ReactJs Test

Create a full-featured admin panel using React with Vite, Redux Toolkit, and a design tool of your choice (Ant Design, Material UI, or Tailwind CSS). The candidate will also implement internationalization with i18next, create utility functions, and perform CRUD operations with a mock API.

Test Instructions:

1. Create an Admin Panel

- Choose a design tool, such as Ant Design, Material UI, or Tailwind CSS.
- Use React, Vite JS, and Redux Toolkit to manage data.
- Implement i18next for internationalization.
- Create utility functions, constants, and helpers as needed.

2. Basic Admin Login, Register, and Forgot Password Pages

- Create basic forms for Login, Register, and Forgot Password with validation.
- Implement the required functionality for these forms (mock authentication).

3. Navigation

- Add a navigation bar with links to the Dashboard, Project, and Estimation modules.

1. Dashboard Module

- Implement the Dashboard using your best knowledge and experience.
- Display relevant information and use appropriate charts or data displays.

2. Project Module

- Implement CRUD operations for projects.
- Add filters for project listing.
- Use a mock API for data fetching and updating.

3. Estimation Module

- Implement CRUD operations for estimations.
- Add filters for estimation listing.
- Use a mock API for data fetching and updating.

ReactJs Test

Estimation consists of multiple sections and items. To manage these:

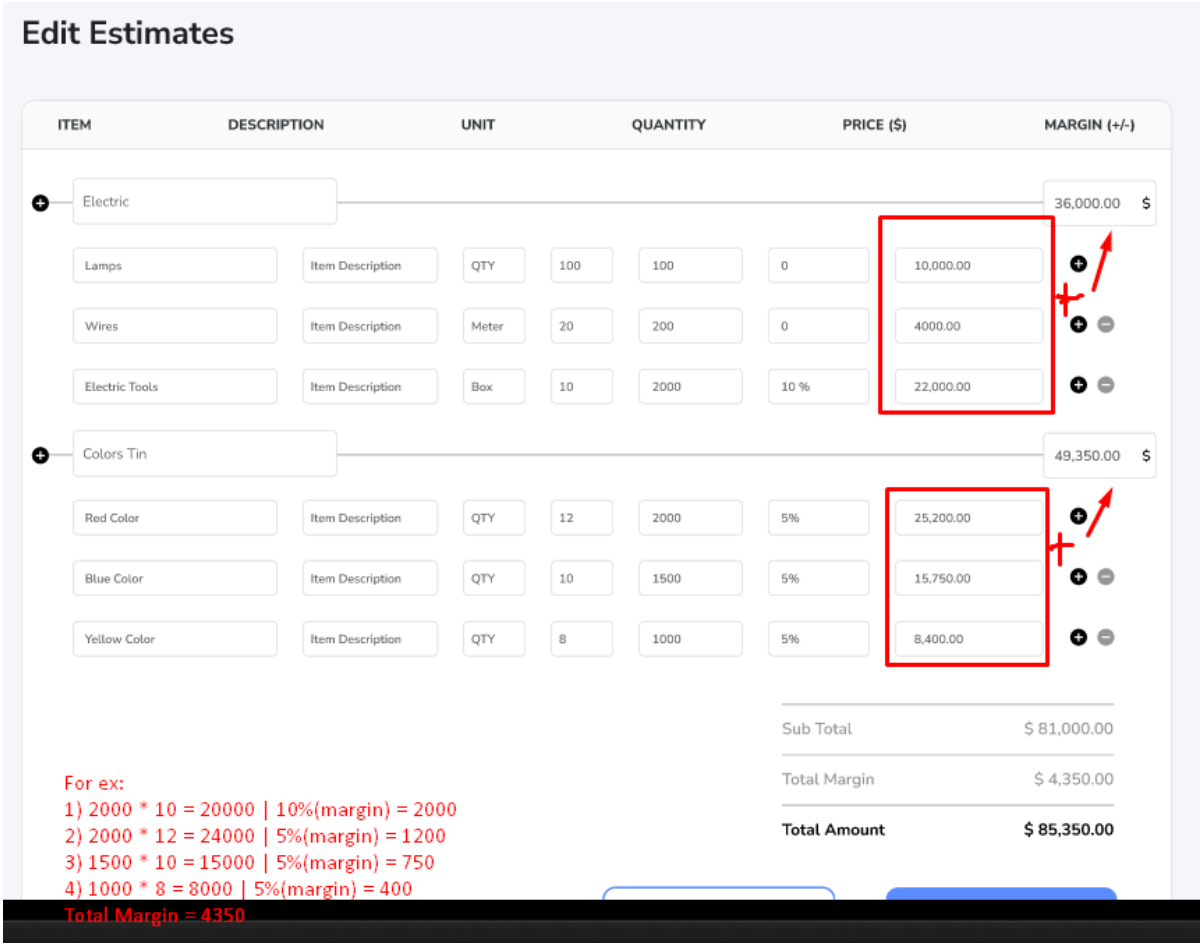
- **Add Sections and Items**: Use the **plus (+)** icon to add sections or items, and the **minus (-)** icon to remove them.

Each **Item** includes the following fields:

- **Title**
- **Description**
- **Unit**
- **Quantity**
- **Price**
- **Margin** (in percentage)
- **Total**

Total Calculation:

- **Item Total** = (Quantity × Price) + (Margin percentage of (Quantity × Price))



ReactJs Test

4. Create Utils, Constants, and Helpers

- Organise the project by creating separate folders for utils, constants, and helpers.
- Write reusable utility functions and constants for the application

5. Internationalization with i18next

- Set up i18next for multi-language support.
- Create language files and integrate them into the application.

6. Create README.md

- Create a detailed README.md file with the following sections:
    - Project setup and installation instructions.
    - Description of the project structure.
    - Explanation of implemented features.
    - Instructions for running the mock API.
    - Notes on design choices and any other relevant information.

Evaluation Criteria

- **Project Structure:** Proper organization and modularity of the code.
- **Functionality:** Implementation of required features and adherence to the provided specifications.
- **Code Quality:** Clean, readable, and maintainable code.
- **Design:** Consistency with the chosen design tool and adherence to the Figma designs.
- **Documentation:** A Comprehensive and clear README.md file.
- **Innovation:** Any additional features or enhancements beyond the basic requirements.

Submission

- Share the GitHub repository link containing the completed project.
- Ensure the repository includes the README.md file with all required details.

Additional Notes

- This test aims to evaluate both technical skills and the ability to follow specifications and documentation.
- Feel free to reach out for any clarifications or additional information required for completing the test.

ReactJs Test

- **Reminder:** This test does **not require** any backend development (e.g., Node.js, Express) or database setup (e.g., MongoDB, PostgreSQL). Use only a **mock API** for demonstrating CRUD operations.

- Please refer to the following Figma link: https://www.figma.com/design/Xe9n7Ot7Rsg0tlJ1Qbj7py/ReactJS-Practical-Test-Design?node-id=0-1&m=dev&t=d9hwhpG2P9448Nir-1