Decentralized Voting System Using Ethereum Blockchain

Project Report 2025 Edition

Prepared by: Zophlic

Table of Contents

Introduction

The Decentralized Voting System using Ethereum Blockchain is a secure and transparent solution for conducting elections at Murang'a University of Technology. This system leverages blockchain technology to ensure tamper-proof voting records, enabling users to cast votes remotely while maintaining anonymity and preventing fraud.

The 2025 Edition of this system introduces significant improvements to the user interface, security features, and overall functionality. The system now supports multiple simultaneous elections, improved account management, and a more intuitive admin experience.

Project Overview

Motivation

Traditional voting systems face numerous challenges including lack of transparency, vulnerability to tampering, and limited accessibility. The Decentralized Voting System addresses these issues by utilizing Ethereum blockchain technology to create an immutable record of votes that can be verified by anyone while maintaining voter privacy.

Objectives

1. Create a secure and transparent voting platform
2. Eliminate intermediaries in the voting process
3. Ensure voter anonymity while preventing double voting
4. Provide real-time vote tallying and results
5. Support multiple simultaneous elections
6. Improve user experience for both administrators and voters
7. Enable multi-device voting

System Architecture

The Decentralized Voting System follows a three-tier architecture:

1. **Frontend Layer**: HTML, CSS, and JavaScript for the user interface
2. **Backend Layer**: Node.js server and FastAPI for database operations

3. **Blockchain Layer**: Ethereum smart contracts for vote storage and verification

Components

Frontend Components

- Admin Portal: For election creation and management
- Voter Interface: For casting votes and viewing results
- Authentication System: For secure user login

Backend Components

- Node.js Server: Serves the web application
- FastAPI Server: Handles database operations
- MySQL Database: Stores user credentials and roles

Blockchain Components

- Smart Contracts: Implemented in Solidity
- Ethereum Network: For deploying and interacting with smart contracts
- MetaMask: For wallet management and transaction signing

Data Flow

1. Administrators create elections through the admin portal
2. Election data is stored on the Ethereum blockchain
3. Voters authenticate using JWT tokens
4. Votes are cast as transactions on the blockchain
5. Results are tallied in real-time from the blockchain data

Implementation Details

Technologies Used

- **Frontend**: HTML5, CSS3, JavaScript, jQuery
- **Backend**: Node.js, FastAPI, Python
- **Database**: MySQL
- **Blockchain**: Ethereum, Solidity, Web3.js
- **Authentication**: JWT (JSON Web Tokens)
- **Development Tools**: Truffle, Ganache, MetaMask

Smart Contracts

The system utilizes two main smart contracts:

1. **Voting Contract**: Handles individual elections
2. Stores candidate information
3. Records votes
4. Prevents double voting
5. Calculates results
6. **Election Factory Contract**: Manages multiple elections
7. Creates new election instances
8. Tracks active elections
9. Manages election lifecycle

Database Schema

The database contains a single table for voter authentication:

```
CREATE TABLE voters (
  voter_id VARCHAR(36) PRIMARY KEY NOT NULL,
  role ENUM('admin', 'user') NOT NULL,
  password VARCHAR(255) NOT NULL
);
```

API Endpoints

The FastAPI server provides the following endpoints:

- `/login`: Authenticates users and issues JWT tokens
- `/verify`: Verifies JWT tokens
- `/voters`: Manages voter information

Key Features

Core Features

- **Secure Authentication**: JWT-based authentication system
- **Tamper-Proof Voting**: Votes stored on the Ethereum blockchain
- **Transparent Process**: All transactions visible on the blockchain
- **Real-Time Results**: Immediate vote tallying
- **Role-Based Access**: Different interfaces for admins and voters

2025 Edition Enhancements

- **Multiple Simultaneous Elections**: Support for running multiple elections concurrently
- **Improved MetaMask Integration**: Better account switching and connection handling
- **DD/MM/YYYY Date Format**: More intuitive date inputs for international users
- **Enhanced UI/UX**: Improved user interface with Murang'a University branding
- **Comprehensive Debugging**: Detailed error messages and troubleshooting tools
- **Multi-Device Support**: Responsive design for voting from various devices

Simplified Storage-Based System

For testing and demonstration purposes, a simplified version was implemented using browser localStorage:

- **Standalone Interfaces**: Independent admin and voter pages
- **Persistent Storage**: Elections and votes stored in localStorage
- **MetaMask Authentication**: Uses MetaMask for identity verification only
- **Double-Vote Prevention**: Tracks voter addresses to prevent multiple votes
- **Real-Time Updates**: Immediate result visualization

Technical Challenges and Solutions

Challenge 1: MetaMask Connection Issues

Problem: Users experienced difficulties connecting to MetaMask, particularly when switching accounts.

Solution: Implemented a direct MetaMask connection script that: - Provides clear connection status indicators - Handles account switching gracefully - Includes comprehensive error messages - Offers a troubleshooting guide for common issues

Challenge 2: Date Format Standardization

Problem: The original system used different date formats across interfaces.

Solution: Standardized on DD/MM/YYYY format with: - Separate input fields for day, month, and year - Input validation for valid dates - Consistent display format throughout the application

Challenge 3: Multiple Election Support

Problem: The original system supported only one active election at a time.

Solution: Implemented an Election Factory contract that: - Creates and tracks multiple election instances - Manages the lifecycle of each election independently - Provides a unified interface for election management

Challenge 4: Browser Compatibility

Problem: The application had inconsistent behavior across different browsers.

Solution: - Implemented feature detection instead of browser detection - Used polyfills for older browsers - Tested extensively across Chrome, Firefox, Safari, and Edge

User Interface

Admin Portal

The admin portal provides the following functionality:

- **Election Creation**: Create new elections with custom names, dates, and candidates
- **Candidate Management**: Add, edit, and remove candidates
- **Election Monitoring**: View active, upcoming, and completed elections
- **Result Visualization**: See vote counts and percentages with graphical representation

Voter Interface

The voter interface offers:

- **Election Browsing**: View available elections with status indicators
- **Voting**: Cast votes in active elections
- **Result Viewing**: See election results with visual representations
- **Vote Verification**: Confirm that votes were recorded correctly

UI Improvements

The 2025 Edition includes several UI improvements:

- **Responsive Design**: Adapts to different screen sizes
- **Intuitive Navigation**: Clear pathways for different user actions
- **Visual Feedback**: Status indicators and confirmation messages
- **Accessibility**: Improved keyboard navigation and screen reader support
- **Error Handling**: Clear error messages with suggested solutions

Security Considerations

Blockchain Security

- **Immutable Records**: Once recorded, votes cannot be altered
- **Distributed Ledger**: No single point of failure
- **Smart Contract Auditing**: Contracts reviewed for vulnerabilities

Application Security

- **JWT Authentication**: Secure token-based authentication
- **Input Validation**: All user inputs validated server-side
- **HTTPS**: Encrypted communication between client and server
- **Password Hashing**: Secure storage of user credentials

User Privacy

- **Anonymous Voting**: Votes linked to Ethereum addresses, not personal identities
- **Private Keys**: Users maintain control of their private keys
- **No Personal Data**: Minimal personal information stored in the system

Testing and Validation

Testing Methodology

- **Unit Testing**: Individual components tested in isolation
- **Integration Testing**: Interactions between components verified
- **End-to-End Testing**: Complete user workflows tested
- **Security Testing**: Vulnerability assessment and penetration testing

Test Scenarios

1. **Election Creation**: Verify administrators can create valid elections
2. **Voting Process**: Ensure voters can cast votes successfully
3. **Double Voting Prevention**: Verify users cannot vote multiple times
4. **Result Calculation**: Confirm accurate vote tallying
5. **Error Handling**: Test system response to invalid inputs and edge cases

Validation Results

The system successfully passed all test scenarios, demonstrating: - Reliable election creation and management - Secure and accurate voting process - Correct result calculation and display - Appropriate error handling and user feedback

Future Enhancements

Planned Improvements

1. **Mobile Application**: Native mobile apps for Android and iOS
2. **Biometric Authentication**: Fingerprint or facial recognition for voter verification
3. **Advanced Analytics**: Detailed voting statistics and patterns
4. **Internationalization**: Support for multiple languages
5. **Offline Voting**: Capability to vote without continuous internet connection

Windows Executable Packaging

The application can be packaged as a Windows executable (.exe or .msi) for easier distribution:

1. **Using Electron**:
2. Wrap the web application in an Electron container
3. Package with electron-builder to create a Windows installer
4. Include all dependencies (Node.js, blockchain tools)
5. **Using Windows Installer XML (WiX)**:
6. Create a traditional Windows installer (.msi)
7. Support for Group Policy deployment
8. Better for enterprise environments

8. Better for enterprise environments

9. **Using NSIS**:

10. Create a customizable installer with NSIS scripts
11. Smaller installer size
12. Extensive scripting capabilities

Conclusion

The Decentralized Voting System using Ethereum Blockchain represents a significant advancement in election technology. By leveraging blockchain's inherent security and transparency, the system provides a trustworthy platform for conducting elections at Murang'a University of Technology.

The 2025 Edition builds upon the original system with improved usability, enhanced features, and better security. The addition of multiple election support and improved MetaMask integration addresses key limitations of the previous version.

The simplified storage-based implementation provides an accessible entry point for testing and demonstration, while the full blockchain implementation offers the security and transparency benefits of distributed ledger technology.

As blockchain technology continues to evolve, this system can serve as a foundation for future voting applications, potentially extending beyond educational institutions to broader electoral systems.

References

1. Ethereum Documentation: https://ethereum.org/en/developers/docs/
2. Web3.js Documentation: https://web3js.readthedocs.io/
3. Solidity Documentation: https://docs.soliditylang.org/
4. FastAPI Documentation: https://fastapi.tiangolo.com/
5. JWT Authentication: https://jwt.io/
6. MetaMask Documentation: https://docs.metamask.io/
7. Truffle Suite Documentation: https://trufflesuite.com/docs/
8. Original Project Repository: https://github.com/Krish-Depani/Decentralized-Voting-System-Using-Ethereum-Blockchain