

Voting Website Project Plan

Group members: Najiyah Dorsey, Darliza Sanchez, and Rapheal Ocran.

Project Overview:

The project entails the creation of an online voting platform that is user friendly and safe. Application users are able to register, log in, vote, and have access to real-time results. This project emphasizes data security, real-time updates, and scalability, assuring a seamless voting process.

Project Scope:

User registration and login functionalities with robust validation and secure password storage.

Voting mechanism: Each user can cast one vote per election. Real-time vote tracking and results display.

Live results visualization using graphical elements. Backend using PHP and MySQL for secure data handling.

Frontend designed for accessibility and ease of use, implemented in HTML and CSS. Optimized MySQL database structure for efficient data retrieval and joins.

Objectives:

Primary Objective: Build a secure platform where users can register, log in, and cast their votes anonymously while viewing live results. Ensure all votes are recorded in real-time without disclosing voter identities. Prevent duplicate votes by implementing database checks. Provide an intuitive user interface to maximize accessibility. Implement secure backend practices to protect sensitive user and voting data.

Key Functional Requirements and Initial Design Plans:

Frontend

Home page: Introduction to the voting and links to key sections.

Registration page: sign up securely.

Login page: Authentication and redirects users to the voting page.

Voting Page: Candidate options and Voting Buttons.

Result Page: Real time visualizations of the voting results.

Backend

Authentication of Users: Check credentials at login.

Recording of Vote: Record votes in the database and reject duplicates.

Results Retrieval: Employ optimized SQL joins that aggregate and present vote counts.

Session Management: Ensure that during sensitive operations, users are indeed logged in.

Database

users table: This holds the credentials of users.

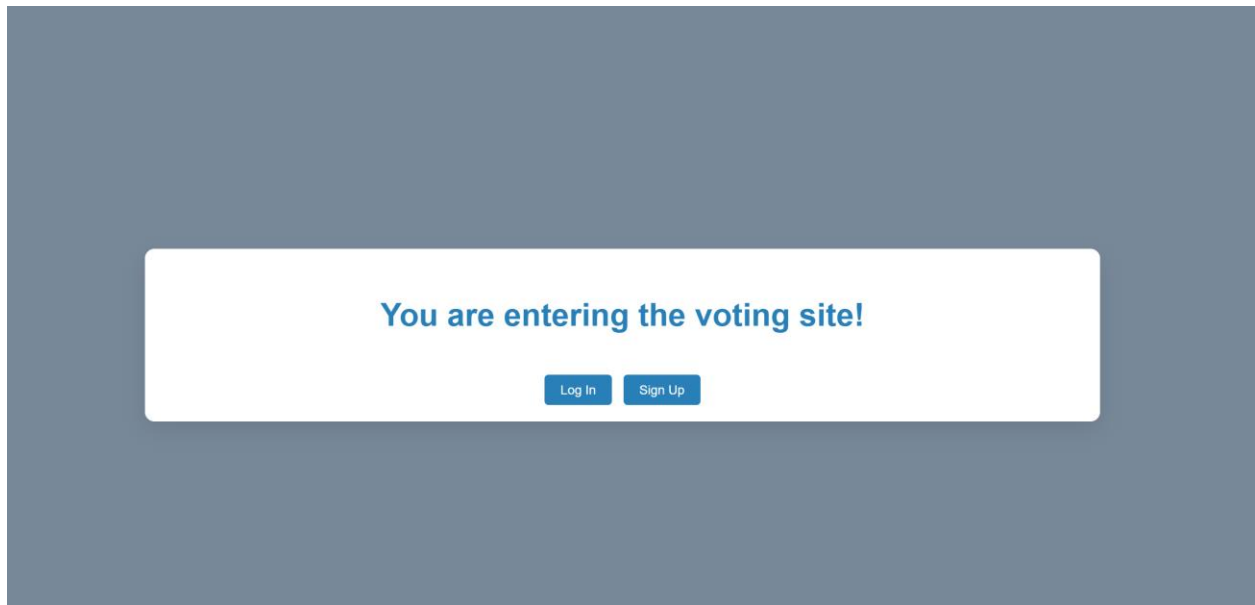
voting table: This would hold votes in anonymity, linked by user IDs.

SQL joins for aggregation of results without revealing voter identity.

User guide

Welcome to the Voting Platform! This guide will help you navigate the application, register, log in, cast your vote, and view results.

First you'll be greeted by the Intropage with options to log in, sign up, or view voting policies.



If you already have an account, you can go ahead and log in if that you can go ahead and sign up.

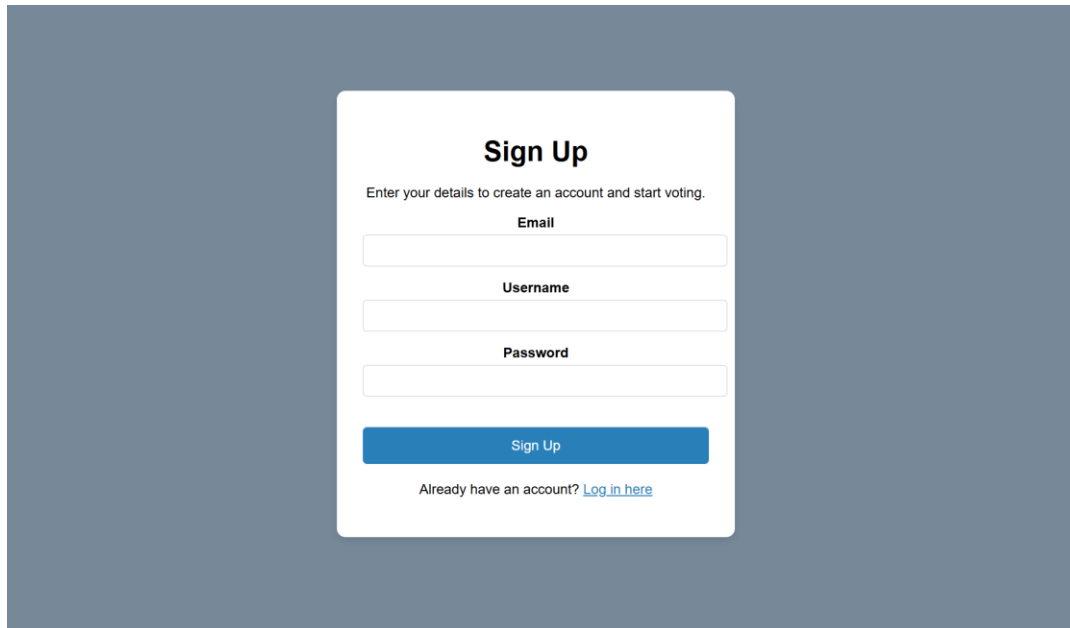
If you're a new user, follow these steps:

Click on the Sign-Up button.

Fill in your email and create a password and username in the registration form. Passwords must be at least 8 characters long.

Click Sign up.

You'll receive a confirmation message indicating successful registration. Return to the log in and use your new credentials.

A sign-up form centered on a dark blue background. The form is white with rounded corners. It has a title 'Sign Up' in bold black text. Below the title is a subtitle 'Enter your details to create an account and start voting.' in a smaller font. There are three input fields: 'Email', 'Username', and 'Password', each with its label above it. Below the input fields is a blue button with the text 'Sign Up' in white. At the bottom of the form, there is a link 'Already have an account? [Log in here](#)' in a smaller font.

Sign Up

Enter your details to create an account and start voting.

Email

Username

Password

Sign Up

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

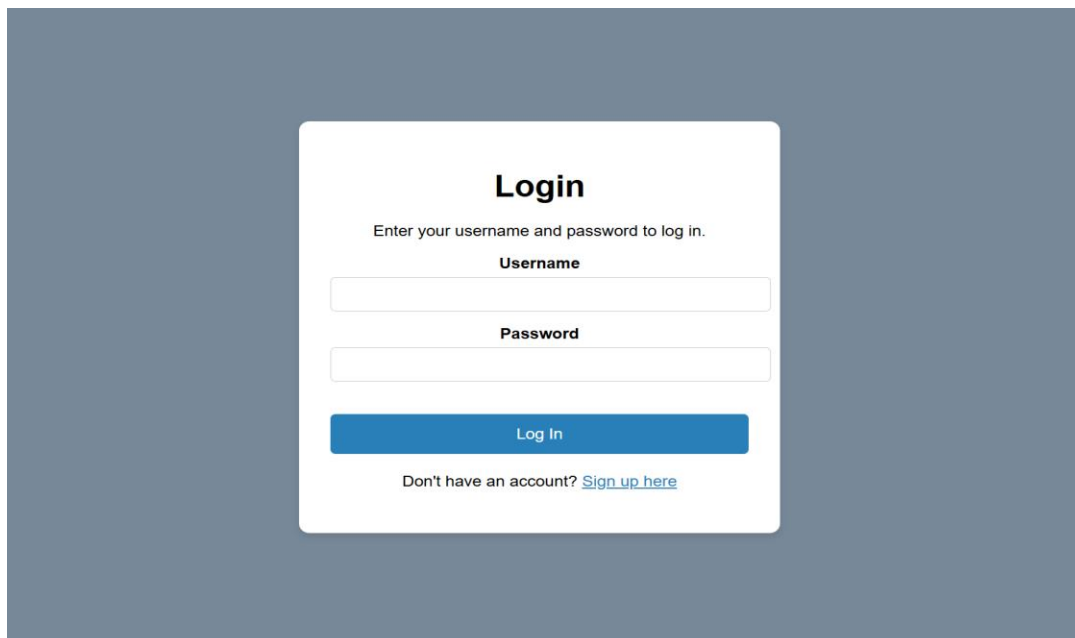
For existing users:

Click on the Log In button.

Enter your registered username and password.

Click Log In.

Upon successful login, you'll be redirected to the voting page.

A login form centered on a dark blue background. The form is white with rounded corners. It has a title 'Login' in bold black text. Below the title is a subtitle 'Enter your username and password to log in.' in a smaller font. There are two input fields: 'Username' and 'Password', each with its label above it. Below the input fields is a blue button with the text 'Log In' in white. At the bottom of the form, there is a link 'Don't have an account? [Sign up here](#)' in a smaller font.

Login

Enter your username and password to log in.

Username

Password

Log In

Don't have an account? [Sign up here](#)


Once logged in:

Review the candidates displayed on the Voting Page.

[Home](#) [Voting Info](#) [Policies](#)


2024 Election

Cast your vote and watch the live results!



Kamala Harris

Vote for Kamala Harris



Donald Trump

Vote for Donald Trump

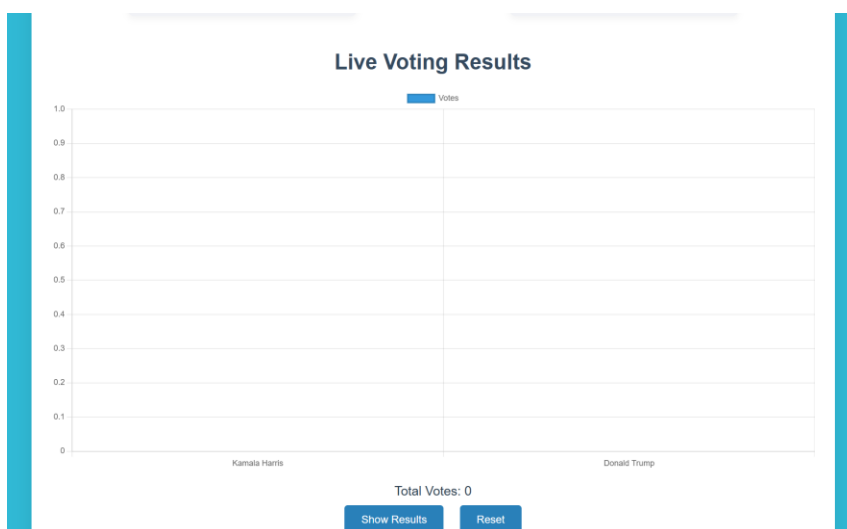
To cast your vote:

Click the Vote button for your chosen candidate.

A confirmation message will appear, indicating that your vote has been recorded.

Note: You can only vote once, and your vote is anonymous.

You can navigate to the live voting results on the same page.




You can use the **Show Results** button to see detailed statistics.

You can also find more information on the given candidates on the Policies tab.

[Home](#) [Voting Info](#) [Policies](#)

2024 Election


Cast your vote and watch the live results!



Kamala Harris

Policies:

- **Tax Reform:** Proposes raising taxes on individuals earning over \$400,000 annually while providing tax credits for first-time homebuyers.
- **Child Tax Credit Expansion:** Plans to increase the child tax credit to \$6,000 for families with newborns, aiming to support low- and middle-income households.
- **Affordable Housing:** Intends to build 3 million new housing units and offer up to \$25,000 in down-payment assistance for first-time homebuyers to address housing affordability.
- **Healthcare:** Supports capping insulin



Donald Trump

Policies:

- **Immigration Reform:** Plans to carry out the largest domestic deportation operation in U.S. history, aiming to remove undocumented immigrants and implement stricter border controls.
- **Tax Policy:** Proposes extending the 2017 Tax Cuts and Jobs Act, reducing the corporate tax rate to 15% for companies manufacturing in the U.S., and eliminating taxes on Social Security benefits.
- **Trade Policy:** - Intends to impose a universal baseline tariff on most foreign goods, with higher tariffs on Chinese imports, to encourage domestic manufacturing and reduce trade deficits.

Thats about it. Thank you for using our Voting Platform! Happy voting!

Interactive Voting System Report

The interactive voting system is a well-structured web application designed to offer users a seamless experience in casting votes and viewing live election results. This report delves into the integration of its key components—HTML, CSS, and JavaScript—highlighting their roles, functionality, and how they work together to create a dynamic and user-friendly platform.

HTML: The Backbone of Content Organization

The HTML files (Homepage.html, Intro.html, and policies.html) form the backbone of the application, providing a well-organized structure for content. The navigation system, consistent across all pages, enables users to switch seamlessly between sections such as "Home," "Voting Info," and "Policies." For instance, the Homepage.html file contains the following navigation bar:

```
<nav class="main-nav">
  <ul>
    <li><a href="intro.html">Home</a></li>
    <li><a href="homepage.html">Voting Info</a></li>
    <li><a href="policies.html">Policies</a></li>
  </ul>
</nav>
```

Each link directs users to specific pages, ensuring smooth navigation throughout the application. This consistent structure is reinforced by styling rules in the style.css file, which ensures the visual uniformity of the navigation bar.

The Homepage.html file serves as the central hub for voting activities, featuring candidate cards with images, descriptions, and voting buttons. The buttons use the onclick attribute to trigger JavaScript functions, such as:

```
<button onclick="vote('A')">Vote for Kamala Harris</button>
```

This connection between HTML and JavaScript facilitates real-time vote tracking. Similarly, the policies.html file provides a detailed breakdown of each candidate's policies, formatted in user-friendly lists. The clear organization of these policies ensures users can easily access and compare the stances of each candidate.

CSS: Enhancing Visual Appeal and Responsiveness

The style.css file brings the application to life by adding visual appeal and ensuring responsiveness across devices. Global styles, such as setting margins, padding, and font

families, establish a clean and consistent aesthetic. For example:

```
* {  
  margin: 0;  
  padding: 0;  
  box-sizing: border-box;  
  font-family: 'Roboto', sans-serif;  
}
```

Interactive elements, such as buttons and candidate cards, feature hover effects that provide visual feedback to users. When users hover over a candidate card, the card slightly lifts, and its shadow intensifies:

```
.candidate-card:hover {  
  transform: translateY(-5px);  
  box-shadow: 0 6px 15px rgba(0, 0, 0, 0.1);  
}
```

This interactivity enhances the user experience, making the interface more engaging. Additionally, media queries in style.css ensure the application adapts gracefully to different screen sizes. For instance, the layout of the voting section changes on smaller screens:

```
@media (max-width: 768px) {  
  .voting-section {  
    flex-direction: column;  
    align-items: center;  
  }  
}
```

These responsive design principles make the application accessible and user-friendly, regardless of the device.

JavaScript: Powering Interactivity

The script.js file, central to the application's functionality, manages vote counting, chart rendering, and real-time updates. When users click a voting button, the vote() function updates the vote counts and refreshes the results:

```
function vote(candidate) {  
  if (candidate === 'A') {  
    votesA++;  
  } else if (candidate === 'B') {  
    votesB++;  
  }  
}
```



```
    updateResults();  
}
```

The `updateResults()` function dynamically modifies the Document Object Model (DOM), ensuring the displayed data reflects the latest vote counts. Additionally, the application employs Chart.js to create a bar chart that visualizes voting results. The chart dynamically updates with each vote, offering users a clear and interactive representation of the election's progress:

```
new Chart(document.getElementById('resultsChart'), {  
  type: 'bar',  
  data: {  
    labels: ['Candidate A', 'Candidate B'],  
    datasets: [{  
      label: 'Votes',  
      data: [votesA, votesB],  
      backgroundColor: ['#2980b9', '#e74c3c']  
    }]  
  }  
});
```

To allow users to restart the voting process, the `resetVotes()` function resets all vote counts and updates the DOM:

```
function resetVotes() {  
  votesA = 0;  
  votesB = 0;  
  updateResults();  
}
```

This integration of JavaScript ensures the application is not only interactive but also responsive to user actions, maintaining accuracy and transparency.

Integration: Creating a Seamless User Experience

The seamless integration of HTML, CSS, and JavaScript creates a cohesive and dynamic user experience. HTML structures the content, providing clear navigation and interactive elements. CSS enhances this structure with visually appealing styles and ensures responsiveness across devices. JavaScript powers the interactivity, enabling real-time updates and visualizations. For instance, when a user casts a vote on `Homepage.html`, the `onclick` attribute triggers a JavaScript function, which updates the vote count...

The user journey begins with the `Intro.html` page, where users can log in or sign up to access the voting system. From there, they navigate to the `Homepage.html` to cast votes and

view live results. They can also explore the candidates' policies in `policies.html`, which provides detailed, well-organized information. This modular design ensures that all components work together harmoniously, creating an intuitive and engaging experience for users.

Conclusion:

The interactive voting system exemplifies a well-designed web application that leverages the strengths of HTML, CSS, and JavaScript. The clear structure provided by HTML, the aesthetic enhancements of CSS, and the dynamic functionality powered by JavaScript come together to create a seamless and engaging platform. This system is not only user-friendly but also scalable and adaptable, making it a robust solution for real-time election tracking. The thoughtful integration of these technologies ensures that.

Signup.php

The signup.php is a PHP code that allows the creation of a secure user sign-up form, handling and validating user input before storing it in a MySQL database.

First The script imports a database connection file to interact with the MySQL database.

```
require 'db_connection.php';
```

Then it Checks if the form was submitted using the POST method.

```
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
```

It moves on to Input Sanitization and Validation which is where the code removes unwanted characters from user input for safety using `filter_var()` sanitizes the email and `htmlspecialchars()` protects against XSS by escaping HTML characters in the username. Then it Ensures the email is in a valid format and Checks that the password is at least 8 characters long.

There is also Password Hashing that Hashes the password using a secure algorithm to prevent storing plaintext passwords.

```
$hashedPassword = password_hash($password, PASSWORD_DEFAULT);
```

Then it proceeds to Prepare and execute a SQL query to check if the email or username is already in use.

```
$stmt = $conn->prepare("SELECT email, username FROM users WHERE email = ? OR  
username = ?");
```

```
$stmt->bind_param("ss", $email, $username);
```

```
$stmt->execute();
```

```
$result = $stmt->get_result();
```

If the email and username are unique, the script inserts the new user into the users table. Prepared statements (`bind_param`) are used to prevent SQL injection.

Logs the user's sign-up event into a `signup_tracker` table.

```
$trackerStmt = $conn->prepare("INSERT INTO signup_tracker (user_id) VALUES (?)");
```

The rest of the page Provides visual feedback to the user about the success or failure of the sign-up process using HTML `<div>` elements styled with inline CSS.

Several challenges had to be overcome in the attempt to make the sign-up system secure and work properly. Prepared statements prevented SQL code injection, password_hash() for secure password storing, and escaping special characters helped to avoid XSS attacks. Validation messages and error handling gave hints on correcting input errors and helped in debugging. Capturing the user ID with \$conn->insert_id in the signup_tracker table was necessary without allowing foreign key constraints to block insertion. The design of a responsive form with inline feedback enhanced its usability, and issues with parameter binding were resolved through thorough debugging of the data types.

Login.php

The following PHP script allows the creation of a secure user login system in the Voting Platform with proper session management, error handling, and security practices. I used chapter 16 from uCerify as a base for both the log in and sign-up portion.

Starts a session to maintain the login state across pages.

```
session_start();
```

Database Connection: This script connects to a MySQL database using default credentials on XAMPP. The 'voting' database holds login information and tracking records. Error Handling: If this connection fails, the script would output an error message via a try/catch block.

```
$servername = "localhost";
```

```
$dbUsername = "root";
```

```
$dbPassword = ""; $dbname = "voting";
```

```
$conn = new mysqli($servername, $dbUsername, $dbPassword, $dbname);
```

Then if (\$_SERVER['REQUEST_METHOD'] === 'POST') { - Handles the form submission when the user submits their login credentials.

```
$username = htmlspecialchars(trim($_POST['username']));
```

```
$password = trim($_POST['password']);
```

Removes unwanted whitespace and encodes special characters to prevent XSS attacks and ensures both username and password fields are filled, displaying an error message if empty.

It also checks if the entered username exists in the database using a prepared statement, preventing SQL injection.

```
$stmt = $conn->prepare("SELECT id, password FROM users WHERE username = ?");  
$stmt->bind_param("s", $username);  
$stmt->execute();  
$result = $stmt->get_result();
```

```
if (password_verify($password, $user['password'])) {
```

Compares the entered password with the hashed password stored in the database using password_verify().

Then it proceeds to Login Tracking. Logs successful login attempts in the login_tracker table with a "Success" status.

```
$logStmt = $conn->prepare("INSERT INTO login_tracker (user_id, status) VALUES (?, 'Success')");
```

Logs failed attempts in the same table with a "Failure" status.

```
$logStmt = $conn->prepare("INSERT INTO login_tracker (user_id, status) VALUES ((SELECT id FROM users WHERE username = ?), 'Failure')");
```

On successful login, session variables store the user's information, and the user is redirected to a secure dashboard or homepage.

```
$_SESSION['user_id'] = $userId;  
$_SESSION['username'] = $username;  
header("Location: homepage.html");
```

```
echo "<div style='color: red; margin-top: 20px;'>Invalid username or password.</div>";
```

Displays an error message if the credentials are invalid.

The rest of the page consists of: The form is styled with CSS for a clean and responsive user interface. Form Fields: This includes fields to input the username and password, and a button to submit the login details and a link guides new users to the registration page if they don't have an account.

Db_connection.php

This PHP code sets up a connection to a MySQL database using the MySQLi extension and provides error handling for debugging.

Database Connection Settings

```
$servername = "localhost";
```

```
$username = "root";
```

```
$password = "";
```

```
$dbname = "voting";
```

```
$conn = new mysqli($servername, $username, $password, $dbname);
```

Establishes a new MySQLi connection to the database using the credentials and database name provided above. \$conn holds the connection object, which is used in subsequent database operations.

```
if ($conn->connect_error) { die("Connection failed: " . $conn->connect_error); }
```

Checks if there is a connection error (connect_error). If an error occurs, the script terminates and displays a detailed error message.

This script establishes a secure and reliable connection to the voting database. It ensures

Error Detection: Any connection issues are instantly identified and reported, and

Debugging: Strict error reporting helps developers troubleshoot database-related errors during development. This script can be included in other PHP files through require or include, such as those handling sign-ups or logins, to provide reusable configuration for the database connection.

Database Setup

Create Database:

SQL code:

```
CREATE DATABASE IF NOT EXISTS voting;
```

This command creates a database named voting if it does not already exist.

Select Database:

SQL code:

```
USE voting;
```

Selects the voting database for subsequent operations.

Transaction and Time Zone Settings

Set SQL Mode:

Sql code:

```
SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
```

Configures SQL mode to disable automatic incrementing of zero values.

Start Transaction:

SQL code:

```
START TRANSACTION;
```

Begins a transaction, ensuring that all operations are executed as a single unit.

Set Time Zone:

SQL code:

```
SET time_zone = "+00:00";
```

Sets the time zone to UTC.

Character Set and Collation

Ensures consistent character encoding using UTF-8:

SQL code:

```
/*!40101 SET NAMES utf8mb4 */;
```

Table Definitions

Users Table:

SQL code:

```
CREATE TABLE IF NOT EXISTS `users` (  
  `id` INT(11) NOT NULL AUTO_INCREMENT,  
  `email` VARCHAR(100) NOT NULL UNIQUE,  
  `username` VARCHAR(50) NOT NULL UNIQUE,  
  `password` VARCHAR(255) NOT NULL,  
  `created_at` TIMESTAMP DEFAULT CURRENT_TIMESTAMP,  
  PRIMARY KEY (`id`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

Stores user information with fields for ID, email, username, password (hashed), and creation timestamp.

Uses InnoDB engine with UTF-8 encoding.

Login Tracker Table:

SQL code:

```
CREATE TABLE IF NOT EXISTS `login_tracker` (  
  `id` INT NOT NULL AUTO_INCREMENT,  
  `user_id` INT NOT NULL,  
  `login_time` TIMESTAMP DEFAULT CURRENT_TIMESTAMP,  
  `status` ENUM('Success', 'Failure') NOT NULL,  
  PRIMARY KEY (`id`),  
  FOREIGN KEY (`user_id`) REFERENCES `users`(`id`) ON DELETE CASCADE  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

Tracks login attempts with fields for ID, user ID, login time, and status.

Foreign key references the users table, cascading deletes.

Signup Tracker Table:

SQL code:

```
CREATE TABLE IF NOT EXISTS `signup_tracker` (  
  `id` INT NOT NULL AUTO_INCREMENT,
```



```
`user_id` INT NOT NULL,  
`signup_time` TIMESTAMP DEFAULT CURRENT_TIMESTAMP,  
PRIMARY KEY (`id`),  
FOREIGN KEY (`user_id`) REFERENCES `users`(`id`) ON DELETE CASCADE  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

Records user sign-ups with fields for ID, user ID, and signup time.
Also references the users table with cascading deletes.

Voting Table:

SQL code:

```
CREATE TABLE IF NOT EXISTS `voting` (  
`id` INT NOT NULL AUTO_INCREMENT,  
`user_id` INT NOT NULL,  
`candidate` VARCHAR(50) NOT NULL,  
`vote_time` TIMESTAMP DEFAULT CURRENT_TIMESTAMP,  
PRIMARY KEY (`id`),  
FOREIGN KEY (`user_id`) REFERENCES `users`(`id`) ON DELETE CASCADE  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

Logs votes with fields for ID, user ID, candidate name, and vote time.
References the users table with cascading deletes.

Policies Table:

SQL code:

```
CREATE TABLE IF NOT EXISTS `policies` (  
`id` INT NOT NULL AUTO_INCREMENT,  
`title` VARCHAR(255) NOT NULL,  
`content` TEXT NOT NULL,  
`last_updated` TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE  
CURRENT_TIMESTAMP,  
PRIMARY KEY (`id`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

Stores voting-related policies with fields for ID, title, content, and last updated timestamp.

Introduction Table:

SQL code:

```
CREATE TABLE IF NOT EXISTS `introduction` (  
  `id` INT NOT NULL AUTO_INCREMENT,  
  `content` TEXT NOT NULL,  
  `last_updated` TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE  
  CURRENT_TIMESTAMP,  
  PRIMARY KEY (`id`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

Contains an overview of the voting platform with fields for ID, content, and last updated timestamp.

Commit Transaction

SQL code:

```
COMMIT;
```

Commits all changes made during the transaction to the database.