Project Scope Document for Frontend Development

Overview

This project is a **voting platform** where **Hosts** (users who organize elections) can create and manage elections. **Voters** will securely log in to cast their votes. The system will provide features such as election creation, email notifications, secure voting, and results visualization.

The **Frontend Developer** will implement a user-friendly interface for the Hosts and Voters, based on the backend API provided.

Key Models and Relationships

User Model

- **Host**: A single host per user.
- **Email**: Matches the host's email and is automatically assigned during account creation.
- Note: Hosts will act as the platform users with dashboards, while the developer will be the superuser/admin.

Host Model

- **Election**: A host can create only one election at a time.
- **Email**: Associated with the host.
- **Date Registered**: Timestamp for when the host account was created.

Election Model

- **Name**: Title of the election.
- **Host**: Linked to a specific host.
- **Candidates**: List of candidates for the election.
- **Is Ended**: Boolean to mark if the election has ended.
- **Is Started**: Boolean to mark if the election has begun.
- **Positions**: Roles available in the election (e.g., President, Vice President).
- **Date Added**: Timestamp for when the election was created.

Candidate Model

- **Name**: Candidate's name.
- **Position**: Linked to one position.
- **Vote Count**: Integer to track the votes received.

Position Model

- **Title**: Name of the position (e.g., President).
- **Candidates**: A position can have multiple candidates.

Voter Model

- **Email**: Voter's email address.
- **Candidate**: The candidate voted for.
- **Is Voted**: Boolean to indicate if the voter has cast their vote.
- **Voting ID**: Unique ID (e.g., matric number).
- **Time Voted**: Timestamp of when the voter cast their vote.

Functionality

Host Features

Sign-Up/Sign-In:

- Hosts will register and log in using their email and password.

Election Creation:

- Create one election at a time with the following details:
  - Election name
  - Candidates and their positions
  - Timeframe for the election
- A button to manually end the election.

Voter Management:

- Add voter details (email and unique ID) manually or by importing a spreadsheet with columns "Email" and "Matric No."
- Generate random passwords for voters.
- Send voter credentials (email, matric number, and password) via predefined emails.

Email Notifications:

- Predefined email templates for:
  - Voter credentials (including election link).
  - Election results.

- Emails distributed through multiple Gmail accounts for load balancing.

Results:

- View election results on the dashboard with bar graphs:
    - Candidate votes for each position.
- Export results to a spreadsheet file.
- Share results via a public link through predefined emails.

Dashboard:

- View election details, results, and voter participation.

Voter Features

Login:

- Login using matric number (Voting ID) and password sent via email.
- Only voters who haven't voted can log in.

Vote:

- Select candidates for each position and submit votes.
- Mark voter as "Voted" after successful submission.

Frontend Requirements

Authentication:

- Implement host and voter login forms.
- Redirect to respective dashboards upon successful login.

Host Dashboard:

- UI for creating elections, managing voters, and viewing results.
- Graphical representation of election results (bar graphs).
- File upload for voter details (spreadsheet support).

Voter Dashboard:

- Simplified UI for voting.
- Alert or notification if the voter has already voted.

Email Notification Integration:

- Display confirmation messages after emails are sent.

Responsiveness:

- Ensure all pages are mobile-friendly.

Result Sharing:

- Public results page accessible through a shared link.

Key Backend Integrations

1. **API Endpoints**:
   - Endpoints for election creation, voter management, and result retrieval.
2. **File Upload**:
   - API to handle spreadsheet imports for voter details.
3. **Email Integration**:
   - REST API triggers for sending predefined emails.

This document outlines the key requirements and scope for the frontend development. Please ensure the UI aligns with the backend API and provides a seamless user experience for both Hosts and Voters.

Here is a comprehensive list of pages required for the frontend development, based on the outlined scope:

Authentication Pages

Host Sign-Up Page

1. Fields: Email, Password, Confirm Password.
2. Includes a link to the login page.

Host Login Page

1. Fields: Email, Password.
2. Includes a "Forgot Password?" link.

Voter Login Page

1. Fields: Matric Number (Voting ID), Password.
2. Includes a notice for voters who've already voted.

Host Dashboard Pages

Dashboard Overview

1. Display a summary of ongoing election. (If non, a create election button and some little overview)

Election Creation Page

1. Fields for election name, timeframe, candidates, and positions.
2. Section for adding positions and candidates dynamically.

Voter Management Page

1. Options to:
    1. Manually add voters (Email and Matric Number).
    2. Upload a spreadsheet to bulk import voter details.
    3. View, edit, or delete voter records.
2. Button to trigger email distribution to voters.

Election Details Page

1. Display all details of a specific election, including:
    1. Candidates and positions and number of votes for each candidates.
    2. Status (Started, Ended, Not Started).
    3. Buttons to manually start or end the election.
2. Email Notification.
    1. Button to send election-related emails (e.g., voter credentials, results).

Results Page

1. Graphical representation of results using bar graphs (e.g., candidates vs. votes).
2. Option to export results to a spreadsheet.
3. Button to share results page via a public link.

Voter Pages

1. **Voting Page**

- Display election details, positions, and corresponding candidates.
- Allow voters to select one candidate per position.
- Button to submit votes with confirmation before final submission.

1. **Vote Confirmation Page**

- Show a preview of voted candidates and success message confirming the vote.
- Include a note that the voter cannot vote again.

Public Pages

1. **Election Results Page (Public)**

- Accessible via a shared link.
- Show election results with graphs and key stats (e.g., total votes, winner for each position).

Error and Utility Pages

1. **Error Pages**

- **404 Page**: For invalid URLs.
- **403 Page**: For unauthorized access attempts.
- **500 Page**: For server errors.

1. **Password Reset Page (Optional)**

- Allow hosts to reset their password if forgotten.
- Fields: Email, New Password, Confirm Password.

1. **Landing Page (Optional)**

- A homepage for the platform (optional if the application is strictly for logged-in users).
- Provide links to sign-up and log-in.

Navigation and Accessibility

- Each page should include navigation elements appropriate to the user type (Host or Voter).
- Mobile responsiveness should be prioritized for all pages.

This structure covers all the required functionalities and ensures a seamless user experience.