**Documentation on architecture and Design :**

The architecture of this project is a simple client-server architecture, where the client interacts with the server to perform different operations.

The ElectionStatistics class acts as the server, which handles all the requests from the client and performs the necessary operations. The client interacts with the server through the methods provided in the ElectionStatistics class, such as enterCandidate(name), castVote(name), countVote(name), listVote(), and getWinner().

1. The class has a private instance variable candidates which is a HashMap, which stores the name of the candidate as a key and their vote count as a value.
2. The enterCandidate(name) method is used to add a new candidate to the candidates HashMap. It takes the name of the candidate as an input, creates a new entry in the HashMap with the candidate's name as the key and 0 as the value, representing the initial vote count.
3. The castVote(name) method is used to cast a vote for a candidate. It takes the name of the candidate as an input, increments the vote count of that candidate by 1 and returns the new vote count.
4. The countVote(name) method is used to view the vote count of a specific candidate. It takes the name of the candidate as an input, retrieves the vote count of that candidate from the candidates HashMap and returns it.
5. The listVote() method is used to view the list of all registered candidates and their vote counts. It converts the candidates HashMap to a JSON object and returns it as a String.
6. The getWinner() method is used to view the name of the candidate who got the largest number of votes. It goes through the candidates HashMap, compares the vote count of each candidate and returns the name of the candidate with the highest vote count.

The project is simple, easy to understand and maintain, and scales well as the number of candidates and vote counts increases.