E-Voting Web Application

Introduction

This document provides a complete and detailed explanation of the E-Voting web application, a full-stack project developed to simulate a secure, role-based digital voting system. The project integrates multiple technologies—including React, PHP, Python, MySQL, and Twilio—to deliver a multi-user platform with authentication, voting mechanisms, administrative tools, and transparent election reporting. The implementation reflects production-like workflows and demonstrates strong understanding of full-stack engineering concepts.

System Overview

The E-Voting platform allows voters, candidates, and admins to interact with the system based on predefined role permissions. It digitizes the core functionalities found in real elections such as registration, verification, vote casting, poll configuration, candidate representation, and results analysis.
Key highlights of the system include:

- OTP-secured login using Twilio via Python scripts
- Role-based dashboards for voters, candidates, and administrators
- Single-vote enforcement with backend validation
- Timer-based voting window with automatic neutral vote submission
- Detailed result viewing with CSV export support
- Admin-controlled approval of new registrations

Mandatory Folder Structure Requirement

To ensure smooth communication between the React frontend and the PHP backend, the backend files must be placed in one specific directory path. This is due to the fact that React components contain hardcoded fetch URLs pointing to this backend location.

Required backend directory path:
C:\Web_App Development\XAMPP\htdocs\evoting

The folder name must be exactly "evoting". If it is renamed or moved elsewhere, the PHP APIs will no longer be accessible to the frontend, resulting in failed data fetching and broken functionality.

Frontend Architecture (ReactJS)

Frontend code location:

C:\Web_App Development\Projects\e_voting\src

The frontend is implemented using React's component-based architecture. It handles UI rendering, routing, input validation, and communication with the backend via REST-like fetch requests. Styling is done using standard CSS files, ensuring a clean and readable interface.
Major responsibilities of the frontend:

- Display user dashboards for voter, candidate, and admin roles
- Collect user credentials and pass them to the backend for OTP authentication
- Render voting screens with countdown timers
- Provide data tables for results at constituency, assembly, and ward levels
- Allow admins to navigate control panels for verification and updates
- Facilitate CSV download using browser-compatible techniques

Backend Architecture (PHP + Python)

Backend code location:

C:\Web_App Development\XAMPP\htdocs\evoting

5.1 PHP Backend

PHP serves as the primary backend engine. It handles API routing, session handling, data validation, and direct interactions with the MySQL database. All core election logic—such as ensuring a voter cannot vote twice—is enforced on the backend to guarantee accuracy and security.

Key responsibilities:

- Processing login and registration data
- Fetching and updating voter/candidate/admin information
- Conducting vote submission and verification
- Managing poll configurations and results

5.2 Python Backend (Twilio OTP)

Python scripts are integrated with Twilio's SMS service to generate and send OTPs for user authentication. Whenever a login request is made using a mobile number, PHP triggers the Python script, which communicates with Twilio to deliver the OTP in real time.

Database Design (MySQL)

Database Name: e_voting

The database is structured to accurately represent voters, candidates, polling information, and new registrations awaiting verification. It maintains data integrity across multiple tables through primary keys, constraints, and logical grouping of attributes.

6.1 voter_database

Contains personal, demographic, and administrative details of verified voters. Each voter record is uniquely identified using voter_uid, aadhar_uid, and mobile number.

6.2 voter_database_temp

Stores temporary entries submitted through online registration forms. Admins review this table to approve or reject applications before moving them into the main voter database.

6.3 candidate_database

Stores candidate information including name, party name, constituency, assembly details, and logo. It also records which election year and ward a candidate is participating in.

6.4 poll

Defines polling configurations for each constituency, assembly, ward, and year. These records are used to determine which voters participate in which polls.

User Roles & Functionality

7.1 Voter

• Logs in via OTP sent to mobile phone
• Views complete personal details from the database
• Casts vote once within the allowed voting window
• Voting timeout triggers a neutral vote submission

7.2 Candidate

• Logs in using secure credentials
• Views personal profile and party information
• Reviews election metadata such as constituency and assembly details

7.3 Admin

• Validates new voter registrations from the temporary table
• Updates voter and candidate details when needed
• Manages poll parameters across all hierarchies
• Generates and downloads detailed CSV data for offline use

Voting Logic & Security Measures

The system integrates multiple layers of security and safeguards to ensure the integrity of the voting process, prevent misuse, and enforce fair usage. Security principles implemented:

- OTP-based authentication using Twilio
- Prevention of duplicate voting through backend checks
- Timer-based voting interface to avoid prolonged sessions
- Neutral vote submission on timeout or inactivity
- Validation of all inputs and requests on the PHP backend

Result Management & CSV Export

Users can explore election details across constituency, assembly, and ward levels. Admin users have access to detailed data extraction utilities that allow results to be downloaded in CSV format for offline review, analytics, or reporting.

Software Tools Used

- XAMPP – Local server environment for PHP and MySQL
- MySQL – Relational database backend
- Node.js – Required for running React development environment
- ReactJS – Frontend UI framework
- Python – Integrated with Twilio for OTP operations
- VS Code – Primary development IDE

Skills Demonstrated

- Full-stack web development
- User authentication systems (OTP-based)
- Multi-language integration (React, PHP, Python)
- REST-like API communication
- SQL schema design and optimization
- Session management and backend validation
- Data export (CSV) implementation
- Secure voting workflow design

Conclusion

This E-Voting system is a robust demonstration of practical software engineering skills. It shows the capability to build an end-to-end application incorporating frontend development, backend logic, database systems, and third-party API integrations. The project mirrors real-world election workflows and demonstrates industry-relevant development practices suitable for professional environments.