1. **Problem Description**

* In many organizations and institutions, the process of conducting elections or surveys is often manual, time-consuming, and prone to errors such as duplicate votes, miscounting, or voter fraud. This traditional method is not only inefficient but also lacks transparency and real-time accessibility for both administrators and participants.
* The **online voting system** aims to address these challenges by providing a secure, efficient, and user-friendly web application. This system allows registered users to sign up, log in, and cast their votes digitally. It eliminates the need for physical ballot papers, reduces administrative overhead, and ensures accurate vote counting.
* By leveraging technology, the system enhances accessibility, allowing users to vote from any location, and promotes trust through data integrity and authentication mechanisms. This project is particularly beneficial for small-scale elections in schools, clubs, or local organizations.

**2. Stakeholders/Users**

**2.1 Voters**

* Individual users who register on the platform as part of a group.
* Can log in and securely cast their votes for the elections they are participating in.

**2.2 Groups**

* Entities such as organizations, clubs, or committees that register to hold elections for their members.
* Responsible for creating elections, defining candidates, and monitoring group-level voting outcomes.

**3. Feasibility Study**

**3.1 Purpose**

The online voting system is needed to streamline the election process, especially for small-scale groups such as clubs, organizations, or committees. Traditional voting methods are often time-consuming, resource-intensive, and susceptible to errors or disputes. By providing a digital solution, the system ensures secure, efficient, and accurate vote management, catering to modern demands for convenience and accessibility.

**3.2 Advantages**

* **Accessibility**: Allows voters to cast their votes from anywhere, removing geographical constraints.
* **Efficiency**: Automates the voting process, reducing the time and effort required to organize and manage elections.
* **Accuracy**: Minimizes human error in vote counting and eliminates issues such as duplicate or invalid votes.
* **Cost-Effective**: Reduces the need for physical resources like paper ballots and voting booths.
* **Environmental Impact**: Promotes eco-friendliness by replacing paper-based methods with a digital system.
* **Transparency**: Provides immediate results and ensures data integrity, fostering trust among participants.

**3.3 Disadvantages**

* **Technical Knowledge Requirement**: Users may need basic technical skills to register and vote.
* **Internet Dependency**: The system requires stable internet access, which might not be available to all users.
* **Data Security Risks**: Although measures can be taken to secure the system, potential vulnerabilities to cyberattacks or data breaches may exist.
* **Limited Scope**: The basic functionality may not support large-scale elections or advanced features without significant upgrades.

## 4. Proposed Solution

### 4.1 Description of the Proposed Solution

The online voting system is a web-based application designed to facilitate secure, efficient, and user-friendly voting for small groups. The system allows registered users to create groups, register voters, and conduct elections within their respective groups.

### 4.2 Key Features of the Proposed Solution

#### **4.2.1 Group Registration and Management**

* Groups can register on the platform to set up elections.
* Each group can define candidates and manage election settings, such as voting duration and participant lists.

#### **4.2.2 User Registration and Authentication**

* Voters sign up using unique credentials and are assigned to specific groups.
* The login system ensures secure access, allowing only authorized users to cast their votes.

#### **4.2.3 Voting Process**

* Voters can view a list of ongoing elections within their group.
* Each voter can cast a single vote per election.
* A confirmation mechanism ensures the vote is recorded accurately.

#### **4.2.4 Real-Time Vote Counting**

* Votes are automatically tallied and stored securely in the system database.
* Results are displayed to voters once the election ends, ensuring transparency.

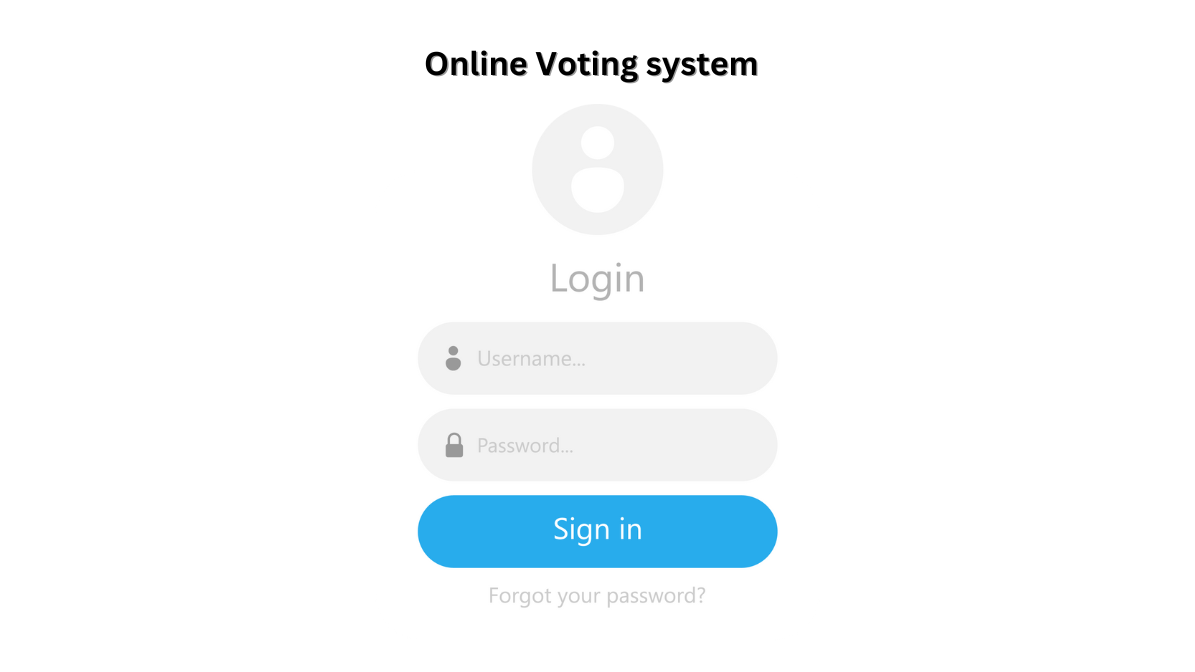
#### **4.2.5 Secure and Scalable Architecture**

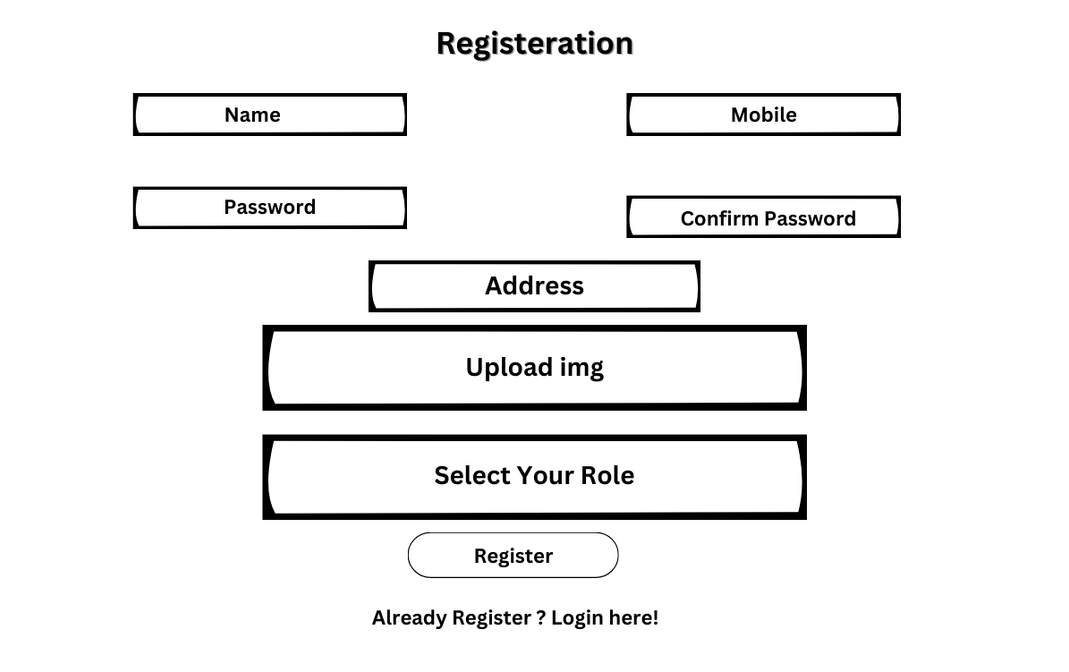
* All data, including voter credentials and votes, is encrypted to maintain security and privacy.
* The system is scalable to support multiple groups conducting elections simultaneously.

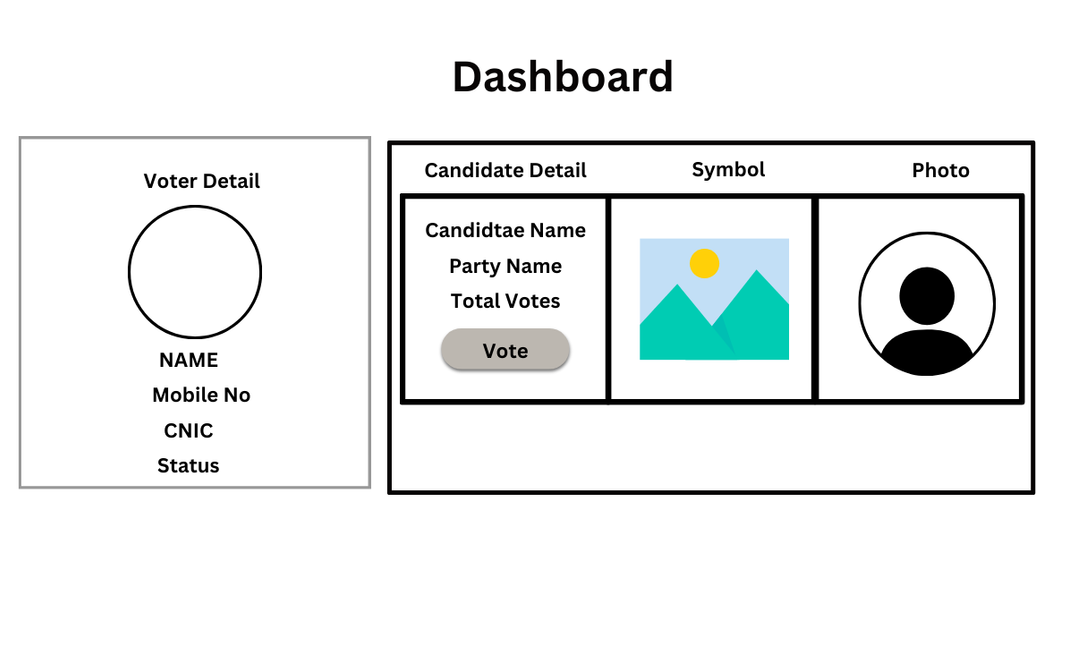
### 4.3 Benefits of the Solution

* **Simplicity**: The interface is intuitive and easy to use, even for non-technical users.
* **Efficiency**: Automates tedious tasks like vote counting and result tabulation.
* **Security**: Implements safeguards to prevent unauthorized access or manipulation.

### 4.4 Front-End Design







## 5. Software Requirement Specification Document

### 5.1 Functional Requirements

These are the specific features and functionalities that the application must provide:

#### **5.1.1 User Registration and Login**

* Users (voters) can create accounts with unique credentials.
* Authentication ensures only registered users can access the system.

#### **5.1.2 Group Creation and Management**

* Groups can register to create and manage their elections.
* Group administrators can define election details, such as candidates and voting deadlines.

#### **5.1.3 Voting Mechanism**

* Voters can participate in elections within their assigned groups.
* Each voter is allowed to cast one vote per election.
* Votes are recorded securely in real time.

#### **5.1.4 Results Display**

* The system calculates and displays election results once voting ends.
* Results are accessible to all voters within the group.

#### **5.1.5 User Dashboard**

* Voters can view a list of ongoing and completed elections.
* Group managers can monitor election status and view results.

### 5.2 Non-Functional Requirements

These define the system's performance and quality attributes:

#### **5.2.1 Performance**

* The system should handle multiple concurrent users and groups without significant delays.
* Voting and result calculations must process within 2 seconds of user interaction.

#### **5.2.2 Reliability**

* The system must have at least 99.9% uptime to ensure availability during elections.
* Proper backup mechanisms must be implemented to avoid data loss.

#### **5.2.3 Security**

* User credentials and votes must be encrypted using industry-standard algorithms.
* The system should prevent unauthorized access or tampering with data.

#### **5.2.4 Scalability**

* The system should support an increasing number of users, groups, and elections as needed.

#### **5.2.5 Usability**

* The user interface must be intuitive and straightforward for non-technical users.
* Provide clear navigation and guidance throughout the system.

#### **5.2.6 Maintainability**

* The system should allow for easy updates to features and security patches.

#### **5.2.7 Compatibility**

* The application must be accessible on commonly used web browsers (e.g., Chrome, Firefox, Edge).

### 5.3 Use Case Diagrams/User Stories

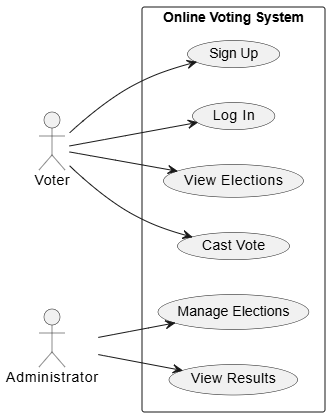
#### **5.3.1 Use Case Diagram**

**Actors**:

* **Voter**: A registered user who can log in, browse elections, and cast votes.
* **Administrator**: Manages the election setup, user registration, and monitors voting results.

**Use Cases**:

1. **Sign Up**: A new user creates an account.
2. **Log in**: The user logs into the system.
3. **View Elections**: The user browses the available elections.
4. **Cast Vote**: The voter selects their choice and casts their vote.
5. **Manage Elections**: The administrator creates and configures elections.
6. **View Results**: The administrator views the results (accessible only after voting ends).



#### **5.3.2 User Stories**

**Voter Stories**:

1. As a **voter**, I want to create an account so that I can participate in elections.
2. As a **voter**, I want to log in securely so that I can access my personalized dashboard.
3. As a **voter**, I want to view a list of available elections so that I can choose where to vote.
4. As a **voter**, I want to cast my vote so that my choice is counted in the election.

**Administrator Stories**:

1. As an **administrator**, I want to create elections so that voters can participate.
2. As an **administrator**, I want to manage voter registrations so that only eligible users can vote.
3. As an **administrator**, I want to view results after voting closes so that I can announce the outcomes.

### 5.4 Assumptions/Constraints

#### **5.4.1 Assumptions**

1. **User Access**:
   * Voters and groups (or parties) have access to the internet and a device capable of running a web browser.
2. **Group Registration**:
   * Groups or parties will provide accurate and complete details during registration, including the list of candidates for their elections.
3. **One Vote per Voter**:
   * Each voter is allowed to cast only one vote per election, with no changes allowed after submission.
4. **System Usage**:
   * The system is intended for small-scale elections with a limited number of groups and voters.
   * All participants will follow the defined rules of the voting process.
5. **Basic System Knowledge**:
   * Users are expected to have a basic understanding of web applications for registration and voting.

#### **5.4.2 Constraints**

1. **Limited Functionality**:
   * The system is designed for basic operations, such as voter and group registration, and voting. Advanced features like voter verification through external databases or detailed election analytics are not included.
2. **Group-Specific Voting**:
   * Voters can only participate in elections organized by the group they are registered with.
3. **Data Integrity**:
   * The system relies on users (both voters and groups) to input accurate data during registration. No external validation mechanisms are in place.
4. **Single Platform**:
   * The application is web-based, with no support for native mobile or offline usage.
5. **Limited Scalability**:
   * The application is not designed to handle large-scale elections or high simultaneous user traffic.
6. **Security Measures**:
   * Basic encryption and secure login features are implemented, but the system does not cater to advanced security threats or sophisticated cyberattacks.

## 6. Project Plan

### 6.1 Timeline and Milestones

**Week 1: Requirements Analysis**

* Define system requirements (functional & non-functional).
* Create and finalize the System Requirements Specification (SRS).
* **Milestone**: Approved SRS document.

**Week 2: System Design**

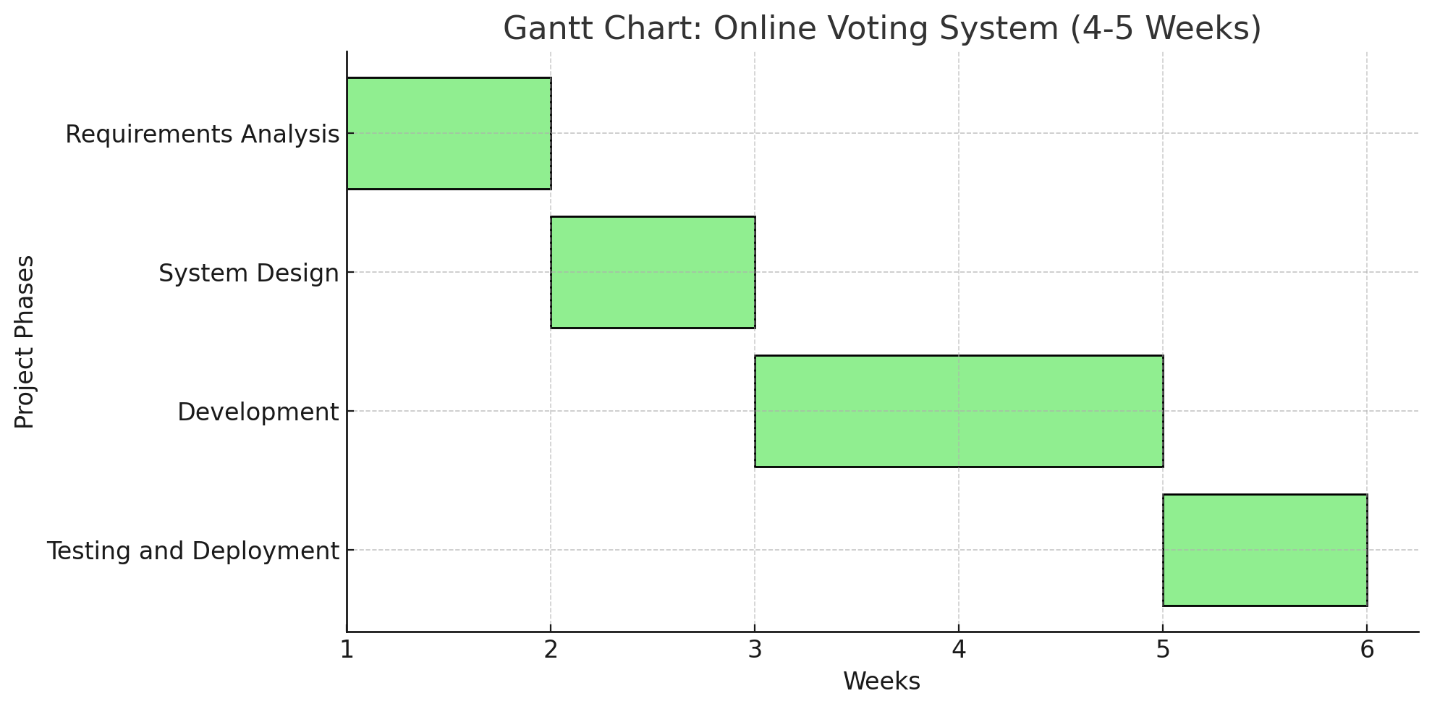
* Design the database schema.
* Create UI wireframes and system architecture.
* **Milestone**: Completed system design documentation.

**Week 3–4: Development**

* Frontend development (sign-up, login, voting interface).
* Backend development (database integration, vote recording).
* Module integration and basic testing.
* **Milestone**: Functional prototype ready.

**Week 5: Testing and Deployment**

* Conduct system testing and resolve bugs.
* Deploy the application on a server.
* Perform user acceptance testing (UAT).
* **Milestone**: Fully deployed system.



### 6.2 Team Responsibilities

#### **6.2.1 Planning**

**6.2.1.1 Requirement Analysis**

* Collect user requirements – **Hamza Zahoor (Team Lead)**
* Define functional and non-functional requirements – **Hamza Zahoor**
* Create System Requirements Specification (SRS) document – **Haseeb Ahmed**

**6.2.1.2 Technology Selection**

* Choose frontend and backend frameworks – **Hamza Zahoor**
* Identify database solutions – **Abdullah Sajid**

#### **6.2.2 Design**

**6.2.2.1 System Design**

* Design database schema – **Abdullah Sajid**
* Create system architecture diagram – **Abdullah Sajid**

**6.2.2.2 UI/UX Design**

* Develop wireframes – **Haseeb Ahmed**
* Finalize the user interface prototype – **Haseeb Ahmed**

#### **6.2.3 Development**

**6.2.3.1 Frontend Development**

* Implement signup/login interface – **Haseeb Ahmed**
* Create voting interface – **Haseeb Ahmed**

**6.2.3.2 Backend Development**

* Develop user authentication system – **Abdullah Sajid**
* Implement vote submission and storage functionality – **Abdullah Sajid**

**6.2.3.3 Database Development**

* Set up and configure the database – **Hamza Zahoor**
* Integrate database with backend – **Abdullah Sajid**

#### **6.2.4 Testing**

**6.2.4.1 Unit Testing**

* Test individual modules – **Hamza Zahoor**

**6.2.4.2 System Testing**

* Ensure system functionality as a whole – **Hamza Zahoor**

**6.2.4.3 User Testing**

* Gather feedback from testers – **Hamza Zahoor**

#### **6.2.5 Deployment**

**6.2.5.1 Deployment Preparation**

* Set up hosting environment – **Abdullah Sajid**
* Configure domain and security protocols – **Abdullah Sajid**

**6.2.5.2 Launch**

* Deploy system to live server – **Abdullah Sajid**

#### **6.2.6 Maintenance**

**6.2.6.1 Bug Fixing**

* Monitor and resolve user-reported issues – **All team members**

**6.2.6.2 Feature Updates**

* Plan and implement new features – **All team members**

### 6.3 Technology Stack and Tools

**Frontend**:

* **HTML5, CSS3, JavaScript** for structure, styling, and interactivity.
* **React** for building a dynamic and responsive user interface.

**Backend**:

* **Php** for server-side scripting.

**Database**:

* **MySql** for database storage (NoSQL database).

**Deployment**:

* **GitHub** for version control and collaboration.

**Testing**:

* **Jest** for unit testing JavaScript code.
* **Postman** for testing API endpoints.

**Other Tools**:

* **Canva** for UI/UX design and wireframing.
* **Jira** for project management and task tracking.