**EVERYONEVOTE**

**OVERVIEW**

Currently the process of voting is almost manual and cost intensive which may be easily replaced by current technologies for ease of access and cost effectiveness.

The online voting system provides a digital platform for voters to cast their votes securely and conveniently. It consists of a frontend user interface (UI) built using React.js and a backend server developed with Node.js and Express.js. The system follows a client-server architecture, with communication between the frontend and backend facilitated through HTTP requests.

We can also introduce higher level encryption method in the same project using blockchain or other security protocols for ensuring the voting goes without any security problems.

**Design Details**

**Server Used:** *Render* – Link: https://everyonevote.onrender.com

**Frontend Deployer Used**: *Vercel* – Link: https://every-one-vote.vercel.app/

**Database Deployment**: MongoDB : ac-oc5xjcm-shard-00-00.9ryiccc.mongodb.net

**API Docs**: *Postman* : <https://app.getpostman.com/join-team?invite_code=66eb3ce8094dff487576b05f3258c580>

**Source Control** : *Github*: [Shubham-Aggarwal-1306/EveryOneVote (github.com)](https://github.com/Shubham-Aggarwal-1306/EveryOneVote)  
  
**Tech Stack:**

1. Database: MongoDB
2. Frontend: React
3. Backend: Express
4. Package Manager: Node

**High Level Design Considerations**

**Frontend:**

**Technologies**: React.js for the user interface.

**Components**:

1. **Sign In Page**: Allows users to sign in using their credentials.
2. **Login Page**: Allows users to log in after signing in.
3. **Home Page**:
   1. **For Voters**: Displays the list of candidates and allows them to cast their votes.
   2. **For Officers**: Displays voting statistics and other relevant information.

**Backend:**

**Technologies**: Node.js with Express.js for building the server.

**API Endpoints**:

* **/users/login**: Handles user sign-in.
* **/candidates**: Manages CRUD operations for candidates.
* **/votes**: Manages voting operations.

**Database**:

MongoDB for storing user data, candidate information, and votes.

**Authentication and Authorization**:

* **User Authentication:**
  + Users are authenticated using their username and password.
  + Upon successful authentication, users receive a JWT (JSON Web Token) for authorization.
* **Role-Based Access Control (RBAC):**
  + Users are assigned roles (e.g., voter, officer) which determine their access to different features.
  + Voters can cast votes and view candidate information.
  + Officers have additional privileges such as viewing voting statistics and managing candidates.

**Security:**

* **JWT Authentication**: Ensures secure communication between the frontend and backend.
* **Password Hashing:** User passwords are hashed before storing them in the database to enhance security.
* **Input Validation**: All user inputs are validated to prevent injection attacks and ensure data integrity.

**Low Level Design Considerations**