**ONLINE VOTING PLATFORM**

(WEB FRAMEWORKS USING REST API)

* Submitted by

Nivedha B

***Table of Content***

|  |  |  |
| --- | --- | --- |
| S. No | TITLE | Page  Number |
| 1 | **ABSTRACT** | 1 |
| 2 | **INTRODUCTION**  2.1 Intended Audience  2.2 Intended Use | 1 |
| 3 | **EXISTING WORK** | 2 |
| 4 | **PROBLEM STATEMENT** | 2 |
| 5 | **PROPOSED WORK** | 2 |
| 6 | **SCOPES**  6.1 In-Scope  6.2 Out-Scope | 2 |
| 7 | **TECHNOLOGY USED** | 3 |
| 8 | **SYSTEM REQUIREMENT**  8.1 Software Requirement  8.2 Hardware Requirement | 3 |

|  |  |  |
| --- | --- | --- |
| 9 | **MODULE VISION**  9.1 User  9.2 Election Admin  9.3 Super Admin | 4 |
| 10 | **FUNCTIONAL REQUIREMENT** | 5 |
| 11 | **NON-FUNCTIONAL REQUIREMENT** | 6 |
| 12 | **FUNCTIONAL OVERVIEW**  12.1 Voter Registration  12.2 Voter Login  12.3 Admin Login  12.4 Election Setup  12.5 Voting  12.6 Results | 6 |
| 13 | **FLOW DIAGRAM**  13.1 Workflow Diagram  13.2 Class Diagram  13.3 ER Diagram  13.4 Sequence Diagram | 6 |
| 14 | **CONCLUSION** | 13 |

**Online Voting Platform**

**Project Documentation**

**1. Abstract**

The Online Electronic Voting Platform ensures secure, transparent, and efficient elections. It allows registered voters to cast their votes remotely while maintaining anonymity and integrity. The system minimizes manual errors and automates vote tallying, enhancing the overall voting experience. Designed for both individual and organizational elections, the platform guarantees accessibility and scalability.

**2. Introduction**

The system introduces a modern approach to conducting elections using digital platforms. It aims to overcome the challenges of traditional voting systems, such as inefficiency, logistical issues, and security risks.

**2.1 Intended Audience**

The platform is aimed at government bodies, private organizations, educational institutions, and other groups seeking a reliable voting solution.

**2.2 Intended Use**

This platform can be used to conduct elections for various purposes, including government elections, school or college council elections, corporate board voting, and community polls.

**3. Existing Work**

Traditional voting systems involve manual voting and counting, which are prone to human errors and security vulnerabilities. Some existing digital voting systems lack robust security measures or scalability, making them unsuitable for larger elections.

**4. Problem Statement**

The traditional voting process is time-consuming, resource-intensive, and susceptible to fraud. The lack of a secure, accessible, and scalable digital platform restricts the ability to conduct seamless elections.

**5. Proposed Work**

The proposed system is a user-friendly online platform for conducting secure and transparent elections. It uses end-to-end encryption for vote casting and counting, ensuring confidentiality and reliability. The system includes features such as voter registration, real-time voting updates, and detailed analytics.

**6. Scope**

**6.1 In-Scope**

* Secure voter registration and verification.
* Role-based access for election administrators.
* Real-time vote casting and result display.
* Integration with external ID verification systems.

**6.2 Out-of-Scope**

* Offline voting mechanism.
* Handling disputes arising outside the digital domain.

**7. Technology Used**

**Backend**

* SpringBoot java

**Database**

* MySql

**8. System Requirements**

**8.1 Software Requirements**

* Windows 10 or above / macOS
* MySql

**8.2 Hardware Requirements**

* 4 GB RAM
* 500 GB Hard Disk
* Intel i3 Processor or higher

**9. Module Vision**

**9.1 User**

* Voter registration
* Login to cast votes
* View election results

**9.2 Election Admin**

* Election setup and management
* Monitoring voter activity
* Tallying and result declaration

**9.3 Super Admin**

* Managing election administrators
* Setting system-level configurations

**10.Functional Requirement:**

|  |  |  |
| --- | --- | --- |
| **FR No** | **Functional Requirement** | **Sub Requirement** |
| FR-1 | User Registration | Registration Through Gmail |
| FR-2 | Society Registration | Societies are registered only by the super admin |
| FR-3 | Profile Completion | User need to add all his details in the profile page |
| FR-4 | Billing Process | User can able to pay their bills using Razorpay payment gateway |

Table 10 – Functional Requirement

**11.Non-Functional Requirement:**

|  |  |  |
| --- | --- | --- |
| **NFR No** | **Non-Functional Requirement** | **Description** |
| NFR-1 | Usability | With an efficient, user friendly UI, users will not have difficulty in using the solution and navigating through the application |
| NFR-2 | Security | By enabling protected routes which provides authentication access to application |
| NFR-3 | Performance | The performance should be faster and user friendly for efficiency |
| NFR-4 | Availability | The model should be available for use always, it can be exported to users and can be run in the local machine |
| NFR-5 | Scalability | This can be developed into an API which can be incorporated by others who can make use of it |

Table 11 – Non-Functional Requirement

**12. Functional Overview**

**12.1 Voter Registration**

Voters register with verified credentials, such as ID proof.

**12.2 Voter Login**

Registered voters log in securely to access elections.

**12.3 Admin Login**

Admins log in to manage election settings and monitor processes.

**12.4 Election Setup**

Admins configure elections, including candidate details and timelines.

**12.5 Voting**

Users can vote for their preferred candidates during the election period.

**12.6 Results**

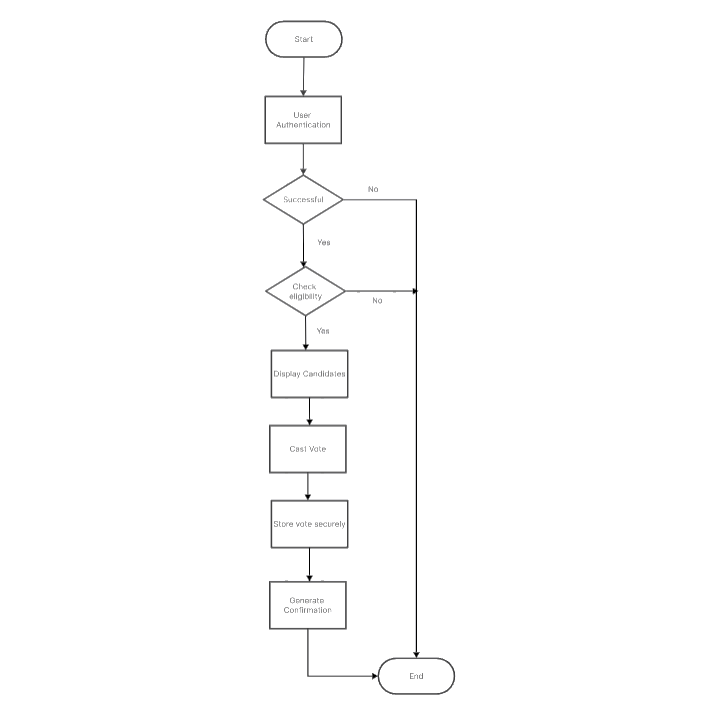
Results are calculated in real-time and displayed once the election concludes.

**13. Flow Diagram:**

**13.1 Workflow Diagram**

This diagram illustrates the operational flow of the electronic voting process. Key stages include:

* **Voter Authentication**: User logs in with valid credentials.
* **Ballot Presentation**: Display of candidate list.
* **Vote Casting**: Secure vote submission.
* **Vote Tallying**: Backend counting and verification.
* **Result Announcement**: Display of final results.

**

*(Workflow diagram :.)*

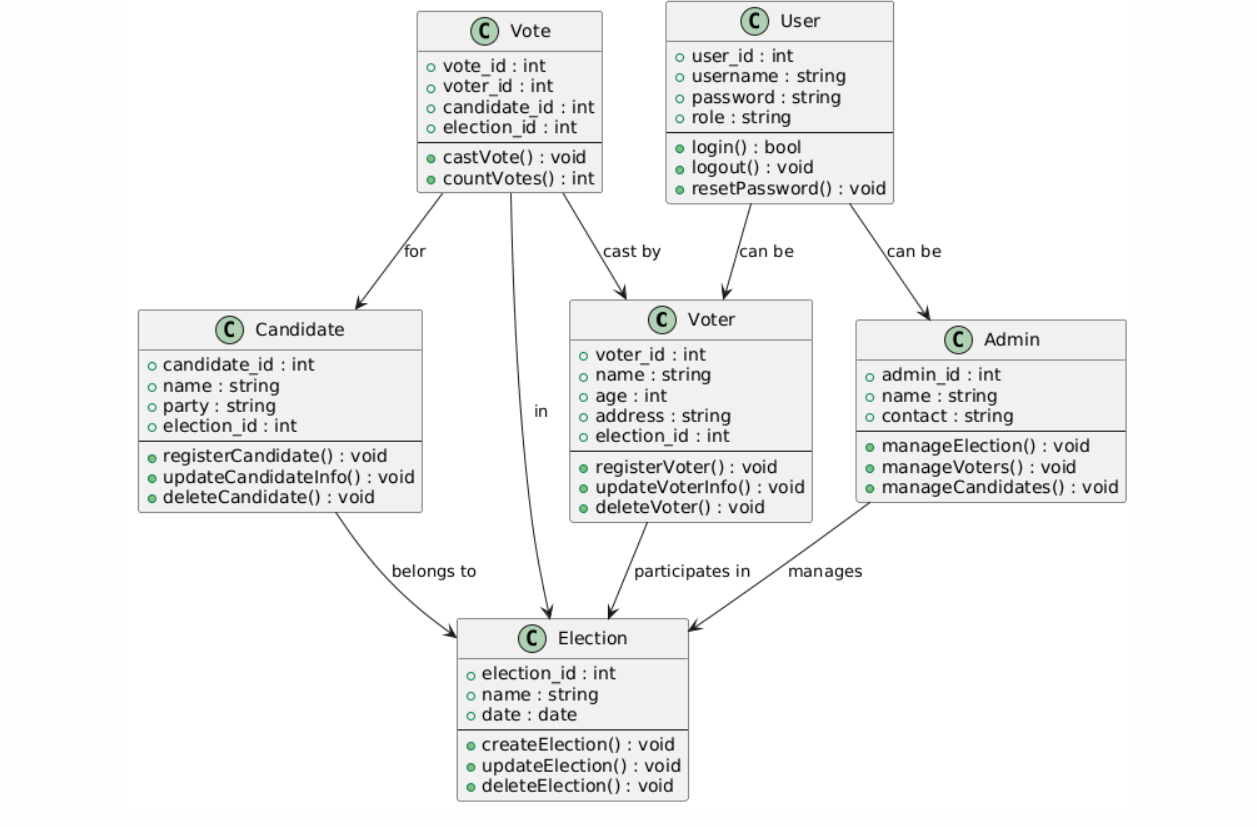
**13.2. Project Class Diagram**

The class diagram highlights the main entities and their relationships within the system:

* **Classes**: Voter, Admin, Candidate, Ballot, Election, and Results.
* **Attributes**:
  + - Voter: voterID, name, age,address,election\_id
    - Vote: voteID,voterID,candidateID,electionID
    - Admin: adminID, name, contact.
    - Candidate: candidateID, name, party, electionID.
    - Election: electionID, name, date.
    - User: user\_id,username,password,role

**Relationships**:

* A Voter participates in an Election
* A Candidate belongs to an Election
* A Vote is cast by a Voter for a Candidate in an Election
* An Admin manages an Election
* A User can either be an Admin or a Voter (for authentication)

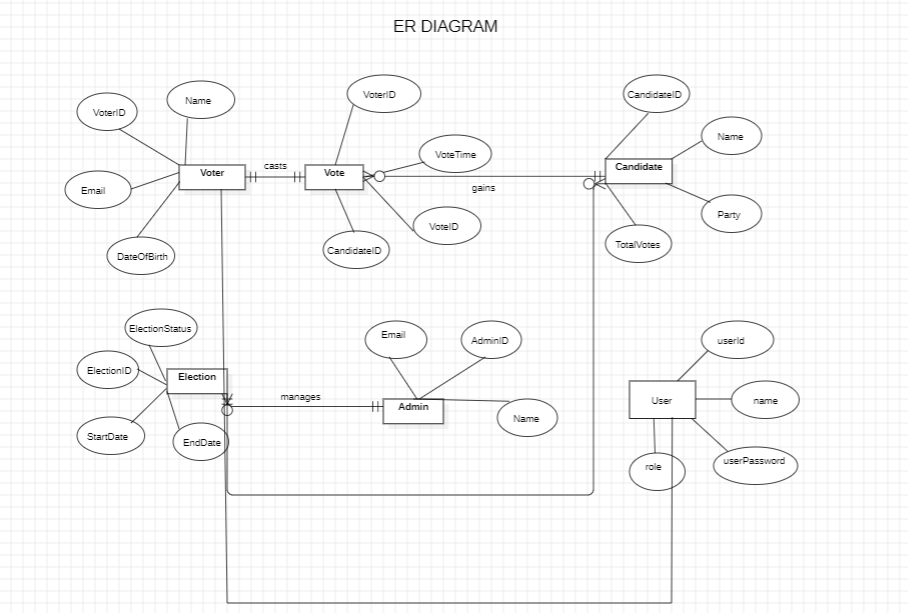


*(Class diagram : Include entities with attributes and relationships like association, aggregation, or inheritance.)*

**13.3. ER Diagram**

The ER diagram depicts the database structure, including:

* **Entities**: Voter, Admin, Candidate, Election, Vote, User.
* **Attributes**:
  + Primary keys like voterID, adminID, candidateID, electionID, userID, and voteID.
  + Attributes such as name, email, date, party, position, and votes.
* **Relationships**:
  + One-to-many between Election and Candidate.
  + Many-to-one between Elections and Admin.
  + One-to-one between Voter and Vote.



*(ER diagram : Reflect entities with keys and attributes, connected with relationship lines showing cardinality.)*

**13.4. Sequence Diagram**

This diagram showcases the sequence of interactions:

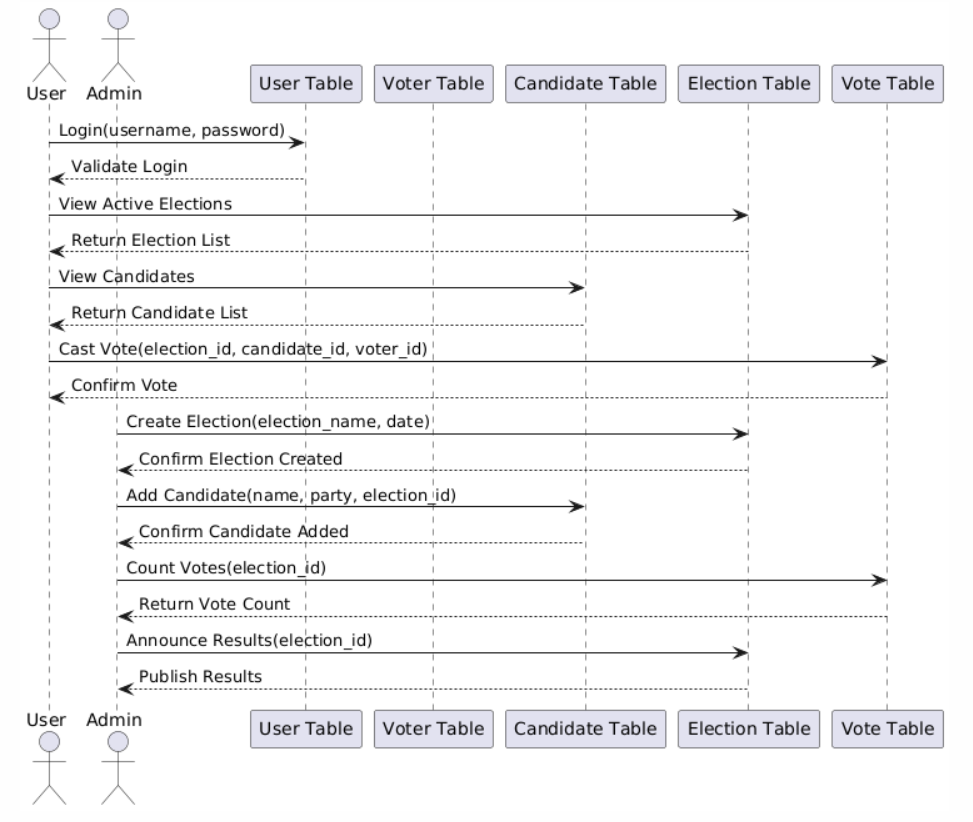
* **Actors**: Voter, System, Admin.
* **Steps**:

**User Actions:**

1. **Login (username, password)** → User enters login credentials.
2. **Validate Login** → System verifies credentials from the **User Table**.
3. **View Active Elections** → User requests a list of active elections.
4. **Return Election List** → System retrieves and displays elections from the **Election Table**.
5. **View Candidates** → User requests a list of candidates.
6. **Return Candidate List** → System fetches and displays candidates from the **Candidate Table**.
7. **Cast Vote (election\_id, candidate\_id, voter\_id)** → User selects a candidate and votes.
8. **Confirm Vote** → System records the vote in the **Vote Table** and confirms.

**Admin Actions:**

1. **Create Election (election\_name, date)** → Admin creates an election.
2. **Confirm Election Created** → Election details are stored in the **Election Table** and confirmed.
3. **Add Candidate (name, party, election\_id)** → Admin adds a candidate to an election.
4. **Confirm Candidate Added** → Candidate details are stored in the **Candidate Table** and confirmed.
5. **Count Votes (election\_id)** → Admin requests the vote count for an election.
6. **Return Vote Count** → System retrieves and returns the vote count from the **Vote Table**.
7. **Announce Results (election\_id)** → Admin announces the results.
8. **Publish Results** → System finalizes and stores the results.

* 

**15. CONCLUSION**

The Online Electronic Voting Platform eliminates the inefficiencies of traditional systems, ensuring secure, scalable, and transparent elections. It enables stakeholders to trust the election process while saving time and resources.