights the major legal, security, and social issues related to implementing e-voting in European elections, offering guidelines for governments.
3. **"Estonian E-Voting System: Case Study"** by RIA (Information System Authority) Estonia (2017).
   * A detailed study of Estonia's e-voting system, which has been successfully implemented in national elections. This report covers security, usability, and adoption rates.

### Theses and Dissertations

1. **"The Implementation of Electronic Voting in Developing Countries: A Case Study of Brazil"** by Ana Barbosa, University of London (2011).
   * This thesis explores how Brazil successfully adopted electronic voting across the country, discussing the challenges faced and solutions implemented.
2. **"Secure and Transparent Online Voting System Using Cryptographic Techniques"** by Mohamed Abbas, Massachusetts Institute of Technology (MIT) (2018).
   * A research study focusing on the cryptographic aspects of ensuring secure and transparent online voting systems.

**Source Code as Manage.py:**

**from django.contrib.auth.models import AbstractUser, UserManager**

**from django.db import models**

**from django.contrib.auth.hashers import make\_password**

**from django.db.models.signals import post\_save**

**from django.dispatch import receiver**

**# Create your models here.**

**class CustomUserManager(UserManager):**

**def \_create\_user(self, email, password, \*\*extra\_fields):**

**email = self.normalize\_email(email)**

**user = CustomUser(email=email, \*\*extra\_fields)**

**user.password = make\_password(password)**

**user.save(using=self.\_db)**

**return user**

**def create\_user(self, email, password=None, \*\*extra\_fields):**

**extra\_fields.setdefault("is\_staff", False)**

**extra\_fields.setdefault("is\_superuser", False)**

**return self.\_create\_user(email, password, \*\*extra\_fields)**

**def create\_superuser(self, email, password=None, \*\*extra\_fields):**

**extra\_fields.setdefault("is\_staff", True)**

**extra\_fields.setdefault("is\_superuser", True)**

**extra\_fields.setdefault("user\_type", 1)**

**extra\_fields.setdefault("last\_name", "System")**

**extra\_fields.setdefault("first\_name", "Administrator")**

**assert extra\_fields["is\_staff"]**

**assert extra\_fields["is\_superuser"]**

**return self.\_create\_user(email, password, \*\*extra\_fields)**

**class CustomUser(AbstractUser):**

**USER\_TYPE = ((1, "Admin"), (2, "Voter"))**

**username = None # Removed username, using email instead**

**email = models.EmailField(unique=True)**

**user\_type = models.CharField(default=2, choices=USER\_TYPE, max\_length=1)**

**created\_at = models.DateTimeField(auto\_now\_add=True)**

**updated\_at = models.DateTimeField(auto\_now=True)**

**USERNAME\_FIELD = "email"**

**REQUIRED\_FIELDS = []**

**objects = CustomUserManager()**

**def \_\_str\_\_(self):**

**return self.last\_name + " " + self.first\_name**

**from django.shortcuts import render, redirect, reverse**

**from .email\_backend import EmailBackend**

**from django.contrib import messages**

**from .forms import CustomUserForm**

**from voting.forms import VoterForm**

**from django.contrib.auth import login, logout**

**# Create your views here.**

**def account\_login(request):**

**if request.user.is\_authenticated:**

**if request.user.user\_type == '1':**

**return redirect(reverse("adminDashboard"))**

**else:**

**return redirect(reverse("voterDashboard"))**

**context = {}**

**if request.method == 'POST':**

**user = EmailBackend.authenticate(request, username=request.POST.get(**

**'email'), password=request.POST.get('password'))**

**if user != None:**

**login(request, user)**

**if user.user\_type == '1':**

**return redirect(reverse("adminDashboard"))**

**else:**

**return redirect(reverse("voterDashboard"))**

**else:**

**messages.error(request, "Invalid details")**

**return redirect("/")**

**return render(request, "voting/login.html", context)**

**def account\_register(request):**

**userForm = CustomUserForm(request.POST or None)**

**voterForm = VoterForm(request.POST or None)**

**context = {**

**'form1': userForm,**

**'form2': voterForm**

**}**

**if request.method == 'POST':**

**if userForm.is\_valid() and voterForm.is\_valid():**

**user = userForm.save(commit=False)**

**voter = voterForm.save(commit=False)**

**voter.admin = user**

**user.save()**

**voter.save()**

**messages.success(request, "Account created. You can login now!")**

**return redirect(reverse('account\_login'))**

**else:**

**messages.error(request, "Provided data failed validation")**

**# return account\_login(request)**

**return render(request, "voting/reg.html", context)**

**def account\_logout(request):**

**user = request.user**

**if user.is\_authenticated:**

**logout(request)**

**messages.success(request, "Thank you for visiting us!")**

**else:**

**messages.error(**

**request, "You need to be logged in to perform this action")**

**return redirect(reverse("account\_login"))**

**from django.shortcuts import render, reverse, redirect**

**from voting.models import Voter, Position, Candidate, Votes**

**from account.models import CustomUser**

**from account.forms import CustomUserForm**

**from voting.forms import \***

**from django.contrib import messages**

**from django.http import JsonResponse, HttpResponse**

**from django.conf import settings**

**import json # Not used**

**from django\_renderpdf.views import PDFView**

**def find\_n\_winners(data, n):**

**"""Read More**

**https://www.geeksforgeeks.org/python-program-to-find-n-largest-elements-from-a-list/**

**"""**

**final\_list = []**

**candidate\_data = data[:]**

**# print("Candidate = ", str(candidate\_data))**

**for i in range(0, n):**

**max1 = 0**

**if len(candidate\_data) == 0:**

**continue**

**this\_winner = max(candidate\_data, key=lambda x: x['votes'])**

**# TODO: Check if None**

**this = this\_winner['name'] + \**

**" with " + str(this\_winner['votes']) + " votes"**

**final\_list.append(this)**

**candidate\_data.remove(this\_winner)**

**return ", &nbsp;".join(final\_list)**

**class PrintView(PDFView):**

**template\_name = 'admin/print.html'**

**prompt\_download = True**

**@property**

**def download\_name(self):**

**return "result.pdf"**

**def get\_context\_data(self, \*args, \*\*kwargs):**

**title = "E-voting"**

**try:**

**file = open(settings.ELECTION\_TITLE\_PATH, 'r')**

**title = file.read()**

**except:**

**pass**

**context = super().get\_context\_data(\*args, \*\*kwargs)**

**position\_data = {}**

**for position in Position.objects.all():**

**candidate\_data = []**

**winner = ""**

**for candidate in Candidate.objects.filter(position=position):**

**this\_candidate\_data = {}**

**votes = Votes.objects.filter(candidate=candidate).count()**

**this\_candidate\_data['name'] = candidate.fullname**

**this\_candidate\_data['votes'] = votes**

**candidate\_data.append(this\_candidate\_data)**

**print("Candidate Data For ", str(**

**position.name), " = ", str(candidate\_data))**

**# ! Check Winner**

**if len(candidate\_data) < 1:**

**winner = "Position does not have candidates"**

**else:**

**# Check if max\_vote is more than 1**

**if position.max\_vote > 1:**

**winner = find\_n\_winners(candidate\_data, position.max\_vote)**

**else:**

**winner = max(candidate\_data, key=lambda x: x['votes'])**

**if winner['votes'] == 0:**

**winner = "No one voted for this yet position, yet."**

**else:**

**"""**

**https://stackoverflow.com/questions/18940540/how-can-i-count-the-occurrences-of-an-item-in-a-list-of-dictionaries**

**"""**

**count = sum(1 for d in candidate\_data if d.get(**

**'votes') == winner['votes'])**

**if count > 1:**

**winner = f"There are {count} candidates with {winner['votes']} votes"**

**else:**

**winner = "Winner : " + winner['name']**

**print("Candidate Data For ", str(**

**position.name), " = ", str(candidate\_data))**

**position\_data[position.name] = {**

**'candidate\_data': candidate\_data, 'winner': winner, 'max\_vote': position.max\_vote}**

**context['positions'] = position\_data**

**print(context)**

**return context**

**def dashboard(request):**

**positions = Position.objects.all().order\_by('priority')**

**candidates = Candidate.objects.all()**

**voters = Voter.objects.all()**

**voted\_voters = Voter.objects.filter(voted=1)**

**list\_of\_candidates = []**

**votes\_count = []**

**chart\_data = {}**

**for position in positions:**

**list\_of\_candidates = []**

**votes\_count = []**

**for candidate in Candidate.objects.filter(position=position):**

**list\_of\_candidates.append(candidate.fullname)**

**votes = Votes.objects.filter(candidate=candidate).count()**

**votes\_count.append(votes)**

**chart\_data[position] = {**

**'candidates': list\_of\_candidates,**

**'votes': votes\_count,**

**'pos\_id': position.id**

**}**

**context = {**

**'position\_count': positions.count(),**

**'candidate\_count': candidates.count(),**

**'voters\_count': voters.count(),**

**'voted\_voters\_count': voted\_voters.count(),**

**'positions': positions,**

**'chart\_data': chart\_data,**

**'page\_title': "Dashboard"**

**}**

**return render(request, "admin/home.html", context)**

**def voters(request):**

**voters = Voter.objects.all()**

**userForm = CustomUserForm(request.POST or None)**

**voterForm = VoterForm(request.POST or None)**

**context = {**

**'form1': userForm,**

**'form2': voterForm,**

**'voters': voters,**

**'page\_title': 'Voters List'**

**}**

**if request.method == 'POST':**

**if userForm.is\_valid() and voterForm.is\_valid():**

**user = userForm.save(commit=False)**

**voter = voterForm.save(commit=False)**

**voter.admin = user**

**user.save()**

**voter.save()**

**messages.success(request, "New voter created")**

**else:**

**messages.error(request, "Form validation failed")**

**return render(request, "admin/voters.html", context)**

**def view\_voter\_by\_id(request):**

**voter\_id = request.GET.get('id', None)**

**voter = Voter.objects.filter(id=voter\_id)**

**context = {}**

**if not voter.exists():**

**context['code'] = 404**

**else:**

**context['code'] = 200**

**voter = voter[0]**

**context['first\_name'] = voter.admin.first\_name**

**context['last\_name'] = voter.admin.last\_name**

**context['phone'] = voter.phone**

**context['id'] = voter.id**

**context['email'] = voter.admin.email**

**return JsonResponse(context)**

**def view\_position\_by\_id(request):**

**pos\_id = request.GET.get('id', None)**

**pos = Position.objects.filter(id=pos\_id)**

**context = {}**

**if not pos.exists():**

**context['code'] = 404**

**else:**

**context['code'] = 200**

**pos = pos[0]**

**context['name'] = pos.name**

**context['max\_vote'] = pos.max\_vote**

**context['id'] = pos.id**

**return JsonResponse(context)**

**def updateVoter(request):**

**if request.method != 'POST':**

**messages.error(request, "Access Denied")**

**try:**

**instance = Voter.objects.get(id=request.POST.get('id'))**

**user = CustomUserForm(request.POST or None, instance=instance.admin)**

**voter = VoterForm(request.POST or None, instance=instance)**

**user.save()**

**voter.save()**

**messages.success(request, "Voter's bio updated")**

**except:**

**messages.error(request, "Access To This Resource Denied")**

**return redirect(reverse('adminViewVoters'))**

**def deleteVoter(request):**

**if request.method != 'POST':**

**messages.error(request, "Access Denied")**

**try:**

**admin = Voter.objects.get(id=request.POST.get('id')).admin**

**admin.delete()**

**messages.success(request, "Voter Has Been Deleted")**

**except:**

**messages.error(request, "Access To This Resource Denied")**

**return redirect(reverse('adminViewVoters'))**

**def viewPositions(request):**

**positions = Position.objects.order\_by('-priority').all()**

**form = PositionForm(request.POST or None)**

**context = {**

**'positions': positions,**

**'form1': form,**

**'page\_title': "Positions"**

**}**

**if request.method == 'POST':**

**if form.is\_valid():**

**form = form.save(commit=False)**

**form.priority = positions.count() + 1 # Just in case it is empty.**

**form.save()**

**messages.success(request, "New Position Created")**

**else:**

**messages.error(request, "Form errors")**

**return render(request, "admin/positions.html", context)**

**def updatePosition(request):**

**if request.method != 'POST':**

**messages.error(request, "Access Denied")**

**try:**

**instance = Position.objects.get(id=request.POST.get('id'))**

**pos = PositionForm(request.POST or None, instance=instance)**

**pos.save()**

**messages.success(request, "Position has been updated")**

**except:**

**messages.error(request, "Access To This Resource Denied")**

**return redirect(reverse('viewPositions'))**

**def deletePosition(request):**

**if request.method != 'POST':**

**messages.error(request, "Access Denied")**

**try:**

**pos = Position.objects.get(id=request.POST.get('id'))**

**pos.delete()**

**messages.success(request, "Position Has Been Deleted")**

**except:**

**messages.error(request, "Access To This Resource Denied")**

**return redirect(reverse('viewPositions'))**

**def viewCandidates(request):**

**candidates = Candidate.objects.all()**

**form = CandidateForm(request.POST or None, request.FILES or None)**

**context = {**

**'candidates': candidates,**

**'form1': form,**

**'page\_title': 'Candidates'**

**}**

**if request.method == 'POST':**

**if form.is\_valid():**

**form = form.save()**

**messages.success(request, "New Candidate Created")**

**else:**

**messages.error(request, "Form errors")**

**return render(request, "admin/candidates.html", context)**

**def updateCandidate(request):**

**if request.method != 'POST':**

**messages.error(request, "Access Denied")**

**try:**

**candidate\_id = request.POST.get('id')**

**candidate = Candidate.objects.get(id=candidate\_id)**

**form = CandidateForm(request.POST or None,**

**request.FILES or None, instance=candidate)**

**if form.is\_valid():**

**form.save()**

**messages.success(request, "Candidate Data Updated")**

**else:**

**messages.error(request, "Form has errors")**

**except:**

**messages.error(request, "Access To This Resource Denied")**

**return redirect(reverse('viewCandidates'))**

**def deleteCandidate(request):**

**if request.method != 'POST':**

**messages.error(request, "Access Denied")**

**try:**

**pos = Candidate.objects.get(id=request.POST.get('id'))**

**pos.delete()**

**messages.success(request, "Candidate Has Been Deleted")**

**except:**

**messages.error(request, "Access To This Resource Denied")**

**return redirect(reverse('viewCandidates'))**

**def view\_candidate\_by\_id(request):**

**candidate\_id = request.GET.get('id', None)**

**candidate = Candidate.objects.filter(id=candidate\_id)**

**context = {}**

**if not candidate.exists():**

**context['code'] = 404**

**else:**

**candidate = candidate[0]**

**context['code'] = 200**

**context['fullname'] = candidate.fullname**

**previous = CandidateForm(instance=candidate)**

**context['form'] = str(previous.as\_p())**

**return JsonResponse(context)**

**def ballot\_position(request):**

**context = {**

**'page\_title': "Ballot Position"**

**}**

**return render(request, "admin/ballot\_position.html", context)**

**def update\_ballot\_position(request, position\_id, up\_or\_down):**

**try:**

**context = {**

**'error': False**

**}**

**position = Position.objects.get(id=position\_id)**

**if up\_or\_down == 'up':**

**priority = position.priority - 1**

**if priority == 0:**

**context['error'] = True**

**output = "This position is already at the top"**

**else:**

**Position.objects.filter(priority=priority).update(**

**priority=(priority+1))**

**position.priority = priority**

**position.save()**

**output = "Moved Up"**

**else:**

**priority = position.priority + 1**

**if priority > Position.objects.all().count():**

**output = "This position is already at the bottom"**

**context['error'] = True**

**else:**

**Position.objects.filter(priority=priority).update(**

**priority=(priority-1))**

**position.priority = priority**

**position.save()**

**output = "Moved Down"**

**context['message'] = output**

**except Exception as e:**

**context['message'] = e**

**return JsonResponse(context)**

**def ballot\_title(request):**

**from urllib.parse import urlparse**

**url = urlparse(request.META['HTTP\_REFERER']).path**

**from django.urls import resolve**

**try:**

**redirect\_url = resolve(url)**

**title = request.POST.get('title', 'No Name')**

**file = open(settings.ELECTION\_TITLE\_PATH, 'w')**

**file.write(title)**

**file.close()**

**messages.success(**

**request, "Election title has been changed to " + str(title))**

**return redirect(url)**

**except Exception as e:**

**messages.error(request, e)**

**return redirect("/")**

**def viewVotes(request):**

**votes = Votes.objects.all()**

**context = {**

**'votes': votes,**

**'page\_title': 'Votes'**

**}**

**return render(request, "admin/votes.html", context)**

**def resetVote(request):**

**Votes.objects.all().delete()**

**Voter.objects.all().update(voted=False, verified=False, otp=None)**

**messages.success(request, "All votes has been reset")**

**return redirect(reverse('viewVotes'))**

**from django.db import models**

**from account.models import CustomUser**

**# Create your models here.**

**class Voter(models.Model):**

**admin = models.OneToOneField(CustomUser, on\_delete=models.CASCADE)**

**phone = models.CharField(max\_length=11, unique=True) # Used for OTP**

**otp = models.CharField(max\_length=10, null=True)**

**verified = models.BooleanField(default=False)**

**voted = models.BooleanField(default=False)**

**otp\_sent = models.IntegerField(default=0) # Control how many OTPs are sent**

**def \_\_str\_\_(self):**

**return self.admin.last\_name + ", " + self.admin.first\_name**

**class Position(models.Model):**

**name = models.CharField(max\_length=50, unique=True)**

**max\_vote = models.IntegerField()**

**priority = models.IntegerField()**

**def \_\_str\_\_(self):**

**return self.name**

**class Candidate(models.Model):**

**fullname = models.CharField(max\_length=50)**

**photo = models.ImageField(upload\_to="candidates")**

**bio = models.TextField()**

**position = models.ForeignKey(Position, on\_delete=models.CASCADE)**

**def \_\_str\_\_(self):**

**return self.fullname**

**class Votes(models.Model):**

**voter = models.ForeignKey(Voter, on\_delete=models.CASCADE)**

**position = models.ForeignKey(Position, on\_delete=models.CASCADE)**

**candidate = models.ForeignKey(Candidate, on\_delete=models.CASCADE)**

**from django.shortcuts import render, redirect, reverse**

**from account.views import account\_login**

**from .models import Position, Candidate, Voter, Votes**

**from django.http import JsonResponse**

**from django.utils.text import slugify**

**from django.contrib import messages**

**from django.conf import settings**

**from django.http import JsonResponse**

**import requests**

**import json**

**# Create your views here.**

**def index(request):**

**if not request.user.is\_authenticated:**

**return account\_login(request)**

**context = {}**

**# return render(request, "voting/login.html", context)**

**def generate\_ballot(display\_controls=False):**

**positions = Position.objects.order\_by('priority').all()**

**output = ""**

**candidates\_data = ""**

**num = 1**

**# return None**

**for position in positions:**

**name = position.name**

**position\_name = slugify(name)**

**candidates = Candidate.objects.filter(position=position)**

**for candidate in candidates:**

**if position.max\_vote > 1:**

**instruction = "You may select up to " + \**

**str(position.max\_vote) + " candidates"**

**input\_box = '<input type="checkbox" value="'+str(candidate.id)+'" class="flat-red ' + \**

**position\_name+'" name="' + \**

**position\_name+"[]" + '">'**

**else:**

**instruction = "Select only one candidate"**

**input\_box = '<input value="'+str(candidate.id)+'" type="radio" class="flat-red ' + \**

**position\_name+'" name="'+position\_name+'">'**

**image = "/media/" + str(candidate.photo)**

**candidates\_data = candidates\_data + '<li>' + input\_box + '<button type="button" class="btn btn-primary btn-sm btn-flat clist platform" data-fullname="'+candidate.fullname+'" data-bio="'+candidate.bio+'"><i class="fa fa-search"></i> Platform</button><img src="' + \**

**image+'" height="100px" width="100px" class="clist"><span class="cname clist">' + \**

**candidate.fullname+'</span></li>'**

**up = ''**

**if position.priority == 1:**

**up = 'disabled'**

**down = ''**

**if position.priority == positions.count():**

**down = 'disabled'**

**output = output + f"""<div class="row"> <div class="col-xs-12"><div class="box box-solid" id="{position.id}">**

**<div class="box-header with-border">**

**<h3 class="box-title"><b>{name}</b></h3>"""**

**if display\_controls:**

**output = output + f""" <div class="pull-right box-tools">**

**<button type="button" class="btn btn-default btn-sm moveup" data-id="{position.id}" {up}><i class="fa fa-arrow-up"></i> </button>**

**<button type="button" class="btn btn-default btn-sm movedown" data-id="{position.id}" {down}><i class="fa fa-arrow-down"></i></button>**

**</div>"""**

**output = output + f"""</div>**

**<div class="box-body">**

**<p>{instruction}**

**<span class="pull-right">**

**<button type="button" class="btn btn-success btn-sm btn-flat reset" data-desc="{position\_name}"><i class="fa fa-refresh"></i> Reset</button>**

**</span>**

**</p>**

**<div id="candidate\_list">**

**<ul>**

**{candidates\_data}**

**</ul>**

**</div>**

**</div>**

**</div>**

**</div>**

**</div>**

**"""**

**position.priority = num**

**position.save()**

**num = num + 1**

**candidates\_data = ''**

**return output**

**def fetch\_ballot(request):**

**output = generate\_ballot(display\_controls=True)**

**return JsonResponse(output, safe=False)**

**def generate\_otp():**

**"""Link to this function**

**https://www.codespeedy.com/otp-generation-using-random-module-in-python/**

**"""**

**import random as r**

**otp = ""**

**for i in range(r.randint(5, 8)):**

**otp += str(r.randint(1, 9))**

**return otp**

**def dashboard(request):**

**user = request.user**

**# \* Check if this voter has been verified**

**if user.voter.otp is None or user.voter.verified == False:**

**if not settings.SEND\_OTP:**

**# Bypass**

**msg = bypass\_otp()**

**messages.success(request, msg)**

**return redirect(reverse('show\_ballot'))**

**else:**

**return redirect(reverse('voterVerify'))**

**else:**

**if user.voter.voted: # \* User has voted**

**# To display election result or candidates I voted for ?**

**context = {**

**'my\_votes': Votes.objects.filter(voter=user.voter),**

**}**

**return render(request, "voting/voter/result.html", context)**

**else:**

**return redirect(reverse('show\_ballot'))**

**def verify(request):**

**context = {**

**'page\_title': 'OTP Verification'**

**}**

**return render(request, "voting/voter/verify.html", context)**

**def resend\_otp(request):**

**"""API For SMS**

**I used https://www.multitexter.com/ API to send SMS**

**You might not want to use this or this service might not be available in your Country**

**For quick and easy access, Toggle the SEND\_OTP from True to False in settings.py**

**"""**

**user = request.user**

**voter = user.voter**

**error = False**

**if settings.SEND\_OTP:**

**if voter.otp\_sent >= 3:**

**error = True**

**response = "You have requested OTP three times. You cannot do this again! Please enter previously sent OTP"**

**else:**

**phone = voter.phone**

**# Now, check if an OTP has been generated previously for this voter**

**otp = voter.otp**

**if otp is None:**

**# Generate new OTP**

**otp = generate\_otp()**

**voter.otp = otp**

**voter.save()**

**try:**

**msg = "Dear " + str(user) + ", kindly use " + \**

**str(otp) + " as your OTP"**

**message\_is\_sent = send\_sms(phone, msg)**

**if message\_is\_sent: # \* OTP was sent successfully**

**# Update how many OTP has been sent to this voter**

**# Limited to Three so voters don't exhaust OTP balance**

**voter.otp\_sent = voter.otp\_sent + 1**

**voter.save()**

**response = "OTP has been sent to your phone number. Please provide it in the box provided below"**

**else:**

**error = True**

**response = "OTP not sent. Please try again"**

**except Exception as e:**

**response = "OTP could not be sent." + str(e)**

**# \* Send OTP**

**else:**

**#! Update all Voters record and set OTP to 0000**

**#! Bypass OTP verification by updating verified to 1**

**#! Redirect voters to ballot page**

**response = bypass\_otp()**

**return JsonResponse({"data": response, "error": error})**

**def bypass\_otp():**

**Voter.objects.all().filter(otp=None, verified=False).update(otp="0000", verified=True)**

**response = "Kindly cast your vote"**

**return response**

**def send\_sms(phone\_number, msg):**

**"""Read More**

**https://www.multitexter.com/developers**

**"""**

**import requests**

**import os**

**import json**

**response = ""**

**email = os.environ.get('SMS\_EMAIL')**

**password = os.environ.get('SMS\_PASSWORD')**

**if email is None or password is None:**

**raise Exception("Email/Password cannot be Null")**

**url = "https://app.multitexter.com/v2/app/sms"**

**data = {"email": email, "password": password, "message": msg,**

**"sender\_name": "OTP", "recipients": phone\_number, "forcednd": 1}**

**headers = {'Content-type': 'application/json', 'Accept': 'text/plain'}**

**r = requests.post(url, data=json.dumps(data), headers=headers)**

**response = r.json()**

**status = response.get('status', 0)**

**if str(status) == '1':**

**return True**

**else:**

**return False**

**def verify\_otp(request):**

**error = True**

**if request.method != 'POST':**

**messages.error(request, "Access Denied")**

**else:**

**otp = request.POST.get('otp')**

**if otp is None:**

**messages.error(request, "Please provide valid OTP")**

**else:**

**# Get User OTP**

**voter = request.user.voter**

**db\_otp = voter.otp**

**if db\_otp != otp:**

**messages.error(request, "Provided OTP is not valid")**

**else:**

**messages.success(**

**request, "You are now verified. Please cast your vote")**

**voter.verified = True**

**voter.save()**

**error = False**

**if error:**

**return redirect(reverse('voterVerify'))**

**return redirect(reverse('show\_ballot'))**

**def show\_ballot(request):**

**if request.user.voter.voted:**

**messages.error(request, "You have voted already")**

**return redirect(reverse('voterDashboard'))**

**ballot = generate\_ballot(display\_controls=False)**

**context = {**

**'ballot': ballot**

**}**

**return render(request, "voting/voter/ballot.html", context)**

**def preview\_vote(request):**

**if request.method != 'POST':**

**error = True**

**response = "Please browse the system properly"**

**else:**

**output = ""**

**form = dict(request.POST)**

**# We don't need to loop over CSRF token**

**form.pop('csrfmiddlewaretoken', None)**

**error = False**

**data = []**

**positions = Position.objects.all()**

**for position in positions:**

**max\_vote = position.max\_vote**

**pos = slugify(position.name)**

**pos\_id = position.id**

**if position.max\_vote > 1:**

**this\_key = pos + "[]"**

**form\_position = form.get(this\_key)**

**if form\_position is None:**

**continue**

**if len(form\_position) > max\_vote:**

**error = True**

**response = "You can only choose " + \**

**str(max\_vote) + " candidates for " + position.name**

**else:**

**# for key, value in form.items():**

**start\_tag = f"""**

**<div class='row votelist' style='padding-bottom: 2px'>**

**<span class='col-sm-4'><span class='pull-right'><b>{position.name} :</b></span></span>**

**<span class='col-sm-8'>**

**<ul style='list-style-type:none; margin-left:-40px'>**

**"""**

**end\_tag = "</ul></span></div><hr/>"**

**data = ""**

**for form\_candidate\_id in form\_position:**

**try:**

**candidate = Candidate.objects.get(**

**id=form\_candidate\_id, position=position)**

**data += f"""**

**<li><i class="fa fa-check-square-o"></i> {candidate.fullname}</li>**

**"""**

**except:**

**error = True**

**response = "Please, browse the system properly"**

**output += start\_tag + data + end\_tag**

**else:**

**this\_key = pos**

**form\_position = form.get(this\_key)**

**if form\_position is None:**

**continue**

**# Max Vote == 1**

**try:**

**form\_position = form\_position[0]**

**candidate = Candidate.objects.get(**

**position=position, id=form\_position)**

**output += f"""**

**<div class='row votelist' style='padding-bottom: 2px'>**

**<span class='col-sm-4'><span class='pull-right'><b>{position.name} :</b></span></span>**

**<span class='col-sm-8'><i class="fa fa-check-circle-o"></i> {candidate.fullname}</span>**

**</div>**

**<hr/>**

**"""**

**except Exception as e:**

**error = True**

**response = "Please, browse the system properly"**

**context = {**

**'error': error,**

**'list': output**

**}**

**return JsonResponse(context, safe=False)**

**def submit\_ballot(request):**

**if request.method != 'POST':**

**messages.error(request, "Please, browse the system properly")**

**return redirect(reverse('show\_ballot'))**

**# Verify if the voter has voted or not**

**voter = request.user.voter**

**if voter.voted:**

**messages.error(request, "You have voted already")**

**return redirect(reverse('voterDashboard'))**

**form = dict(request.POST)**

**form.pop('csrfmiddlewaretoken', None) # Pop CSRF Token**

**form.pop('submit\_vote', None) # Pop Submit Button**

**# Ensure at least one vote is selected**

**if len(form.keys()) < 1:**

**messages.error(request, "Please select at least one candidate")**

**return redirect(reverse('show\_ballot'))**

**positions = Position.objects.all()**

**form\_count = 0**

**for position in positions:**

**max\_vote = position.max\_vote**

**pos = slugify(position.name)**

**pos\_id = position.id**

**if position.max\_vote > 1:**

**this\_key = pos + "[]"**

**form\_position = form.get(this\_key)**

**if form\_position is None:**

**continue**

**if len(form\_position) > max\_vote:**

**messages.error(request, "You can only choose " +**

**str(max\_vote) + " candidates for " + position.name)**

**return redirect(reverse('show\_ballot'))**

**else:**

**for form\_candidate\_id in form\_position:**

**form\_count += 1**

**try:**

**candidate = Candidate.objects.get(**

**id=form\_candidate\_id, position=position)**

**vote = Votes()**

**vote.candidate = candidate**

**vote.voter = voter**

**vote.position = position**

**vote.save()**

**except Exception as e:**

**messages.error(**

**request, "Please, browse the system properly " + str(e))**

**return redirect(reverse('show\_ballot'))**

**else:**

**this\_key = pos**

**form\_position = form.get(this\_key)**

**if form\_position is None:**

**continue**

**# Max Vote == 1**

**form\_count += 1**

**try:**

**form\_position = form\_position[0]**

**candidate = Candidate.objects.get(**

**position=position, id=form\_position)**

**vote = Votes()**

**vote.candidate = candidate**

**vote.voter = voter**

**vote.position = position**

**vote.save()**

**except Exception as e:**

**messages.error(**

**request, "Please, browse the system properly " + str(e))**

**return redirect(reverse('show\_ballot'))**

**# Count total number of records inserted**

**# Check it viz-a-viz form\_count**

**inserted\_votes = Votes.objects.filter(voter=voter)**

**if (inserted\_votes.count() != form\_count):**

**# Delete**

**inserted\_votes.delete()**

**messages.error(request, "Please try voting again!")**

**return redirect(reverse('show\_ballot'))**

**else:**

**# Update Voter profile to voted**

**voter.voted = True**

**voter.save()**

**messages.success(request, "Thanks for voting")**

**return redirect(reverse('voterDashboard'))**

**from django import forms**

**from .models import \***

**from account.forms import FormSettings**

**class VoterForm(FormSettings):**

**class Meta:**

**model = Voter**

**fields = ['phone']**

**class PositionForm(FormSettings):**

**class Meta:**

**model = Position**

**fields = ['name', 'max\_vote']**

**class CandidateForm(FormSettings):**

**class Meta:**

**model = Candidate**

**fields = ['fullname', 'bio', 'position', 'photo']**

**from django.contrib.auth.backends import ModelBackend**

**from django.contrib.auth import get\_user\_model**

**class EmailBackend(ModelBackend):**

**def authenticate(self, username=None, password=None, \*\*kwargs):**

**UserModel = get\_user\_model()**

**try:**

**user = UserModel.objects.get(email=username)**

**except UserModel.DoesNotExist:**

**return None**

**else:**

**if user.check\_password(password):**

**return user**

**return None**

**from django.utils.deprecation import MiddlewareMixin**

**from django.urls import reverse**

**from django.shortcuts import redirect**

**from django.contrib import messages**

**class AccountCheckMiddleWare(MiddlewareMixin):**

**def process\_view(self, request, view\_func, view\_args, view\_kwargs):**

**modulename = view\_func.\_\_module\_\_**

**user = request.user # Who is the current user ?**

**if user.is\_authenticated:**

**if user.user\_type == '1': # Admin**

**if modulename == 'voting.views':**

**error = True**

**if request.path == reverse('fetch\_ballot'):**

**pass**

**else:**

**messages.error(**

**request, "You do not have access to this resource")**

**return redirect(reverse('adminDashboard'))**

**elif user.user\_type == '2': # Voter**

**if modulename == 'administrator.views':**

**messages.error(**

**request, "You do not have access to this resource")**

**return redirect(reverse('voterDashboard'))**

**else: # None of the aforementioned ? Please take the user to login page**

**return redirect(reverse('account\_login'))**

**else:**

**# If the path is login or has anything to do with authentication, pass**

**if request.path == reverse('account\_login') or request.path == reverse('account\_register') or modulename == 'django.contrib.auth.views' or request.path == reverse('account\_login'):**

**pass**

**elif modulename == 'administrator.views' or modulename == 'voting.views':**

**# If visitor tries to access administrator or voters functions**

**messages.error(**

**request, "You need to be logged in to perform this operation")**

**return redirect(reverse('account\_login'))**

**else:**

**return redirect(reverse('account\_login'))**

**#!/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', 'e\_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()**

**# Generated by Django 5.0.6 on 2024-07-04 12:46**

**import account.models**

**import django.utils.timezone**

**from django.db import migrations, models**

**class Migration(migrations.Migration):**

**initial = True**

**dependencies = [**

**('auth', '0012\_alter\_user\_first\_name\_max\_length'),**

**]**

**operations = [**

**migrations.CreateModel(**

**name='CustomUser',**

**fields=[**

**('id', models.AutoField(auto\_created=True, primary\_key=True, serialize=False, verbose\_name='ID')),**

**('password', models.CharField(max\_length=128, verbose\_name='password')),**

**('last\_login', models.DateTimeField(blank=True, null=True, verbose\_name='last login')),**

**('is\_superuser', models.BooleanField(default=False, help\_text='Designates that this user has all permissions without explicitly assigning them.', verbose\_name='superuser status')),**

**('first\_name', models.CharField(blank=True, max\_length=150, verbose\_name='first name')),**

**('last\_name', models.CharField(blank=True, max\_length=150, verbose\_name='last name')),**

**('is\_staff', models.BooleanField(default=False, help\_text='Designates whether the user can log into this admin site.', verbose\_name='staff status')),**

**('is\_active', models.BooleanField(default=True, help\_text='Designates whether this user should be treated as active. Unselect this instead of deleting accounts.', verbose\_name='active')),**

**('date\_joined', models.DateTimeField(default=django.utils.timezone.now, verbose\_name='date joined')),**

**('email', models.EmailField(max\_length=254, unique=True)),**

**('user\_type', models.CharField(choices=[(1, 'Admin'), (2, 'Voter')], default=2, max\_length=1)),**

**('created\_at', models.DateTimeField(auto\_now\_add=True)),**

**('updated\_at', models.DateTimeField(auto\_now=True)),**

**('groups', models.ManyToManyField(blank=True, help\_text='The groups this user belongs to. A user will get all permissions granted to each of their groups.', related\_name='user\_set', related\_query\_name='user', to='auth.group', verbose\_name='groups')),**

**('user\_permissions', models.ManyToManyField(blank=True, help\_text='Specific permissions for this user.', related\_name='user\_set', related\_query\_name='user', to='auth.permission', verbose\_name='user permissions')),**

**],**

**options={**

**'verbose\_name': 'user',**

**'verbose\_name\_plural': 'users',**

**'abstract': False,**

**},**

**managers=[**

**('objects', account.models.CustomUserManager()),**

**],**

**),**

**]**

**"""**

**Activate virtualenv for current interpreter:**

**Use exec(open(this\_file).read(), {'\_\_file\_\_': this\_file}).**

**This can be used when you must use an existing Python interpreter, not the virtualenv bin/python.**

**""" # noqa: D415**

**from \_\_future\_\_ import annotations**

**import os**

**import site**

**import sys**

**try:**

**abs\_file = os.path.abspath(\_\_file\_\_)**

**except NameError as exc:**

**msg = "You must use exec(open(this\_file).read(), {'\_\_file\_\_': this\_file}))"**

**raise AssertionError(msg) from exc**

**bin\_dir = os.path.dirname(abs\_file)**

**base = bin\_dir[: -len("bin") - 1] # strip away the bin part from the \_\_file\_\_, plus the path separator**

**# prepend bin to PATH (this file is inside the bin directory)**

**os.environ["PATH"] = os.pathsep.join([bin\_dir, \*os.environ.get("PATH", "").split(os.pathsep)])**

**os.environ["VIRTUAL\_ENV"] = base # virtual env is right above bin directory**

**os.environ["VIRTUAL\_ENV\_PROMPT"] = "" or os.path.basename(base) # noqa: SIM222**

**# add the virtual environments libraries to the host python import mechanism**

**prev\_length = len(sys.path)**

**for lib in "../lib/python3.10/site-packages".split(os.pathsep):**

**path = os.path.realpath(os.path.join(bin\_dir, lib))**

**site.addsitedir(path.decode("utf-8") if "" else path)**

**sys.path[:] = sys.path[prev\_length:] + sys.path[0:prev\_length]**

**sys.real\_prefix = sys.prefix**

**sys.prefix = base**

**"""Patches that are applied at runtime to the virtual environment."""**

**from \_\_future\_\_ import annotations**

**import os**

**import sys**

**from contextlib import suppress**

**VIRTUALENV\_PATCH\_FILE = os.path.join(\_\_file\_\_)**

**def patch\_dist(dist):**

**"""**

**Distutils allows user to configure some arguments via a configuration file:**

**https://docs.python.org/3/install/index.html#distutils-configuration-files.**

**Some of this arguments though don't make sense in context of the virtual environment files, let's fix them up.**

**""" # noqa: D205**

**# we cannot allow some install config as that would get packages installed outside of the virtual environment**

**old\_parse\_config\_files = dist.Distribution.parse\_config\_files**

**def parse\_config\_files(self, \*args, \*\*kwargs):**

**result = old\_parse\_config\_files(self, \*args, \*\*kwargs)**

**install = self.get\_option\_dict("install")**

**if "prefix" in install: # the prefix governs where to install the libraries**

**install["prefix"] = VIRTUALENV\_PATCH\_FILE, os.path.abspath(sys.prefix)**

**for base in ("purelib", "platlib", "headers", "scripts", "data"):**

**key = f"install\_{base}"**

**if key in install: # do not allow global configs to hijack venv paths**

**install.pop(key, None)**

**return result**

**dist.Distribution.parse\_config\_files = parse\_config\_files**

**# Import hook that patches some modules to ignore configuration values that break package installation in case**

**# of virtual environments.**

**\_DISTUTILS\_PATCH = "distutils.dist", "setuptools.dist"**

**# https://docs.python.org/3/library/importlib.html#setting-up-an-importer**

**class \_Finder:**

**"""A meta path finder that allows patching the imported distutils modules."""**

**fullname = None**

**# lock[0] is threading.Lock(), but initialized lazily to avoid importing threading very early at startup,**

**# because there are gevent-based applications that need to be first to import threading by themselves.**

**# See https://github.com/pypa/virtualenv/issues/1895 for details.**

**lock = [] # noqa: RUF012**

**def find\_spec(self, fullname, path, target=None): # noqa: ARG002**

**if fullname in \_DISTUTILS\_PATCH and self.fullname is None:**

**# initialize lock[0] lazily**

**if len(self.lock) == 0:**

**import threading**

**lock = threading.Lock()**

**# there is possibility that two threads T1 and T2 are simultaneously running into find\_spec,**

**# observing .lock as empty, and further going into hereby initialization. However due to the GIL,**

**# list.append() operation is atomic and this way only one of the threads will "win" to put the lock**

**# - that every thread will use - into .lock[0].**

**# https://docs.python.org/3/faq/library.html#what-kinds-of-global-value-mutation-are-thread-safe**

**self.lock.append(lock)**

**from functools import partial**

**from importlib.util import find\_spec**

**with self.lock[0]:**

**self.fullname = fullname**

**try:**

**spec = find\_spec(fullname, path)**

**if spec is not None:**

**# https://www.python.org/dev/peps/pep-0451/#how-loading-will-work**

**is\_new\_api = hasattr(spec.loader, "exec\_module")**

**func\_name = "exec\_module" if is\_new\_api else "load\_module"**

**old = getattr(spec.loader, func\_name)**

**func = self.exec\_module if is\_new\_api else self.load\_module**

**if old is not func:**

**with suppress(AttributeError): # C-Extension loaders are r/o such as zipimporter with <3.7**

**setattr(spec.loader, func\_name, partial(func, old))**

**return spec**

**finally:**

**self.fullname = None**

**return None**

**@staticmethod**

**def exec\_module(old, module):**

**old(module)**

**if module.\_\_name\_\_ in \_DISTUTILS\_PATCH:**

**patch\_dist(module)**

**@staticmethod**

**def load\_module(old, name):**

**module = old(name)**

**if module.\_\_name\_\_ in \_DISTUTILS\_PATCH:**

**patch\_dist(module)**

**return module**

**sys.meta\_path.insert(0, \_Finder())**