# CHAPTER SIX

## SYSTEM IMPLEMENTATION

### Tools Used for Coding and Testing

Developing a secure and efficient e-voting system requires a robust set of tools for both coding and testing. The selection of these tools plays a crucial role in ensuring the system's functionality, scalability, and security. This section discusses the programming languages, frameworks, databases, and testing tools used in the implementation of the e-voting system.

#### **Coding Tools**

### The development of the e-voting system involves both frontend and backend technologies to create a seamless and interactive voting experience.

#### **Frontend Development**

* **React.js** – A popular JavaScript library used for building user interfaces. It enables efficient rendering of UI components and enhances the responsiveness of the voting system.
* **HTML & CSS** – These are used for structuring and styling the web pages, ensuring a user-friendly interface.
* **Tailwind CSS** – A utility-first CSS framework that allows for rapid UI design with minimal code.

**Backend Development**

* **Node.js with Express.js** – A lightweight and scalable server-side framework used to manage user authentication, voting processes, and database interactions.
* **MySQL** – A relational database management system (RDBMS) used to store voter records, election results, and authentication data.
* **JWT (JSON Web Token)** – A secure method for handling user authentication and session management.
* **Bcrypt.js** – A library for hashing passwords to enhance system security.

#### **2. Testing Tools**

To ensure the reliability and accuracy of the e-voting system, various testing tools are employed:

* **Postman** – Used for API testing to verify that backend endpoints work correctly.

**Testing Approach and Data Used**

##### **Introduction**

Testing is a crucial phase in the system implementation process, ensuring that the e-voting system functions correctly, securely, and efficiently. This section outlines the approach used for testing, the types of data utilized, and how different testing methods validate the accuracy, security, and usability of the system.

**1. Testing Approach**

The testing process follows a structured approach, including various levels of testing:

* **Unit Testing** – Each component (e.g., login system, vote submission, results display) is tested individually to ensure it functions correctly. Jest is used for frontend unit testing, while Mocha and Chai are used for backend validation.
* **Integration Testing** – Different system modules are combined and tested as a group. This ensures that the frontend properly communicates with the backend and database. Postman is used to test API requests and responses.
* **Functional Testing** – The system is tested against the defined requirements to confirm that it performs as expected. This includes scenarios like registering voters, casting votes, and tallying results.
* **Security Testing** – This involves testing for vulnerabilities such as SQL injection, unauthorized access, and data breaches. Tools like OWASP ZAP and penetration testing techniques help identify security flaws.
* **Performance and Load Testing** – The system is tested under different loads to ensure it can handle a high number of concurrent users without crashing.
* **User Acceptance Testing (UAT)** – Real users, including students and administrators, interact with the system in a controlled environment to ensure it meets their needs.

**2. Data Used for Testing**

To evaluate system functionality, different datasets are utilized:

* **Mock User Data** – Test users are created with different access levels (voters and admins) to validate authentication, registration, and role-based access control.
* **Simulated Election Data** – Multiple candidates and election events are generated to test the voting process and result computation.
* **Invalid Data Scenarios** – The system is tested with incorrect or malicious inputs (e.g., duplicate votes, incorrect login credentials) to verify error handling and security enforcement.

### **Proposed Change-Over Techniques**

The transition from the existing voting system to the newly developed e-voting system is a critical phase that ensures minimal disruption and a smooth adoption process. The choice of a change-over technique determines how the new system replaces the old one. This section discusses various change-over methods and selects the most suitable approach for implementing the e-voting system.

**1. Change-Over Techniques**

Several change-over techniques can be used for system implementation:

* **Direct Change-Over (Big Bang Approach)**  
  In this method, the old system is completely replaced by the new system at a specific date. It is fast and cost-effective but risky, as any failure in the new system can disrupt operations.
* **Parallel Running**  
  Both the old and new systems operate simultaneously for a period until the new system is fully validated. This method reduces risks by allowing users to compare outputs, but it is resource-intensive and may cause redundancy.
* **Phased Implementation**  
  The new system is introduced gradually, module by module, until the entire system is deployed. This approach allows for gradual adaptation and early issue detection but may take longer to complete.
* **Pilot Change-Over**  
  The new system is initially implemented in a limited scope (e.g., a specific department or group of users). If successful, it is expanded to other areas. This method reduces risk but may require additional time for full deployment.

**2. Recommended Change-Over Method**

For the e-voting system, the **parallel running** approach is recommended. Since elections are crucial, running both the old and new systems concurrently ensures that any failures in the new system do not affect the voting process. This allows for validation and user training before fully switching to the new system.