Project Documentation for Online Voting System

1. Project Overview

The Online Voting System is a web-based application that allows users to cast votes for elections remotely. The system provides an easy-to-use platform for voters, enabling them to participate in elections securely and efficiently. The application is built using **Java Maven**, **Servlet**, and **MySQL** for the backend.

Technologies Used:

- **Frontend**: HTML, CSS, JavaScript
- **Backend**: Java, Servlet
- **Database**: MySQL
- **Build Tool**: Maven

2. Project Goals

- To provide a secure and transparent way for people to vote online.
- To allow users to register, log in, and cast their votes.
- To ensure only authenticated users can vote once in an election.
- To allow administrators to manage election data, including creating elections, candidates, and viewing the results.

3. Functional Requirements

1. **User Registration and Login**:
    - Users can register with their email, name, and password.
    - Users can log in to the system securely using their registered credentials.
2. **Admin Panel**:
    - Admins can create elections with different candidates.
    - Admins can view the results of the elections in real-time.
3. **Voting Process**:
    - After logging in, users can view available elections and candidates.
    - Users can vote for a candidate in an election and submit their vote.
4. **Vote Security**:
    - Only authenticated users can vote.
    - Each user can vote once per election.
5. **Result Calculation**:

- The system calculates the number of votes for each candidate.
- Admins can view the election results, including the number of votes each candidate received.

4. System Design

Architecture:

- **Model-View-Controller (MVC)** architecture is used:
    - **Model**: Represents data and business logic (e.g., users, elections, candidates, and votes).
    - **View**: The user interface displayed to the user (HTML, JSP).
    - **Controller**: Handles user input and interacts with the model (Servlets).

Database Design:

1. **Tables**:
    - **Users**: Stores user information (id, email, password, name, etc.).
    - **Elections**: Stores election information (id, name, description, date).
    - **Candidates**: Stores candidate information (id, election_id, name, party).
    - **Votes**: Stores vote data (user_id, election_id, candidate_id).
2. **Relationships**:
    - **Users** can vote for **Candidates** in a specific **Election**.
    - **Elections** can have multiple **Candidates**.
    - **Votes** are linked to both **Users** and **Candidates**.

5. Non-Functional Requirements

1. **Security**:
    - User authentication using secure password hashing.
    - Protection against SQL injection and other common vulnerabilities.
    - HTTPS protocol for secure communication.
2. **Scalability**:
    - The system should be able to handle a large number of users and votes without significant performance degradation.
3. **Performance**:

- System should respond within 10 seconds for most operations (such as voting or fetching election details).

4. **Usability**:
    - User-friendly interface for easy navigation and voting.
    - Admin interface should allow for simple election creation and result viewing.

6. System Flow

1. **User Registration**:
    - User accesses the registration page and fills in personal details.
    - User submits the form, and the system stores the details in the database.
2. **User Login**:
    - The user enters their credentials.
    - The system checks credentials and grants access if valid.
3. **Voting**:
    - After logging in, the user views active elections and selects a candidate.
    - The user submits their vote, and the system records it.
4. **Admin Panel**:
    - Admin can create elections, assign candidates, and view results.
    - Admin views the number of votes for each candidate.

7. Data Flow Diagram (DFD)

- **Level 0 (Context Diagram)**: The system interacts with users, admins, and the MySQL database.
- **Level 1 (Detailed Diagram)**: Depicts the flow of data between user interactions, authentication, voting, and result management.

8. Implementation

Maven Configuration:

- The project is managed with Maven for dependency management and building.
- **pom.xml** includes dependencies for Servlet API, MySQL connector, and other libraries required for the project.

9. Testing

Test Cases:

- **User Registration**: Test the registration with valid/invalid inputs.
- **User Login**: Test the login functionality with correct and incorrect credentials.
- **Voting**: Test the voting process by ensuring users can only vote once.
- **Admin Panel**: Test the creation of elections, candidates, and result viewing.

Tools:

- JUnit for unit testing Java code.
- Selenium for end-to-end UI testing.

10. Deployment

- **Server**: The application is deployed on an Apache Tomcat server.
- **Database**: The MySQL database is hosted on a local or cloud server.
- **Steps**:
    1. Install Tomcat and MySQL.
    2. Build the project with Maven (mvn clean install).
    3. Deploy the WAR file to Tomcat.
    4. Configure database connection in web.xml.

11. Conclusion

This Online Voting System provides an easy and secure method for users to participate in elections online. With the use of Java, Servlet, and MySQL, the system ensures robust performance and scalability for handling a large number of votes. The application is designed to be secure, with proper authentication, and offers real-time election results.