Election Meeting System Specifications

Project Overview:

The task is to improve and complete the Election Meeting System, which is a secure voting application for an employee-owned company. The system includes functionality for voting on business proposals, scheduling elections, and securely managing member participation with dynamic membership changes. It is built using Wails, with a Go backend and a Vanilla JavaScript/HTMX/CSS frontend. The system utilizes Shamir's Secret Sharing for voting security, ensuring anonymity and reliable decision-making within the organization.

Project Scope:

The developer will complete and enhance the three-page Wails app, which includes:

1. Login Page: A secure login page for employees to authenticate and access the election system.

2. Survey Page: A page that presents a 7-question survey that forms the basis for voting in election meetings.

3. Election Times and Results Page: A page that displays the scheduled election times, ongoing election events, and the results of past elections.

Detailed Code Specifications:

1. Login Page:

• Implement a secure login mechanism using industry-standard encryption.

• Employees must log in using a unique identifier (e.g., employee ID or email) and a password.

• Add a multi-factor authentication (MFA) option for added security (e.g., email or SMS verification).

• Integrate session management to ensure users are automatically logged out after a period of inactivity.

• Develop error handling and notifications for invalid logins.

2. Survey Page:

• Display the 7-question survey that allows employees to vote on specific topics, such as:

1. Nomination and removal of business owners.

2. Scheduling of upcoming elections.

3. Proposals for company spending and software features.

4. The reconstruction threshold for decision-making.

• Ensure that each survey response is encrypted using Shamir's Secret Sharing.

• Implement client-side validation to ensure that each question is answered properly.

• Develop a mechanism to allow employees to review and confirm their responses before submission.

• Securely submit survey data to the Go backend, ensuring anonymity through the use of distributed shares.

• Automatically add any proposed questions, features, or changes in voting thresholds to the next meeting agenda, ensuring an iterative process for improvement.

3. Election Times and Results Page:

• Display upcoming election schedules with times in a table format, supporting internationalization and time zone adjustments.

• Develop a feature for real-time updates, showing when an election is actively taking place.

• Add a results section that displays the outcomes of past elections, ensuring all results are displayed anonymously and securely.

• Implement a mechanism to update election timing and allow admins to adjust the schedule.

• Integrate visual indicators to show whether an election has met the reconstruction threshold (based on the number of votes required for a decision).

• Adjust the meeting times dynamically based on what has been voted on, ensuring that election schedules reflect the decisions made by the participants.

• Ensure that the calendar view underlines the number representing the meeting date.

• On days when a survey/election is not being held, display the content from https://romogi.com/EMPLOYEE-OWNER-VOTING/election-times-and-results.html.

• On election days, display the content from https://romogi.com/EMPLOYEE-OWNER-VOTING/survey.html.

• Modify the existing time zone button so that it correctly adjusts the displayed time zone, ensuring that users see the correct times on the calendar and clocks without needing to think in international time.

NOTE: Each web page must have the Romogi logo as the favicon, and the app must have the Romogi icon as the button to press.

NOTE: It needs to look good on mobile and desktop/laptop computers. And must be available on Windows and Linux (Debian).

4. Backend Development (Go):

• Develop a membership management system that allows for adding or removing members dynamically. This will involve adjusting the Shamir's Secret Sharing scheme accordingly.

• Implement the Shamir's Secret Sharing algorithm for secure, anonymous voting. Each vote should be split into shares and distributed amongst voting members, ensuring privacy and security.

• Manage changes to voting thresholds when membership changes, ensuring the system remains consistent and secure.

• Implement endpoints for the login, survey submission, and fetching election schedules/results.

• Ensure all data is stored securely, including encrypted user credentials, survey responses, and election results.

5. Frontend Development (Vanilla JS/HTMX/CSS):

• Enhance the user interface for accessibility and ease of use, ensuring that it follows a monochrome, high contrast design.

• Utilize HTMX to dynamically load survey questions, election results, and other content without requiring full page refreshes.

• Improve the overall design to make sure input fields are grayed out until ready to be edited, with clear placeholder values that are removed upon interaction.

• Ensure the UI elements are responsive, providing a seamless experience across different devices (desktop, tablet, and mobile).

Other Requirements:

• Testing: Develop unit and integration tests for both backend and frontend components. Ensure all core features are tested, including login security, voting, encryption, and dynamic membership management.

• Documentation: Provide clear and concise documentation, including:

- Installation and deployment instructions.

- How to manage and configure membership changes.

- Code-level documentation for critical functions (e.g., the Shamir's Secret Sharing implementation).

Tools and Technologies:

• Wails for the app framework.

• Golang (Go) for backend development.

• Vanilla JavaScript/HTMX/CSS for the frontend.

• Shamir's Secret Sharing for secure voting and dynamic threshold adjustment.

Deliverables:

• Completed and enhanced three-page app as per the scope.

• A comprehensive testing suite covering all features.

• Full documentation for deployment, usage, and future maintenance.

The hired developer is expected to provide weekly progress updates, along with a roadmap detailing milestones for each phase of development.