STUDENT ELECTION SYSTEM

A MINI-PROJECT REPORT

Submitted by

KALAISELVI S	**220701116**
KEERTHIKA B	**220701126**

in partial fulfillment of the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

RAJALAKSHMI ENGINEERING COLLEGE, CHENNAI

An Autonomous Institute
Thandalam
Chennai - 602105
2023-2024

BONAFIDE CERTIFICATE

Certified that this project report “**STUDENT ELECTION SYSTEM**” is the bonafide work of “ **KALAISELVI S(220701116), KEERTHIKA B(220701126)**” who carried out the project work under my supervision.

Submitted for the Practical Examination held on ______________________

SIGNATURE	**SIGNATURE**
Dr.R.SABITHA **Professor and II year Academic Head,** **Computer Science and Engineering,** **Rajalakshmi Engineering College (Autonomous),** **Thandalam, Chennai - 602 105.**	**Dr.G.Dharani Devi** **Associate Professor,** **Computer Science and Engineering,** **Rajalakshmi Engineering College (Autonomous),** **Thandalam, Chennai - 602 105.**

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ABSTRACT

The Student Election System is an advanced web-based platform developed to streamline and secure the election process within student organizations. The system is built using a combination of HTML, CSS, JavaScript, PHP, and SQL, ensuring a robust and dynamic user experience. This project aims to eliminate the inefficiencies and vulnerabilities of traditional paper-based voting systems by offering a comprehensive digital solution.

Key features of the Student Election System include multiple modules for user registration and login, tailored for voters, candidates, and administrators. Voters can register and log in to cast their votes electronically, candidates can log in to manage their profiles and campaign details, and administrators can oversee the entire election process. The front-end interface, designed with HTML and CSS, provides a user-friendly and responsive experience across various devices. JavaScript is utilized to enhance interactivity and client-side validation, ensuring a smooth and seamless user experience.

The back-end functionality is powered by PHP, which handles server-side processing, form submissions, and business logic. SQL is employed for database management, storing all relevant data such as user credentials, votes, and election results securely. The system's architecture ensures secure authentication and authorization processes, with each voter receiving a unique identification number to prevent multiple voting.

In conclusion, the Student Election System utilizes modern web technologies to create an efficient, secure, and transparent platform for student elections. This integration enhances the integrity and accessibility of the election process, ensuring accurate and real-time results.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

The student election system is designed to streamline the process of conducting student elections within educational institutions. This system leverages technology to ensure a fair, transparent, and efficient election process. By transitioning from traditional paper-based voting to a digital platform, the system aims to enhance accessibility, reduce administrative burden, and increase student participation.

1.2. OBJECTIVES

The primary objectives of the student election system are:

1. Efficiency: To expedite the election process by reducing the time and resources required for organizing and conducting elections.

2. Transparency: To ensure the integrity of the election by providing a transparent voting process that is easy to audit.

3. Accessibility: To make the voting process more accessible to all students, allowing them to vote from anywhere within a specified timeframe.

4. Security: To protect the election data and ensure that the votes are secure, preventing any form of tampering or fraud.

5. User-Friendliness: To create an intuitive interface that is easy for students and administrators to use, ensuring high levels of participation and satisfaction.

6. Data Accuracy: To minimize human errors in vote counting and result tabulation, ensuring accurate election outcomes.

1.3. MODULES

The student election system is composed of several key modules, each serving a specific function to ensure the smooth operation of the election process:

1. User Authentication Module

Voter Registration:

Allows students to register as voters by providing necessary identification and details.

Voter Login:

Enables registered voters to securely log in to the system using their credentials.

Candidate Login:

Provides a login interface for candidates to access their profiles and manage their campaign information.

Admin Login:

Allows administrators to log in to the system to manage the election process and view detailed statistics.

2. Candidate Management Module

- Allows administrators to manage candidate nominations and verify their eligibility.
- Provides a platform for candidates to submit their profiles and campaign information.

3. Voting Module

- Facilitates the voting process, allowing students to cast their votes securely and anonymously.
- Implements measures to prevent multiple voting and ensure the integrity of each vote.

4. Results Tabulation Module

- Automates the counting of votes and the calculation of election results.
- Provides real-time updates on voting progress and final results.

5. Admin Dashboard

- Displays voting percentage details and other key statistics to administrators.
- Provides tools for administrators to monitor election activities and ensure everything runs smoothly.

6. About Page

- Provides information about the student election system, its purpose, and how it works.
- Includes FAQs, contact information, and support resources for users.

CHAPTER 2

SURVEY OF TECHNOLOGIES

2.1 SOFTWARE DESCRIPTION

ARCHITECTURE DESCRIPTION

Presentation Layer

Responsible for the user interface and interaction.

Implements web pages, forms, and interfaces for users.

Utilizes HTML for content structure, CSS for styling, and JavaScript for interactivity.

PHP used for server-side processing and dynamic content generation.

Application Layer

Manages the business logic and workflow of the system.

Processes user requests, coordinates data flow, and interacts with the database.

Implements routing, request handling, and data manipulation logic.

Primarily consists of PHP scripts running on the server.

Data Access Layer

Handles interactions with the database.

Executes SQL queries for data retrieval, storage, and manipulation.

Ensures data integrity and security.

Utilizes MySQL or PostgreSQL as the relational database management system.

Database Layer

Stores and organizes system data, including user profiles, candidate information, and voting records.

Manages data persistence and retrieval operations.

Provides a structured storage mechanism for efficient data handling.

Relies on MySQL or PostgreSQL databases for data storage and management.

MODEL-VIEW-CONTROLLER (MVC) DESCRIPTION

Model

Represents the data and business logic of the application.

Defines data structures, database interactions, and application logic.

Validates user input, enforces business rules, and maintains data integrity.

Includes models for users, candidates, votes, etc.

View

Renders the user interface and presents data to users.

Implements HTML templates for layout and structure.

Applies CSS stylesheets for visual presentation and user experience enhancement.

Includes JavaScript for client-side interactivity and dynamic content manipulation.

Controller

Acts as an intermediary between the model and the view.

Handles user requests, invokes appropriate methods in the model, and prepares data for presentation in the view.

Implements routing logic, request handling, and response generation.

Coordinates the interaction between the user interface and the underlying data model.

COMPONENT DESCRIPTION

User Authentication Component

Manages user authentication and access control.

Validates user credentials and manages user sessions.

Ensures secure access to system resources and functionalities.

Candidate Management Component

Handles candidate nominations, eligibility verification, and campaign management.

Allows candidates to submit their profiles and campaign information.

Provides administrative tools for managing candidate data and profiles.

Voting Component

Facilitates the voting process for registered users.

Presents ballot options to voters and records their votes securely.

Implements measures to prevent multiple voting and ensure the integrity of the voting process.

Results Tabulation Component

Automates the counting of votes and calculation of election results.

Generates real-time updates on voting progress and final results.

Provides tools for administrators to monitor election outcomes and analyze voting patterns.

Admin Dashboard Component

Offers a centralized interface for administrators to monitor and manage the election process.

Displays key statistics, voting percentages, and other relevant information.

About Page Component

Provides information about the Student Election System, its purpose, and functionalities.

Includes FAQs, contact information, and support resources for users.

Enhances user understanding and engagement with the system.

2.2. LANGUAGES

1. index.php

Description: Main landing page. Provides navigation to various sections like voter login, candidate registration, and admin login.

2. about.php

Description: Page containing information about the election system.

3. adminLogin.php

Description: PHP script and HTML form for admin login.

4. adminLogout.php

Description: PHP script for admin logout functionality.

5. dashboard.php

Description: Admin dashboard for managing the election process, viewing voter and candidate details, and election results.

6. loginSubmit.php

Description: PHP script to handle login submissions for both voters and candidates.

7. logout.php

Description: PHP script for logging out users (voters and candidates).

8. master.css

Description: CSS file containing styles for the entire application.

9. registerCandidate.php

Description: PHP script and HTML form for candidate registration.

10. registerCandidateScript.php

Description: PHP script to handle candidate registration form submissions and insert data into the database.

11. registerVoter.php

Description: PHP script and HTML form for voter registration.

12. registerVoterScript.php

Description: PHP script to handle voter registration form submissions and insert data into the database.

13. result.php

Description: Page for displaying the election results.

14. studentvote.sql

Description: SQL file to set up the database schema for the Student Election System.

15. update.php

Description: PHP script for updating voter or candidate details.

16. vote.php

Description: PHP script and HTML form for casting votes.

17. voteCaste.php

Description: PHP script to handle vote submissions and record them in the database.

18. voterLogin.php

Description: PHP script and HTML form for voter login.

CHAPTER 3

REQUIREMENTS AND ANALYSIS

3.1 REQUIREMENT SPECIFICATION

USER STORIES

1.As a user, I want to register as a voter by providing my identification and necessary details, so that I can participate in the election.

2.As a user, I want to securely log in to the system using my credentials, so that I can access the voting platform.

3.As a user, I want to log in to the system using my credentials as a candidate, so that I can manage my profile and campaign information.

4.As a user, I want to securely log in to the system as an administrator, so that I can manage the election process and view detailed statistics.

5.As a user, I want to manage candidate nominations and verify their eligibility (as an administrator), so that I can ensure only eligible candidates participate.

6.As a user, I want to submit my profile and campaign information through the system (as a candidate), so that students can view my candidacy details.

7.As a user, I want to cast my vote securely and anonymously, so that my vote is confidential and counted correctly.

8.As a user, I want the system to prevent multiple voting, so that the integrity of the election is maintained.

9. As a user, I want the system to automate the counting of votes, so that the results are calculated efficiently and accurately.1.As a user, I want to view detailed voting percentage details and other key statistics on the dashboard, so that I can monitor election activities (as an administrator).

10. As a user, I want to read information about the student election system, so that I understand its purpose and how it works.

3.2 HARDWARE AND SOFTWARE REQUIREMENTS

HARDWARRE REQUIREMENTS :

1.Server

- A dedicated server or cloud-based hosting capable of handling the expected traffic and data processing requirements of the system.
- Suggested specifications:
- CPU: Multi-core processor (e.g., Intel Xeon, AMD Ryzen)
- RAM: At least 8GB (16GB recommended)
- Storage: SSD storage for faster data access
- Network: Stable internet connection with sufficient bandwidth

3. Database Server

A separate database server to store user credentials, candidate information, and voting data securely.

4. Network Infrastructure

Reliable networking equipment including switches, routers, and firewalls to ensure secure communication between clients and servers.

Network security measures such as firewalls, intrusion detection/prevention systems, and regular security updates.

5. Client Devices

Devices for users to access the system including desktop computers, laptops, tablets, and smartphones.

Compatible web browsers for accessing the system's web interface.

SOFTWARE REQUIREMENTS :

1. Operating System

- For the server: Linux distribution (e.g., Ubuntu Server, CentOS) or Windows Server, depending on the system administrator's preference and expertise.
- For client devices: Compatibility with popular operating systems like Windows, macOS, iOS, Android, and Linux.

2.Web Server

Apache, Nginx, or Microsoft Internet Information Services (IIS) to serve web pages and handle HTTP requests.

3.Programming Languages

- Server-side scripting language such as PHP, Python, or Node.js for dynamic web page generation and server-side logic.
- Client-side scripting languages like JavaScript for interactive features.

4.Database Management System (DBMS)

MySQL, PostgreSQL, MongoDB, or other relational or NoSQL databases for storing and managing data.

5.Security Software

- SSL/TLS certificates for encrypting data transmitted over the network.
- Security protocols and best practices for user authentication, authorization, and data protection.
- Regular security updates and patches to address potential vulnerabilities.

6.Other Dependencies

- Additional software libraries, frameworks, or modules required by the chosen programming languages and development tools.
- Version control system (e.g., Git) for managing code changes and collaboration among developers.

7.Monitoring and Management Tools

- Monitoring tools for tracking server performance, network traffic, and system health.
- Backup and disaster recovery solutions to protect against data loss and system failures.

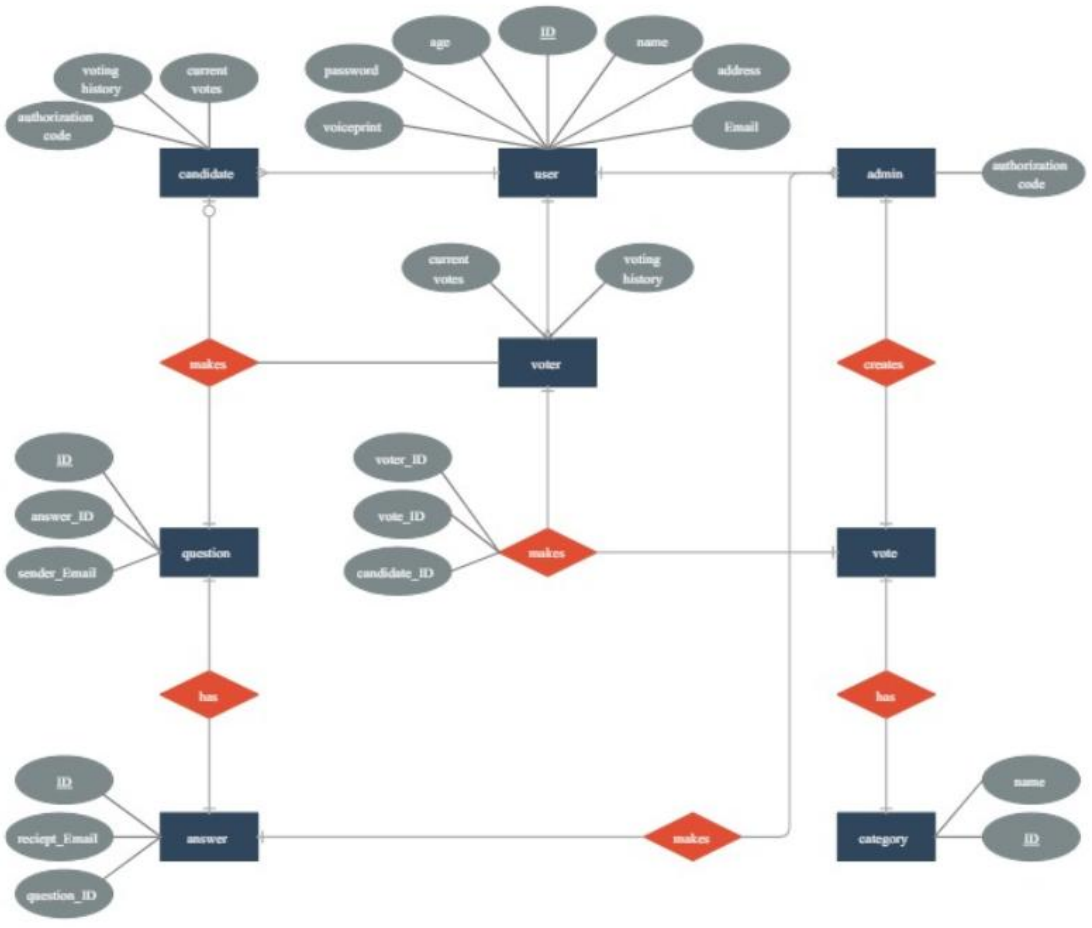
3.3 ARCHITECTURE DIAGRAM

```
            +-----------------------------------+
            |         Client Devices            |
            | (Desktops, Laptops, Tablets, |
            |        Smartphones)               |
            +---------------+-------------------+
                            |
                            |
            +---------------v-------------+
            |          Web Server         |
            |       (Apache/Nginx)        |
            +---------------+-------------+
                        |
             / \        |
            /   \       |
           /     \      |
          /       \     |
         /         \    |
        /           \   |
+-------------v---+  +---v-------------+  +---------------------+
| Authentication |  | Candidate      |  | Voting            |
|   Module       |  | Management     |  | Module            |
+----------------+  |   Module       |  +---------------------+
                +    ---------------    +         |
                          |                  |
            +---------------v-----------v---------------+
            | Results Tabulation Module                 |
            |  Admin Dashboard                          |
            |                                           |
            +-------------------------------------------+
                             |
                             |
            +--------------v-------------+
            |     Database Server       |
            |        (MySQL)            |
            +--------------+-------------+
                            |
            +--------------v--------------+
            |    Database Tables          |
            | Candidates, Parties,        |
            |   Voters, Votes, etc.)      |
            +----------------------------+
```

The architecture diagram illustrates the structure and interactions of the student election system. Users, including voters, candidates, and administrators, interact with the system through client devices, such as desktops, laptops, tablets, and smartphones. These devices communicate with the web server, which hosts the application and handles HTTP requests. The system comprises functional modules like authentication, candidate management, voting, results tabulation, and administration. These modules interact with a database server, storing and managing data related to candidates, parties, voters, and votes. Users access the system's functionalities through a web interface provided by the web server.

Overall, the architecture ensures smooth and secure operation of the election process, from voter registration to result tabulation, with data integrity maintained throughout.

3.4 ER DIAGRAM



This Entity-Relationship (ER) diagram represents a student election system. Here is a brief explanation of its components:

1.Entities and Attributes

- User: Has attributes like ID, name, age, password, address, email, and voiceprint.
- Admin: Has attributes ID and authorization code.
- Candidate: Associated with attributes like voting history, current votes, and authorization code.
- Voter: Linked to current votes and voting history.
- Question: Attributes include ID, answer_ID, and sender_Email.
- Answer: Attributes include ID, receipt_Email, and question_ID.
- Vote: Connected with vote_ID and candidate_ID.
- Category: Has attributes name and ID.

2. Relationships

Makes:

- A user can become a voter, thereby making votes.
- A voter can cast votes, associating with a vote and candidate.
- A candidate, being a user, can submit questions.
- A question can have answers.
- An answer is tied to a specific question.
- Creates: Admins can create votes.
- Has: Votes belong to categories.

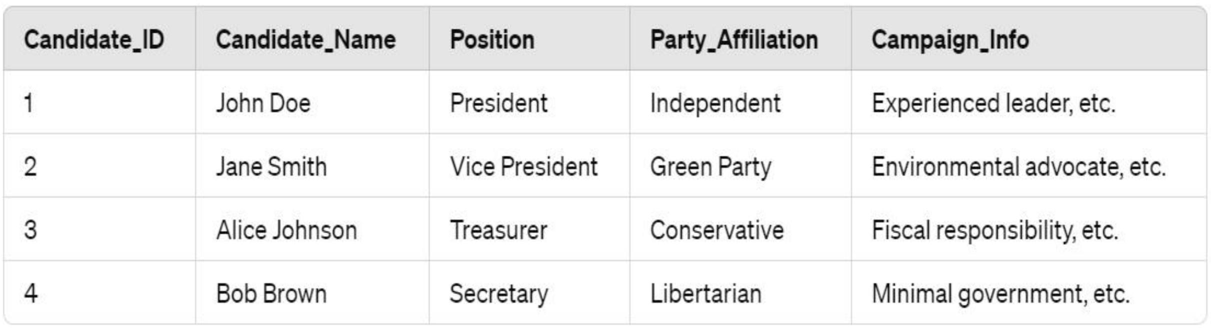
3. Cardinality and Participation

- User can be a voter, candidate, or admin.
- Admin can only create votes.

- Voter relationship indicates that a user must become a voter to participate in voting.
- Candidate relationship suggests users can stand as candidates in the election.
- Questions are made by candidates and can receive answers.
- Answers are linked to specific questions and have associated email information.
- Votes are categorized under specific categories.

3.5 NORMALISATION

CANDIDATE TABLE



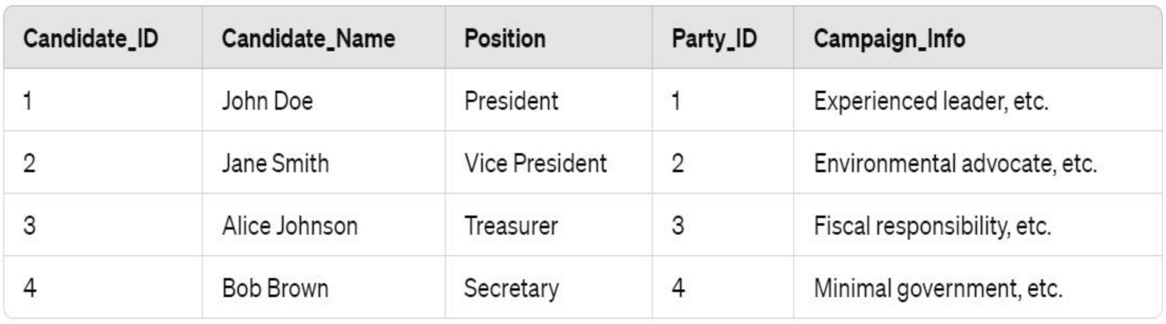
Candidate_ID	Candidate_Name	Position	Party_Affiliation	Campaign_Info
1	John Doe	President	Independent	Experienced leader, etc.
2	Jane Smith	Vice President	Green Party	Environmental advocate, etc.
3	Alice Johnson	Treasurer	Conservative	Fiscal responsibility, etc.
4	Bob Brown	Secretary	Libertarian	Minimal government, etc.

1NF (FIRST NORMAL FORM)

- In 1NF, each column should contain atomic (indivisible) values, and each row should be unique.
- Our table already meets the criteria for 1NF as each cell contains a single value, and there are no repeating groups within the rows.

2NF (SECOND NORMAL FORM)

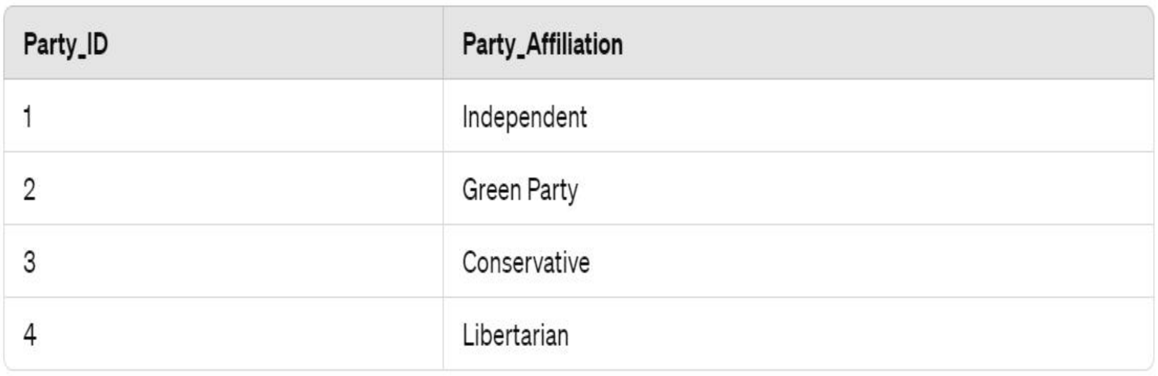
- In 2NF, the table should be in 1NF, and all attributes should be fully functionally dependent on the primary key.
- To achieve 2NF, we need to identify the candidate's primary key and ensure that all other attributes depend on that primary key. In this case, let's assume **Candidate_ID** is the primary key.



Candidate_ID	Candidate_Name	Position	Party_ID	Campaign_Info
1	John Doe	President	1	Experienced leader, etc.
2	Jane Smith	Vice President	2	Environmental advocate, etc.
3	Alice Johnson	Treasurer	3	Fiscal responsibility, etc.
4	Bob Brown	Secretary	4	Minimal government, etc.

- We've removed the non-key attribute Party_Affiliation and created a new table for parties.
- Party_ID now references the primary key of the new parties table.

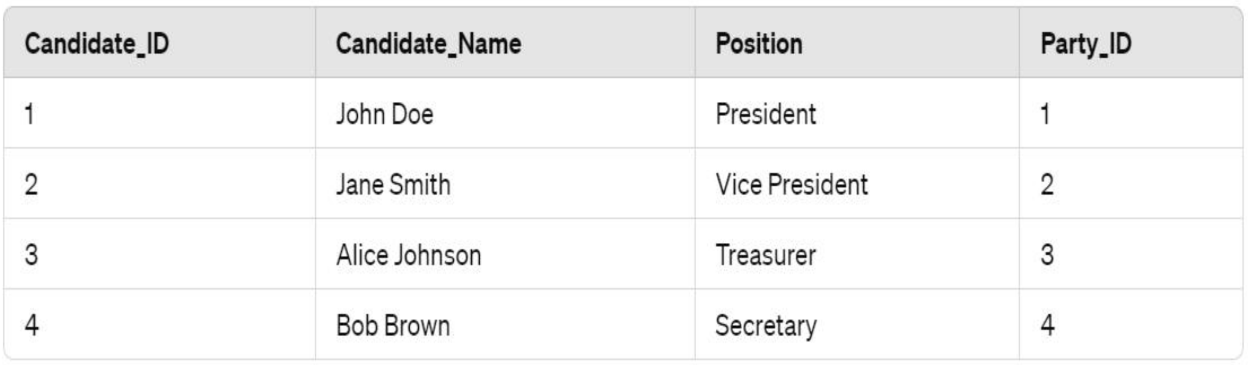
PARTIES TABLE



Party_ID	Party_Affiliation
1	Independent
2	Green Party
3	Conservative
4	Libertarian

3NF (THIRD NORMAL FORM)

- In 3NF, the table should be in 2NF, and there should be no transitive dependencies.
- To achieve 3NF, we need to remove any attributes that are not directly dependent on the primary key.



Candidate_ID	Candidate_Name	Position	Party_ID
1	John Doe	President	1
2	Jane Smith	Vice President	2
3	Alice Johnson	Treasurer	3
4	Bob Brown	Secretary	4

- We've removed the non-key attribute Campaign_Info, as it's not directly dependent on the candidate's primary key.

Now, the table is in 3NF with no transitive dependencies.

This normalization process helps ensure data integrity and reduces redundancy in the database schema.

CHAPTER 4
PROGRAM CODE

ABOUT PAGE (about.php):

```
<!DOCTYPE html>

<html>

<head>

        <title>About |SES</title>

                <linkhref="https://fonts.googleapis.com/css?family=Secular+One"
rel="stylesheet">

  <link rel="stylesheet" type="text/css" href="master.css">

  <style type="text/css">

        .content{max-width: 650px;

                border: 5px solid #e1e1e1;

                padding: 20px;

                text-align: center;

                margin-left: auto;

                margin-right: auto;}

  </style></head><body>

<h1>Student Election System </h1>

<div class="content">

        Student Election System is a online election system, which can be used for normal
class election processes. For example, election of a

        class Representative, Student secretary, and other posts.

        <p>When it comes to Voting by students, this platform will serve the better way. First
you have to have a admin account.There can only be one admin account, and it has superuser
access to database. Next, all the voters have to first register in the forum and then he/she can
cast vote. One can cast only one vote, identified by user's unique ID/ student ID. </p>

        <p>So use it for your real life purposes, and have fun.</p>

        <p>Site wide look and feel can be changed in master.css</p>

</div>
```

```
<br>
<p style="text-align: center; "><a href="index.php" style=" background-color: #000aff;
padding: 8px 25px;">Back</a></p></body></html>
```

DASHBOARD PAGE (dashboard.php):

```
<?php
session_start();
if(!isset($_SESSION['vid']))
{ header("location: voterLogin.php"); }
?>
<!DOCTYPE html>
<link href="https://fonts.googleapis.com/css?family=Secular+One" rel="stylesheet">
<link rel="stylesheet" type="text/css" href="master.css">
<?php
//$id=$_GET['a'];
$id = $_SESSION['vid'];
$db = mysqli_connect("localhost","root","","StudentVote");
$sql = "SELECT * FROM users WHERE id=$id";
$result = mysqli_query($db,$sql);
$row = mysqli_fetch_array($result,MYSQLI_ASSOC);
?>
<html>
  <head>
    <meta charset="UTF-8">
    <title></title>
    <style>
      body{ background-color: #27ace854;  }
      input{  width: 250px;
        padding: 5px;
        margin: 5px;
        border-radius: 10px;
```

```
          }
          hr{  align: center;
             width: 500px;   }
       </style>  </head>  <body><center>
       <h1>Student Election System </h1>
                   <h3>  Voter's Dashboard </h3>    <hr>
        <p     ><b>     Hello,     <span     style="text-transform:     uppercase;"><?php     echo
$row['name']; ?></span></b></p>
                   <p>Student ID: <?php echo $row['studentId']; ?> </p><hr>
                   <?php
                            $voted = $row['voted'];
                            if($voted==1){echo "<b>You have already voted.</b>";
                   }else{echo "You have not voted. Please Vote";
                                     echo "<h2><a href='vote.php'>VOTE HERE</a></h2>";}?>
       <div class="jumbotron">
                   <p></p></div><hr>
       <h3>Update Your Details</h3>
       <form action="update.php" method="post">
       <input   type="text"   placeholder="Enter   your   Registered   Email:"   name="email3"
value="<?php echo $row['studentId']; ?>" readonly >
          <br>
  <input type="text" placeholder=" NEW Name" name="name2" ><br>
  <input type="password" placeholder=" NEW Password" name="pass2" value=""> <br>
 <input type="submit" name="submitUpd" value="Update"> </form><hr>
<h3><a href="logout.php">LOGOUT</a></h3>
 <h3> <a href="index.php">Goto HOME</a></h3> </center> <?php  ?></body></html>
```

ADMIN LOGIN PAGE(adminLogin.php) :

```
<?php
if(!isset($_SESSION['adminLoggedin']))
{ session_start();} ?>
```

```
<!DOCTYPE html>
<html>
<head>
        <title>Admin Login</title>
        <style type="text/css">
                body{text-align: center;}
                .container{margin-top: 10%;
                        border:20px solid #e4e4e4;
                        padding-bottom: 5%;
                        padding-right: 5%;
                        padding-left: 5%;
                        display: inline-block;
                        border-radius: 20px;}
                input{padding:5px;
                        border-radius: 5px;
                        margin:10px;
                        width: 200px;}
                form{border: 5px solid #c1ded0;
                        padding:15px;
                        display: inline-block;
                        border-radius: 10px;}
                h2{color: #ffe300;}
        </style>
        <link href="https://fonts.googleapis.com/css?family=Secular+One" rel="stylesheet">
<link rel="stylesheet" type="text/css" href="master.css">
</head>
<body>
        <h1>Student Election System </h1>
        <div class="container">
                <h2>Login to Admin Panel</h2>
                <form target="" method="post">
                <input        type="text"        name="username"        placeholder="Username"
value=""><br>
                <input        type="password"        name="pass"        placeholder="Password"
value=""><br>
                <input type="submit" name="loginbtn" value="Login"></form></div>
<p><a href="index.php">Go to Home</a></p>
</body>
</html>
<?php
if(isset($_POST['loginbtn'])){
        if($_POST['username']=="admin" & $_POST['pass']=='admin'){
                echo "login Successful";
                echo "<script>alert('login Successful')</script>";
                $_SESSION['adminLoggedin']="ok";
                header("location: result.php");
        }else{echo "<script>alert('Invalid Credentials.')</script>";}} ?>
```

CANDIDATE REGISTRATION PAGE (registerCandidate.php):

```
<!DOCTYPE html>
<html><head>
<meta charset="UTF-8"> <title></title>
 <style>
  body{ background-color: #27ace854;  }
   input{   width: 250px;
            padding: 10px;
            margin: 5px;
            border-radius: 10px;}
        hr{align: center;
            width: 500px;    }
      </style>
<link href="https://fonts.googleapis.com/css?family=Secular+One" rel="stylesheet">
```

```
<link rel="stylesheet" type="text/css" href="master.css"> </head> <body><center>
 <h1>Student Election System </h1>
<h3> <mark> Candidate Registration Portal </mark></h3>
<h3> <a href="index.php">Goto HOME</a></h3> <hr>
 <h3>New Candidate's Details</h3>
 <form action="registerCandidateScript.php" method="post">
 <input type="text" placeholder="Name" name="name" ><br>
   <input type="email" placeholder="Email" name="email"><br>
<input type="text" placeholder="Contact" name="contact"> <br>
<input class="btn btn-primary" type="submit" name="submit"
value="Register"></form><hr> </center><?php  ?> </body></html>
```

VOTER REGISTRATION PAGE(registerVoter.php):

```
<!DOCTYPE html>
<html> <head>
<meta charset="UTF-8">
 <title>Home | Student Voting System</title>
 <link href="https://fonts.googleapis.com/css?family=Secular+One" rel="stylesheet">
<style>
 body{color: white;  }
input{ width: 250px;
          padding: 10px;
          margin: 5px;
         border-radius: 10px;}
 hr{ align: center;
      width: 500px;}</style>
        <link rel="stylesheet" type="text/css" href="master.css"> </head> <body><center>
        <h1>Student Election System</h1>
            <h3>New User Registration </h3>
            <h3> <a href="index.php">Goto HOME</a></h3><hr>
        <h3>New Record Insertion</h3>
        <form action="registerVoterScript.php" method="post">
            <input type="text" placeholder="Name" name="name" > <br>
            <input type="text" placeholder="StudentId" name="sid"> <br>
            <input type="password" placeholder="Password" name="pass"> <br>
            <input type="text" placeholder="Contact" name="contact"> <br>
            <input class="btn btn-primary" type="submit" name="submit"
value="Register"></form> <hr></center><?php  ?> </body></html>
```

VOTER LOGIN PAGE(voterLogin.php) :

```
 <!DOCTYPE html>
<html><head>
<meta name="viewport" content="width=device-width, initial-scale=1">
<link href="https://fonts.googleapis.com/css?family=Secular+One" rel="stylesheet">
<link rel="stylesheet" type="text/css" href="master.css">
<style>
input[type=text], input[type=password] {
  width: 100%;
  padding: 12px 20px;
  margin: 8px 0;
  display: inline-block;
  border: 1px solid #ccc;
  box-sizing: border-box;}
button {  background-color: cyan;
  color: white;
  padding: 14px 20px;
  margin: 8px 0;
  border: none;
  cursor: pointer;
  width: 100%;}

</style><?php
```

```
if(!isset($_SESSION))
{session_start();}?> <center>
<h1>Student Election System</h1>
<h4>A Student Election Campaign Management Portal. Voting Made Easy.</h4></center>
<p><center></head><body><h2>Voter Login Form</h2>
<form action="loginSubmit.php" method="post">
 <div class="parent"><div class="container">
   <label for="uname"><b>Username</b></label>
    <input type="text" placeholder="Enter StudentID" name="uname" required>
    <label for="psw"><b>Password</b></label>
    <input type="password" placeholder="Enter Password" name="psw" required>
            <p><mark><?php if(isset($_GET['error'])){ echo $_GET['error']; } ?></mark> </p>
    <button type="submit" >Login</button>
   <a href="index.php">New User? Register </a></div>  </div></form>
```

VOTER LOGOUT PAGE (logout.php):

```
<?php
session_start();
session_unset();
if(!isset($_SESSION['vid'])){
    header("location: index.php");
}
session_destroy();
header("location: index.php");
?>
```

VOTE PAGE (vote.php):

```
<?php
session_start();
 ?>
```

```
<link href="https://fonts.googleapis.com/css?family=Secular+One" rel="stylesheet"> <?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "studentVote";
echo "<h1>Student Election System </h1>";
echo"<h2>Please vote your candidate.</h2>";
echo "<h2>Registed Candidates are:<br></h2>";
$conn = new mysqli($servername, $username, $password, $dbname);
if ($conn->connect_error) {
   die("Connection failed: " . $conn->connect_error);}
$sql = "SELECT id, name, email FROM candidate";
$result = $conn->query($sql);
?>
<form action='voteCaste.php' method='POST'>
   <table class="table">
<?php
if ($result->num_rows > 0) {
   // output data of each row
   while($row = $result->fetch_assoc()) {
       echo "<tr><label>";
          echo "<input type=\"radio\" name=\"chosen_candidate\" value=\"".$row['id']."\">";
       echo " ID: ". $row["id"]. " ,  Name: ". $row["name"]. " , Email: " . $row["email"] .
"<br><hr>";
       echo "</label></tr>"; }
} else {echo "0 results";}
$conn->close();
?> </table>
<input type="submit" value="Vote Now" class="btn"></form>
<style>
body{backgroud-color:#345;
```

```
        font-family: "Secular One", serif;
        text-align: center;
        max-width: 750px;
        margin-right: auto;
        margin-left: auto;}
    h1{  color: aqua;}
    .btn{
        padding:5px 15px;
        background-color: #00ffd2;
    }
</style>
```

VOTE RESULT PAGE (result.php):

```
<?php
session_start();
if(!isset($_SESSION['adminLoggedin']))
{ header("location: adminLogin.php"); }
 ?>
<head>
<link                                                                    rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"
integrity="sha384-
ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T"
crossorigin="anonymous">
<link href="https://fonts.googleapis.com/css?family=Secular+One" rel="stylesheet">
</head><h1>Student Election System </h1>
<?php
$conn = mysqli_connect("localhost", "root", "", "studentVote") or die(mysqli_error($conn));
$query = "Select id, name, email, voteCount FROM candidate";
$result = mysqli_query($conn, $query) or die(mysqli_error($conn));
$totalRows = mysqli_num_rows($result);
echo "<h2 style='text-align:center;'>Voting Result</h2>";
echo "<h5>Total Number of candidates: ".$totalRows."</h5>";
```

```
$sum2=0;
while ($row2 = mysqli_fetch_assoc($result)){
   $sum2 += $row2['voteCount'];}
echo "<h5>Total Number of casted vote till Now: <b>$sum2</b></h5> <br><br>";
mysqli_data_seek($result, 0);
?><table class="table table-striped table-bordered table-hover">
                    <tr>  <th>ID</th>  <th>Name</th>  <th>Email</th>  <th>Total  Votes
(casted)</th> <th>Vote Percentage</th></tr>
<?php
          if($totalRows>0){while($row = mysqli_fetch_array($result)){
?>
                    <tr><td><?php echo $row['id'];?> </td>
                              <td><?php echo $row['name'];?> </td>
                              <td><?php echo $row['email'];?> </td>
                              <td><b><?php echo $row['voteCount'];?></b> </td>
                              <td><?php                                              echo
round(votePercent($row['voteCount']),2)." %"; ?></td></tr><?php}}
          else{echo "No results. No registered candidate.";}
          function votePercent($castedVote){
                    global $sum2;
                    $percent = $castedVote * 100 /$sum2; // sum2 is total casted vote
                    return $percent;}?></table>
<a href="adminLogout.php" class="btn">LOGOUT</a>
<style type="text/css">
          body{padding-left: 10%;
                    padding-right: 10%;
                    text-align: center;
                    font-family: "Secular One", serif;}
          table{text-align: center;}
          h1 {font-size: 36px;
            font-family: "Secular One", serif;
            color: aqua;}
        .btn{padding:5px 15px;
                background-color: #00ffd2;}}
</style>
```

master.css :

```
* { background: #1c1c1c;
  font-family: "Secular One", serif;
  color: #fff;}
.parent{text-align: center;
}
.container{border:5px solid #343434;
        display: inline-block;
        padding-left: 6%;
        padding-right: 6%;
        margin: 0 auto;}
h1 {  text-align: center;
  font-size: 36px;
  font-family: "Secular One", serif;
  color: aqua;}
a.gradient {color: #65b8a6;
  color: #fff;
  text-decoration: none;}
a.gradient:focus,
a.gradient:hover {background: -webkit-gradient(
    linear,
    left top,
    right top,
    from(#62a07b),
    to(#4f8b89)  );
  background: linear-gradient(to right, #62a07b, #4f8b89);
  -webkit-background-clip: text;
  -webkit-text-fill-color: transparent;
  -webkit-box-decoration-break: clone;
  box-decoration-break: clone;
  text-shadow: none;}
a.underline { color: #fff;}
a.underline { position: relative;
  transition: background-color 0.5s ease;
  -webkit-transition: background-color 0.5s ease;
  color: #fff;
  z-index: 1;
  white-space: nowrap;}
a.underline:hover {background: #613860;}
a.underline:after {content: "";
  width: 100%;
  height: 4px;
  background: #613860;
  position: absolute;
  bottom: -4px;
  left: 0;
  z-index: -1;}
```

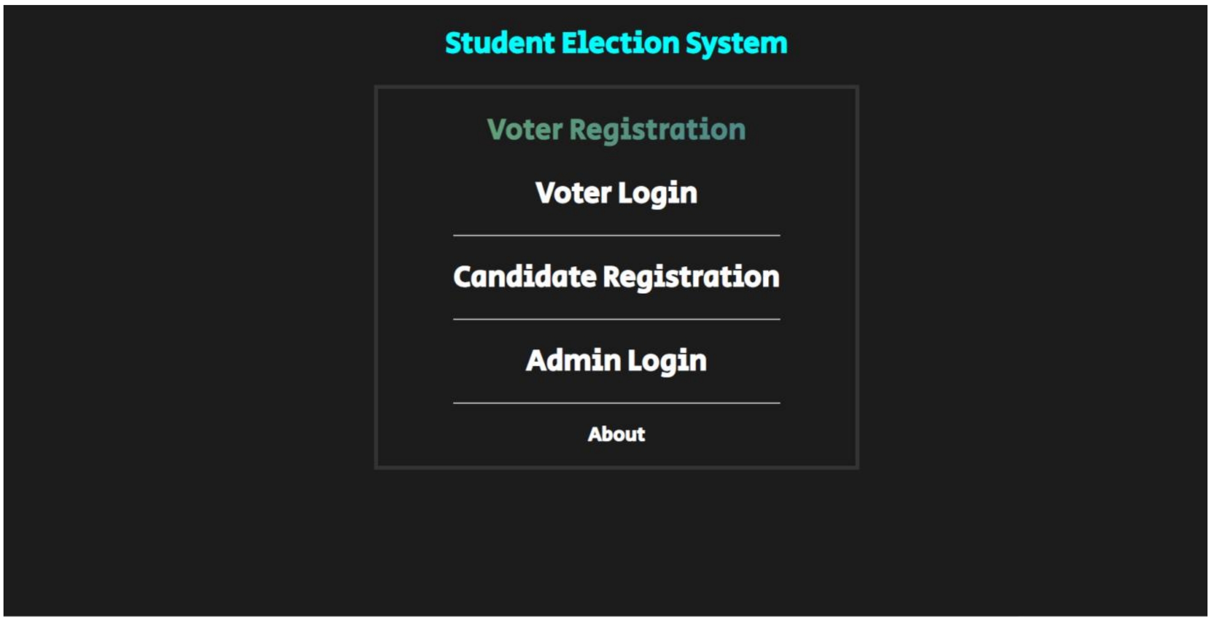
STUDENTVOTE.SQL :

```
SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
SET AUTOCOMMIT = 0;
START TRANSACTION;
SET time_zone = "+00:00";
CREATE TABLE `candidate` (
  `id` int(11) NOT NULL,
  `name` varchar(50) NOT NULL,
  `email` varchar(50) NOT NULL,
  `contact` varchar(50) NOT NULL,
  `voteCount` int(11) DEFAULT '0'
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
INSERT INTO `candidate` (`id`, `name`, `email`, `contact`, `voteCount`) VALUES
(1, 'kishore', 'kishore@gmail.com', '1234567890', 7),
(2, 'kumar', 'kumar@gmail.com', '98654312', 3),
(3, 'harik', 'harik@dgmail.com', '987654', 1),
(4, 'Neelesh Plaparthy', 'neelu@gmail.com', '987654320', 15),
(5, 'len', 'len@gmail.com', '123457783', 1),
(6, 'len', 'len@gmail.com', '123457783', 1);
CREATE TABLE `users` (
  `id` int(11) NOT NULL,
  `name` varchar(50) NOT NULL,
  `studentId` varchar(50) NOT NULL,
  `pass_word` varchar(50) NOT NULL,
  `mobileNumber` varchar(15) NOT NULL,
  `voted` int(5) DEFAULT '0'
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
INSERT INTO `users` (`id`, `name`, `studentId`, `pass_word`, `mobileNumber`, `voted`)
VALUES
(2, 'Hari', '16EI111', '12345', '9876543210', 1),
(3, 'kishore', '123', '1234', '1234567890', NULL),
(4, 'asdf', '321', 'asd', '987465', 1),
(5, 'asdf', '9874', '789987', '987465', 0),
(6, 'kumar', '3210', '12345', '987987', 1),
(7, 'new', 'new', '123', '987654', 1),
(8, 'harikishore', '45', '987', '98765465', 1),
(9, 'hari kishore', '16EI111', '1234', '8300496930', 1),
(10, 'gau', '16IT114', '120', '987456332', 1),
(11, '', '', '', '', 0),
(12, 'SJKASHKA', '16EC159', '123456', '48923409', 1),
(13, 'purushottam banerjee', '16CS138', '123', '12345', 1),
(14, 'subbu kona', '16CS123', '123', '9866441201', 1),
(15, '', '', '', '', 0),
(16, 'hello', '1234', '1234', '1234556776', 0),
(17, 'hello', '1234', '1234', '1234556776', 0),
(18, 'hello', '1234', '1234', '1234556776', 0),
(19, 'hello', '1234', '1234', '1234556776', 0),
(20, 'hello', '1234', '1234', '1234556776', 0),
(21, 'hello', '1234', '1234', '1234556776', 0),
(22, 'hello', '1234', '1234', '1234556776', 0),
(23, 'Raushan', '16EE139', '12345', '987654321', 1);
ALTER TABLE `candidate`
  ADD PRIMARY KEY (`id`);
ALTER TABLE `users`
  ADD PRIMARY KEY (`id`);
ALTER TABLE `candidate`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=7;
ALTER TABLE `users`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=24;
COMMIT;
```

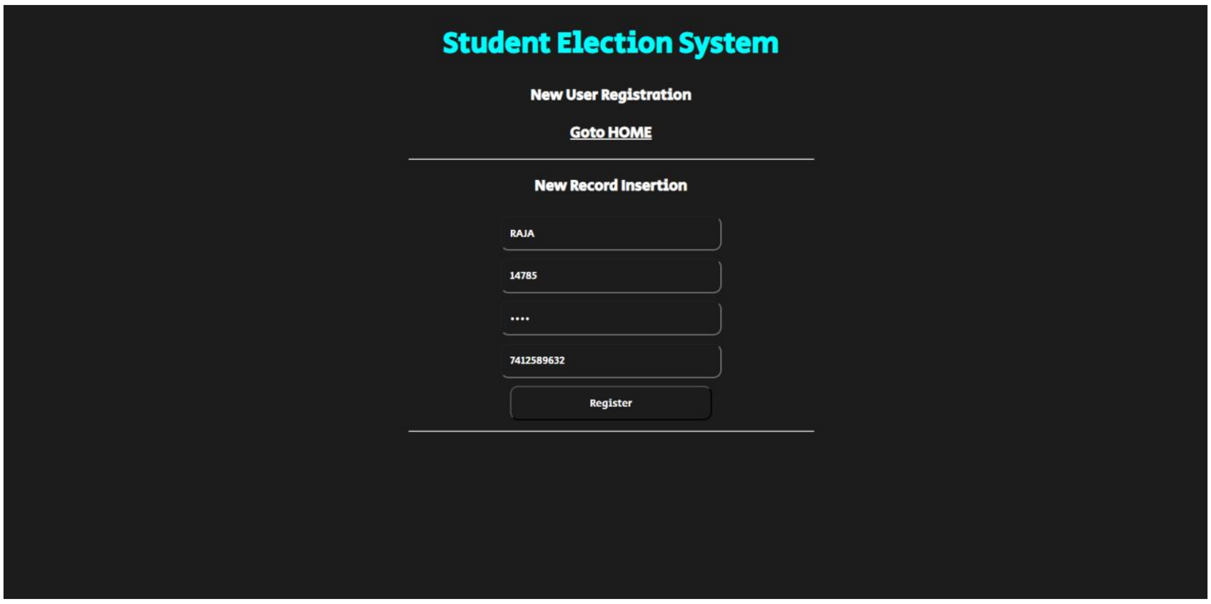
CHAPTER 5

RESULTS AND DISCUSSION

DASHBOARD PAGE :

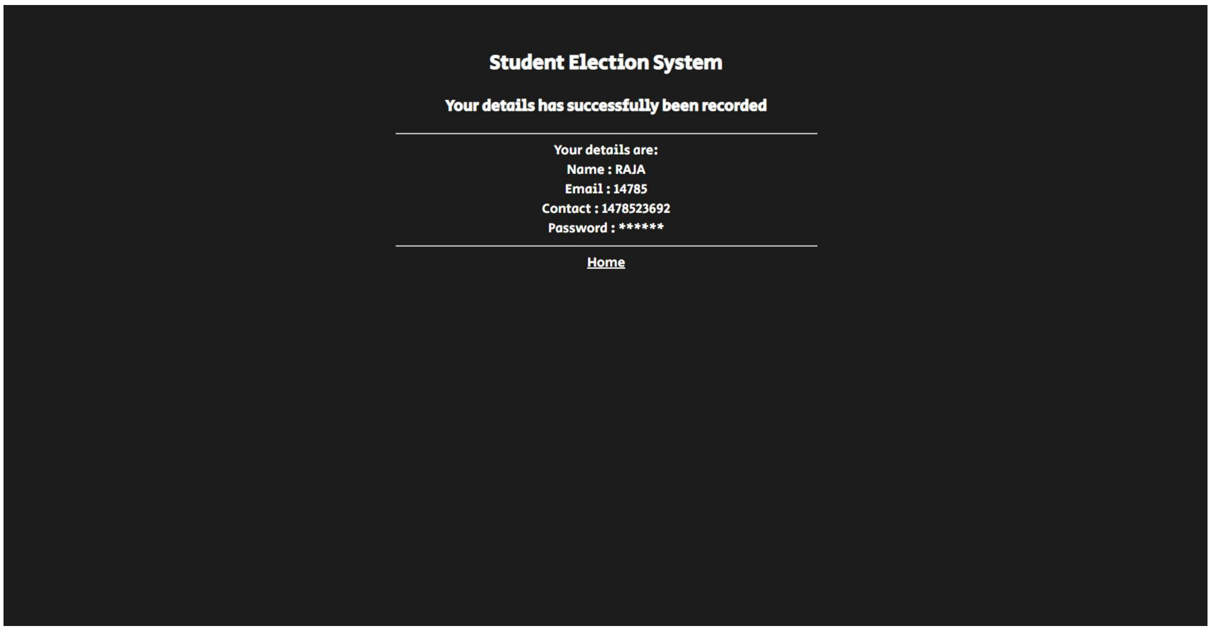


VOTER REGISTERATION PAGE :



RAJA
14785
••••
7412589632

VOTER REGISTRATION COMPLETION:

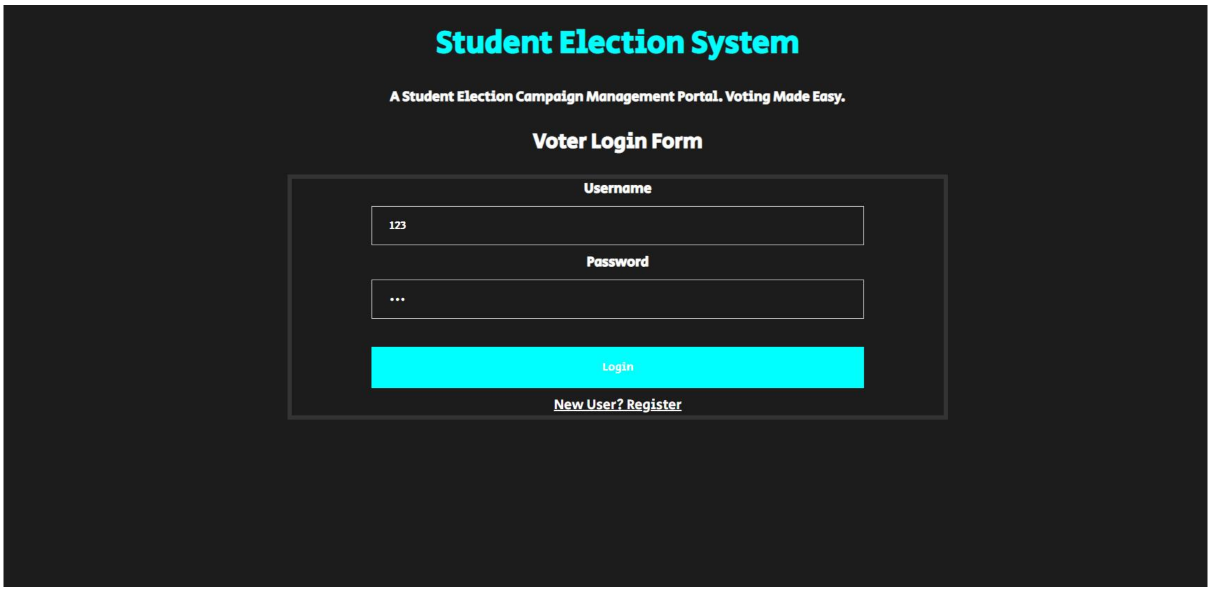


Student Election System

Your details has successfully been recorded

Your details are:
Name : RAJA
Email : 14785
Contact : 1478523692
Password : ******

VOTER LOGIN PAGE :



Student Election System

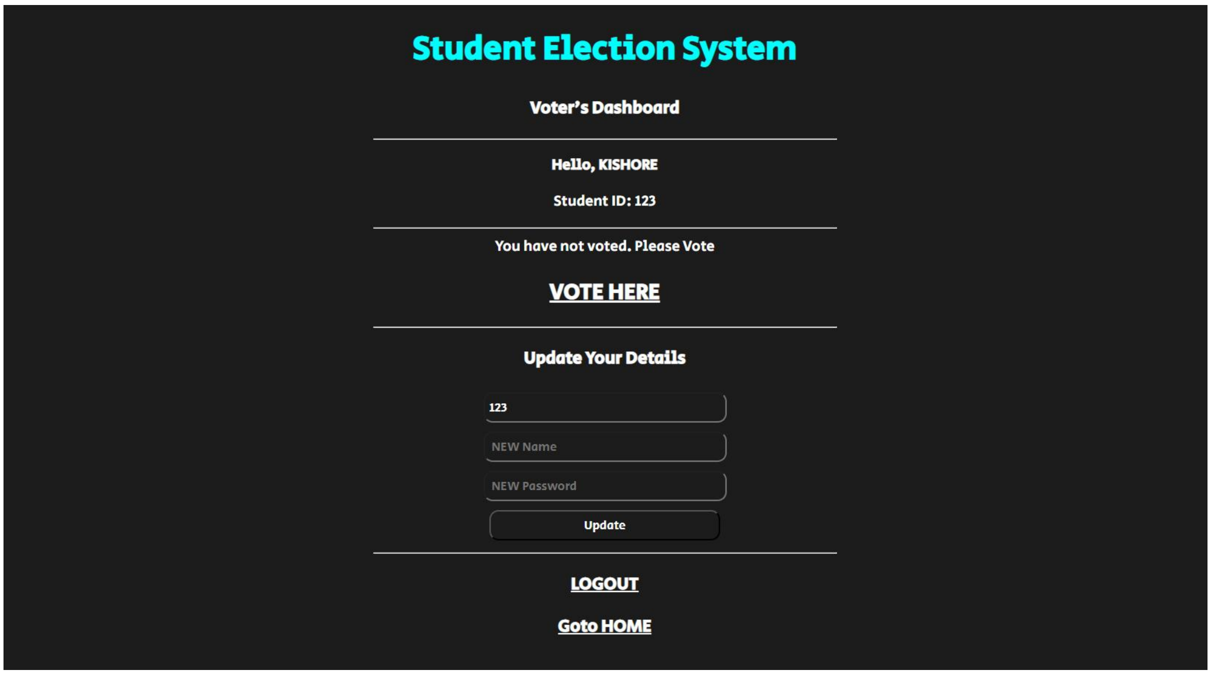
A Student Election Campaign Management Portal. Voting Made Easy.

Voter Login Form

Username: 123

Password: •••

VOTER'S DASHBOARD PAGE :



Student Election System

Voter's Dashboard

Hello, KISHORE

Student ID: 123

You have not voted. Please Vote

VOTE HERE

Update Your Details

123

NEW Name

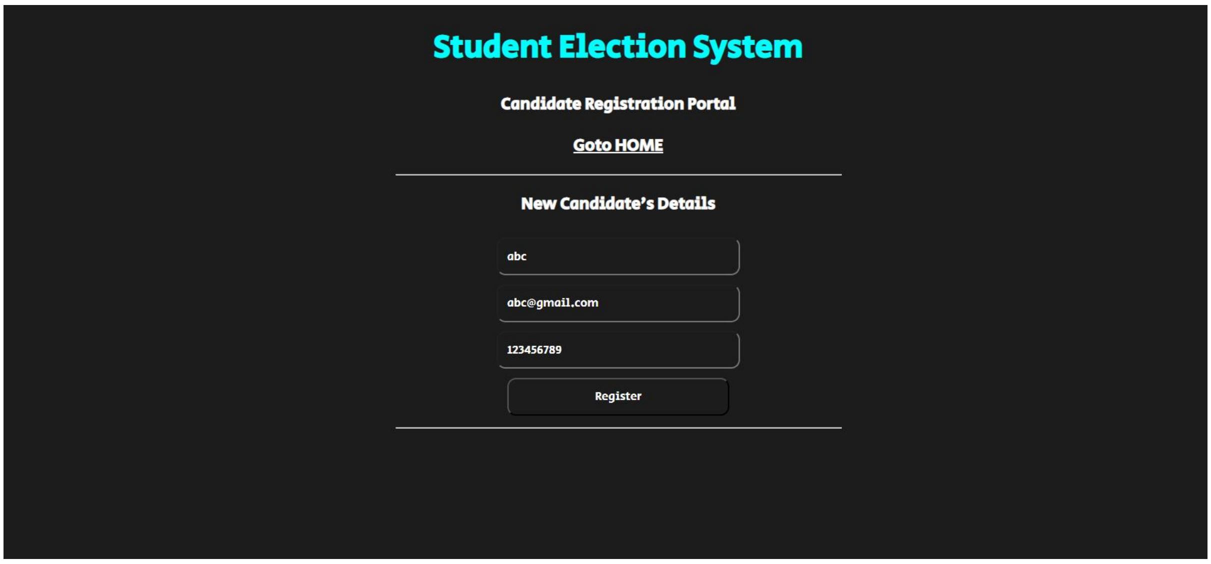
NEW Password

Update

LOGOUT

Goto HOME

CANDIDATE REGISTRATION PAGE :



Student Election System

Candidate Registration Portal

Goto HOME

New Candidate's Details

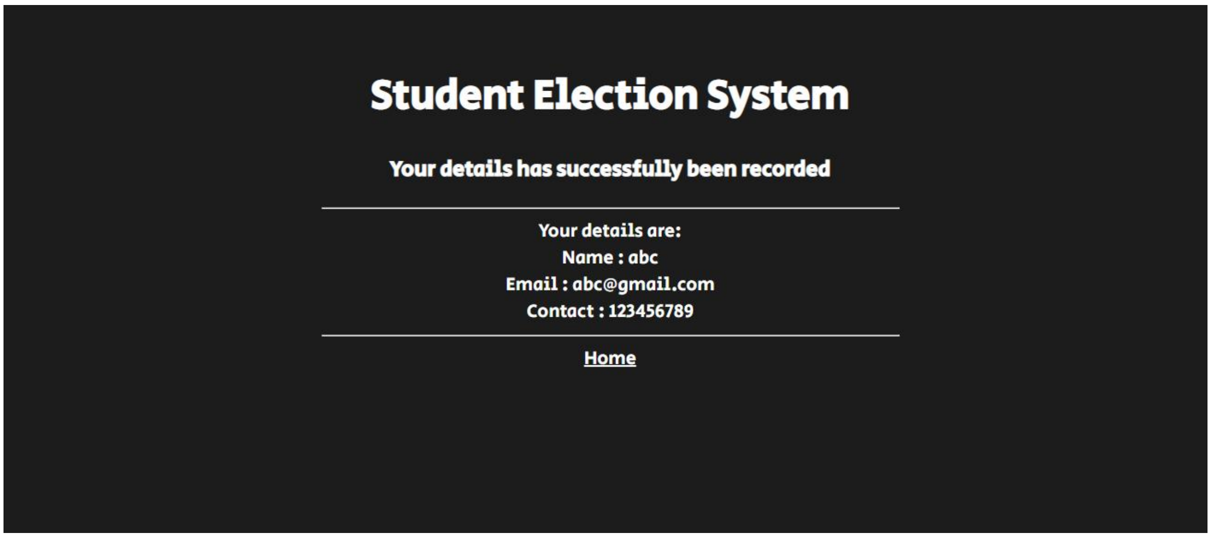
abc

abc@gmail.com

123456789

Register

CANDIDATE REGISTRATION CONFIRMATION PAGE :



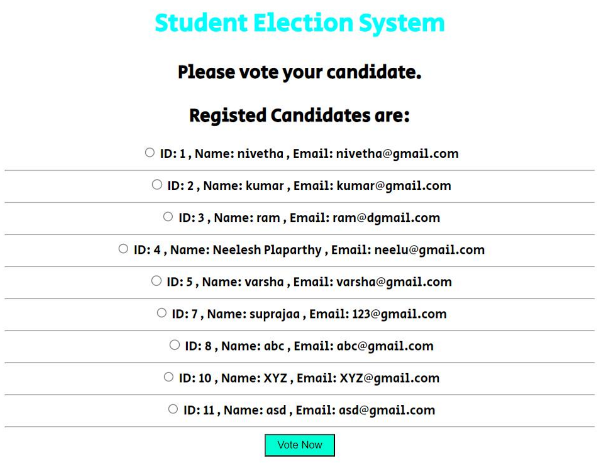
Student Election System

Your details has successfully been recorded

Your details are:
Name : abc
Email : abc@gmail.com
Contact : 123456789

Home

VOTING PAGE :



Student Election System

Please vote your candidate.

Registed Candidates are:

○ ID: 1 , Name: nivetha , Email: nivetha@gmail.com

○ ID: 2 , Name: kumar , Email: kumar@gmail.com

○ ID: 3 , Name: ram , Email: ram@dgmail.com

○ ID: 4 , Name: Neelesh Plaparthy , Email: neelu@gmail.com

○ ID: 5 , Name: varsha , Email: varsha@gmail.com

○ ID: 7 , Name: suprajaa , Email: 123@gmail.com

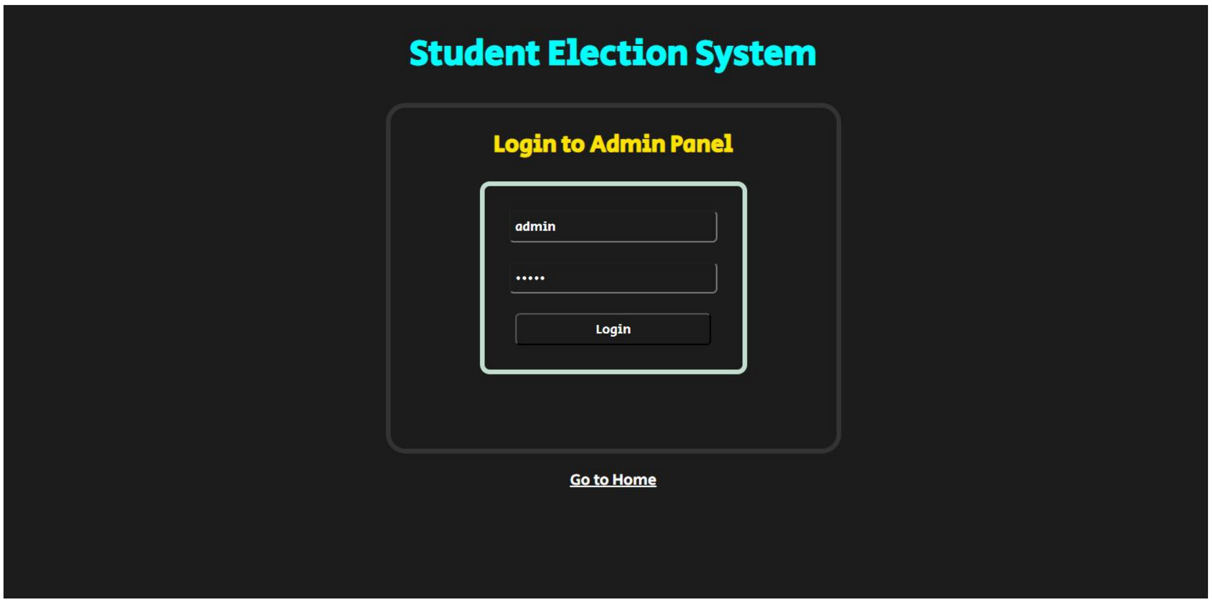
○ ID: 8 , Name: abc , Email: abc@gmail.com

○ ID: 10 , Name: XYZ , Email: XYZ@gmail.com

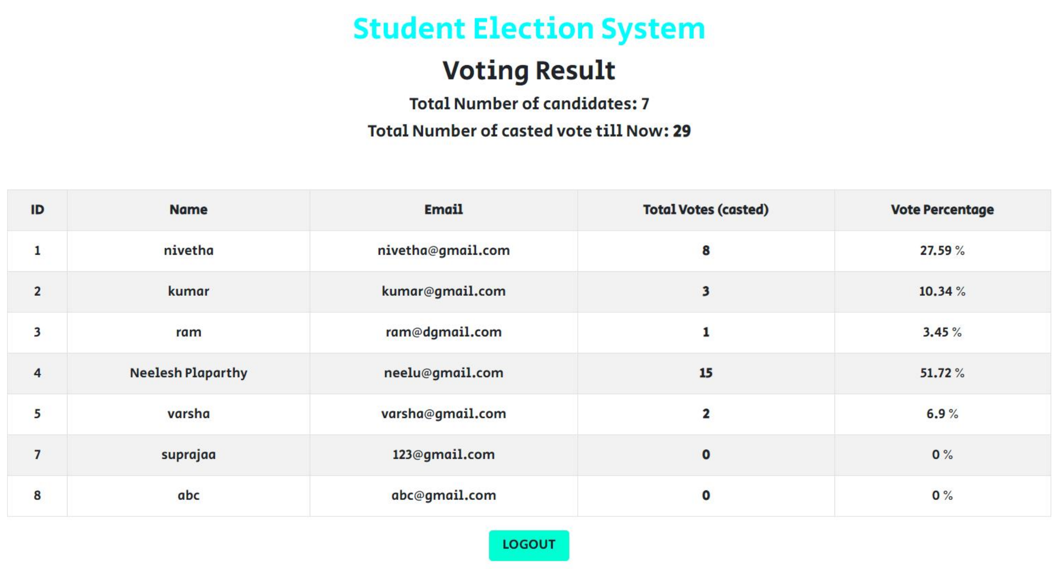
○ ID: 11 , Name: asd , Email: asd@gmail.com

Vote Now

ADMIN LOGIN PAGE :



RESULT PAGE :


Student Election System

Voting Result

Total Number of candidates: 7

Total Number of casted vote till Now: 29

ID	Name	Email	Total Votes (casted)	Vote Percentage
1	nivetha	nivetha@gmail.com	8	27.59 %
2	kumar	kumar@gmail.com	3	10.34 %
3	ram	ram@dgmail.com	1	3.45 %
4	Neelesh Plaparthy	neelu@gmail.com	15	51.72 %
5	varsha	varsha@gmail.com	2	6.9 %
7	suprajaa	123@gmail.com	0	0 %
8	abc	abc@gmail.com	0	0 %

CONCLUSION

The Student Election System project successfully provides a streamlined and efficient platform for conducting student elections online. By leveraging web technologies such as HTML, CSS, JS, PHP, and MySQL, this system ensures a smooth and secure election process from candidate registration to final result declaration.

Key Achievements:

User-Friendly Interface: The system features an intuitive interface, making it easy for students, candidates, and administrators to navigate and perform their respective roles.

Secure Voting Process: Implementing user authentication and secure vote casting ensures the integrity of the election process, preventing fraud and unauthorized access.

Efficient Management: The admin dashboard provides real-time insights into voting statistics and allows for effective management of voter and candidate information.

Comprehensive Documentation: Detailed project documentation, including installation steps and usage guidelines, ensures that users and developers can easily set up and maintain the system.

This project demonstrates the potential of web-based solutions in modernizing traditional processes, offering convenience and reliability. Future enhancements could include additional security features, more robust data analytic, and expanded functionality to handle larger scale elections.

Overall, the Student Election System stands as a robust example of how technology can enhance democratic processes in educational institutions, fostering greater student participation and leadership development.

REFERENCES

WEBSITES :

1. HTML - https://www.w3schools.com/html/

2. CSS - https://www.w3schools.com/css/

3. JAVASCRIPT - https://www.w3schools.com/js/

4. PHP - https://www.php.net/manual/en/index.php

5. MYSQL - https://www.w3schools.com/MySQL/default.asp

6. PHP YOUTUBE LINK - https://www.youtube.com/watch?v=zZ6vybT1HQs

7. MYSQL YOUTUBE LINK - https://www.youtube.com/watch?reload=9&v=HI4NZB1XR9c

8. How to use PHP in HTML - https://www.geeksforgeeks.org/how-to-use-php-in-html/

9. XAMPP - https://www.apachefriends.org/download.html

10. phpMyAdmin -- https://www.phpmyadmin.net/

BOOKS :

1. MYSQL BOOK - Efficient MySQL Performance: Best Practices and Techniques By Daniel Nichter

2. PHP AND MYSQL - Head First PHP & MySQL Paperback by Lynn Beighley, Michael Morrison

3. Learning PHP, MySQL, JavaScript, CSS & HTML5: A Step-by-Step Guide to Creating Dynamic Websites 3rd Edition by Robin Nixon