

# **Voting Application**

**A PROJECT REPORT**

**for**

**Mini Project (KCA353)**

**Session (2024-25)**

**Submitted by**

**Rishav Kishore**

**University Roll No 2300290140140**

**Satyam Singh**

**University Roll No 2300290140163**

**Submitted in partial fulfilment of the  
Requirements for the Degree of**

## **MASTER OF COMPUTER APPLICATION**

**Under the Supervision of**

**Dr Vipin Kumar**

**(Associate Professor)**



**Submitted to**

**Department Of Computer Applications**

**KIET Group of Institutions, Ghaziabad**

**Uttar Pradesh-201206**

**FEBRUARY 2024**

# CERTIFICATE

Certified that **Rishav Kishore (2300290140140)**, **Satyam Singh (2300290140163)** have carried out the project work having “**Voting Application**” (**Mini Project KCA353**) for **Master of Computer Application** from Dr. A.P.J. Abdul Kalam Technical University (AKTU) (formerly UPTU), Lucknow under my supervision. The project report embodies original work, and studies are carried out by the student himself and the contents of the project report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

**Date:**

**Rishav Kishore (2300290140140)**

**Satyam Singh (2300290140163)**

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Date:

**Dr Vipin Kumar**

**Associate Professor**

**Department of Computer Applications**

**KIET Group of Institutions, Ghaziabad**

**Dr. Arun Tripathi**

**Head**

**Department of Computer Applications**

**KIET Group of Institutions, Ghaziabad**

# ABSTRACT

The Voting Application is a MERN stack-based project aimed at simplifying the decision-making process for selecting a course in a classroom setting. The application is designed to accommodate the increasing number of students and ensures that each student gets a fair opportunity to vote. Each student can cast their vote only once for their preferred course, ensuring transparency and fairness in the process.

Once the voting is completed, the course that receives the maximum number of votes is finalized and launched for the class. This system automates and streamlines the selection process, removing the need for manual decision-making and improving participation and engagement among students.

This project incorporates features like user authentication, real-time vote updates, and secure data handling, providing a seamless and reliable voting experience for students. The application serves as an efficient tool to make democratic decisions in an academic environment.

# ACKNOWLEDGEMENTS

Success in life is never attained single-handedly. My deepest gratitude goes to my project supervisor, **Dr Vipin Kumar** for her guidance, help, and encouragement throughout my project work. Their enlightening ideas, comments, and suggestions.

Words are not enough to express my gratitude to **Dr. Arun Kumar Tripathi**, Professor and Head, Department of Computer Applications, for his insightful comments and administrative help on various occasions.

Fortunately, I have many understanding friends, who have helped me a lot on many critical conditions.

Finally, my sincere thanks go to my family members and all those who have directly and indirectly provided me with moral support and other kind of help. Without their support, completion of this work would not have been possible in time. They keep my life filled with enjoyment and happiness.

**Rishav Kishore**

**Satyam Singh**

# TABLE OF CONTENTS

Certificate	i
Abstract	ii
Acknowledgements	iii
Table of Contents	iv-v
List of Tables	vi
List of Figures	vii
1 Introduction	1-2
1.1 Background	1
1.2 Project overview	1
1.3 Objective	1
1.4 Key features	2
1.5 Scope of the project	2
2 Problem identification & feasibility study	3-4
2.1 Problem Identification	3
2.2 Feasibility Study	3
2.2.1 Technical Feasibility	3
2.2.2 Operational Feasibility	3
2.2.3 Economic Feasibility	4
3 Requirement Analysis	5-6
3.1 Introduction	5
3.2 Functional Requirements	5
3.2.1 User Module	5

3.2.2	Administrator Module	5
3.2.3	Package Module	5
3.3	Non-Functional Requirements	5
3.3.1	Performance	5
3.3.2	Security	6
3.3.3	Scalability	6
3.3.4	Usability	6
4	Hardware and Software Specification	7-9
4.1	Hardware Specification	8
4.2	Software Specification	9
5	Choice of Tools & Technology	10-15
5.1	React	12
5.2	MongoDB	13
5.3	Data Flow Diagram	14
5.4	Context Level Diagram	12-15

6	DataBase	16-18
	6.1 User	17
	6.2 Vote	18
7	Form Design	19-25
	7.1 Login	19
	7.2 Signup	20
	7.3 User	21
	7.3.1 Home Page	22
	7.3.2 Course Page	23
	7.3.3 Vote Page	24
	7.3.4 Result Page	25
8	Coding	23-36
9	Testing	37-40
	9.1 Introduction	
	9.2 Types of Testing	
	Future Scope and Further Enhancement of the Project	
	Conclusion & References	

# CHAPTER 1

## INTRODUCTION

### 1.1 Background

In classrooms with a growing number of students, selecting a course to be taught often becomes a challenging and time-consuming process. Traditional methods, such as verbal discussions or manual polling, can be inefficient, biased, or prone to errors, leading to dissatisfaction among students. To address these challenges, this project introduces a digital solution that leverages modern technology to automate and democratize the decision-making process.

### 1.2 Project Overview

The Voting Application is a dynamic and interactive web-based platform built with the MERN stack (MongoDB, Express.js, React.js, Node.js) that facilitates a democratic process for course selection in a classroom setting. The application empowers students to cast a single vote for their preferred course, ensuring fairness and preventing duplicate votes. The course with the highest votes is selected for implementation in the class.



### **1.3 Objective**

The primary objective of the Voting Application is to create a fair and transparent system for course selection in a classroom setting. By allowing each student to cast one vote for their preferred course, the application ensures equal participation and eliminates bias or manual intervention in the decision-making process.

### **1.4 Key Features**

Voting Application include a number of features

The user interface is simple and intuitive, making it easy for students to participate in the voting process. Additionally, the application is scalable, capable of handling a growing number of students without performance issues. MongoDB is used for storing votes securely, ensuring data integrity and preventing any tampering. Lastly, the application promotes transparency by displaying unbiased and accurate results, building trust among users. Together, these features make the Voting Application an efficient and reliable tool for course selection in academic environments.

The user interface is simple and intuitive, making it easy for students to participate in the voting process. Additionally, the application is scalable, capable of handling a growing number of students without performance issues. MongoDB is used for storing votes

securely, ensuring data integrity and preventing any tampering.

### **1.5 Scope of the project**

The scope of the Voting Application is to provide a digital platform for course selection that simplifies and automates the voting process in educational environments. This application is designed to serve both small and large classrooms, enabling students to easily cast their votes and ensuring a fair and democratic process. The application handles everything from vote collection to course finalization, providing an efficient and user-friendly solution for academic institutions.

The scope of this project can be expanded to support various use cases beyond course selection, such as voting for events, elections, or any decision-making process that requires student participation. The platform can be scaled to accommodate more students and additional features like email notifications, detailed analytics, and support for multiple categories of votes.

## **CHAPTER 2**

# **PROBLEM IDENTIFICATION & FEASIBILITY STUDY**

### **2.1 Problem Identification**

Identifying potential problems in a voting application website is crucial for ensuring a smooth user experience. Common issues include poor user interface and experience, limited search and filtering options, incomplete or inaccurate information, booking and payment issues, ineffective customer support, poor mobile responsiveness, limited personalization, and security concerns. Addressing these issues through regular user testing, feedback gathering, and continuous improvement efforts can enhance the overall user satisfaction and usability of the website.

### **2.2 Feasibility Study**

Before initiating the development of the traveling website, a thorough feasibility study

was conducted to evaluate the feasibility and practicality of the proposed platform. This study examined technical, operational, and economic aspects to ensure the viability of the project.

### **2.2.1 Technical Feasibility**

Assess the technical requirements and capabilities needed to develop and maintain the website. Consider factors such as web hosting, scalability, security measures, and integration with third-party services like booking systems and payment gateways.

### **2.2.2 Operational Feasibility**

Assess the practicality of managing and operating the website on a day-to-day basis. Consider factors such as staffing requirements, content management processes, customer support mechanisms, and strategic partnerships with travel suppliers and service providers.

### **2.2.3 Economic Feasibility**

The economic study of a voting application website involves assessing its financial viability and sustainability through various analyses. This includes evaluating costs associated with development, maintenance, and operations, as well as identifying potential revenue streams such

as booking commissions, advertising, and subscription fees. Market research is conducted to understand demand, competition, and growth opportunities, while financial projections and ROI calculations provide insights into the website's potential performance. Risk assessment and sensitivity analysis help identify and mitigate potential challenges, ensuring the website's resilience in fluctuating market conditions. Overall, the economic study informs stakeholders about the feasibility and profitability of the traveling website, guiding strategic decisions for its successful implementation and operation.

# CHAPTER 3

## Requirement Analysis

### 3.1 Introduction

The success of any application depends on a comprehensive understanding of its requirements. This section outlines the functional and non-functional requirements of the "Voting Application for College Purpose." The application is designed to simplify and democratize course selection by providing a transparent and efficient voting platform for students and administrators.

### 3.2 Functional Requirements

The functional requirements define the core functionalities and features of the application.

#### 3.2.1 User Module

**User Registration and Login:** Allows students to register, log in, and access their profiles securely.

**Vote Submission:** Enables students to vote for their preferred subjects.

**View Results:** Provides users with access to the voting results in real time or after a specified voting period.

**Profile Management:** Users can update personal details and manage their accounts.

#### 3.2.2 Administrator Module

**Admin Login:** Secure login functionality for administrators.

**Manage Subjects:** Allows administrators to add, update, or delete subjects available for voting.

**Monitor Votes:** Provides tools to track and analyze voting activity.

**Generate Results:** Enables the generation of voting results, ensuring accuracy and transparency.

### **3.2.3 Package Module**

**Voting Configuration:** Allows customization of voting parameters such as start and end times.

**Export Data:** Administrators can export voting data for analysis or record-keeping purposes.

## **3.3 Non-Functional Requirements**

The non-functional requirements ensure the application's overall quality, performance, and user experience.

### **3.3.1 Performance**

The application should handle concurrent user interactions efficiently, even during peak usage.

Voting results should be processed and displayed with minimal delay.

### **3.3.2 Security**

User data, including login credentials and voting choices, must be securely stored and transmitted using encryption.

The application should include role-based access control to protect sensitive administrator functions.

### **3.3.3 Scalability**

The system should support an increasing number of users and voting events without significant performance degradation.

The design should allow easy integration of additional features in the future.

### **3.3.4 Usability**

The application must have a user-friendly interface that is easy to navigate for students and administrators.

The system should provide clear instructions and error messages to guide users effectively.



# CHAPTER 4

## Hardware and Software Specifications

### 4.1 Hardware Specification

The following hardware components are required to develop and run the "Voting Application for College Purpose":

#### Development Environment:

**Processor:** Intel Core i5 or higher / AMD Ryzen 5 or higher

**RAM:** 8 GB (16 GB recommended for smoother performance during development)

**Storage:** 256 GB SSD or higher

**Display:** Full HD monitor (1920x1080 resolution)

**Peripherals:** Keyboard, mouse, and optional external webcam for online collaboration

#### Deployment Environment (Server):

**Processor:** Quad-core server-grade processor (e.g., Intel Xeon or AMD EPYC)

**RAM:** 16 GB or higher

**Storage:** 512 GB SSD or higher (to store application data and logs)

**Network:** High-speed internet with at least 100 Mbps bandwidth

## 4.2 Software Specification

The following software tools and platforms are required for developing, testing, and deploying the application:

### Development Tools:

**Operating System:** Windows 10/11, macOS, or Linux

**Frontend Development:** React.js, Node.js (runtime environment), and a code editor like Visual Studio Code

**Styling:** CSS3, Tailwind CSS, or other UI libraries/frameworks

**Backend Development:** Node.js with Express.js framework

**Database:** MongoDB (NoSQL) or MySQL (relational database)

**Version Control:** Git with GitHub or GitLab

### Testing Tools:

**Unit Testing:** Jest, Mocha

**Integration Testing:** Postman (for API testing), Selenium (for UI automation)

## **Deployment Tools:**

**Hosting Platform:** AWS, Google Cloud, or Azure (optional: Firebase for simpler deployments)

**Web Server:** Nginx or Apache

**Domain:** Custom domain for accessing the application

# CHAPTER 5

## Choice of Tools & Technology

### 5.1 React

React is a JavaScript library used for building user interfaces. It is ideal for creating dynamic and interactive web applications like the "Voting Application for College Purpose."

#### Key Features:

**Component-Based Architecture:** Allows reusability of components, reducing development effort.

**Virtual DOM:** Enhances performance by efficiently updating only the necessary parts of the UI.

**State Management:** Provides tools like React Hooks and Context API for managing application state.

**Cross-Platform:** Ensures the application runs seamlessly across devices and browsers.

### 5.2 MongoDB

MongoDB is a NoSQL database that provides flexibility and scalability for storing application data.

#### Key Features:

**Document-Oriented Storage:** Data is stored in JSON-like BSON format, making it suitable for dynamic data structures.

**Scalability:** Easily handles large volumes of data with horizontal scaling.

**Query Language:** Simple and powerful query capabilities for efficient data retrieval.

**High Performance:** Optimized for read and write operations.

### 5.3 Data Flow Diagram

A Data Flow Diagram (DFD) represents the flow of information within the system. The DFD for the "Voting Application for College Purpose" includes:

#### Level 0:

**Entities:** Students, Administrators

**Processes:** User Registration, Voting, Result Generation

**Data Stores:** User Database, Voting Data

#### Level 1 (Detailed Processes):

##### User Registration Process:

Input: User details

Output: Registered user profile stored in the database

##### Voting Process:

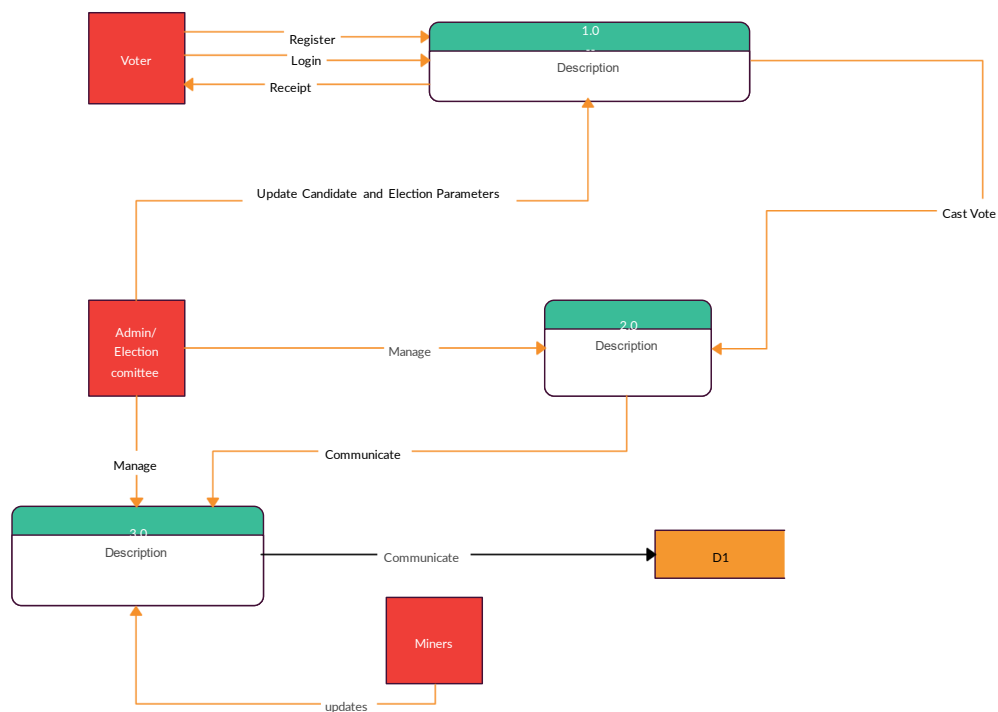
Input: User's subject preference

Output: Vote recorded in the database

## Result Generation:

Input: Voting data

Output: Final results displayed to users



## 5.4 Context Level Diagram

The Context Level Diagram represents the system's interaction with external entities.

## Components:

### Students:

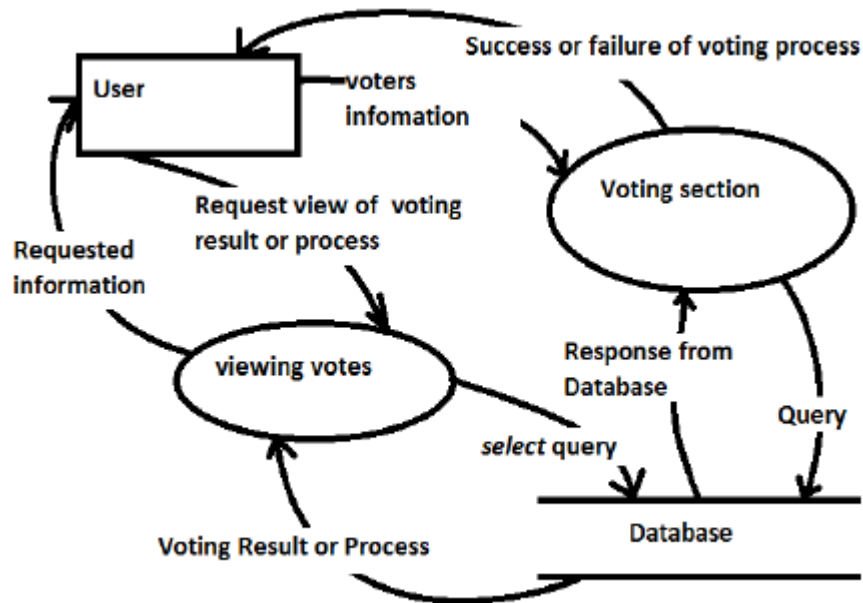
Can register, vote, and view results.

#### 1. Administrators:

Can manage subjects, monitor votes, and generate results.

#### 2. System:

Manages all data flow between entities and databases.



# CHAPTER 6

## DataBase

### 6.1 User

The **User** collection/table stores information about students and administrators who interact with the application.

#### **Attributes:**

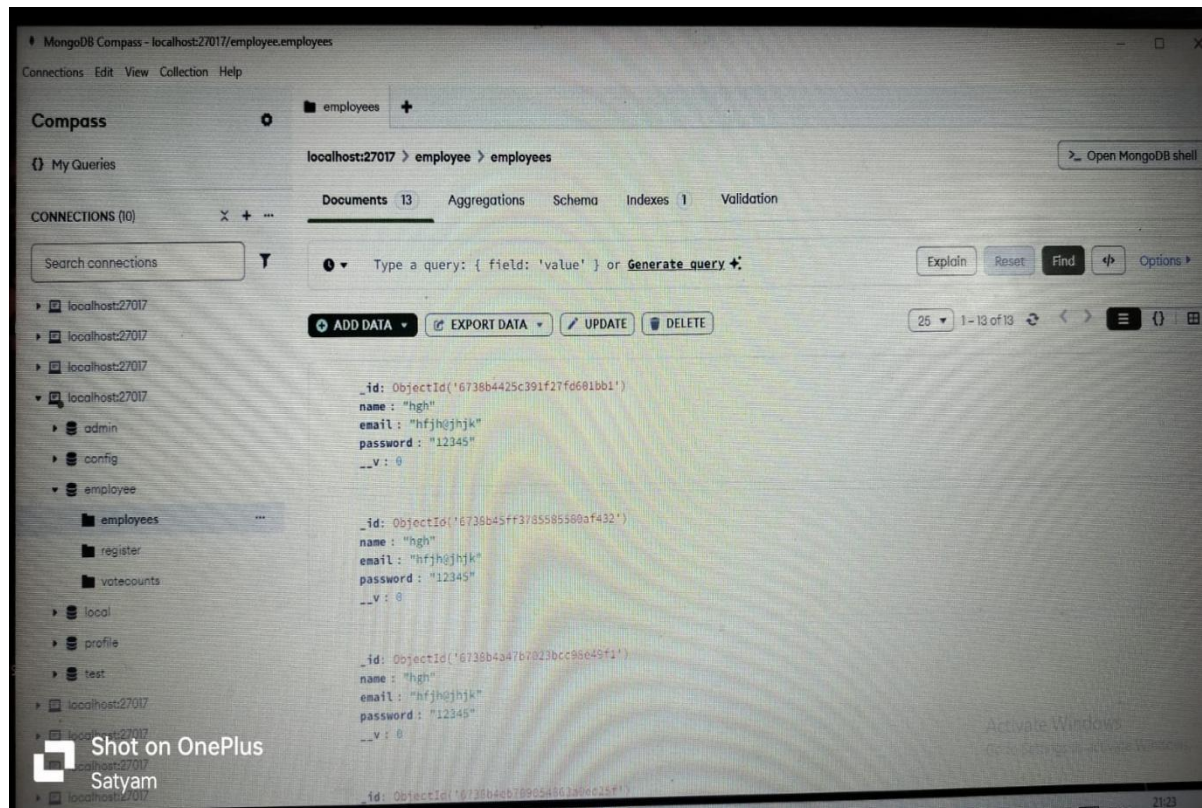
**user\_id** : A unique identifier for each user.

**name**: The full name of the user.

**email**: The email address of the user (used for login and communication).

**password**: The hashed password for secure authentication.





## 6.2 Vote

The **Vote** collection/table stores data related to the voting process.

### Attributes:

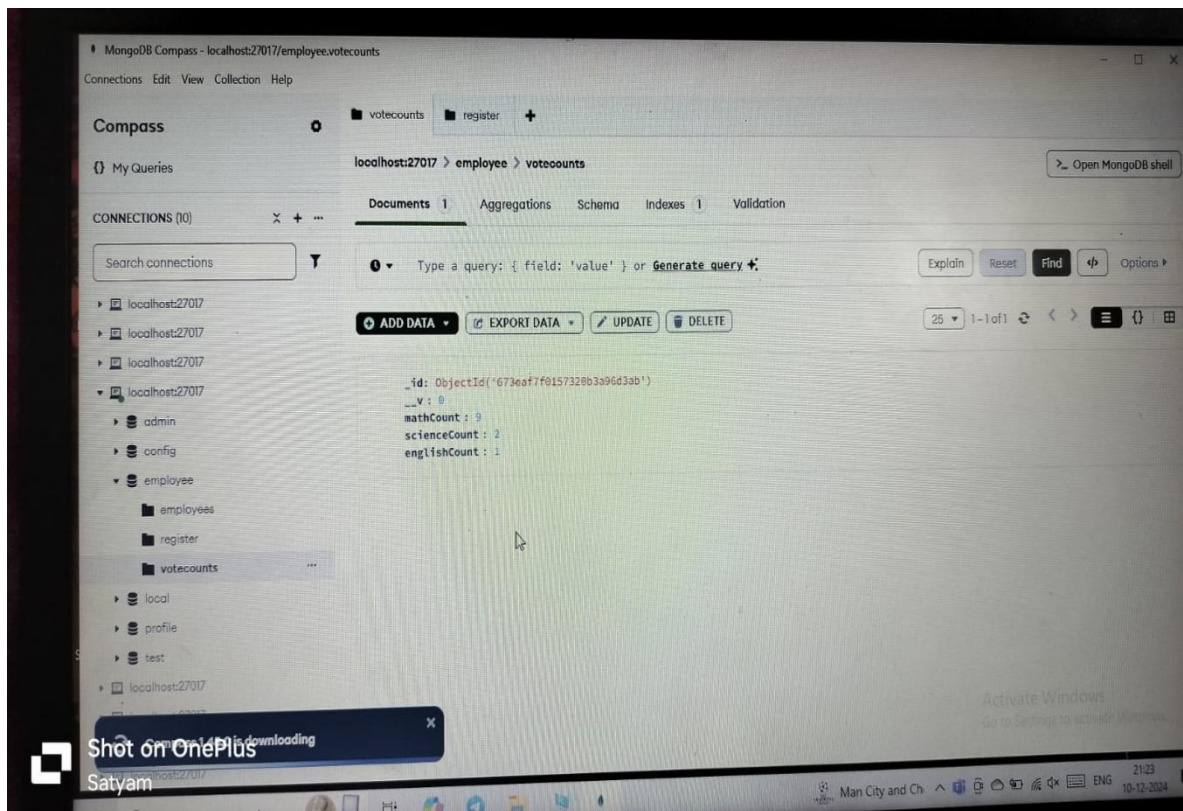
**vote\_id** : A unique identifier for each vote record.

**user\_id**: Links the vote to the user who cast it.

**subject\_id** : References the subject or course for which the vote was cast.

**vote\_timestamp**: The exact time the vote was submitted.

**status** : Indicates the vote's validity



# CHAPTER 7

## Form Design

### 7.1 Login

The login form is designed to authenticate users securely.

Features:

#### Fields:

Email

Password

#### Functionalities:

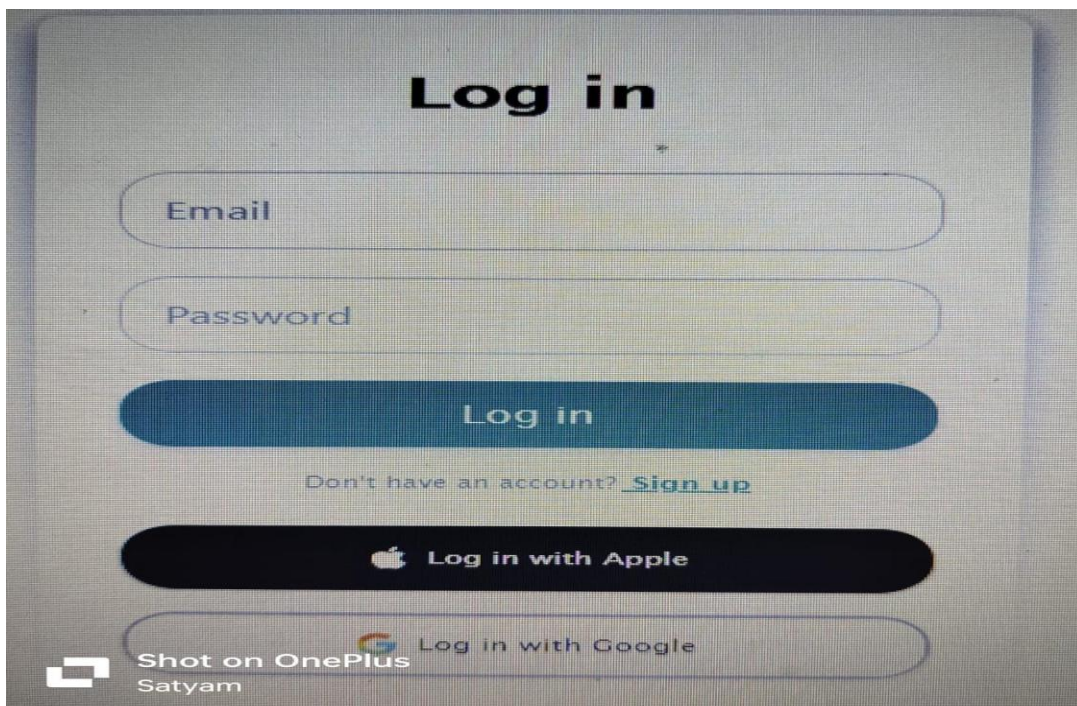
"Forgot Password" link for recovery.

"Remember Me" checkbox for persistent login.

#### UI/UX Considerations:

Minimalist design with clear labels and error messages.

Secure password masking.



## 7.2 Signup

The signup form allows new users to register on the platform.

### Features:

### Fields:

Name

Email

Password

Confirm Password

### Functionalities:

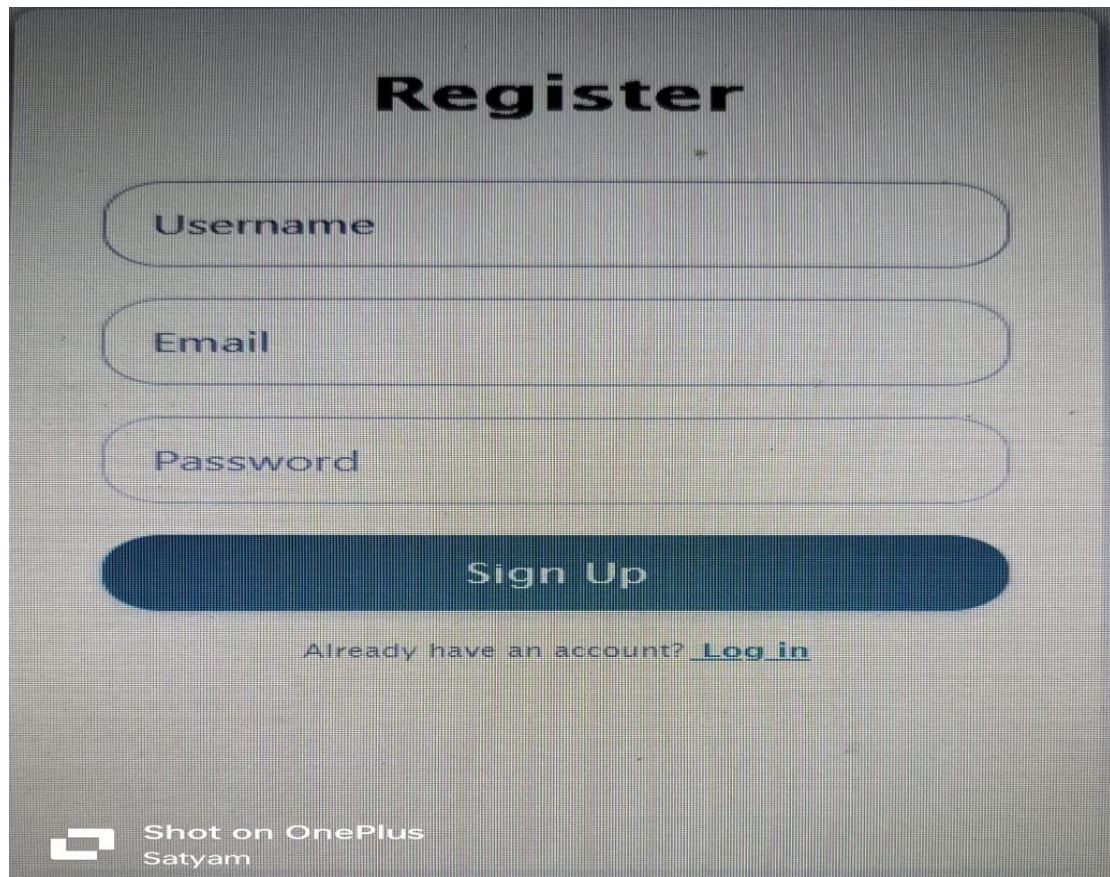
Password strength indicator.

Validation for unique email and proper format.

**UI/UX Considerations:**

Guided error messages for invalid inputs



A screenshot of a mobile application's registration screen. At the top, the word "Register" is displayed in a large, bold, black font. Below it are three rounded rectangular input fields, each with a label: "Username", "Email", and "Password". These labels are in a dark blue font. Underneath the input fields is a prominent dark blue button with the text "Sign Up" in white. Below the button, there is a line of text: "Already have an account? [Log in](#)". At the bottom left of the screen, there is a small logo consisting of two overlapping squares, followed by the text "Shot on OnePlus" and "Satyam" in a smaller font.

**Register**


Username

Email

Password

**Sign Up**

Already have an account? [Log in](#)

 Shot on OnePlus  
Satyam

## 7.3 User

### 7.3.1 Home Page

The home page serves as the main dashboard for users.

## Components:

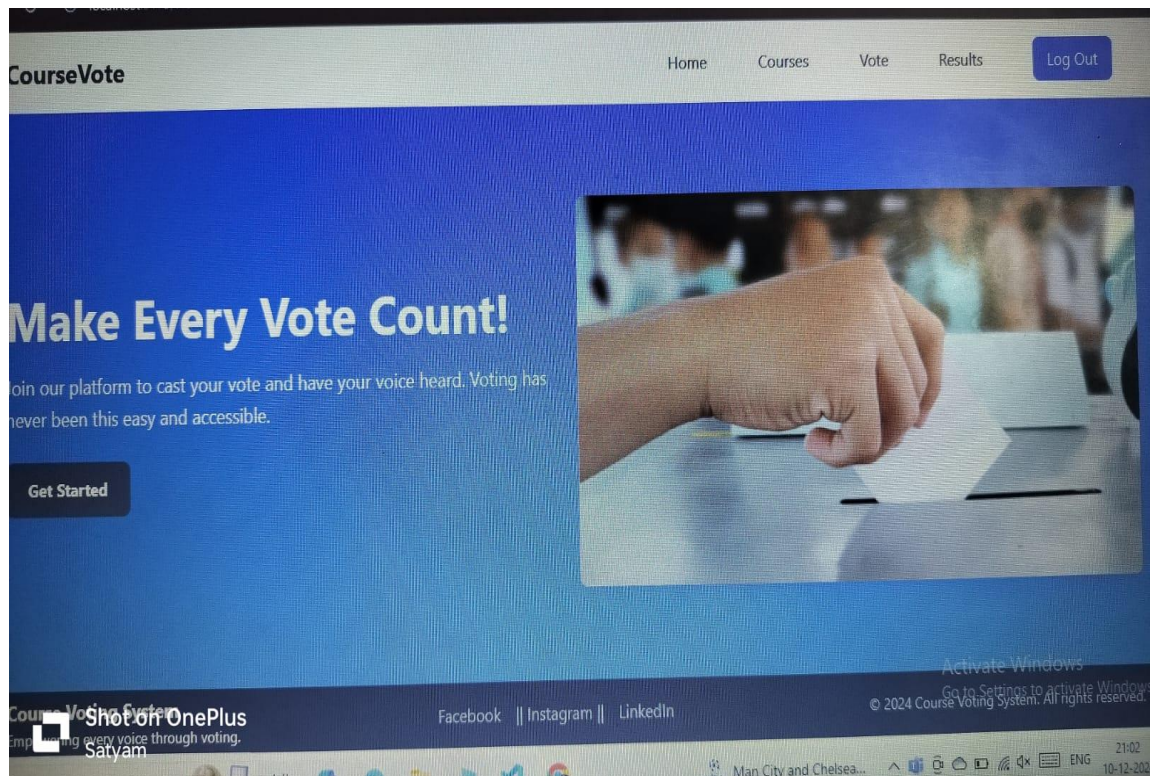
Welcome message with user's name.

Navigation bar (links to Course Page, Vote Page, Result Page, Profile).

Announcement or system updates section (optional).

## Design:

Clean layout with a responsive design.



### 7.3.2 Course Page

The course page lists all available courses for voting.

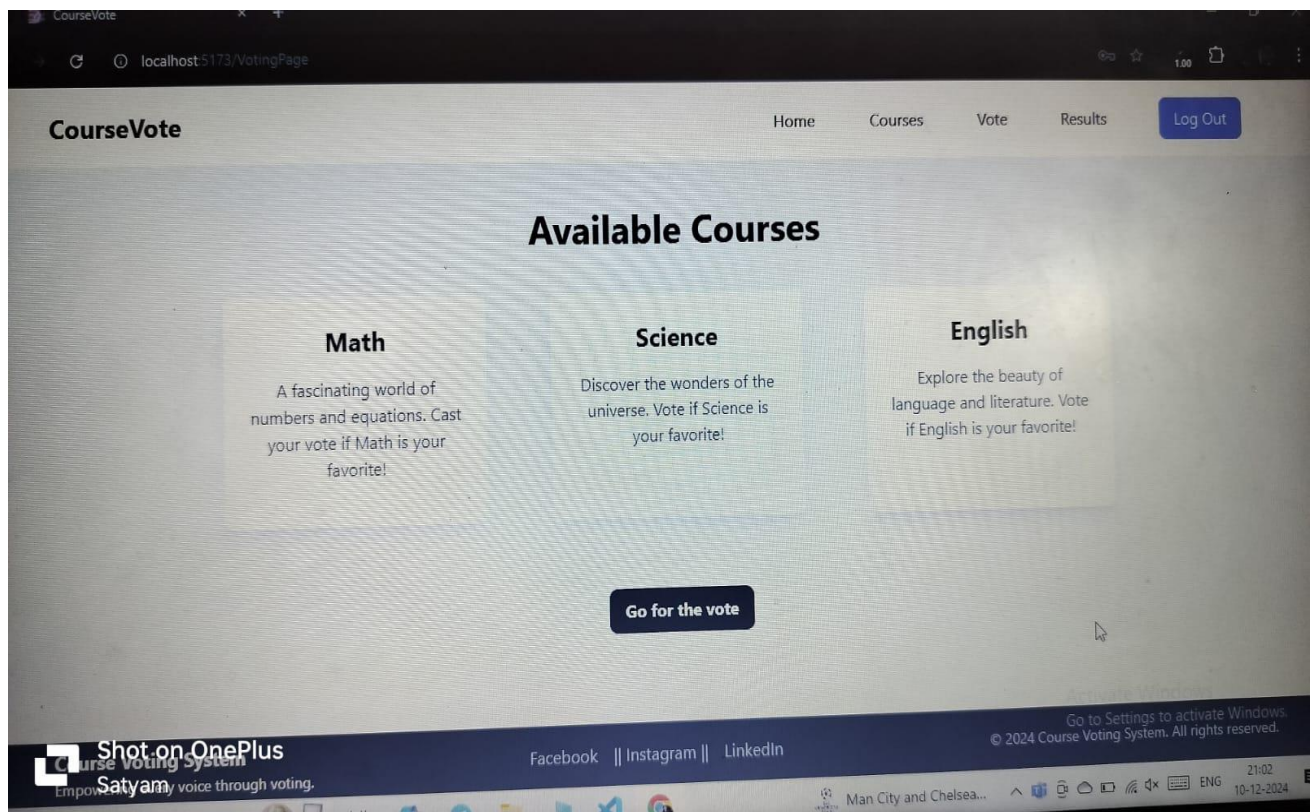
#### Components:

List or grid view of courses with brief descriptions.

"Vote Now" button beside each course.

#### Design:

Visually appealing cards for each course with dynamic hover effects





### 7.3.3 Vote Page

The vote page is where users can cast their votes.

#### Components:

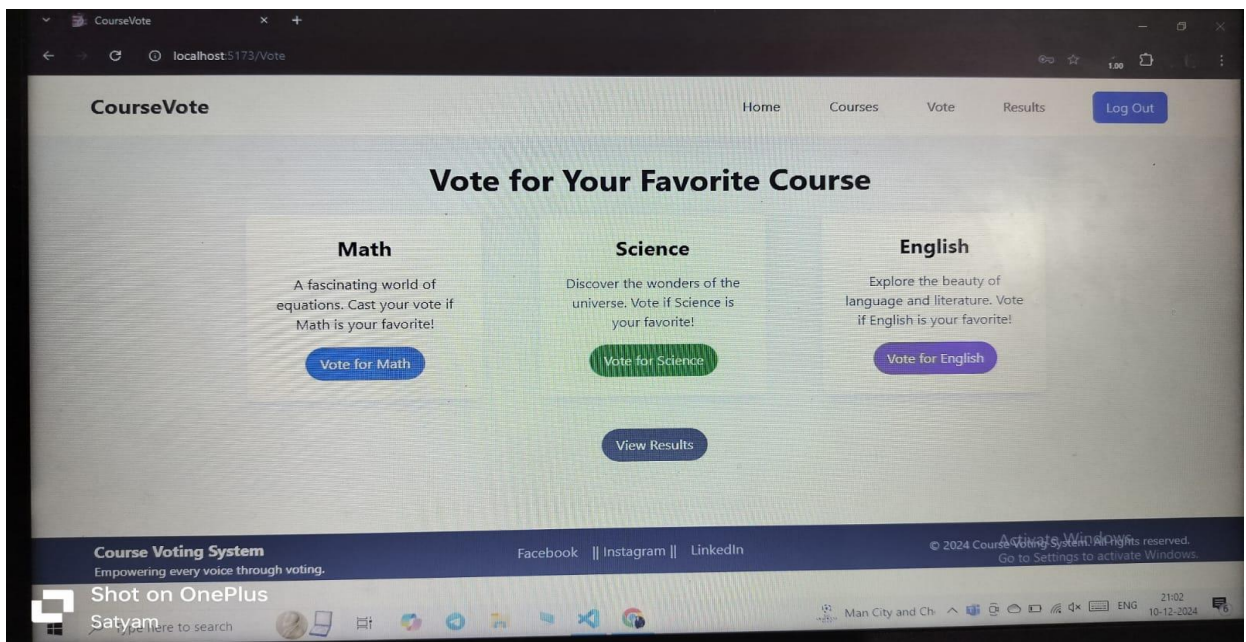
Dropdown or list of available courses.

Confirmation modal before finalizing the vote.

#### Functionalities:

Validation to ensure a single vote per user.

Live status of votes cast (optional).



### 7.3.4 Result Page

The result page displays voting outcomes.

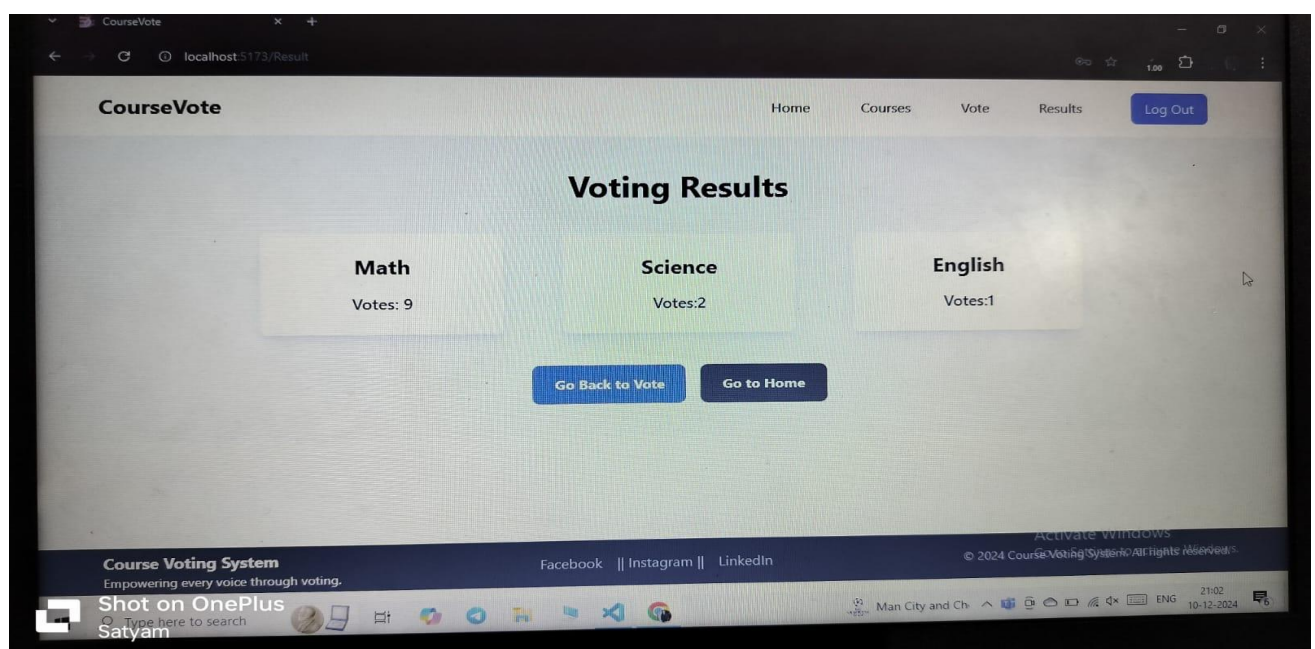
#### Components:

Bar chart or pie chart showing vote distribution.

Ranked list of courses based on votes.

#### Functionalities:

Results visibility toggle (e.g., "Available after voting ends").



## CHAPTER 8

### CODING

#### MAIN

```
import React from 'react'
```

```
import ReactDOM from 'react-dom/client'
```

```
import App from './App.jsx'
```

```
import './index.css'
```

```
ReactDOM.createRoot(document.getElementById('root')).render(  

```

```
  <React.StrictMode>
```

```
    <App />
```

```
  </React.StrictMode>,  

```

```
)
```

## **APP**

```
import React from "react";
```

```
import { BrowserRouter as Router, Route, Routes } from "react-router-dom";
```

```
import VotingPage from "./VotingPage.jsx";
```

```
import Home from "./Home.jsx";
```

```
import Signup from "./Signup.jsx";
```

```
import Login from "./Login.jsx";
```

```
import Vote from "./Vote.jsx";
```

```
import Result from "./Result.jsx";
```

```
import Logout from "./Logout.jsx";
```

```
const App = () => {
```

```
  return (
```

```
    <Router>
```

```
      <div className="App">
```

```
        <Routes>
```

**<Route path="/" element={<Signup />} />**

**<Route path="/Home" element={<Home />} />**

**<Route path="/VotingPage" element={<VotingPage />} />**

**<Route path="/Signup" element={<Signup />} />**

**<Route path="/Login" element={<Login />} />**

**<Route path="/Vote" element={<Vote />} />**

**<Route path="/Result" element={<Result />} />**

**<Route path="/Logout" element={<Logout />} />**

**</Routes>**

**</div>**

**</Router>**

**);**

**};**

**export default App;**

**HEADER**

```
import React from 'react'
```

```
import { NavLink, Outlet } from 'react-router-dom';
```

```
const Header = () => {
```

```
  return (
```

```
    <div className="bg-gray-100 font-sans" >
```

```
      <header className="bg-white shadow-lg">
```

```
        <div className="container mx-auto px-6 py-4 flex justify-between items-center">
```

```
          <h2 className="text-2xl font-bold text-gray-800">CourseVote</h2>
```

```
          <nav className="flex space-x-4">
```

```
            <ul className="flex items-center gap-14 mr-5">
```

```
              <NavLink to="/Home"><li className="text-black hover:text-blue-800">Home</li></NavLink>
```

```
              <NavLink to="/VotingPage"> <li className="text-black hover:text-blue-800">Courses</li></NavLink>
```

```
              <NavLink to="/Vote"> <li className="text-black hover:text-blue-800">Vote</li></NavLink>
```

```
      <NavLink to="/Result"><li className="text-black hover:text-blue-800">Results</li></NavLink>
```

```
      <NavLink to="/Signup"><button className="bg-blue-600 text-white px-4 py-2 rounded-lg hover:bg-gray-200 hover:text-black transition duration-300">Log Out</button></NavLink>
```

```
    </ul>
```

```
  </nav>
```

```
</div>
```

```
</header>
```

```
</div>
```

```
)
```

```
}
```

```
export default Header
```

```
HOME
```

```
import React from 'react';
```

```
import Header from './Header.jsx';
```

```
import Footer from './Footer.jsx';
```

```
import { NavLink } from 'react-router-dom';
```

```
const Home = () => {
```

```
  return (
```

```
    <>
```

```
    <Header/>
```

```
    <div className="bg-gray-100 font-sans">
```

```
      <section className="bg-blue-600 text-white py-20">
```

```
        <div className="container mx-auto px-6 flex flex-col md:flex-row items-center justify-between">
```

```
          <div className="w-full md:w-1/2">
```

```
            <h1 className="text-5xl font-bold leading-tight mb-4">Make Every Vote Count!</h1>
```

```
            <p className="text-lg mb-6">Join our platform to cast your vote and have your voice heard. Voting has never been this easy and accessible.</p>
```

```
            <NavLink to="/Vote"><button className="bg-gray-600 text-white-600 px-6 py-3 rounded-lg font-bold hover:bg-gray-200 hover:text-black transition duration-300">Get Started</button></NavLink>
```

```
          </div>
```



```
    <div className="w-full md:w-1/2 mt-8 md:mt-0">

    </div>

</div>

</section>

</div>

<Footer/>

</>

)

}

export default Home;

LOGIN

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

```
import { Link, useNavigate } from 'react-router-dom';
```

```
import './Login.css';
```

```
import axios from 'axios';
```

```
const Login = () => {
```

```
  const [name, setName] = useState()
```

```
  const [email, setEmail] = useState()
```

```
  const [password, setPassword] = useState()
```

```
  const navigate = useNavigate ();
```

```
  const handleSubmit = (e) =>{
```

```
    e.preventDefault()
```

```
    axios.post('http://localhost:3001/Login',{name,email,password})
```

```
    .then(result => {console.log(result)
```

```
      if(result.data == "Login Successful") {
```

```
        navigate('/Home')
```

```
}

else alert("Invalid Login Credentials")

})

.catch(err => console.log(err))

}

return (

  <div className="login">

    <div className="form-container">

      <p className="title">Log in</p>

      <form className="form">

        <input type="email" className="input" placeholder="Email" onChange={(e)=>
setEmail(e.target.value)} />

        <input type="password" className="input" placeholder="Password" onChange={(e)=>
setPassword(e.target.value)} />

        <button type="submit" onClick={handleSubmit} className="form-btn">Log in</button>

      </form>

    </div>

  </div>

)
```

`<p className="sign-up-label">`

`Don't have an account?<span className="sign-up-link"><Link to="/signup"`  
`className="sign-up-link"> Sign up</Link></span>`

`</p>`

`<div className="buttons-container">`

`<div className="apple-login-button">`

`<svg stroke="currentColor" fill="currentColor" strokeWidth="0" className="apple-icon"`  
`viewBox="0 0 1024 1024" height="1em" width="1em" xmlns="http://www.w3.org/2000/svg">`

`<path d="M747.4 535.7c-.4-68.2 30.5-119.6 92.9-157.5-34.9-50-87.7-77.5-157.3-82.8-65.9-5.2-`  
`138 38.4-164.4 38.4-27.9 0-91.7-36.6-141.9-36.6C273.1 298.8 163 379.8 163 544.6c0 48.7 8.9 99 26.7`  
`150.8 23.8 68.2 109.6 235.3 199.1 232.6 46.8-1.1 79.9-33.2 140.8-33.2 59.1 0 89.7 33.2 141.9 33.2 90.3-`  
`1.3 167.9-153.2 190.5-221.6-121.1-57.1-114.6-167.2-114.6-170.7zm-105.1-305c50.7-60.2 46.1-115`  
`44.6-134.7-44.8 2.6-96.6 30.5-126.1 64.8-32.5 36.8-51.6 82.3-47.5 133.6 48.4 3.7 92.6-21.2 129-`  
`63.7z"></path>`

`</svg>`

`<span>Log in with Apple</span>`

`</div>`

`<div className="google-login-button">`

```
<svg stroke="currentColor" fill="currentColor" strokeWidth="0" version="1.1" x="0px"
y="0px" className="google-icon" viewBox="0 0 48 48" height="1em" width="1em"
xmlns="http://www.w3.org/2000/svg">
```

```
<path fill="#FFC107" d="M43.611,20.083H42V20H24v8h11.303c-1.649,4.657-6.08,8-
11.303,8c-6.627,0-12-5.373-12-12
```

```
c0-6.627,5.373-12,12-12c3.059,0,5.842,1.154,7.961,3.039l5.657-
5.657C34.046,6.053,29.268,4,24,4C12.955,4,4,12.955,4,24
```

```
c0,11.045,8.955,20,20,20c11.045,0,20-8.955,20-
20C44,22.659,43.862,21.35,43.611,20.083z"></path>
```

```
<path fill="#FF3D00"
d="M6.306,14.691l6.571,4.819C14.655,15.108,18.961,12,24,12c3.059,0,5.842,1.154,7.961,3.039l5.657
-5.657
```

```
C34.046,6.053,29.268,4,24,4C16.318,4,9.656,8.337,6.306,14.691z"></path>
```

```
<path fill="#4CAF50" d="M24,44c5.166,0,9.86-1.977,13.409-5.192l-6.19-
5.238C29.211,35.091,26.715,36,24,36
```

```
c-5.202,0-9.619-3.317-11.283-7.946l-6.522,5.025C9.505,39.556,16.227,44,24,44z"></path>
```

```
<path fill="#1976D2" d="M43.611,20.083H42V20H24v8h11.303c-0.792,2.237-2.231,4.166-
4.087,5.571
```

```
c0.001-0.001,0.002-0.001,0.003-
0.002l6.19,5.238C36.971,39.205,44,34,44,24C44,22.659,43.862,21.35,43.611,20.083z"></path>
```

`</svg>`

`<span>Log in with Google</span>`

`</div>`

`</div>`

`</div>`

`</div>`

`);`

`};`

`export default Login;`

**LOGOUT**

`import React from 'react';`

`import { NavLink } from 'react-router-dom';`

`const Logout = () => {`

`return (`

```
<div className="h-full bg-gray-100 flex flex-col items-center justify-center">
```

```
<h1 className="text-4xl font-bold text-gray-800 mb-6">You Have Been Logged Out</h1>
```

```
<p className="text-gray-600 text-lg mb-10">Thank you for using CourseVote. See you  
again!</p>
```

```
<div className="flex space-x-4">
```

```
<NavLink to="/Signup">
```

```
<button className="bg-blue-600 text-white px-6 py-3 rounded-lg hover:bg-blue-500  
transition duration-300"
```

```
aria-label="Go to Sign Up">Sign Up</button>
```

```
</NavLink>
```

```
<NavLink to="/Login">
```

```
<button className="bg-gray-600 text-white px-6 py-3 rounded-lg hover:bg-gray-500  
transition duration-300"
```

```
aria-label="Go to Home">Log In</button>
```

```
</NavLink>
```

```
</div>
```

```
</div>
```

```
);
```

```
};
```

```
export default Logout;
```

## RESULT

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

```
import { NavLink } from 'react-router-dom';
```

```
import Header from './Header';
```

```
import Footer from './Footer';
```

```
import axios from 'axios';
```

```
const Result = () => {
```

```
  const[mathData,setMathData]=useState(0)
```

```
  const[englishData,setEnglishData]=useState(0)
```



```
const[scienceData,setScienceData]=useState(0)
```

```
useEffect(() => {
```

```
  axios.get('http://localhost:3001/Math')
```

```
  .then(resp=>{
```

```
    console.log(resp)
```

```
    const{mathCount}=resp.data
```

```
    setMathData(mathCount)
```

```
  })
```

```
},[])
```

```
useEffect(() => {
```

```
  axios.get('http://localhost:3001/Science')
```

```
  .then(resp=>{
```

```
    console.log(resp)
```

```
    const{scienceCount}=resp.data
```

```
    setScienceData(scienceCount)
```

```
)
```

```
},[])
```

```
useEffect(() => {
```

```
  axios.get('http://localhost:3001/English')
```

```
    .then(resp=>{
```

```
      console.log(resp)
```

```
      const{englishCount}=resp.data
```

```
      setEnglishData(englishCount)
```

```
    })
```

```
  },[])
```

```
  return (
```

```
    <>
```

```
    <Header/>
```

```
    <div className="h-full bg-gray-100 flex flex-col items-center py-10">    <h1 className="text-4xl font-bold text-gray-800 mb-10">Voting Results</h1>
```

```
<div className="flex flex-wrap justify-center gap-16">
```

```
  { /* Math Results */ }
```

```
<div className="bg-white rounded-lg shadow-lg p-6 w-64 text-center">
```

```
  <h2 className="text-2xl font-bold text-gray-800 mb-4">Math</h2>
```

```
  <p className="text-gray-700 font-semibold text-lg">Votes: {mathData}</p>
```

```
</div>
```

```
  { /* Science Results */ }
```

```
<div className="bg-white rounded-lg shadow-lg p-6 w-64 text-center">
```

```
  <h2 className="text-2xl font-bold text-gray-800 mb-4">Science</h2>
```

```
  <p className="text-gray-700 font-semibold text-lg">Votes:{scienceData} </p>
```

```
</div>
```

```
  { /* English Results */ }
```

```
<div className="bg-white rounded-lg shadow-lg p-6 w-64 text-center">
```

```
  <h2 className="text-2xl font-bold text-gray-800 mb-4">English</h2>
```

**<p className="text-gray-700 font-semibold text-lg">Votes:{englishData}</p>**

**</div>**

**</div>**

**<div className="mt-10 flex space-x-4">**

**<NavLink to="/Vote">**

**<button className="bg-blue-600 font-bold text-white px-6 py-3 rounded-lg hover:bg-gray-200 hover:text-black transition duration-300"**

**aria-label="Go Back to Vote">Go Back to Vote</button>**

**</NavLink>**

**<NavLink to="/Home">**

**<button className="bg-gray-600 font-bold text-white px-6 py-3 rounded-lg hover:bg-gray-200 hover:text-black transition duration-300"**

**aria-label="Go to Home">Go to Home</button>**

**</NavLink>**

**</div>**

**</div>**

```
<div className='mt-32'>
```

```
<Footer/>
```

```
</div>
```

```
</>
```

```
);
```

```
};
```

```
export default Result;
```

**SIGN-UP**

```
import {useState} from 'react';
```

```
import './Sign.css';
```

```
import { Link } from 'react-router-dom';
```

```
import axios from 'axios';
```

```
import { useNavigate } from 'react-router-dom';
```

```
const Signup = () => {

  const [name, setName] = useState()

  const [email, setEmail] = useState()

  const [password, setPassword] = useState()

  const navigate = useNavigate();


  const handleSubmit = (e) =>{

    e.preventDefault()

    axios.post('http://localhost:3001/register',{name,email,password})

    .then(result => {console.log(result)

      navigate('/login')

    })

    .catch(err => console.log(err))

  }

  return (

    <div className="signup">
```

```
<div className="form-container">
```

```
<p className="title">Register</p>
```

```
<form className="form" onSubmit={handleSubmit}>
```

```
<input type="text" className="input" placeholder="Username" onChange={(e)=>
setName(e.target.value)} />
```

```
<input type="email" className="input" placeholder="Email" onChange={(e)=>
setEmail(e.target.value)} />
```

```
<input type="password" className="input" placeholder="Password" onChange={(e)=>
setPassword(e.target.value)} />
```

```
<button type="submit" className="form-btn">Sign Up</button>
```

```
</form>
```

```
<p className="sign-up-label">
```

```
Already have an account?<span className="sign-up-link"><Link to="/login"
className="sign-up-link"> Log in</Link></span>
```

```
</p>
```

```
</div>
```

```
</div>
```

```
);
```

```
};
```

```
export default Signup
```

**Vote**

```
import React, { useState } from "react";
```

```
import { NavLink } from "react-router-dom";
```

```
import Header from "../Header";
```

```
import axios from "axios";
```

```
import { ToastContainer, toast } from "react-toastify";
```

```
import "react-toastify/dist/ReactToastify.css";
```

```
import Footer from "../Footer";
```

```
const Vote = () => {
```

```
  const [voted, setVoted] = useState(false);
```



```
const [mathCounter, setMathCount] = useState(0);

const [englishCounter, setEnglishCount] = useState(0);

const [scienceCounter, setScienceCount] = useState(0);

const scienceVote = () => {

  const updateScienceCount = scienceCounter + 1;

  setScienceCount(updateScienceCount);

  try {

    axios.post("http://localhost:3001/Science", {

      scienceCount: updateScienceCount,

    });

    toast.success("Vote successfully");

    setVoted(true);

    console.log(updateScienceCount);

  } catch (err) {

    toast.error("Error voting");

  }
}
```

```
};
```

```
const englishVote = () => {
```

```
    const updateEnglishCount = englishCounter + 1;
```

```
    setEnglishCount(updateEnglishCount);
```

```
    try {
```

```
        axios.post('http://localhost:3001/English', {
```

```
            englishCount: updateEnglishCount,
```

```
        });
```

```
        toast.success("Vote successfully");
```

```
        setVoted(true);
```

```
        console.log(updateEnglishCount);
```

```
    } catch (err) {
```

```
        toast.error("Error voting");
```

```
        console.log("Error", err);
```

```
}
```

```
};
```

```
const mathVote = (e) => {
```

```
    const updateCount = mathCounter + 1;
```

```
    setMathCount(updateCount);
```

```
    try {
```

```
        axios.post("http://localhost:3001/Math", { mathCount: updateCount });
```

```
        toast.success("Vote successfully");
```

```
        setVoted(true);
```

```
        console.log(updateCount);
```

```
    } catch (err) {
```

```
        toast.error("Error voting");
```

```
        console.log("Error", err);
```

```
    }
```

```
};
```

```
const handleVote = (subject) => {
```

```
  setVotes((prevVotes) => ({
```

```
    ...prevVotes,
```

```
    [subject]: prevVotes[subject] + 1,
```

```
  }));
```

```
};
```

```
return (
```

```
<>
```

```
<Header />
```

```
<div className="h-full bg-gray-100 flex flex-col items-center py-8">
```

```
<h1 className="text-4xl font-bold text-gray-800 mb-6">
```

```
  Vote for Your Favorite Course
```

```
</h1>
```

```
<div className="flex flex-wrap justify-center gap-16">
```

```
<div className="bg-white rounded-lg shadow-lg p-6 w-64 text-center transform hover:scale-105 transition duration-300">
```

```
<h2 className="text-2xl font-bold text-gray-800 mb-4">Math</h2>
```

```
<p className="text-gray-800 mb-4">
```

A fascinating world of equations. Cast your vote if

Math is your favorite!

```
</p>
```

```
<button
```

```
onClick={() => mathVote()}
```

```
disabled={voted}
```

```
className={`bg-blue-600 text-white px-4 py-2 rounded-full hover:bg-blue-500 transition duration-300 ${
```

```
voted ? "cursor-not-allowed" : ""
```

```
}`}
```

```
aria-label="Vote for Math"
```

```
>
```

Vote for Math

```
</button>
```

</div>

{/\* Science Card \*/}

<div className="bg-white rounded-lg shadow-lg p-6 w-64 text-center transform hover:scale-105 transition duration-300">

<h2 className="text-2xl font-bold text-gray-800 mb-4">Science</h2>

<p className="text-gray-600 mb-4">

Discover the wonders of the universe. Vote if Science is your

favorite!

</p>

<button

onClick={() => scienceVote()}

disabled={voted}

className={`bg-green-600 text-white px-4 py-2 rounded-full hover:bg-green-500 transition duration-300 \${voted ? 'cursor-not-allowed' : ''}`} aria-label="Vote for Science"

>

Vote for Science{" "}

</button>

</div>

{/\* English Card \*/}

<div className="bg-white rounded-lg shadow-lg p-6 w-64 text-center transform hover:scale-105 transition duration-300">

<h2 className="text-2xl font-bold text-gray-800 mb-4">English</h2>

<p className="text-gray-600 mb-4">

Explore the beauty of language and literature. Vote if English is

your favorite!

</p>

<button

onClick={() => englishVote()}

disabled={voted}

className={`bg-purple-600 text-white px-4 py-2 rounded-full hover:bg-purple-500 transition duration-300 \${

voted ? "cursor-not-allowed" : ""

}}}

aria-label="Vote for English"

>

{' '}

Vote for English{' '}

</button>

</div>

</div>

<div className="mt-10">

<NavLink to="/Result">

<button

className={`bg-gray-600 text-white px-4 py-2 rounded-full hover:bg-gray-500 transition  
duration-300

}}}

aria-label="View Results"

>

View Results{' '}

</button>



```
</NavLink>
```

```
</div>
```

```
</div>
```

```
<ToastContainer />
```

```
<div className="mt-12">
```

```
<Footer />
```

```
</div>
```

```
</>
```

```
);
```

```
};
```

```
export default Vote;
```

```
Voting-Page
```

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

```
import { NavLink } from 'react-router-dom';
```

```
import Header from './Header';
```

```
import Footer from './Footer';
```

```
const VotingPage = () => {
```

```
  return (
```

```
    <>
```

```
    <Header/>
```

```
    <div className="h-full bg-gray-100 flex flex-col items-center py-10">
```

```
      <h1 className="text-4xl font-bold text-gray-800 mb-10">Available Courses</h1>
```

```
      <div className="flex flex-wrap justify-center gap-16">
```

```
        {/* Math Card */}
```

```
        <div className="bg-white rounded-lg shadow-lg p-6 w-64 text-center transform hover:scale-105 transition duration-300">
```

```
          <h2 className="text-2xl font-bold text-gray-800 mb-4">Math</h2>
```

```
          <p className="text-gray-600 mb-4">A fascinating world of numbers and equations. Cast  
your vote if Math is your favorite!</p>
```

</div>

{/\* Science Card \*/}

<div className="bg-white rounded-lg shadow-lg p-6 w-64 text-center transform hover:scale-105 transition duration-300">

<h2 className="text-2xl font-bold text-gray-800 mb-4">Science</h2>

<p className="text-gray-600 mb-4">Discover the wonders of the universe. Vote if Science is your favorite!</p>

</div>

{/\* English Card \*/}

<div className="bg-white rounded-lg shadow-lg p-6 w-64 text-center transform hover:scale-105 transition duration-300">

<h2 className="text-2xl font-bold text-gray-800 mb-4">English</h2>

<p className="text-gray-600 mb-4">Explore the beauty of language and literature. Vote if English is your favorite!</p>

</div>

</div>

```
      <NavLink to="/Vote"> <button className='bg-gray-800 mt-16 font-bold text-white px-4 py-2
rounded-lg hover:bg-gray-200 hover:text-black transition duration-300''>Go for the
vote</button></NavLink>
```

```
</div>
```

```
<div className='mt-10'>
```

```
<Footer/>
```

```
</div>
```

```
</>
```

```
);
```

```
};
```

```
export default VotingPage;
```

```
EMPLOYEE
```

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

```
const EmployeeSchema = new mongoose.Schema({  
  
  name: String,  
  
  email: String,  
  
  password: String  
  
})
```

```
const EmployeeModel = mongoose.model('Employee', EmployeeSchema)  
  
module.exports = EmployeeModel
```

**VOTE**

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

```
const voteCountSchema=new mongoose.Schema({
```

```
  mathCount:{
```

```
    type:Number,

    required:true

  },

  englishCount:{

    type:Number,

    required:true

  },

  scienceCount:{

    type:Number,

    required:true

  }

})

const voteCountModel= mongoose.model('voteCount',voteCountSchema);

module.exports = voteCountModel;
```

**SERVER**

```
const express = require('express')
```

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

```
const cors = require('cors')
```

```
const voteCountModel = require('./models/vote')
```

```
const EmployeeModel = require('./models/employee')
```

```
const app = express()
```

```
app.use(express.json())
```

```
app.use(cors())
```

```
try{
```

```
//
```

```
mongoose.connect('mongodb+srv://sattyam232106:Satyam7991@mycluster.pbqrk.mongodb.net/');
```

```
mongoose.connect('mongodb://localhost:27017/employee');
```

```
console.log("Mongodb connected");
```

```
}catch(e){
```

```
console.log(e,"Connection failed");
```

```
}
```

```
app.post('/Math', async(req,res)=>{
```

```
console.log(req.body);
```

```
try{
```

```
const result = await voteCountModel.findOneAndUpdate(
```

```
{},
```

```
{ $inc: { mathCount: 1 } },
```

```
{ new: true, upsert: true }
```

```
);
```

```
console.log('Vote updated:', result)
```

```
res.status(200).json({ error: 'update vote sucess' });
```



```
}catch(err){

    console.error('Error updating vote:', err);

    res.status(500).json({ error: 'Failed to update vote' });

}

})

app.post('/Science', async(req,res)=>{

    console.log(req.body);

    try{

        const result = await voteCountModel.findOneAndUpdate(

            {},

            { $inc: { scienceCount: 1 } },

            { new: true, upsert: true }

        );

        console.log('Vote updated:', result)

        res.status(200).json({ error: 'update vote sucess' });

    }
```

```
}catch(err){

    console.error('Error updating vote:', err);

    res.status(500).json({ error: 'Failed to update vote' });

}

})

app.post('/English', async(req,res)=>{

    console.log(req.body);

    try{

        const result = await voteCountModel.findOneAndUpdate(

            {},

            { $inc: { englishCount: 1 } },

            { new: true, upsert: true }

        );

        console.log('Vote updated:', result)

        res.status(200).json({ error: 'update vote sucess' });
```

```
    }catch(err){

        console.error('Error updating vote:', err);

        res.status(500).json({ error: 'Failed to update vote' });

    }

})

app.get('/Math',async(req,res)=>{

    try{

        const result = await voteCountModel.findOne();

        res.json(result)}

    catch(err){

        console.error('Error fetching vote:', err);

        res.status(500).json({ error: 'Failed to fetch vote' });

    }

})

app.get('/English',async(req,res)=>{
```

```
    try{

        const result = await voteCountModel.findOne();

        res.json(result)}

    catch(err){

        console.error('Error fetching vote:', err);

        res.status(500).json({ error: 'Failed to fetch vote' });

    }

})

app.get('/Science',async(req,res)=>{

    try{

        const result = await voteCountModel.findOne();

        res.json(result)}

    catch(err){

        console.error('Error fetching vote:', err);

        res.status(500).json({ error: 'Failed to fetch vote' });

    }

})
```

```
})
```

```
app.post('/Login', (req, res) =>{

  const {email, password} = req.body;

  EmployeeModel.findOne({email: email})

  .then(user =>{

    if(user){

      if(user.password === password){

        res.json( 'Login Successful')

      }else{

        res.json('Invalid Password')

      }

    }

  }

  else{

    res.json('User does not exist')
```

```
}
```

```
})
```

```
})
```

```
app.post('/register',async(req, res)=>{
```

```
  await EmployeeModel.create(req.body)
```

```
  .then(employees => res.json(employees))
```

```
  .catch(err => res.json(err))
```

```
})
```

```
app.listen(3001,()=>{
```

```
  console.log('Server running on port 3001')
```

```
})
```

## **CHAPTER- 09**

### **TESTING**

#### **Introduction**

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionalities of components, sub-assemblies, and/or a finished product it is the process of exercising software with the intent of ensuring that the software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement.

#### **Types of Testing**

##### **Unit Testing**

Unit testing focuses verification effort on the smallest unit of software design, the module. The unit testing, we have is white box oriented and some modules the steps are conducted in parallel.

##### **Integration Testing**

Testing is done for each module. After testing all the modules, the modules are integrated and testing of the final system is done with the test data, specially designed to show that the system will operate successfully in all its aspects conditions. Thus, the system testing

is a confirmation that all is correct and an opportunity to show the user that the system works.

The purpose of integration testing is to verify functional, performance and reliability requirements placed on major design items. These "design items", i.e. assemblages (or groups of units), are exercised through their interfaces using black box testing, success and error cases being simulated via appropriate parameter and data inputs. Simulated usage of shared data areas and inter-process communication is tested and individual subsystems are exercised through their input interface.

## System Testing

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration

# **FUTURE SCOPE AND FURTHER ENHANCEMENT OF THE PROJECT**

## **Future Scope of the Voting Application**

### **1. Scalability for Larger Audiences:**

- Expand the application to handle larger-scale elections, such as university



elections, corporate voting, or even local government elections.

- Use cloud technologies (e.g., AWS, Azure) to scale the infrastructure to accommodate a high number of users and votes.

## **2. Multi-Platform Support:**

- Extend the application to work seamlessly across different platforms, including mobile apps (iOS and Android), web browsers, and desktop applications.
- Use responsive design and Progressive Web App (PWA) technologies to enhance accessibility.

## **3. Real-Time Voting Analytics:**

- Implement dashboards that provide real-time data visualization, such as voting trends, participation rates, and live leaderboards.

## **4. Integration with Blockchain:**

- Leverage blockchain technology to enhance security, transparency, and immutability in the voting process, ensuring tamper-proof elections.

## **5. AI-Powered Insights:**

- Use AI/ML to analyze voting data for insights such as voter turnout prediction, user preferences, and anomaly detection (e.g., identifying suspicious voting patterns).

## **Further Enhancements**

### **1. Anonymous Voting:**

- Ensure voter anonymity by separating user identities from their votes using

advanced cryptographic techniques.

## **2. Role-Based Access Control:**

- Introduce roles for administrators, moderators, and voters with specific permissions to manage the voting process efficiently.

## **3. Offline Voting Support:**

- Provide an option to collect votes offline and sync them to the server once the internet connection is restored.

## **4. Notification System:**

- Add features like email, SMS, or app notifications to remind users about voting deadlines or confirm successful vote submissions.

## **5. Integration with External Systems:**

- Allow integration with external systems like Google Forms, government ID verification services, or university portals for voter registration and authentication.

# CONCLUSION & REFERNCES

The voting application is a highly functional and innovative project that simplifies the voting process, ensuring fairness, transparency, and accessibility. By leveraging modern web technologies, it provides a reliable and secure platform for conducting elections efficiently. The project addresses critical challenges like user authentication, vote accuracy, and real-time result computation, making it suitable for small-scale elections such as classroom or organizational voting. With its potential for future enhancements, such as integration with blockchain for tamper-proof elections, AI-based analytics for insights, and multi-platform compatibility, the application can evolve into a robust and scalable system. The adoption of such features can transform it into a versatile solution capable of handling large-scale elections, ensuring inclusivity, and fostering trust in the democratic process. Overall, this project represents a significant step toward digitizing and streamlining election systems.

## **References**

Coding phase: -

1. React

Referenced Sites:

[www.w3school.com](http://www.w3school.com)

[www. https://react.dev/learn](https://react.dev/learn)