# ACKNOWLEDGEMENT

We would like to express our sincere gratitude to our supervisor, **Ms Sarita Neupane** for her invaluable guidance, encouragement, and continuous support throughout the development of our project, “**DeVote – A Decentralized Voting System using Blockchain.**” Her insightful feedback and constructive suggestions were crucial in shaping our project and enhancing our learning.

We are grateful to the Department of Information Technology for providing the resources, technical support, and a positive environment that enabled us to carry out this project effectively.

We would also like to thank our friends and classmates for their encouragement and constructive discussions during the development process.

Finally, we express our heartfelt thanks to our families for their constant motivation and support throughout this academic journey.

c

# ABSTRACT

DeVote is a decentralized electronic voting system using blockchain to ensure secure, transparent, and immutable elections. It allows eligible voters to register and vote using their email and date of birth, ensuring only authorized users can vote while maintaining anonymity.

The system uses React with Vite for the frontend and Go-based blockchain with PBFT consensus for the backend, ensuring transparency and tamper-resistance in vote recording and counting. The blockchain core functionality is implemented and tested, while frontend development and systematic testing are in progress.

This project aims to provide a practical, secure alternative to traditional voting systems in Nepal.

***Keywords:***

**TABLE OF CONTENTS**

[ACKNOWLEDGEMENT i](#_Toc203225287)

[ABSTRACT ii](#_Toc203225288)

[LIST OF TABLES v](#_Toc203225289)

[LIST OF FIGURES v](#_Toc203225290)

[LIST OF ABBREVIATIONS v](#_Toc203225291)

[1. INTRODUCTION 1](#_Toc203225292)

[1.1 Introduction 1](#_Toc203225293)

[1.1 Problem Statement 1](#_Toc203225294)

[1.2 Objectives 1](#_Toc203225295)

[1.3 Scope and Limitation 1](#_Toc203225296)

[1.4 Development Methodology 2](#_Toc203225297)

[1.5 Report Organizations 2](#_Toc203225298)

[2. BACKGROUND STUDY AND LITERATURE REVIEW 4](#_Toc203225299)

[2.1 Background Study 4](#_Toc203225300)

[2.2 Literature Review 4](#_Toc203225301)

[3. SYSTEM ANALYSIS 5](#_Toc203225302)

[3.1 Requirement Analysis 5](#_Toc203225303)

[3.1.1 Functional Requirements 5](#_Toc203225304)

[Use Case Table: 6](#_Toc203225305)

[3.1.2 Non-functional Requirements 7](#_Toc203225306)

[3.2 FEASIBILITY ANALYSIS 7](#_Toc203225307)

[3.2.1 Technical Feasibilty 7](#_Toc203225308)

[3.2.2 Operational Feasibilty 7](#_Toc203225309)

[3.2.3 Economic Feasibilty 7](#_Toc203225310)

[3.2.4 Schedule Feasibilty 7](#_Toc203225311)

[3.3 ANALYSIS 7](#_Toc203225312)

[3.3.1 Class Diagram 7](#_Toc203225313)

[3.3.2 Sequence Diagram 7](#_Toc203225314)

[4. SYSTEM DESIGN 8](#_Toc203225315)

[4.1 Design 8](#_Toc203225316)

[4.1.1 System Architecture 8](#_Toc203225317)

[4.1.4 Flowchart of the working mechanisms 9](#_Toc203225318)

[4.2 Algorithm Details 9](#_Toc203225319)

[5. IMPLEMENTATION AND TESTING 10](#_Toc203225320)

[5.1 Implementation 10](#_Toc203225321)

[5.1.1 Tools Used 10](#_Toc203225322)

[5.1.2 Implementation Detail 10](#_Toc203225323)

[5.2 Testing 10](#_Toc203225324)

[5.2.1 Unit Testing 10](#_Toc203225325)

[5.2.2 Test Case for System Testing 11](#_Toc203225326)

[5.3 Result Analysis 11](#_Toc203225327)

[6. CONCLUSION 11](#_Toc203225328)

[6.1 Progress 11](#_Toc203225329)

[6.2 Ongoing 11](#_Toc203225330)

[6.3 Remaining 11](#_Toc203225331)

[REFERENCES 12](#_Toc203225332)

[APPENDICES 13](#_Toc203225333)

[SCREENSHOTS 13](#_Toc203225334)

[Homepage 13](#_Toc203225335)

[Login Form 13](#_Toc203225336)

[Register Form 14](#_Toc203225337)

[Admin Dashboard 14](#_Toc203225338)

[SOURCE CODE 15](#_Toc203225339)

# LIST OF TABLES

[Table 1: Use Case of DeVote 6](#_Toc203127290)

# LIST OF FIGURES

[Figure i:Use case diagram of DeVote 5](#_Toc203126952)

[Figure ii: Sequence Diagram of DeVote 8](#_Toc203126953)

[Figure iii: Homepage 15](#_Toc203126954)

[Figure iv: Login Page 15](#_Toc203126955)

[Figure v: Register 16](#_Toc203126956)

[Figure vi: Admin Dashboard 16](#_Toc203126957)

[Figure vii: User Dashboard 17](#_Toc203126958)

# LIST OF ABBREVIATIONS

# INTRODUCTION

## 1.1 Introduction

In the past few years, the need for a secure, tamper-proof, and transparent voting is increasing day by day. Traditional voting methods are prone to manipulation, inefficiency, and lack of trust among the voters. To address these challenges, this project proposes DeVote, a voting system based on blockchain technology. DeVote aims to provide a secure vote casting, prevent vote tampering, and provide vote verifiability to ensure every voter’s vote is registered in the blockchain.

## 1.1 Problem Statement

Existing voting systems which are paper-based are centralized which often lacks transparency, are vulnerable to fraud and provides insufficient vote verifiability. Therefore, a modern system that guarantees vote integrity, provides anonymity, and ensures only eligible voters can vote once is necessary.

## 1.2 Objectives

* To develop a secure, user-friendly blockchain voting system with vote immutability, voter anonymity (zk-SNARKs), verifiability (Merkle trees), and Voter ID-based authentication with decentralized validation (PBFT).

## 1.3 Scope and Limitation

This project focuses on developing a voting system using blockchain that ensures secure, tamper-proof and verifiability. The key areas covered in this project are:

**Voter Registration:** The system allows new user to register using email, voter-Id, and password. It checks whether the voter ID exists in the pre-defined voter list to verify eligibility.

**Login:** Users that are registered in the system can login using their verified credentials like email and password.

**Blockchain based voting storage:** Votes are securely stored in the blockchain to ensure immutability and transparency.

**Anonymity and Privacy**: In the blockchain, the voter’s identities are not linked with their votes where the system ensures privacy through cryptographic measures and hashing.

**One Voter, one vote**: The system ensures only registered user can vote only once.

**Admin Dashboard for Candidate Management**: Admin users can add, update, and delete candidates through a secure admin interface.

**Result Generation and Verification**: After the voting ends, the system generates results based on the votes recorded in the blockchain and allows the users to verify their vote using Merkle tree.

**No integration with Government Voter databases**: The system uses a static internal voter list for verification. It does not connect to real-time national or governmental databases for Voter ID validation.

**Limited Scalability**: The system is built for small to medium-scale elections such as academic institutions or local elections.

## 1.4 Development Methodology

## 1.5 Report Organizations

The project report is divided into 6 chapters which is further divided into different headings.

#### Chapter 1

In the Introduction section, an introduction of the project is given. It also includes objectives, problem statements, scope and limitation, and methodology.

#### Chapter 2

In the background study and literature review section, general concepts and terms that are necessary for understanding the system is reviewed and previous research done in this field of study is discussed which is the starting point of this project.

#### Chapter 3

In the System Analysis section, the architecture and requirements of the system is discussed by breaking down a system into components, analyzing their interactions, and identifying important modules. This uses Use case diagram, Entity Relationship Diagrams, Context Diagram and Data Flow Diagrams. A feasibility study is present for understanding the future requirements and external factors that affect the project.

#### Chapter 4

In the System Design section, the system is described in detail. The system architecture demonstrates the entire system structure, the data dictionary includes the structure of all the data that is stored in the system, and the flow charts illustrate important processes in the system.

#### Chapter 5

In Implementation and Testing section, description of procedure, methods, algorithm is included alongside the explanation of functionalities of the system as different modules. including the tools that are used to build the system and tools that support the development. It also includes unit and system testing used to validate system’s functioning.

#### Chapter 6

In Conclusion, the progress of the project is described along with current ongoing development process and remaining work on the project.

# 2. BACKGROUND STUDY AND LITERATURE REVIEW

## 2.1 Background Study

## 2.2 Literature Review

# 3. SYSTEM ANALYSIS

## 3.1 Requirement Analysis

### 3.1.1 Functional Requirements

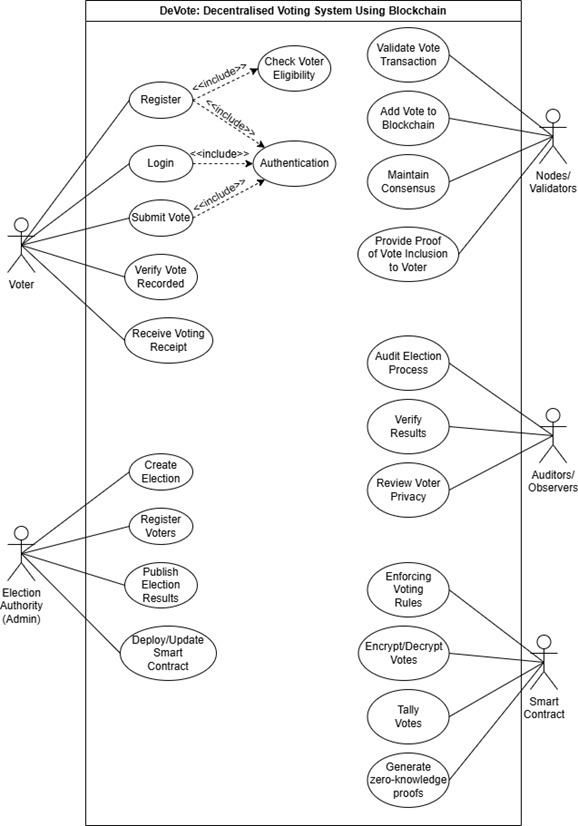


Figure 3.1.1: Use case diagram of DeVote

## Use Case Table:

**Table 1: Use Case of DeVote**

|  |  |
| --- | --- |
| **Use Case** | **Description** |
| Register | Allows a voter to create an account by entering email and voter id. |
| Login | Enables the voter to login into the system using verified username and password. |
| Submit Vote | Allows a verified voter to cast their vote for a selected candidate. |
| Verify Vote recorded | Allows the voter to check the voter whether their vote was successfully recorded in the system. |
| Create Election | Allows the admin to initiate new election process. |
| Registered voters | The admin can upload or register eligible voters to the system’s internal voter list. |
| Publish Election result | Enables the admin to release the final vote tally to the public. |
| Check voter eligibility | Checks if a registering user id exists in the system voter list. |
| Authentication | Verifies voter identities during registration and login. |
| Validate vote transaction | Validators ensure the vote transaction is valid and follows system rule |
| Add vote to blockchain | Records the verified vote into the blockchain securely |
| Maintain consensus | Ensures all the nodes/validators agree on the blockchain state. |
| Proof of vote inclusion | Sends a cryptographic proof (Merkle path) to ensure the voter their vote is included. |
| Audit election process | Auditors/ viewers reviews the fairness and integrity of the voting process. |
| Verify results | Auditors validate the correctness of the final election result |
| Review voter privacy | Ensure that voter identity is protected and not traceable. |
| Encrypt/Decrypt votes | Votes are encrypted before being stored and decrypted for counting security. |
| Tally votes | After voting ends, votes are counted to determine the final results. |
| Generate z-k proofs | Used to prove vote validity and inclusion without revealing voter’s identity. |

### 3.1.2 Non-functional Requirements

## 3.2 FEASIBILITY ANALYSIS

### 3.2.1 Technical Feasibilty

### 3.2.2 Operational Feasibilty

### 3.2.3 Economic Feasibilty

### 3.2.4 Schedule Feasibilty

## 3.3 ANALYSIS

### 3.3.1 Class Diagram

### 3.3.2 Sequence Diagram

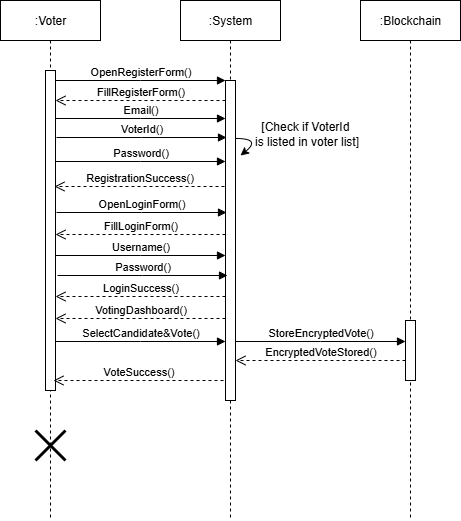


Figure 3.3.1 : Sequence Diagram of DeVote

# 4. SYSTEM DESIGN

## 4.1 Design

### 4.1.1 System Architecture

### 4.1.4 Flowchart of the working mechanisms

## 4.2 Algorithm Details

**4.2.1 Secure Hash Algorithm-256**



# 5. IMPLEMENTATION AND TESTING

## 5.1 Implementation

### 5.1.1 Tools Used

The following tools and technologies were utilized for the implementation of this project:

1. **Visual Studio Code**: It is IDE which is used for writing and debugging code for both frontend and backend. It supports extensions for Go, ReactJS, and version control integration.
2. **Web Browser Dev Tool**: Web developer tools are employed to inspect API calls, debug JavaScript, and verify correct data between the front end and backend.
3. **Vite**: Vite is used as the development build tool, offering fast development server startup times and efficient hot-module replacement (HMR), allowing for rapid iterations on the front-end UI during development.
4. **Git**: Git is used for version control, allowing for efficient collaboration, tracking changes, and maintaining a clean history of development. It ensures that all updates, whether in the front-end (ReactJS/TypeScript), backend (Go), or infrastructure (Docker/PostgreSQL configurations), are properly managed, reviewed, and deployed.
5. **React**:
6. **Go**:
7. **Postman**:

### 5.1.2 Implementation Detail

## 5.2 Testing

### 5.2.1 Unit Testing

The purpose of unit test is to validate that the different functions in a module are behaving according to the specification.

### 5.2.2 Test Case for System Testing

## 5.3 Result Analysis

# 6. CONCLUSION

## 6.1 Progress

## 6.2 Ongoing

## 6.3 Remaining

# REFERENCES

# APPENDICES

## SCREENSHOTS

### Homepage

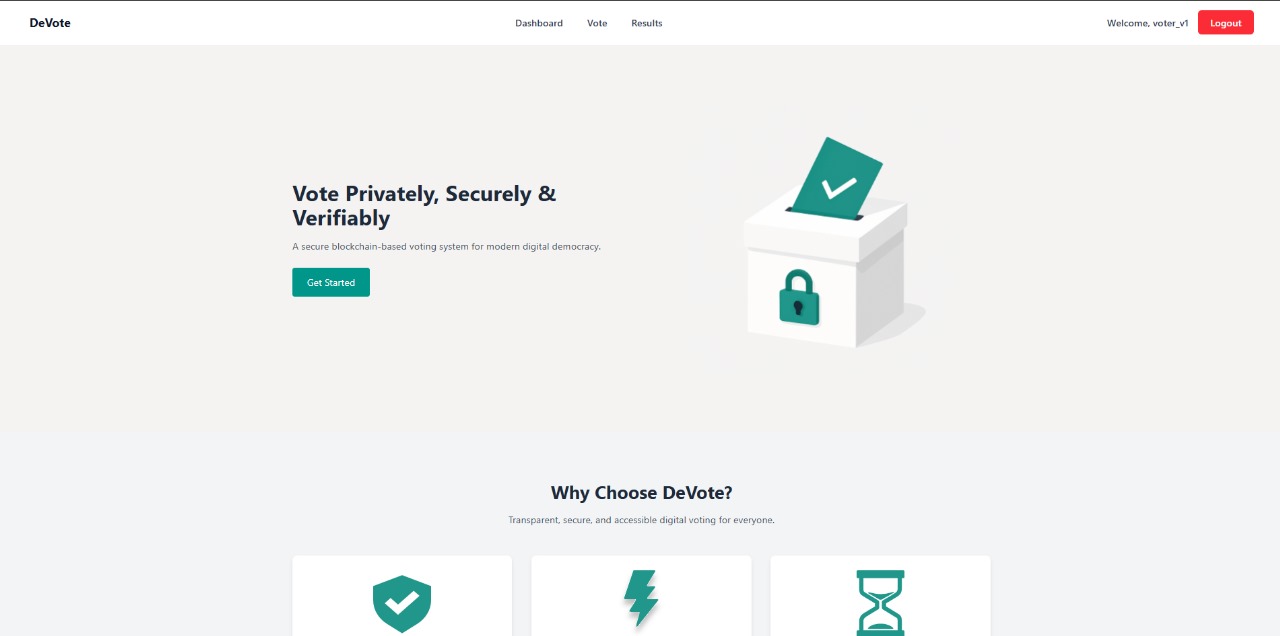


Figure iii: Homepage

### Login Form

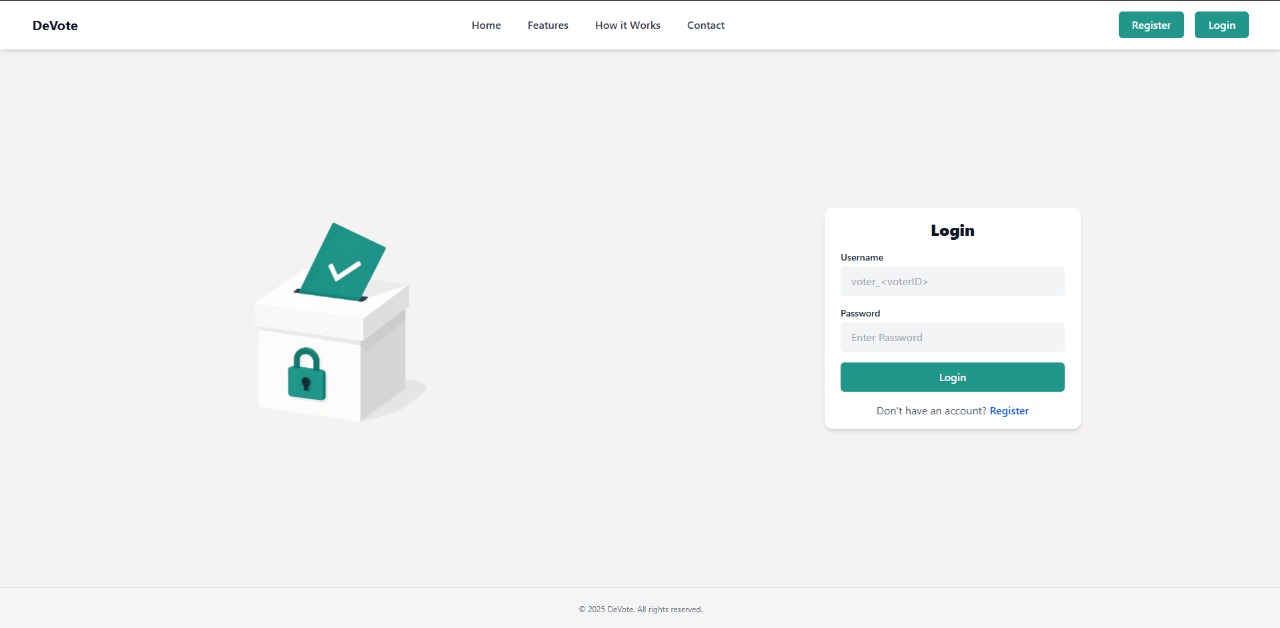


Figure iv: Login Page

### Register Form

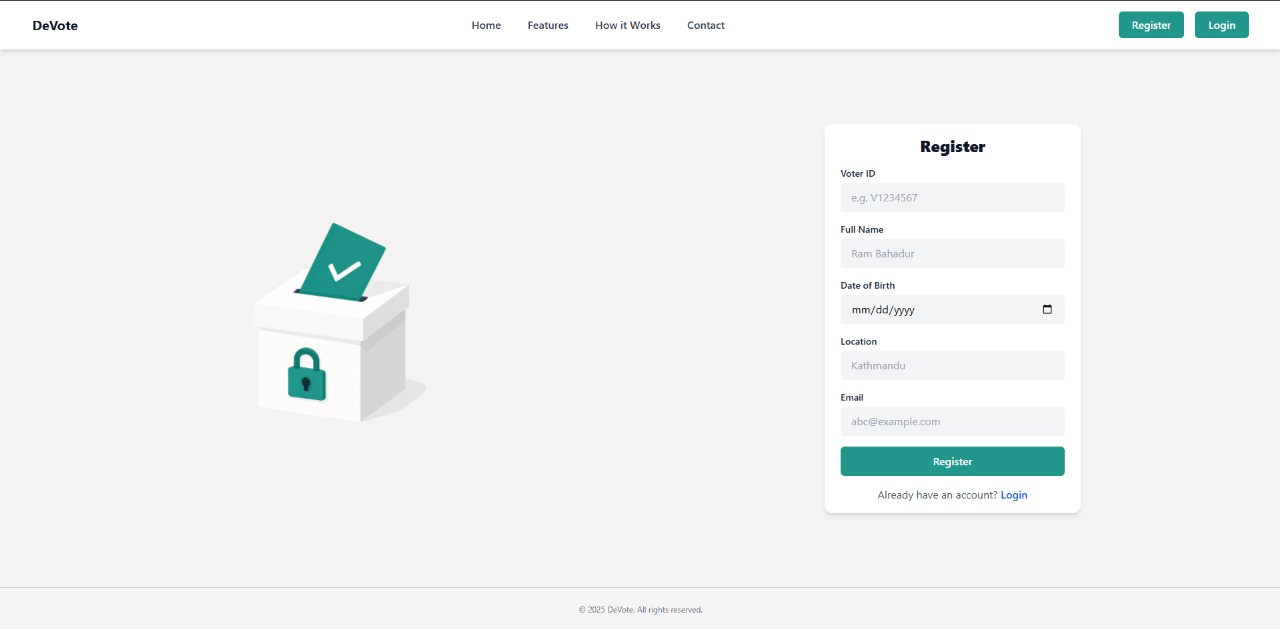


Figure v: Register

### Admin Dashboard

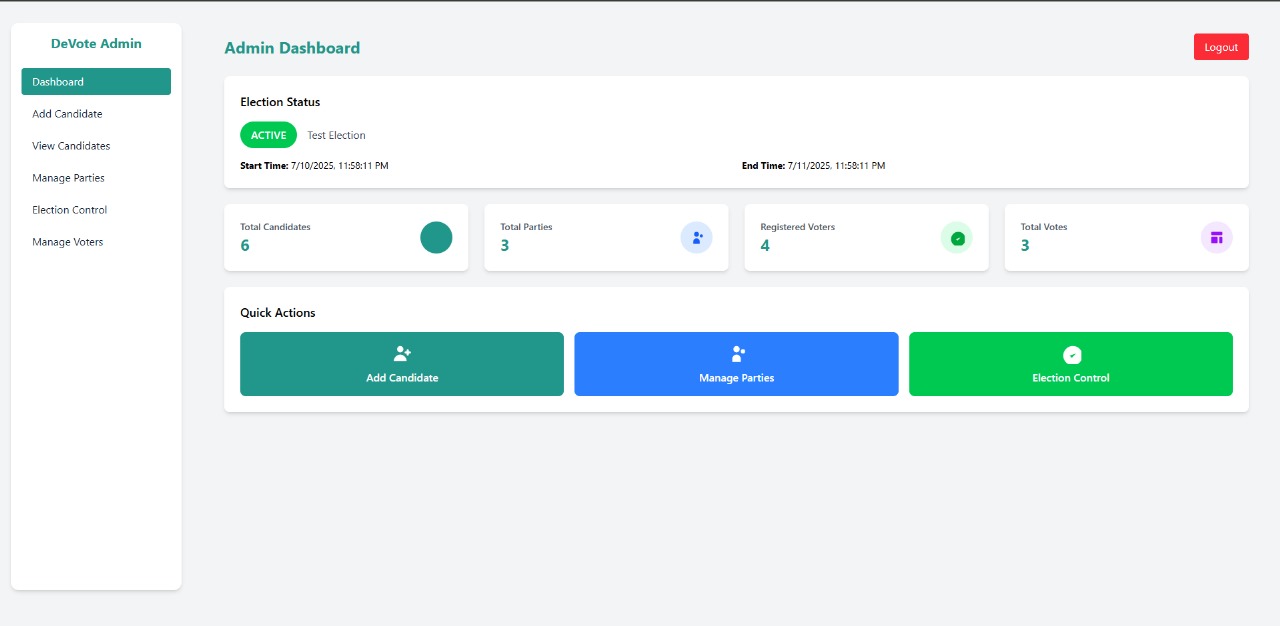


Figure vi: Admin Dashboard

**User Dashboard**

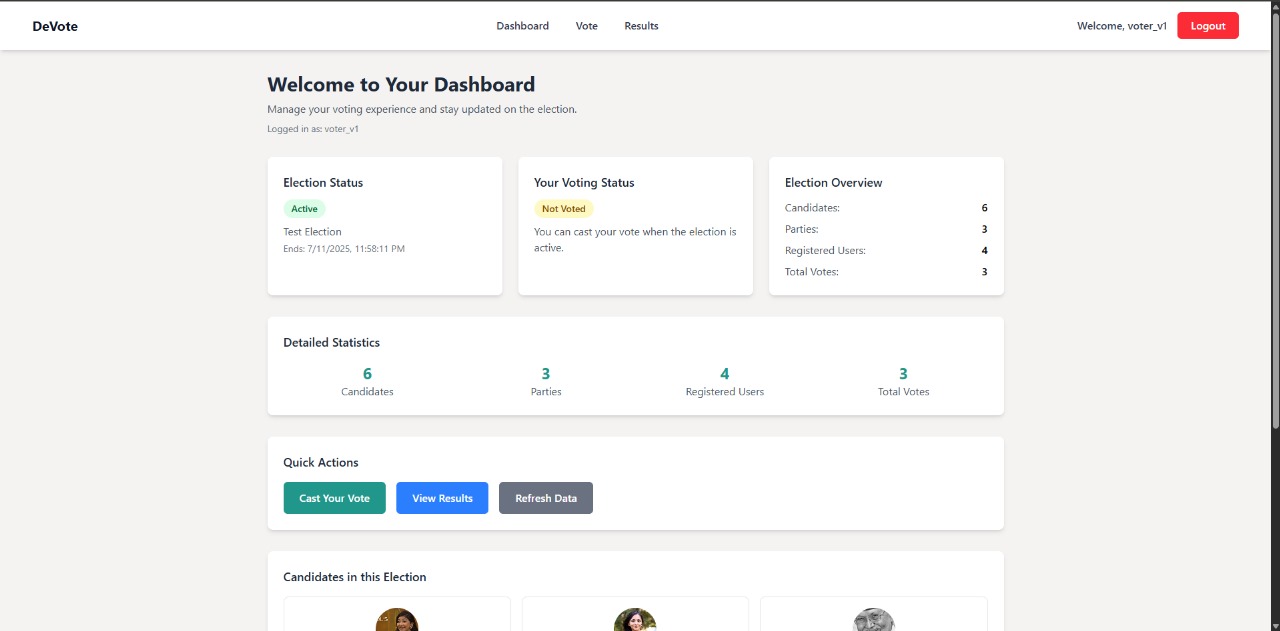


Figure vii: User Dashboard

## SOURCE CODE