

ELECTION COMMISSION

A Project Report for Industrial Training and Internship

Submitted by

TRISHA GHOSH

SUDIPTA DEY

SUBHRA GHOSH

In the partial fulfillment of the award of the degree of

BCA

In the

BCA Department

Of

BENGAL SCHOOL OF TECHNOLOGY & MANAGEMENT



Ardent Computech Pvt. Ltd.





CERTIFICATE FROM SUPERVISOR

This is to certify that “**Trisha ghosh**,”**232661010072**”, “**Sudipta Dey**”,
”**232661010067**”, “**Subhra Ghosh**,”**232661010066**” have completed the
project titled "**Election Commission**" under my supervision during the period
from “**30/06/2025**” to “**01/08/2025**” which is in partial fulfillment of
requirements for the award of the **BCA** degree and submitted to the
Department of “**BCA**” of “**Bengal School Of Technology & Management**”.

Signature of the

Supervisor Date:

10/08/2025

Name of the Project Supervisor:





BONAFIDE CERTIFICATE

Certified that this project work was carried out under my supervision
“*Election Commission*” is the bonafide work of

Name of the student: Trisha Ghosh

Signature:

Name of the student: Sudipta Dey

Signature:

Name of the student: Subhra Ghosh

Signature:

SIGNATURE

Name :

PROJECT MENTOR

SIGNATURE

Name:

EXAMINERS

Ardent Original Seal



ACKNOWLEDGEMENT

The achievement that is associated with the successful completion of any task would be incomplete without mentioning the names of those people whose endless cooperation made it possible. Their constant guidance and encouragement made all our efforts successful.

We take this opportunity to express our deep gratitude towards our project mentor, **Subhajit Santra** for giving such valuable suggestions, guidance and encouragement during the development of this project work.

Last but not the least we are grateful to all the faculty members of **Ardent Computech Pvt. Ltd.** for their support.

Content Page

1.	COMPANY PROFILE-----	1
2.	INTRODUCTION-----	2
2A.	OBJECTIVE-----	3
2B.	SCOPE-----	4
3.	SYSTEM ANALYSIS-----	5
3A.	IDENTIFICATION OF NEED-----	6
1.	Need for Secure Voting System-----	6
2.	Need for Efficient Electoral Process-----	6
3.	Need for Improved Accessibility-----	6
4.	Need for Real-time Results-----	6
5.	Need for Secure Voter Registration-----	6
6.	Need for Effective Election Management-----	6
3B.	FEASIBILITY STUDY-----	7
3C.	WORKFLOW-----	8
□	Waterfall Model Design-----	8
□	Iterative Waterfall Design-----	8
•	Advantages-----	9
•	Disadvantages-----	9
•	Applications-----	10
3D.	STUDY OF THE SYSTEM-----	11
•	Register-----	11
•	Login-----	11
•	Voting-----	11
•	E-Voter Card(E-EPIC)-----	11
•	Admin Dashboard-----	11

3E. INPUT AND OUTPUT-----	12
<input type="checkbox"/> Input-----	12
<input type="checkbox"/> Output-----	12
3F. SOFTWARE REQUIREMENT SPECIFICATIONS-	13
• Functional Requirements-----	13
• Non- Functional Requirements-----	13
• System Requirements-----	13
• Interfaces-----	13
• Documentation-----	13
4. SYSTEM DESIGN-----	15
4A. DATA FLOW DIAGRAM-----	16
DFD NOTATION-----	17
DED EXAMPLE-----	17
Database Input Output-----	17
Rules for constructing a Data Flow Diagram-----	18
• LEVEL 0 DFD OR CONTEXT DIAGRAM-----	18
<input type="checkbox"/> LEVEL 1 DFD-----	19
<input type="checkbox"/> LEVEL 1 DFD-----	20
4B. SEQUENCE DIAGRAM-----	21
How to draw use case Diagram?-----	25
4C. SCHEMA DIAGRAM-----	27
5. UI SNAPSHOT-----	28
<input type="checkbox"/> FRONTEND-----	28
<input type="checkbox"/> CODE-----	28
CODE-----	32
<input type="checkbox"/> BACKEND-----	98
6. CONCLUSION-----	102

7.	FUTURE SCOPE & FURTHER ENHANCEMENTS-----	103
<input type="checkbox"/>	FUTURE SCOPE-----	103
<input type="checkbox"/>	FURTHER ENHANCEMENT-----	140
1.	Biometric Authentication-----	140
2.	Mobile OTP Verification-----	140
3.	Offline Voting Mode-----	140
4.	AI-Powered Fraud Detection-----	140
5.	Multi-Language Support-----	140
6.	Enhanced Admin Analytics-----	140
7.	Accessibility Features-----	140

1. COMPANY PROFILE

ARDENT (Ardent Computech Pvt. Ltd.), formerly known as Ardent Computech Private Limited, is an ISO 9001:2015 certified Software Development and Training Company based in India. Operating independently since 2003, the organization has recently undergone a strategic merger with ARDENT Technologies, enhancing its global outreach and service offerings.

ARDENT Technologies

ARDENT Technologies delivers high-end IT services across the UK, USA, Canada, and India. Its core competencies lie in the development of customized application software, encompassing end-to-end solutions including system analysis, design, development, implementation, and training. The company also provides expert consultancy and electronic security solutions. Its clientele spans educational institutions, entertainment companies, resorts, theme parks, the service industry, telecom operators, media, and diverse business sectors.

ARDENT Collaborations

ARDENT Collaborations, the Research, Training, and Development division of ARDENT (Ardent Computech Pvt. Ltd.), offers professional IT-enabled services and industrial training programs. These are tailored for freshers and professionals from B.Tech, M.Tech, MBA, MCA, BCA, and MSc backgrounds. ARDENT (Ardent Computech Pvt. Ltd.) provides Summer Training, Winter Training, and Industrial Training to eligible candidates. High-performing students may qualify for stipends, scholarships, and additional benefits based on performance and mentor recommendations.

Associations and Accreditations

ARDENT (Ardent Computech Pvt. Ltd.) is affiliated with the National Council of Vocational Training (NCVT) under the Directorate General of Employment & Training (DGET), Ministry of Labour & Employment, Government of India. The institution upholds strict quality standards under ISO 9001:2015 certification and is dedicated to bridging the gap between academic knowledge and industry skills through innovative training programs.

2. INTRODUCTION

Electra is a state-of-the-art election management system designed for the Election Commission, utilizing the MERN stack—MongoDB, Express.js, React, and Node.js—to deliver a robust, scalable, and user-friendly platform. This application empowers voters with seamless registration, online voting, and e-Voter Card (E-EPIC) downloads, while providing administrators with robust tools to manage candidates, voter lists, and election results. By leveraging these modern web technologies, Electra streamlines voter registration, polling, real-time analytics, and secure authentication processes, ensuring transparency, efficiency, and security in all electoral operations. With its seamless integration of the MERN stack, Electra sets new standards for reliability and inclusivity in election administration, empowering both officials and citizens alike to participate confidently in the democratic process. Designed to meet the challenges of modern democracy, Electra integrates technology-driven solutions for voter registration, polling, result tabulation, and electoral roll management. The project aims to empower citizens, facilitate free and fair elections, and uphold the values of inclusivity and integrity in democratic governance. Through innovative features and robust safeguards, Electra promises to set new standards in election administration, reflecting the Election Commission's commitment to advancing democratic participation.

2A.OBJECTIVE

The primary objectives of the Election Commission Project using MERN stack are multifaceted, aiming to revolutionize the electoral process. Firstly, the project seeks to develop a secure and transparent voting system, ensuring the integrity and confidentiality of the voting process through a web-based application. Additionally, it aims to automate electoral processes, streamlining voter registration, candidate management, and election management to reduce manual errors and increase efficiency. The project also focuses on providing real-time results and analytics, enabling election officials to access accurate and timely information for informed decision-making. Furthermore, it strives to enhance the voter experience by creating a user-friendly interface for registration, voting, and accessing election information. Lastly, the system is designed to ensure scalability and reliability, handling large volumes of data and traffic while maintaining high availability and performance.

Specifically, the project objectives include designing and implementing a secure voter registration system, developing a candidate management system, creating an election management system, implementing a secure online voting system, and developing a results and analytics module. By achieving these objectives, the Election Commission Project aims to modernize the electoral process, increase transparency, and enhance the overall voting experience, ultimately contributing to the integrity and efficiency of democratic processes.

2B.SCOPE

The scope of the Election Commission Project using MERN stack encompasses several key areas:

Core Functionalities

- 1. Voter Registration System:** Develop a secure system for voter registration, allowing citizens to register and update their information.
- 2. Candidate Management System:** Create a platform for managing candidate information, including profiles and election data.
- 3.Election Management System:** Design a system for scheduling and managing elections, including polling stations and voting schedules.
- 4.Online Voting System:** Implement a secure online voting system, ensuring the integrity and confidentiality of the voting process.
- 5.Results and Analytics Module:** Develop a module for displaying real-time election results and analytics.

Technical Requirements

- 1.Frontend:** Utilize React.js for building a user-friendly and dynamic interface.
- 2. Backend:** Leverage Node.js and Express.js for managing server-side logic, API integration, and database operations.
- 3.Database:** Employ MongoDB for storing and managing large volumes of election-related data.

Key Features

- 1.Security and Transparency:** Ensure the security and transparency of the voting process through robust authentication and authorization mechanisms.
- 2. Real-time Updates:** Provide real-time updates on election results and analytics.
- 3.User Experience:** Design an intuitive and user-friendly interface for voters, candidates, and election officials.

Benefits

- 1.Increased Transparency:** Enhance the transparency of the electoral process through real-time updates and secure voting mechanisms.
- 2.Improved Efficiency:** Streamline electoral processes, reducing manual errors and increasing productivity.
- 3. Enhanced User Experience:** Provide a seamless and user-friendly experience for voters, candidates, and election officials..

3.SYSTEM ANALYSIS

3A.IDENTIFICATION OF NEED

System analysis is a crucial phase in the development of our project Electra App involving creating an efficient and user-friendly platform.

The Election Commission Project using Mern stack aims to address the limitations and challenges of traditional voting systems.

Key Identification Needs

- 1. Need for Secure Voting System:** To ensure the integrity and confidentiality of the voting process.
- 2. Need for Efficient Electoral Process:** To reduce manual errors and increase productivity.
- 3. Need for Improved Accessibility:** To enable voters to cast votes from anywhere, reducing geographical constraints.
- 4. Need for Real-time Results:** To provide accurate and timely election results.
- 5. Need for Secure Voter Registration:** To ensure secure registration process for voters.
- 6. Need for Effective Election Management:** To schedule and manage elections efficiently.

3B.FEASIBILITY STUDY

Feasibility Study for Election Commission Project Using MERN Stack

The Election Commission Project using MERN stack has been evaluated for feasibility from technical, operational, economic, and schedule perspectives. Here's a summary of the findings:

The project is technically feasible due to the popularity and scalability of the MERN stack, which can handle large volumes of data and traffic while providing robust security features. Operationally, the system is feasible as it will provide a user-friendly interface, ensuring high user acceptance, and training and support will be provided to stakeholders.

From an economic perspective, the project is feasible as it will reduce manual errors and increase productivity, resulting in cost savings. The MERN stack's scalability will also enable the system to scale up or down as needed, reducing infrastructure costs.

In terms of schedule feasibility, a realistic timeline will be developed, and key milestones will be identified and tracked to ensure the project stays on schedule.

It is recommended to proceed with development, involving all stakeholders and conducting regular reviews and testing to ensure the system meets the required standards and is delivered on time. The proposed system will enhance the electoral process, improve voter experience, and provide a reliable platform for election management. With careful planning and execution, the project is expected to achieve its objectives and deliver significant benefits to the electoral process.

3C.WORKFLOW

This Document plays a vital role in the development life cycle (SDLC) as it describes the complete requirements of the system. It is meant for use by the developers and will be the basic during the testing phase. Any changes made to the requirements in the future will have to go through a formal change approval process.

The Waterfall Model was the first Process Model to be introduced. It is also referred to as a linear-sequential life cycle model. It is very simple to understand and use. In a waterfall model, each phase must be completed before the next phase can begin and there is no overlapping in the phases.

The waterfall model is the earliest SDLC approach that was used for software development. The waterfall Model illustrates the software development process in a linear sequential flow; hence it is also referred to as a linear-sequential life cycle model. This means that any phase in the development process begins only if the previous phase is complete. In the waterfall model phases do not overlap.

○ Waterfall Model Design:

The waterfall approach was the first SDLC Model to be used widely in Software Engineering to ensure the success of the project. In "The Waterfall" approach, the whole process of software development is divided into separate phases. In the Waterfall model, typically, the Outcome of one phase acts as the input for the next phase sequentially.

○ Iterative Waterfall Design:

Definition: The Iterative Waterfall Model is a variation of the traditional Waterfall model, which is a linear and sequential software development methodology. In the Iterative Waterfall Model, the development process is divided into small, manageable cycles, allowing for the revisiting and refinement of phases before progressing to the next stage. It combines the systematic structure of the Waterfall model with the flexibility of iterative development.

The sequential phases in Iterative Waterfall model are:

- **Requirement Gathering and Analysis:** All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification doc.
- **System Design:** The requirement specifications from the first phase are studied in this phase and system design is prepared. System Design helps in specifying hardware and system requirements and also helps in defining overall system architecture.
- **Implementation:** With inputs from system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality which is referred to as Unit Testing.
- **Integration and Testing:** All the units developed in the implementation phase are integrated into a system after testing each unit. Post integration the entire system is tested for any faults and failures.
- **Deployment of the system:** Once the functional and non-functional testing is done, the product is deployed in the customer environment or released into the market.
- **Maintenance:** Some issues come up in the client environment. To fix those issues patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

All these phases are cascaded to each other in progress and are seen as flowing steadily downwards (like a waterfall) through the phases. The next phase is started only after the defined set of goals are achieved for the previous phase and it is signed off, so the name "Iterative Waterfall Model". In this model, phases do not overlap.

○ Advantages:

1. Flexibility: Iterations permit adjustments based on feedback.

2. Early Delivery: Partial systems can be delivered incrementally.

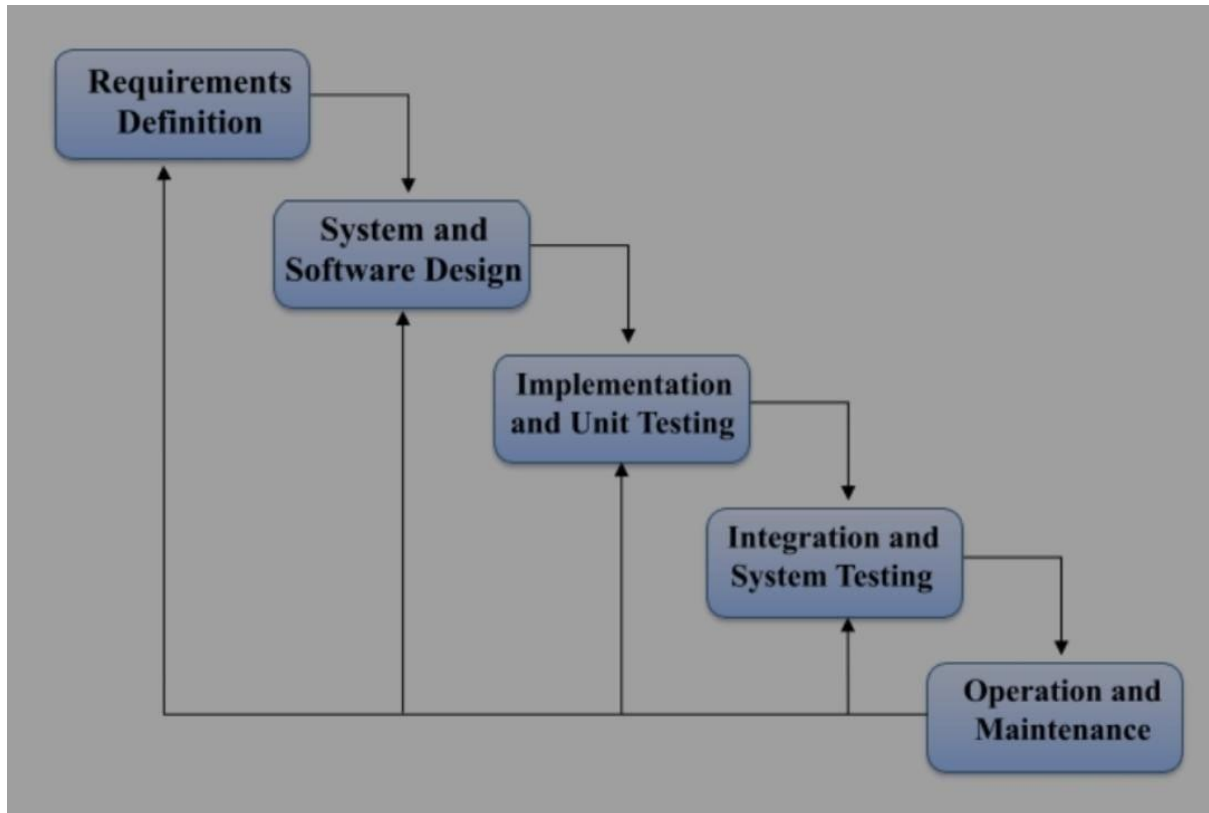
3. Risk Management: Identifying and addressing issues early in the process.

○ **Disadvantages:**

1. Increased Complexity: The iterative nature can make the process more complex.

2. Potential for Scope Creep: Frequent iterations may lead to scope changes,

3. Resource Intensive: Continuous revisiting of phases may demand more resources.



○ Applications:

The Iterative Waterfall Model is suitable for projects with evolving or unclear requirements. It is commonly used in software development projects where regular feedback and refinement are essential.

Additionally, it is applicable in scenarios where partial system delivery is beneficial, allowing stakeholders to assess progress and make adjustments.

3D. STUDY OF THE SYSTEM

Modules: The modules used in the software are as follows:

- **Register:**
Voters can register themselves on the platform, providing necessary details for verification.
- **Login:**
Registered voters can log in securely using their credentials, ensuring only authorized access.
- **Voting:**
Voters can cast their votes electronically through the platform.
- **E-Voter Card (E-EPIC):**
Digital voter cards can be generated and managed through the system.
- **Admin Dashboard:**
Administrators can manage elections, monitor voting processes, and view results.

3E.INPUT AND OUTPUT

The main inputs,outputs and the major function in the details are :

○ Input:

- Voter Registration: Voter details (name, age, address, etc.).
- Login Credentials: Username and password for secure login.
- Voting: Voter's choice of candidate or option.
- Admin Input: Election schedules, candidate information, and other administrative data.

○ Output:

- Voter ID Card (E-EPIC): Digital voter card with unique ID and details.
- Voting Confirmation: Confirmation message or receipt after successful voting.
- Election Results: Tabulated results of the election, including vote counts and winner(s).
- Admin Dashboard: Real-time data and analytics on voter turnout, election progress, and results.

3F.SOFTWARE REQUIREMENT SPECIFICATION:

Software Requirements Specification for Election Commission Project Using MERN Stack

- **Functional Requirements**

- A. **Voter Registration:** The system shall allow voters to register themselves with necessary details.
- B. **Login and Authentication:** The system shall provide secure login and authentication mechanisms for voters and administrators.
- C. **Voting:** The system shall enable voters to cast their votes electronically.
- D. **E-Voter Card (E-EPIC) Generation:** The system shall generate digital voter cards with unique IDs.
- E. **Admin Dashboard:** The system shall provide an admin dashboard for managing elections, monitoring voting processes, and viewing results.
- F. **Results Management:** The system shall calculate and display election results in real-time.

- **Non-Functional Requirements**

- A. **Security:** The system shall ensure the confidentiality, integrity, and availability of voter data and election results.
- B. **Performance:** The system shall handle a large number of concurrent users and transactions efficiently.
- C. **Usability:** The system shall provide an intuitive and user-friendly interface for voters and administrators.
- D. **Scalability:** The system shall be scalable to accommodate increasing voter numbers and election complexity.
- E. **Reliability:** The system shall ensure high availability and minimize downtime.

- **System Requirements:**

A. **Hardware Requirements:** Servers with sufficient processing power, memory, and storage.

B. **Software Requirements:** MERN stack (MongoDB, Express.js, React.js, Node.js) and compatible operating system.

C. **Network Requirements:** Secure and reliable network infrastructure for data transmission.

- **Interfaces:**

A. **User Interface:** Web-based interface for voters and administrators.

B. **Database Interface:** Interface for interacting with the MongoDB database.

C. **API Interface:** API endpoints for data exchange and integration with other systems.

- **Documentation:**

A. **User Manual:** Documentation for voters and administrators on using the system.

B. **System Documentation:** Technical documentation for developers and maintainers, including system architecture, design, and implementation details.

C. **API Documentation:** Documentation for API endpoints, including request and response formats, and usage examples.

By specifying these requirements, the Election Commission Project using MERN stack can be developed to meet the needs of voters, administrators, and stakeholders, ensuring a secure, efficient, and reliable electoral process.

4.SYSTEM DESIGN

4A.DATA FLOW DIAGRAM

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modeling its process aspects. A DFD is often used as a preliminary step to create an overview of the system, which can later be elaborated.

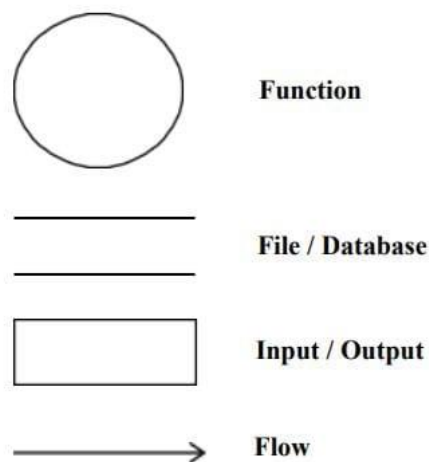
DFD can also be used for the visualization of data processing (structured design). A DFD shows what kind of information will be input to and output from the system, where the data will come from and go to, and where the data will be stored. It does not show information about the timing of the process or information about whether processes will operate in sequence or in parallel (which is shown on a flowchart).

This context-level DFD is next "exploded", to produce a Level 1 DFD that shows some of the detail of the system being modeled. The Level 1 DFD shows how the system is divided into sub-systems (processes), each of which deals with one or more of the data flows to or from an external agent, and which together provide all of the functionality of the system as a whole. It also identifies internal data stores that must be present for the system to do its job and shows the flow of data between the various parts of the system.

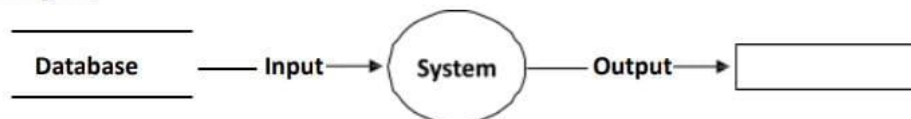
Data flow diagrams are one of the three essential perspectives of the structured-systems analysis and design method SSADM. The sponsor of a project and the end users will need to be briefed and consulted throughout all stages of a system's evolution. With a data flow diagram, users can visualize how the system will operate, what the system will accomplish, and how the system will be implemented. The old system's data-flow diagrams can be drawn up and compared with.

How any system is developed can be determined through a data flow diagram model. In the course of developing a set of leveled data flow diagrams, the analyst/designer is forced to address how the system may be decomposed into component sub-systems and to identify the transaction data in the data model. Data flow diagrams can be used in both the Analysis and Design phase of the SDLC. There are different notations to draw data flow diagrams. Defining different visual representations for processes, data stores, data flow, and external entities.

DFD Notation:



DFD Example:



Steps to Construct Data Flow Diagram:

Four Steps are generally used to construct a DFD.

- Process should be named and referred for easy reference. Each name should be representative of the reference.
- The destination of flow is from top to bottom and from left to right.
- When a process is distributed into lower-level details they are numbered.
- The names of data stores, sources, and destinations are written in capital letters.

Rules for constructing a Data Flow Diagram:

- Arrows should not cross each other.
- Squares, Circles, and Files must bear a name,
- Decomposed data flow squares and circles can have the same names.
- Draw all data flow around the outside of the diagram.

LEVEL 0 DFD OR CONTEXT DIAGRAM:

[X] (Optional)

| "Sentiment Data"

v

[Election Commission Portal]

| | |

"Registration Data" | "Election Settings" |

"Vote Data" | "Management Commands" |

| | | |

v | v |

[Voter] <---- "E-EPIC" [Admin] <---- "Election Results"

| "Vote Confirmation" |

| |

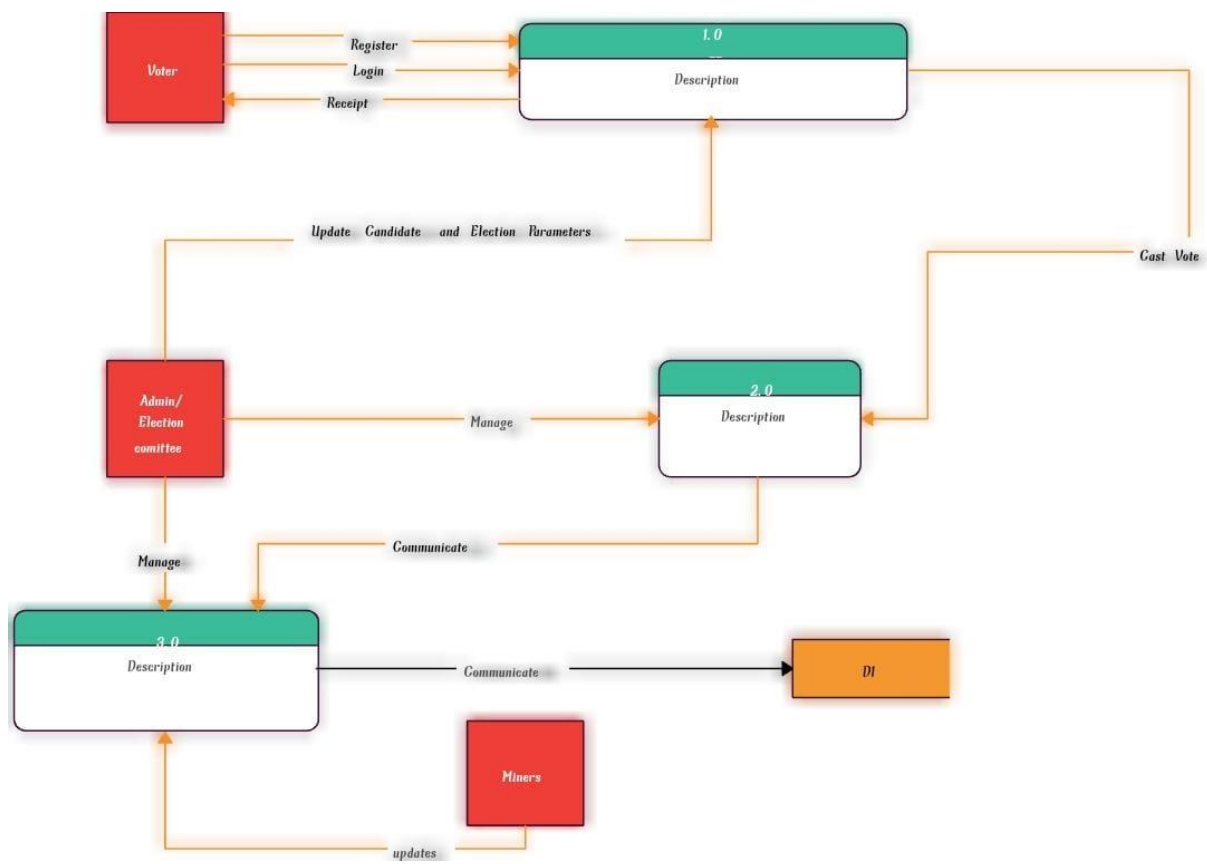
|-----|

| |

"Voter Details" [Voter Database] "Candidate Details" [Candidate Database]

"Updated Voter Info" "Candidate List"

- **LEVEL 1 DFD:**



4B.SEQUENCE DIAGRAM

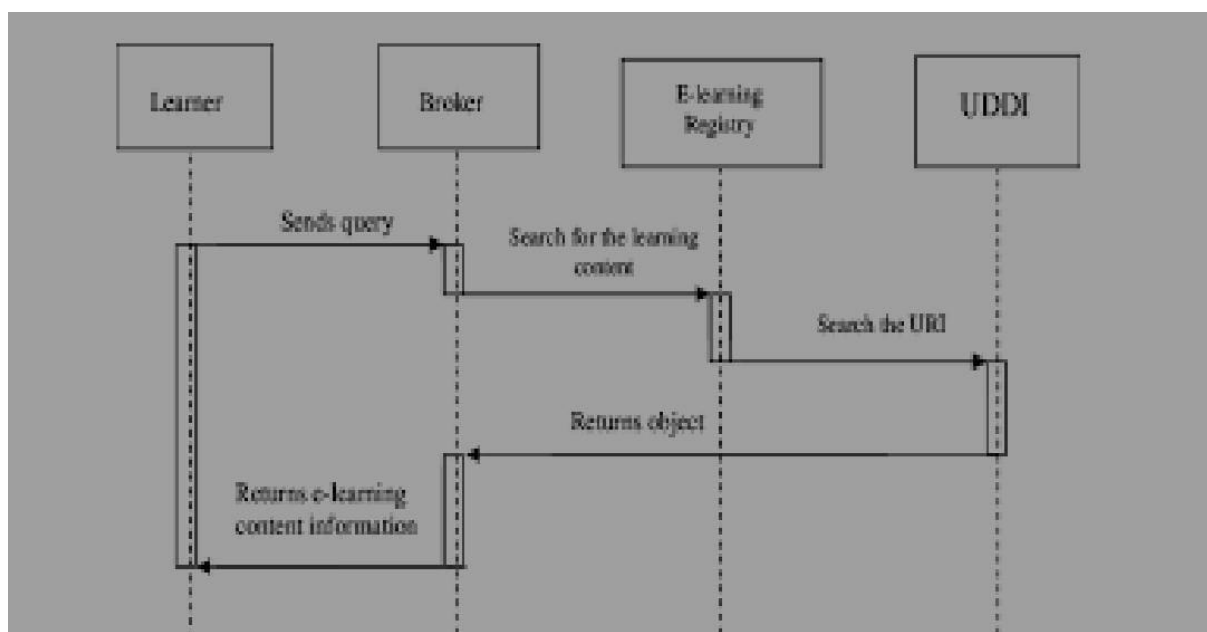
A Sequence diagram is an interaction diagram that shows how processes operate with one another and what is their order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in a time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development.

Sequence diagrams are sometimes called event diagrams or event scenarios

A sequence diagram shows, as parallel vertical lines (lifelines), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.

A sequence diagram is the most common kind of interaction diagram, which focuses on the message interchange between several life lines A sequence diagram describes an interaction by focusing on the sequence of messages that are exchanged, along with their corresponding occurrence specifications on the lifelines.

The following nodes and edges are typically drawn in a UML sequence diagram: lifeline, execution specification, message, fragment, interaction, state invariant, continuation, and destruction occurrence.



A Use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved. A use case diagram can identify the different types of users of a system and the different use cases and will often be accompanied by other types of diagrams as well.

So only static behavior is not sufficient to model a system rather dynamic behavior is more important than static behavior. In UML there are five diagrams available to model dynamic nature and a use case diagram is one of them. Now as we have to discuss that the use case diagram is dynamic in nature there should be some internal or external factors for making the interaction.

These internal and external agents are known as actors. So, use case diagrams consist of actors, use cases, and their relationships. The diagram is used to model the system/subsystem of an application. A single-use case diagram captures a particular functionality of a system.

So, to model the entire system numbers of use case diagrams are used. The purpose of a use case diagram is to capture the dynamic aspect of a system. But this definition is too generic to describe the purpose.

Because the other four diagrams (activity, sequence, collaboration, and State chart) are also having the same purpose. So, we will look into some specific purpose that will distinguish it from the other four diagrams.

Use case diagrams are used to gather the requirements of a system including internal and external influences. These requirements are mostly design requirements. So, when a system is analyzed to gather its functionalities use cases are prepared and actors are identified.

Now when the initial task is complete use case diagrams are modeled to present the outside view. So, in brief, the purposes of use case diagrams can be as follows:

- Used to gather requirements of a system.
- Used to get an outside view of a system.
- Identify external and internal factors influencing the system.

How to draw Use Case Diagram?

Use case diagrams are considered for high level requirement analysis of a system. So, when the requirements of a system are analyzed, the functionalities are captured in use cases.

So, we can say that use cases are nothing but the system functionalities written in an organized manner. Now the second things which are relevant to the use cases are the actors. Actors can be defined as something that interacts with the system.

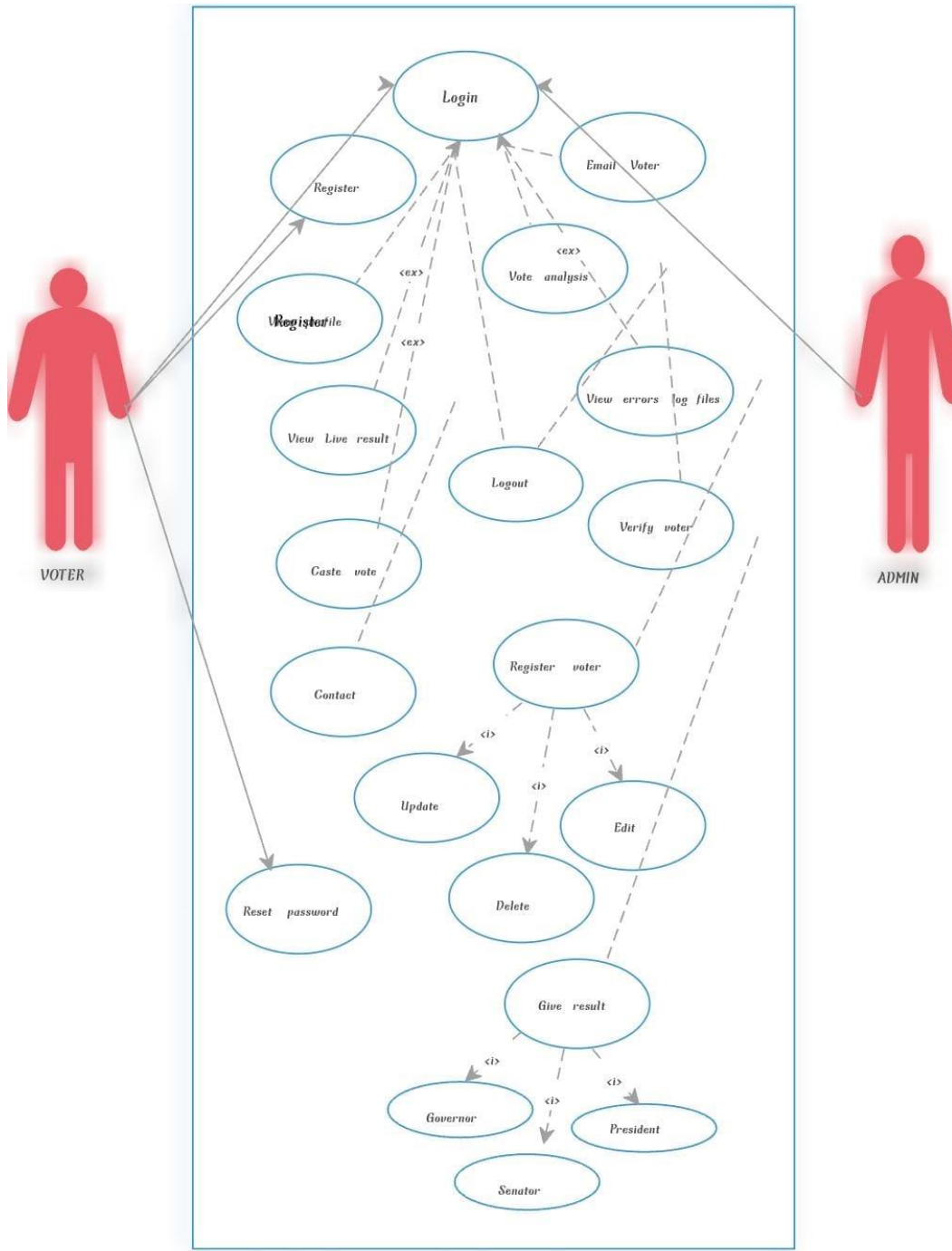
The actors can be human user, some internal applications or may be some external applications. So, in a brief when we are planning to draw use case diagram, we should have the following items identified.

- Functionalities to be represented as a use case
- Actors
- Relationships among the use cases and actors.

Use case diagrams are drawn to capture the functional requirements of a system. So, after identifying the above items we have to follow the following guidelines to draw an efficient use case diagram.

- The name of a use case is very important. So, the name should be chosen in such a way so that it can identify the functionalities performed.
- Give a suitable name for actors.
- Show relationships and dependencies clearly in the diagram.
- Do not try to include all types of relationships. Because the main purpose of the diagram is to identify requirements.
- Use note whenever required to clarify some important point.

- **USE CASE DIAGRAM**



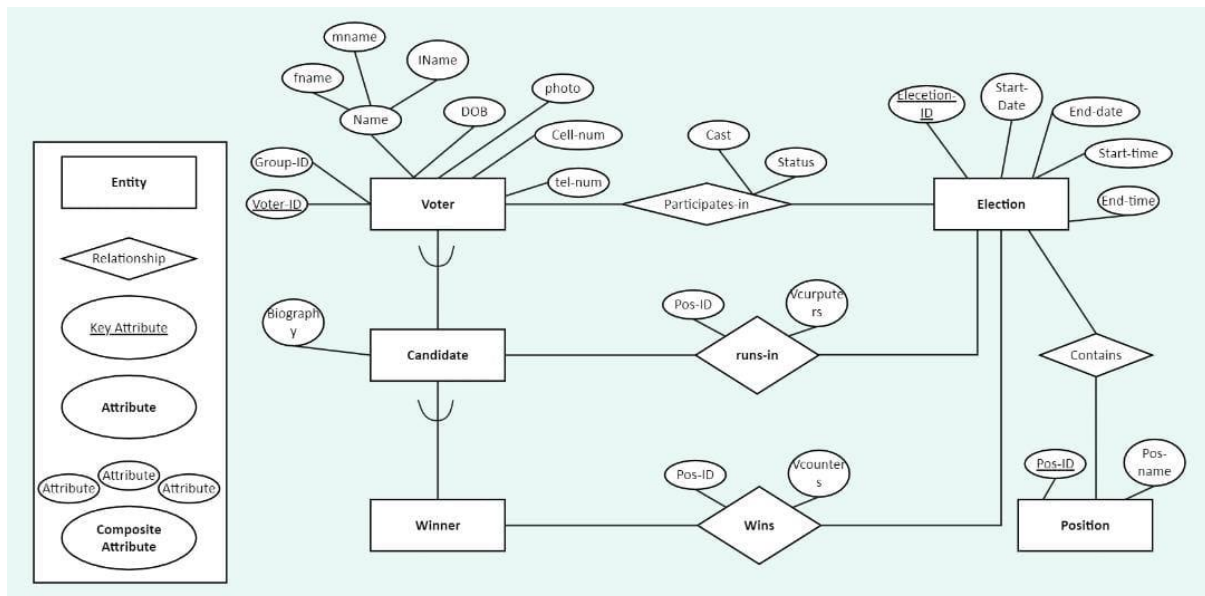
USE-CASE DIAGRAM FOR ONLINE VOTING SYSTEM

4D.SCHEMA DIAGRAM

The schema is an abstract structure or outline representing the logical view of the database as a whole. Defining categories of data and relationships between those categories, database schema design makes data much easier to retrieve, consume, manipulate, and interpret.

DB schema design organizes data into separate entities, determines how to create relationships between organized entities, and influences the applications of constraints on data. Designers create database schema to give other database users, such as programmers and analysts, a logical understanding of data.

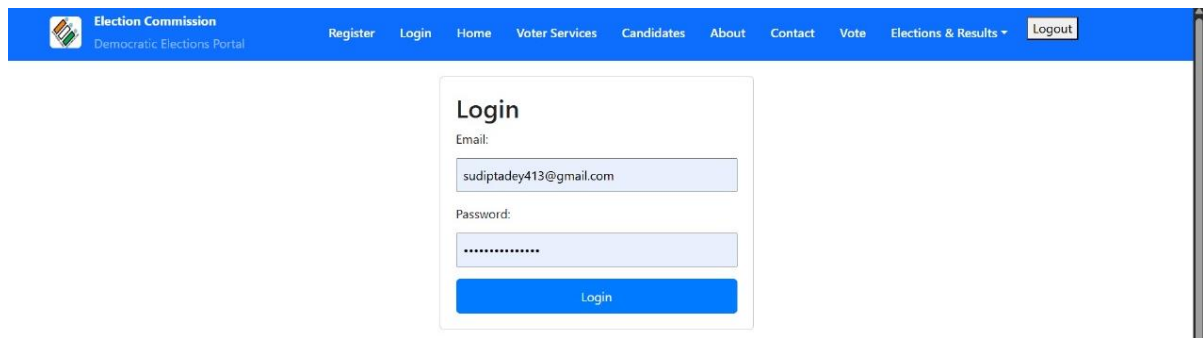
- **SCHEMA DIAGRAM:**



5.UI SNAPSHOT

❖ FRONTEND

1) LOGIN PAGE:



The screenshot shows the login page of the Election Commission Democratic Elections Portal. The header is blue with the logo on the left and navigation links: Register, Login, Home, Voter Services, Candidates, About, Contact, Vote, Elections & Results, and Logout. The main content area is white and contains a 'Login' form. The form has two input fields: 'Email' with the value 'sudiptadey413@gmail.com' and 'Password' with masked characters. A blue 'Login' button is at the bottom of the form.

✓ CODE

```
import React, { useState } from 'react';
import { useNavigate } from 'react-router-dom';
import axios from 'axios';

const Login = () => {
  const [form,setForm] = useState({email:"",password:"});
  const navigate = useNavigate();

  const hc =(e)=> {
    setForm({ ...form,[e.target.name]:e.target.value });
  }
  const hs = async(e) => {
    e.preventDefault();
    const res = await axios.post('http://localhost:5000/api/auth/login',form);
    localStorage.setItem('token',res.data.token);
    navigate('/home');
  }

  return (
    <div style={{ maxWidth: '400px', margin: '20px auto', padding: '20px', border: '1px
    solid #ccc', borderRadius: '5px' }}>
```

```

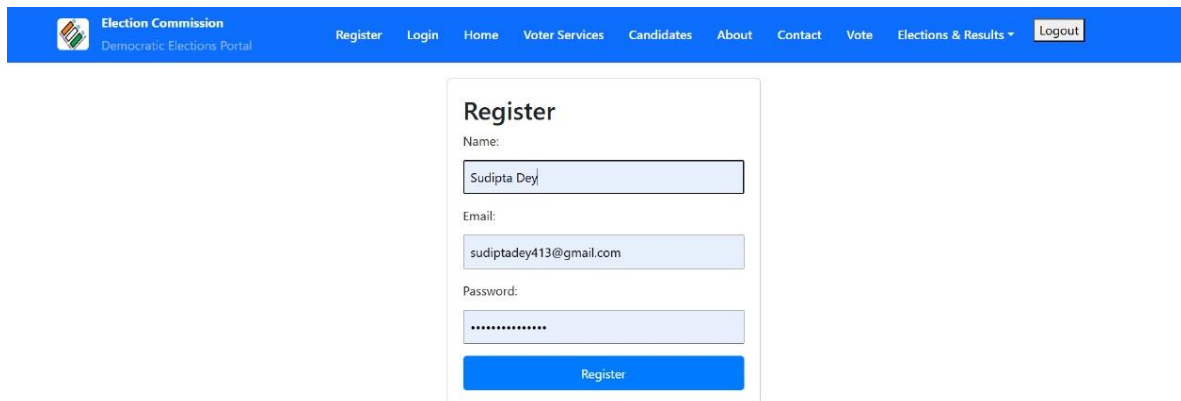
<h2>Login</h2>

<form onSubmit={hs}>
  <div style={{ marginBottom: '15px' }}>
    <label>Email:</label>
    <input
      type="email"
      name="email"
      value={form.email}
      onChange={hc}
      required
      style={{ width: '100%', padding: '8px', marginTop: '5px' }}
    />
  </div>
  <div style={{ marginBottom: '15px' }}>
    <label>Password:</label>
    <input
      type="password"
      name="password"
      value={form.password}
      onChange={hc}
      required
      style={{ width: '100%', padding: '8px', marginTop: '5px' }}
    />
  </div>
  <button type="submit" style={{ width: '100%', padding: '10px', backgroundColor:
'#007bff', color: 'white', border: 'none', borderRadius: '5px' }}>
    Login
  </button>
</form>
</div>
);
};

export default Login;

```

2)REGISTER PAGE:



✓ CODE

```
import React, { useState } from 'react';
import { useNavigate } from 'react-router-dom';
import axios from 'axios';

const Register = () => {
  const [form,setForm] = useState({ name:"",email:"",password:"});
  const navigate = useNavigate();

  const hc =(e)=> {
    setForm({ ...form,[e.target.name]:e.target.value });
  }

  const hs = async(e) => {
    e.preventDefault();
    await axios.post('http://localhost:5000/api/auth/register',form);
    alert('Register sucessfully');
    navigate('/login');
  }

  return (
    <div style={{ maxWidth: '400px', margin: '20px auto', padding: '20px', border: '1px solid #ccc', borderRadius: '5px' }}>
      <h2>Register</h2>

      <form onSubmit={hs}>
        <div style={{ marginBottom: '15px' }}>
          <label>Name:</label>
          <input
            type="text"
            name="name"
```

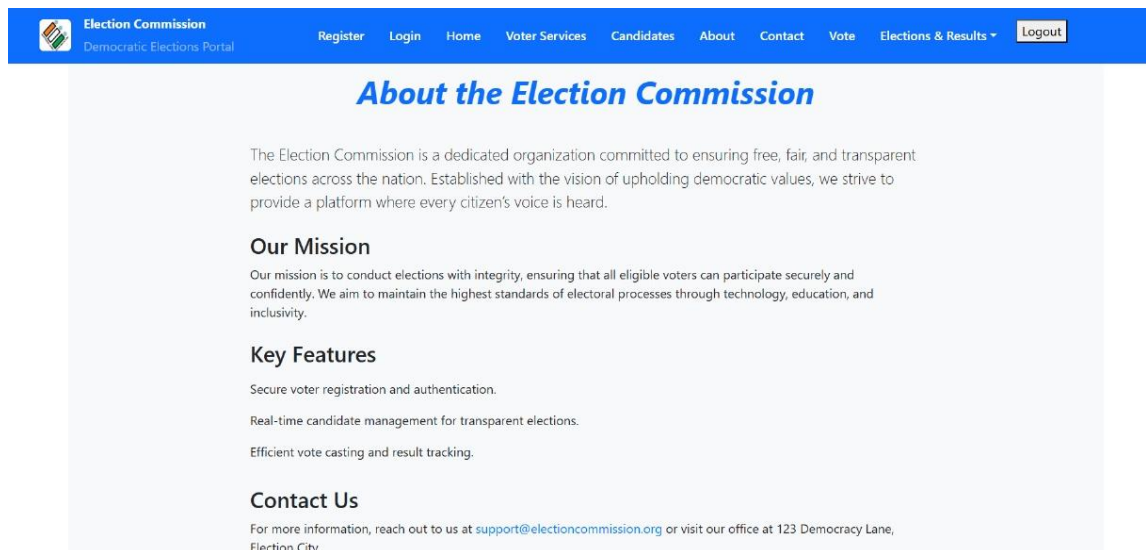
```

        value={ form.name }
        onChange={ hc }
        required
        style={{ width: '100%', padding: '8px', marginTop: '5px' }}
      />
    </div>
    <div style={{ marginBottom: '15px' }}>
      <label>Email:</label>
      <input
        type="email"
        name="email"
        value={ form.email }
        onChange={ hc }
        required
        style={{ width: '100%', padding: '8px', marginTop: '5px' }}
      />
    </div>
    <div style={{ marginBottom: '15px' }}>
      <label>Password:</label>
      <input
        type="password"
        name="password"
        value={ form.password }
        onChange={ hc }
        required
        style={{ width: '100%', padding: '8px', marginTop: '5px' }}
      />
    </div>
    <button type="submit" style={{ width: '100%', padding: '10px', backgroundColor:
'#007bff', color: 'white', border: 'none', borderRadius: '5px' }}>
      Register
    </button>
  </form>
</div>
);
};

export default Register;

```

3)ABOUT:



✓ CODE

```
import React from 'react';
import './About.css'; // Ensure this file exists

const About = () => {
  return (
    <div className="container about-page py-5">
      <h1 className="text-center mb-4 text-primary fw-bold">About the Election
Commission</h1>
      <div className="row justify-content-center">
        <div className="col-md-8">
          <p className="lead">
            The Election Commission is a dedicated organization committed to ensuring
            free, fair, and transparent elections across the nation. Established with the vision of
            upholding democratic values, we strive to provide a platform where every citizen's
            voice is heard.
          </p>
          <h3 className="mt-4">Our Mission</h3>
          <p>
            Our mission is to conduct elections with integrity, ensuring that all eligible
            voters can participate securely and confidently. We aim to maintain the highest
            standards of electoral processes through technology, education, and inclusivity.
          </p>
          <h3 className="mt-4">Key Features</h3>
          <ul className="list-group">
            <li className="list-group-item">Secure voter registration and
```

authentication.

<li className="list-group-item">Real-time candidate management for transparent elections.

<li className="list-group-item">Efficient vote casting and result tracking.

<h3 className="mt-4">Contact Us</h3>

<p>

For more information, reach out to us at support@electioncommission.org or visit our office at 123 Democracy Lane, Election City.

</p>

</div>

</div>

</div>

);

};

export default About;

4)CONTACT:

The screenshot shows the 'Contact Us' page of the Election Commission Democratic Elections Portal. The page has a blue header with the logo and navigation links: Register, Login, Home, Voter Services, Candidates, About, Contact, Vote, Elections & Results, and Logout. The main content area is light gray and contains the following text:

We're here to assist you with any questions or concerns regarding the election process. Feel free to reach out to us using the form below or through our contact details.

Contact Information
Email: support@electioncommission.org
Phone: +1-800-ELECTION (800-353-8246)
Address: 123 Democracy Lane, Election City, EC 12345

Send Us a Message

Name

Email

Message

```
import React, { useState } from 'react';
import './Contact.css'; // Ensure this file exists

const Contact = () => {
  const [formData, setFormData] = useState({
    name: "",
    email: "",
    message: "",
  });
  const [success, setSuccess] = useState("");
  const [error, setError] = useState("");

  const handleChange = (e) => {
    setFormData({ ...formData, [e.target.name]: e.target.value });
  };

  const handleSubmit = async (e) => {
    e.preventDefault();
    setSuccess("");
    setError("");

    if (!formData.name || !formData.email || !formData.message) {
      setError('All fields are required');
```

```

    return;
  }

  try {
    // Simulate sending the form data to the backend (replace with actual
    API call)
    await new Promise((resolve) => setTimeout(resolve, 1000)); // Mock
    delay
    setSuccess('Message sent successfully! We will get back to you
    soon.');
```

setFormData({ name: "", email: "", message: "" }); // Reset form
 } catch (err) {
 setError('Failed to send message. Please try again later.');

```

  }
};

return (
  <div className="container contact-page py-5">
    <h1 className="text-center mb-4 text-primary fw-bold">Contact
    Us</h1>
    <div className="row justify-content-center">
      <div className="col-md-8">
        <p className="lead">
          We're here to assist you with any questions or concerns regarding
          the election process. Feel free to reach out to us using the form below or
          through our contact details.
        </p>
        <h3 className="mt-4">Contact Information</h3>
        <p>
          <strong>Email:</strong> <a
          href="mailto:support@electioncommission.org">support@electioncom
          mission.org</a><br />
          <strong>Phone:</strong> +1-800-ELECTION (800-353-
          8246)<br />
          <strong>Address:</strong> 123 Democracy Lane, Election City,
          EC 12345
        </p>

```




```
<h3 className="mt-4">Send Us a Message</h3>
{error && <p style={{ color: 'red' }}>{error}</p>}
{success && <p style={{ color: 'green' }}>{success}</p>}
<form onSubmit={handleSubmit}>
  <div className="mb-3">
    <label className="form-label">Name</label>
    <input
      type="text"
      className="form-control"
      name="name"
      value={formData.name}
      onChange={handleChange}
      required
    />
  </div>
  <div className="mb-3">
    <label className="form-label">Email</label>
    <input
      type="email"
      className="form-control"
      name="email"
      value={formData.email}
      onChange={handleChange}
      required
    />
  </div>
  <div className="mb-3">
    <label className="form-label">Message</label>
    <textarea
      className="form-control"
      name="message"
      value={formData.message}
      onChange={handleChange}
      rows="4"
      required
    />
  </div>
</form>
```

```
        <button type="submit" className="btn btn-primary w-100">
          Send Message
        </button>
      </form>
    </div>
  </div>
</div>
);
};

export default Contact;
```

5)CANDIDATE



Election Commission

Democratic Elections Portal

Register

Login

Home

Voter Services

Candidates

About

Contact

Vote

Elections & Results

Logout

CANDIDATE LIST

NAME	PARTY	CONSTITUENCY	CANDIDATE ID
Rachana Banerjee	Trinamool Congrees	Hooghly	CAN1001
Locket Chatterjee	Bjp	Hooghly	CAN1002
Manadip Ghosh	CPIM	Hooghly	CAN1003
Abdul Suhel	Nirdol	Hooghly	CAN1004
Pratul Chandra Saha	Indian National Congress	Hooghly	CAN1005
Ajanta Sarkar	Bharatiya National Janata Dal	Hooghly	CAN1006
Trisha Ghosh	Nirdal	Hooghly	CAN1007
Subhra Ghosh	Nirdal	Hooghly	CAN1008
Sudipta Dey	Nirdal	Hooghly	CAN1009

```
import React, { useState, useEffect } from 'react';
import axios from 'axios';
import './Candidates.css';

const Candidates = () => {
  const [formData, setFormData] = useState({
    name: "",
    party: "",
    constituency: "",
    candidateId: "",
  });
  const [candidates, setCandidates] = useState([]);
  const [loading, setLoading] = useState(true);

  const token = localStorage.getItem('token');

  const handleChange = (e) => {
    setFormData({ ...formData, [e.target.name]: e.target.value });
  };

  const handleSubmit = async (e) => {
    e.preventDefault();
```

```

console.log('Submitting formData:', formData);

// Validation check
if (
  !formData.name.trim() ||
  !formData.party.trim() ||
  !formData.constituency.trim() ||
  !formData.candidateId.trim()
) {
  alert('All fields are required');
  return;
}

try {
  await axios.post(
    'http://localhost:5000/api/candidates',
    formData,
    {
      headers: { Authorization: Bearer ${token} },
    }
  );
  alert('Candidate registered successfully');
  setFormData({ name: "", party: "", constituency: "", candidateId: "" });
  fetchCandidates();
} catch (err) {
  console.error('Error creating candidate:', err.response || err);
  alert(err.response?.data?.message || 'Failed to register candidate');
}
};

const fetchCandidates = async () => {
  try {
    const apiUrl = import.meta.env.VITE_API_URL || 'http://localhost:5000';
    const response = await axios.get(`${apiUrl}/api/candidates`, {
      headers: { Authorization: Bearer ${token} },
    });
    setCandidates(response.data);
  } catch (err) {
    console.error('Error fetching candidates:', err);
    alert('Failed to fetch candidates');
  } finally {
    setLoading(false);
  }
};

```

```

useEffect(() => {
  fetchCandidates();
}, []);

if (loading) return <div>Loading...</div>;

return (
  <div className="container candidates-page py-5">
    <h2 className="mb-4 text-center text-primary fw-bold">Manage
Candidates</h2>
    <div className="row justify-content-center">
      <div className="col-md-6 col-lg-5">
        <div className="card shadow border-0 mb-4">
          <div className="card-body p-4">
            <h4 className="mb-3">Add New Candidate</h4>
            <form onSubmit={handleSubmit}>
              <div className="mb-3">
                <label className="form-label">Name</label>
                <input
                  type="text"
                  className="form-control"
                  name="name"
                  value={formData.name}
                  onChange={handleChange}
                  required
                />
              </div>
              <div className="mb-3">
                <label className="form-label">Party</label>
                <input
                  type="text"
                  className="form-control"
                  name="party"
                  value={formData.party}
                  onChange={handleChange}
                  required
                />
              </div>
              <div className="mb-3">
                <label className="form-label">Constituency</label>
                <input
                  type="text"
                  className="form-control"

```

```

        name="constituency"
        value={ formData.constituency }
        onChange={ handleChange }
        required
      />
    </div>
    <div className="mb-4">
      <label className="form-label">Candidate ID</label>
      <input
        type="text"
        className="form-control"
        name="candidateId"
        value={ formData.candidateId }
        onChange={ handleChange }
        required
      />
    </div>
    <button type="submit" className="btn btn-primary w-100">
      Add Candidate
    </button>
  </form>
</div>
</div>
</div>

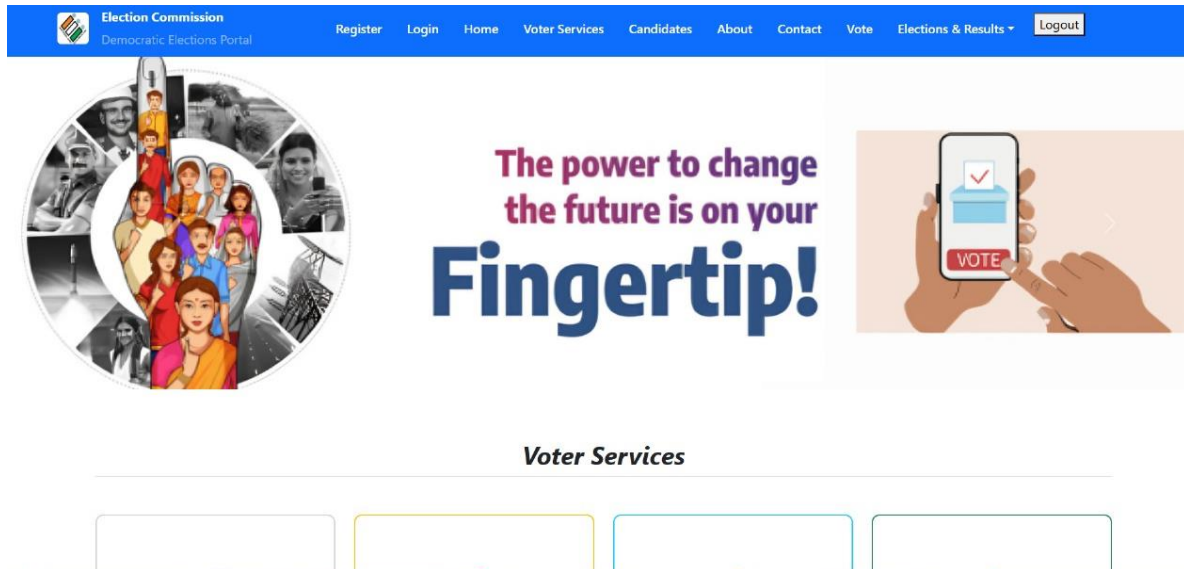
<div className="col-md-8">
  <h4 className="mb-3">Candidate List</h4>
  <table className="table table-striped">
    <thead>
      <tr>
        <th>Name</th>
        <th>Party</th>
        <th>Constituency</th>
        <th>Candidate ID</th>
      </tr>
    </thead>
    <tbody>
      { candidates.length > 0 ? (
        candidates.map((candidate) => (
          <tr key={ candidate._id || candidate.candidateId }>
            <td>{ candidate.name }</td>
            <td>{ candidate.party }</td>
            <td>{ candidate.constituency }</td>
            <td>{ candidate.candidateId }</td>
          </tr>
        ))
      ) : null }
    </tbody>
  </table>

```

```
        </tr>
      ))
    ): (
      <tr>
        <td colSpan="4" className="text-center">
          No candidates found.
        </td>
      </tr>
    )}
  </tbody>
</table>
</div>
</div>
</div>
);
};

export default Candidates;
```

6)HOME:



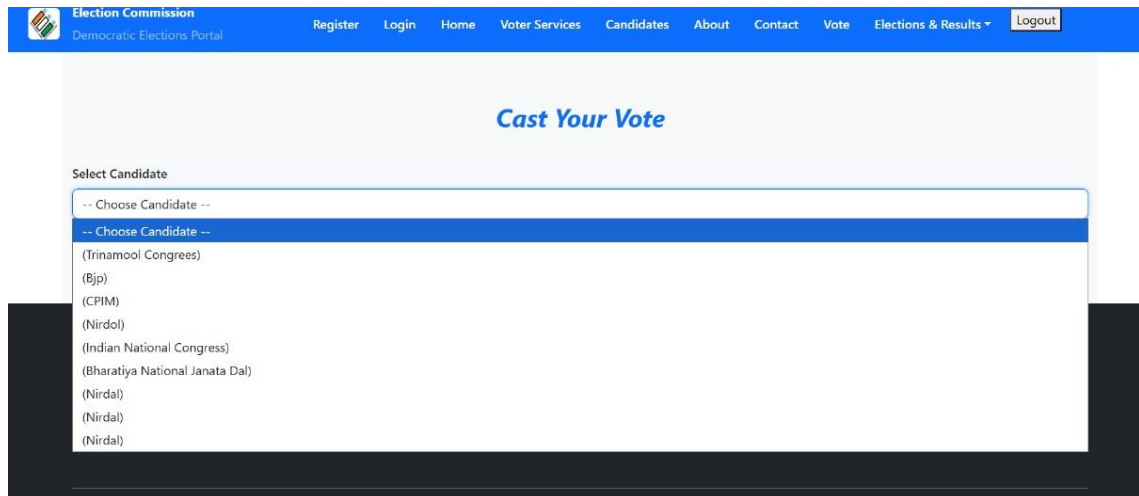
✓ CODE:

```
import React from 'react'
import CarouselSlider from '../Components/CarouselSlider'
import HomePage from '../Components/HomePage'

const Home = () => {
  return <>
    <CarouselSlider/>
    <HomePage/>
  </>
}

export default Home
```


7)VOTE:



✓ CODE:

```
import React, { useState, useEffect } from 'react';
import { useNavigate } from 'react-router-dom';
import axios from 'axios';
import './Vote.css';
```

```
const Vote = () => {
  const [candidates, setCandidates] = useState([]);
  const [selectedCandidate, setSelectedCandidate] = useState("");
  const [error, setError] = useState("");
  const [success, setSuccess] = useState("");
  const navigate = useNavigate();
```

```
const API_BASE = import.meta.env.VITE_API_URL || 'http://localhost:5000';
```

```
// Fetch candidates
```

```
useEffect(() => {
  const fetchCandidates = async () => {
    try {
      const res = await axios.get(`${API_BASE}/api/candidates, {
        headers: {
          Authorization: Bearer ${localStorage.getItem('token')},
        },
      });
    }
  };
});
```

```

        setCandidates(res.data);
    } catch {
        setError('Failed to fetch candidates');
    }
};
fetchCandidates();
}, [API_BASE]);

// Submit vote
const handleSubmit = async (e) => {
    e.preventDefault();
    if (!selectedCandidate) {
        setError('Please select a candidate');
        return;
    }
    try {
        setError("");
        await axios.post(
            `${API_BASE}/api/vote`,
            { candidateId: selectedCandidate },
            {
                headers: {
                    Authorization: Bearer ${localStorage.getItem('token')},
                },
            }
        );
        setSuccess('👍 Vote cast successfully!');
        setTimeout(() => navigate('/'), 2000);
    } catch (err) {
        setError(err.response?.data?.message || 'Failed to cast vote');
    }
};

return (
    <div className="container vote-page py-5">
        <h2 className="mb-4 text-center text-primary fw-bold">Cast Your Vote</h2>

        {error && <p className="text-danger">{error}</p>}
        {success && <p className="text-success">{success}</p>}


        <form onSubmit={handleSubmit}>
            <div className="mb-3">
                <label className="form-label">Select Candidate</label>

```

```
    <select
      className="form-control"
      value={selectedCandidate}
      onChange={(e) => setSelectedCandidate(e.target.value)}
    >
      <option value="">-- Choose Candidate --</option>
      {candidates.map((candidate) => (
        <option key={candidate._id} value={candidate._id}>
          {candidate.fullName} ({candidate.party})
        </option>
      ))}
    </select>
  </div>
  <button type="submit" className="btn btn-primary w-100">
    Submit Vote
  </button>
</form>
</div>
);
};

export default Vote;
```

8)TRACK APPLICATION:

**Election Commission**
Democratic Elections Portal

RegisterLoginHomeVoter ServicesCandidatesAboutContactVoteElections & ResultsLogout

Track Application Status


Application ID

Voter ID

Mobile Number

Search

No applications found.

**Election Commission**
Democratic Elections Portal

Your trusted source for democratic election information and services in India.

Quick Links
About
Voter Services
Candidates
Contact

Contact Us
✉ info@electioncommission.in
f t @

✓ CODE:

```
import React, { useState } from 'react';
import axios from 'axios';

export default function TrackApplication() {
  const [query, setQuery] = useState({
    applicationId: "",
    voterId: "",
    mobile: "",
  });
  const [result, setResult] = useState(null);
  const [loading, setLoading] = useState(false);
  const [error, setError] = useState("");

  const handleChange = (e) => {
    setQuery({ ...query, [e.target.name]: e.target.value });
  };

  const handleSubmit = async (e) => {
    e.preventDefault();
    setError("");
    setResult(null);

    if (!query.applicationId && !query.voterId && !query.mobile) {
```

```

setError('Please enter Application ID or Voter ID or Mobile number');
return;
}

setLoading(true);
try {
  // Use full backend URL or setup proxy in React to avoid hardcoding
  const response = await axios.get('http://localhost:5000/api/applications/search', {
    params: {
      applicationId: query.applicationId || undefined,
      voterId: query.voterId || undefined,
      mobile: query.mobile || undefined,
    },
  });

  if (Array.isArray(response.data) && response.data.length > 0) {
    setResult(response.data);
  } else {
    setResult([]);
    setError('No application found matching the given details.');
```

```

    />
  </div>
  <div className="mb-3">
    <label htmlFor="voterId" className="form-label">Voter ID</label>
    <input
      type="text"
      id="voterId"
      name="voterId"
      value={query.voterId}
      onChange={handleChange}
      className="form-control"
      placeholder="Enter Voter ID"
    />
  </div>
  <div className="mb-3">
    <label htmlFor="mobile" className="form-label">Mobile Number</label>
    <input
      type="tel"
      id="mobile"
      name="mobile"
      value={query.mobile}
      onChange={handleChange}
      className="form-control"
      placeholder="Enter Mobile Number"
    />
  </div>
  <button type="submit" className="btn btn-primary" disabled={loading}>
    {loading ? 'Searching...' : 'Search'}
  </button>
</form>

{error && <div className="alert alert-danger">{error}</div>}

{result && result.length > 0 ? (
  <div className="card">
    <div className="card-header">Application Result</div>
    <div className="card-body">
      {result.map((app) => (
        <div key={app._id} className="mb-3">
          <p><strong>Application ID:</strong> {app.applicationId || 'N/A'}</p>
          <p><strong>Voter ID:</strong> {app.voterId || 'N/A'}</p>
          <p><strong>Mobile:</strong> {app.mobile || 'N/A'}</p>
          <p><strong>Status:</strong> {app.status || 'Pending'}</p>
          <hr />
        </div>
      ))}
    </div>
  </div>
)}

```

```
        </div>
    )}
</div>
</div>
): (
    !loading && !error && <p>No applications found.</p>
)}
</div>
);
}
```

9)VOTER SEARCH:

Election Commission
Democratic Elections Portal

Register Login Home Voter Services Candidates About Contact Vote Elections & Results Logout

Voter Search

Full Name: Sudipta Dey Voter ID: VOTE20089 Mobile Number: 8170842477

Search

Search Results

FULL NAME	VOTER ID	AADHAR NO	MOBILE	DATE OF BIRTH
Sudipta Dey	VOTE20089	2025 8009 3570	8170842477	4/2/2005

✓ CODE:

```
import React, { useState } from "react";
import axios from "axios";
import "bootstrap/dist/css/bootstrap.min.css";

const VoterSearch = () => {
  const [searchParams, setSearchParams] = useState({
    name: "",
    voterId: "",
    mobile: "",
  });
  const [results, setResults] = useState([]);
  const [loading, setLoading] = useState(false);

  const handleChange = (e) => {
    setSearchParams({ ...searchParams, [e.target.name]: e.target.value });
  };

  const handleSearch = async (e) => {
    e.preventDefault();
    setLoading(true);
    try {
      const response = await axios.get("http://localhost:5000/api/voter/search", {
        params: searchParams,
      });
      setResults(response.data);
    } catch (error) {
      console.error(error);
      alert("Error fetching search results");
    }
  }
}
```



```

    setLoading(false);
};

return (
  <div className="container py-4">
    <div className="card shadow-lg p-4">
      <h3 className="text-center text-primary mb-4">🔍 Voter Search</h3>
      <form onSubmit={handleSearch} className="row g-3">
        <div className="col-md-4">
          <label className="form-label">Full Name</label>
          <input
            type="text"
            className="form-control"
            name="name"
            value={searchParams.name}
            onChange={handleChange}
            placeholder="Enter full name"
          />
        </div>
        <div className="col-md-4">
          <label className="form-label">Voter ID</label>
          <input
            type="text"
            className="form-control"
            name="voterId"
            value={searchParams.voterId}
            onChange={handleChange}
            placeholder="Enter voter ID"
          />
        </div>
        <div className="col-md-4">
          <label className="form-label">Mobile Number</label>
          <input
            type="text"
            className="form-control"
            name="mobile"
            value={searchParams.mobile}
            onChange={handleChange}
            placeholder="Enter mobile number"
          />
        </div>
        <div className="col-12 text-center mt-3">
          <button
            type="submit"
            className="btn btn-primary px-4"

```

```

disabled={loading}>
  {loading ? "Searching..." : "Search"}
</button>
</div>
</form>
</div>

{results.length > 0 && (
  <div className="card mt-4 shadow-lg p-4">
    <h4 className="mb-3 text-success">Search Results</h4>
    <div className="table-responsive">
      <table className="table table-bordered table-striped">
        <thead className="table-dark">
          <tr>
            <th>Full Name</th>
            <th>Voter ID</th>
            <th>Aadhar No</th>
            <th>Mobile</th>
            <th>Date of Birth</th>
          </tr>
        </thead>
        <tbody>
          {results.map((voter) => (
            <tr key={voter._id}>
              <td>{voter.fullName}</td>
              <td>{voter.voterId}</td>
              <td>{voter.aadharNo}</td>
              <td>{voter.mobile}</td>
              <td>{new Date(voter.dob).toLocaleDateString()}</td>
            </tr>
          ))}
        </tbody>
      </table>
    </div>
  </div>
)}
{results.length === 0 && !loading && (
  <p className="text-center text-muted mt-3">No results found</p>
)}
</div>
);
};

export default VoterSearch;

```

10)APP.JSX

```
import { BrowserRouter as Router, Routes, Route, Navigate, Router } from
"react-router-dom"
import React from 'react'
import Header from './Components/Header';
import 'bootstrap/dist/css/bootstrap.min.css';
import Home from "./Pages/Home";
import About from "./Pages/About";
import Contact from "./Pages/Contact";

import VoterServices from "./Pages/VoterServices";
import Candidates from "./Pages/Candidates";
import GeneralResults from "./Pages/GeneralResults";
import StateResults from "./Pages/StateResults";

import ForgotPassword from "./Pages/ForgotPassword";
import Footer from "./Components/Footer";

import PrivateRoute from "./utils/PrivateRoute";
import Register from "./Pages/Register.jsx";
import Login from "./Pages/Login.jsx";
import VoterRegistrationForm from "./Pages/VoterRegistrationForm.jsx";
import VoterSearch from "./Pages/VoterSearch.jsx";

import Vote from "./Pages/Vote.jsx";
import TrackApplication from "./Pages/TrackApplication.jsx";
import VoterProfile from "./Pages/VoterProfile.jsx";

const App = () => {
  return (
    <BrowserRouter>
      <Header />
      <Routes>

        <Route path="/" element={<Register />}></Route>
        <Route path="/login" element={<Login/>}></Route>
        <Route element={<PrivateRoute />}>
          <Route path="/home" element={<Home />} />
          <Route path="/voter-services" element={<VoterServices />} />
        </Route>
      </Routes>
    </BrowserRouter>
  )
}
```

```
<Route path="/candidates" element={<Candidates />} />
<Route path="/about" element={<About />} />
<Route path="/contact" element={<Contact />} />
<Route path="/forgot-password" element={<ForgotPassword />} />
<Route path="/results/general" element={<GeneralResults />} />
<Route path="/results/state" element={<StateResults />} />
<Route path="/voter-registration" element={<VoterRegistrationForm />} />
<Route path="/search" element={<VoterSearch />} />
<Route path="/vote" element={<Vote />} />
<Route path="/track-application" element={<TrackApplication />} />
<Route path="/voter-profile" element={<VoterProfile />} />
```

```
</Route>
</Routes>
```

```
<Footer />
```

```
</BrowserRouter>
);
}
```

```
export default App;
```

❖ BACKEND

❖ Db.js:

```
const mongoose = require('mongoose');

const connectdb = async () => {
  try {
    if (!process.env.MONGO_URI) {
      throw new Error('MONGO_URI is not defined in .env file');
    }
    await mongoose.connect(process.env.MONGO_URI);
    console.log('Connected to MongoDB');
  } catch (error) {
    console.error('MongoDB connection error:', error.message);
    process.exit(1); // Exit process on connection failure
  }
};

module.exports = connectdb;
```

❖ server.js:

```
require('dotenv').config();
const express = require('express');
const cors = require('cors');
const jwt = require('jsonwebtoken');
const path = require('path');
const connectdb = require('./config/db');

// ===== Connect to MongoDB =====
connectdb();

// ===== Initialize Express =====
const app = express();

// ===== Middleware =====
app.use(cors({ origin: 'http://localhost:5173' })); // Allow frontend URL
app.use(express.json());
app.use(express.urlencoded({ extended: true }));

// Serve uploaded images
app.use("/uploads", express.static(path.join(__dirname, "uploads")));

// ===== JWT Authentication Middleware =====
const authenticateToken = (req, res, next) => {
  const token = req.headers['authorization']?.split(' ')[1];
  if (!token) return res.sendStatus(401);

  jwt.verify(token, process.env.JWT_SECRET || 'fyugjjjj7y76868', (err, user) => {
    if (err) return res.sendStatus(403);
    req.user = user;
    next();
  });
};

// ===== Import Routes =====
const authRoute = require('./routes/AuthRoute');
const areaRoute = require('./routes/AreaRoute');
const candidateRoutes = require('./routes/CandidateRoute');

const electionRoute = require('./routes/ElectionRoutes');
const feedbackRoute = require('./routes/FeedbackRoute');
const reportRoute = require('./routes/ReportRoutes');
```

```
const voteRoute = require('./routes/VoteRoute');
const voterRoute = require('./routes/VoterRoute'); // ☒ Import normally, no (upload)

// ===== Example Protected Test Route =====
app.post('/api/voter/test', authenticateToken, (req, res) => {
  res.send('Voter data received and authenticated.');
```

});

```
// ===== Use Routes =====
app.use('/api/auth', authRoute);
app.use('/api/area', areaRoute);
app.use('/api/candidates', candidateRoutes);
app.use('/api/election', electionRoute);
app.use('/api/feedback', feedbackRoute);
app.use('/api/report', reportRoute);
app.use('/api/vote', voteRoute);
app.use('/api/voter', voterRoute);

// ===== Start Server =====
const PORT = process.env.PORT || 5000;
app.listen(PORT, () => {
  console.log(☒ Server running on http://localhost:${PORT});
});
```

❖ auth.controller:

```
const User = require('../models/User');
const jwt = require('jsonwebtoken');
const bcrypt = require('bcryptjs');

//register
exports.register = async(req,res) => {
  const {name,email,password} = req.body;
  try {
    const hashedpassword = await bcrypt.hash(password,15);
    const user = await User.create({name,email,password:hashedpassword});
    res.status(201).json({ message:'user register successfully'});
  } catch(err){
    res.status(400).json({ message:'user already exist'});
  }
};

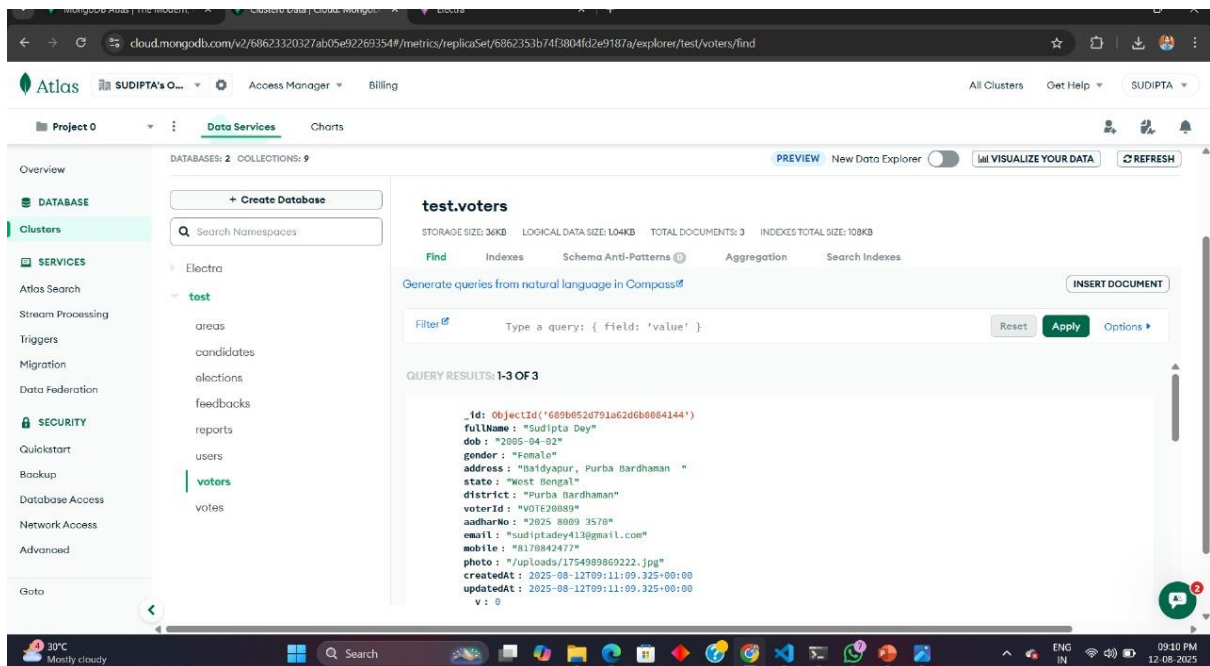
//login
exports.login = async(req,res) => {
  const {email,password} = req.body;
  try {
    const user = await User.findOne({email});
    if(!user) return res.status(400).json({ message:'invalid credentials'});

    const match = await bcrypt.compare(password,user.password);
    if(!match) return res.status(400).json({ message:'invalid credentials'});

    const token =
    jwt.sign({userId:user._id},process.env.JWT_SECRET,{expiresIn:'30d'});
    res.status(201).json({ token});
  } catch(err){
    res.status(401).json({ message:'Login failed'});
  }
};

//dashboard
exports.dashboard = (req,res)=>{
  res.json({ message:'Welcome to dashboard'});
};
```


❖ VOTER:



```
const Vote = require('../models/Vote');
```

```
// Get all votes (Admin view with voter & candidate details)
```

```
exports.getVotes = async (req, res) => {
```

```
  try {
```

```
    const votes = await Vote.find()
```

```
      .populate('voterId', 'name email') // Show name & email of voter
```

```
      .populate('candidateId', 'name party') // Show name & party of candidate
```

```
      .populate('electionId', 'title date'); // Show election title & date
```

```
    res.status(200).json({
```

```
      success: true,
```

```
      count: votes.length,
```

```
      votes
```

```
    });
```

```
  } catch (error) {
```

```
    res.status(500).json({ message: 'Error fetching votes', error: error.message });
```

```
  }
```

```
};
```

```
// Cast a new vote
```

```
exports.castVote = async (req, res) => {
```

```
  try {
```

```
    const { voterId, candidateId, electionId } = req.body;
```

```
    // Validate required fields
```

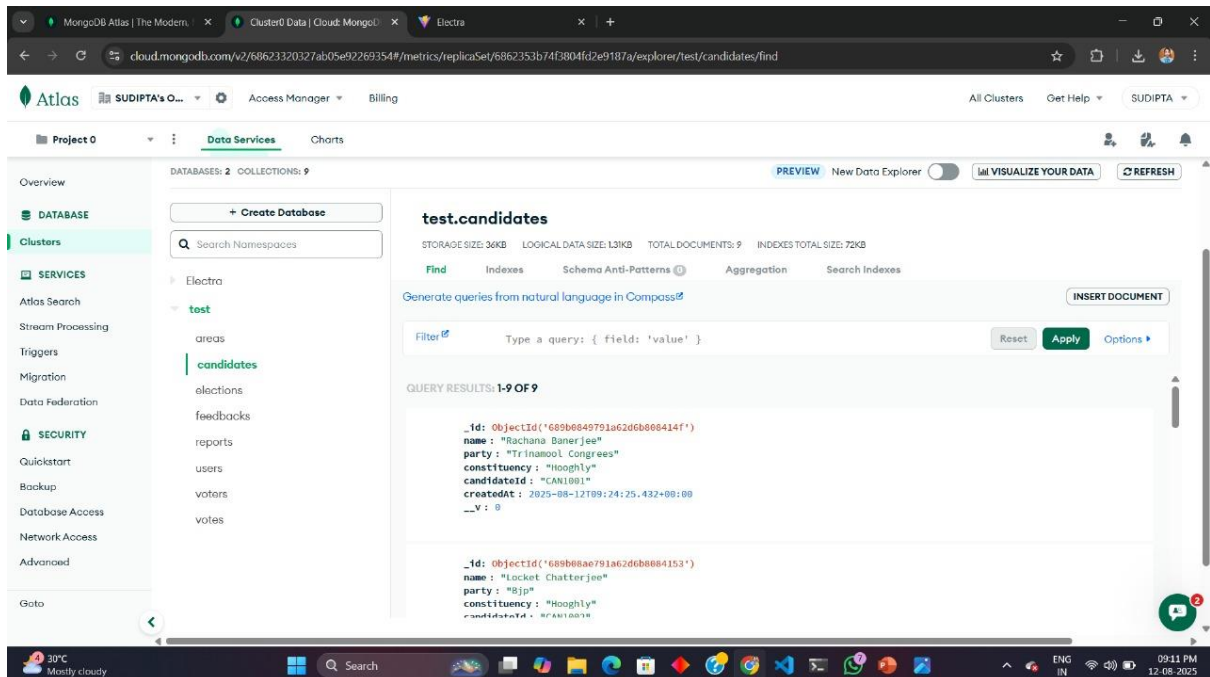
```
if (!voterId || !candidateId || !electionId) {
  return res.status(400).json({ message: 'All fields are required' });
}

// Check if the voter already voted in this election
const existingVote = await Vote.findOne({ voterId, electionId });
if (existingVote) {
  return res.status(400).json({ message: 'You have already voted in this election' });
}

// Save new vote
const newVote = new Vote({ voterId, candidateId, electionId });
const savedVote = await newVote.save();

res.status(201).json({
  success: true,
  message: 'Vote cast successfully',
  vote: savedVote
});
} catch (error) {
  res.status(400).json({ message: 'Error casting vote', error: error.message });
}
};
```

❖ CANDIDATE:



```
const Candidate = require('../models/Candidate');
```

```
// Get all candidates
```

```
exports.getCandidates = async (req, res) => {  
  try {  
    const candidates = await Candidate.find();  
    res.status(200).json(candidates);  
  } catch (error) {  
    res.status(500).json({ message: 'Error fetching candidates', error: error.message });  
  }  
};
```

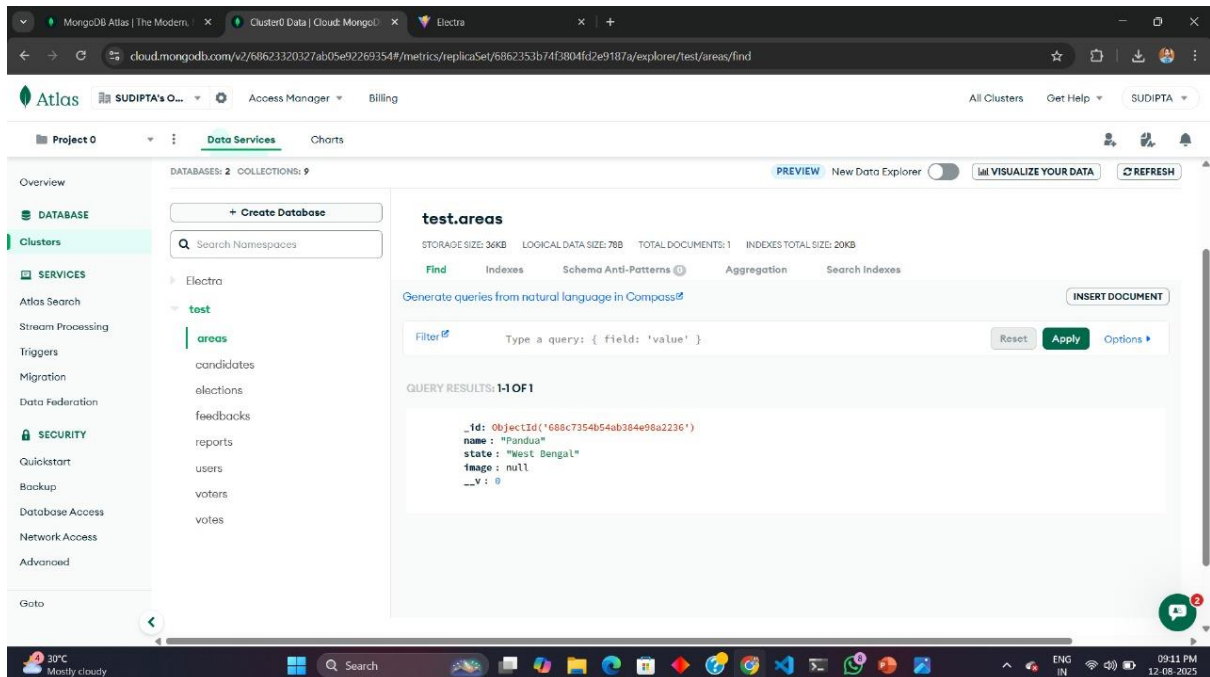
```
// Create a new candidate
```

```
exports.createCandidate = async (req, res) => {  
  try {  
    console.log('Received candidate data:', req.body);  
    const { name, party, constituency, candidateId } = req.body;  
  
    if (!name || !party || !constituency || !candidateId) {  
      return res.status(400).json({ message: 'All fields are required' });  
    }  
  }  
};
```

```
// Optional: check duplicate candidateId
const existing = await Candidate.findOne({ candidateId });
if (existing) {
  return res.status(400).json({ message: 'Candidate ID already exists' });
}

const newCandidate = new Candidate({ name, party, constituency, candidateId });
const savedCandidate = await newCandidate.save();
res.status(201).json(savedCandidate);
} catch (error) {
  console.error('Error creating candidate:', error);
  res.status(400).json({ message: 'Error creating candidate', error: error.message });
}
};
```

❖ AREA:



```
const Area = require('../models/Area');
const fs = require('fs');
const path = require('path');

//create area
exports.createArea = async (req, res) => {
  const { name, state } = req.body;
  const image = req.file ? req.file.filename : null;
  const area = new Area({ name, state, image });
  await area.save();
  res.json(area);
};

//view
exports.getAreas = async (req, res) => {
  const areas = await Area.find();
  res.json(areas);
}

//updated
exports.updatedArea = async (req, res) => {
  const { name, state } = req.body;
  const area = await Area.findById(req.params.id);
  if (!area) return res.status(404).json({ message: 'Area not found' });

  //delete old image
```

```
if (req.file && area.image) {
  const filepath = path.join(__dirname, '../uploads', area.image);
  if (fs.existsSync(filepath)) fs.unlinkSync(filepath)
}
area.name = name || area.name;
area.state = name || area.city;
if (req.file) area.image = req.file.filename;
const updated = await area.save();
res.json(updated);
}
//delete
exports.deleteArea = async (req, res) => {
  const area = await Area.findById(req.params.id);
  if (!area) return res.status(404).json({ message: 'Area not found' });

  //delete old image
  if (area.image) {
    const filepath = path.join(__dirname, '../uploads', area.image);
    if (fs.existsSync(filepath)) fs.unlinkSync(filepath)
  }
  await area.remove();
  res.json({ message: 'Area deleted' });
}
```

❖ **CONCLUSION**

The Election Commission Project using MERN stack is a comprehensive and innovative solution for managing electoral processes. By leveraging the power of MongoDB, Express.js, React.js, and Node.js, the system provides a secure, efficient, and scalable platform for voter registration, online voting, and election management.

The project's key features, including voter registration, E-Voter Card generation, and admin dashboard, ensure a streamlined and transparent electoral process. The system's security measures, such as authentication and authorization, protect the integrity of the voting process and maintain voter confidentiality.

Overall, the Election Commission Project using MERN stack has the potential to revolutionize the way elections are conducted, making the process more accessible, efficient, and reliable. With its robust architecture and user-friendly interface, the system can be a valuable asset for electoral commissions and governments worldwide.

❖ FUTURE SCOPE & FURTHER ENHANCEMENTS

❖ Future Scope :

The Election Commission Project using MERN stack has a promising future scope, with potential enhancements and expansions:

1. Integration with Emerging Technologies

- **Blockchain:** Integrate blockchain technology to ensure transparency, security, and immutability of votes.
- **Artificial Intelligence (AI):** Leverage AI for voter sentiment analysis, election forecasting, and anomaly detection.

2. Enhanced Security Measures

- **Biometric Authentication:** Implement biometric authentication (e.g., facial recognition, fingerprint scanning) for enhanced voter verification.
- **Advanced Encryption:** Employ advanced encryption techniques to protect voter data and ensure confidentiality.

3. Mobile Application Development

- **Mobile Voting App:** Develop a mobile app for voters to cast votes using their smartphones, increasing accessibility and convenience.

4. Data Analytics and Insights

- **Real-time Analytics:** Provide real-time analytics and insights on voter turnout, election trends, and results.
- **Data Visualization:** Utilize data visualization tools to present complex data in an intuitive and actionable manner.

5. Integration with Other Government Systems

- **Integration with Voter ID Systems:** Integrate with existing voter ID systems to streamline voter registration and verification.
- **Integration with Government Databases:** Integrate with government databases to ensure data accuracy and reduce duplication of efforts.

❖ FURTHER ENHANCEMENTS

Further Enhancements for Election Commission Project Using MERN Stack

To further improve the Election Commission Project using MERN stack, consider the following enhancements:

1. Biometric Authentication

- **Secure Voter Verification:** Implement biometric authentication (e.g., facial recognition, fingerprint scanning) to ensure secure voter verification and prevent identity theft.

2. Mobile OTP Verification

- **Enhanced Security:** Provide mobile OTP verification to add an extra layer of security for voters and prevent unauthorized access.

3. Offline Voting Mode

- **Increased Accessibility:** Develop an offline voting mode that allows voters to cast their votes without an internet connection, syncing data when connectivity is restored.

4. AI-Powered Fraud Detection

- **Real-time Anomaly Detection:** Utilize AI-powered fraud detection to identify and prevent electoral malpractices in real-time, ensuring the integrity of the voting process.

5. Multi-Language Support

- **Inclusive Platform:** Provide support for multiple languages to cater to diverse voter populations, ensuring that language barriers do not hinder the voting process.

6. Enhanced Admin Analytics

- **Data-Driven Insights:** Offer enhanced analytics and insights for administrators to track voter turnout, election trends, and results, enabling data-driven decision-making.

7. Accessibility Features

- **Inclusive Design:** Implement accessibility features to ensure that the platform is usable by voters with disabilities, promoting equal participation in the electoral process.

By incorporating these enhancements, the Election Commission Project using MERN stack can become more secure, accessible, and efficient, ultimately strengthening the democratic process.

