Front-End Developer Assessment Task Requirements Document

Project Overview:

The task is to build a financial dashboard application with multiple views for the user. The application should have responsive design and functionality as shown in the provided **Figma Link**. The main purpose of this application is to display an overview of financial activities, card details, transactions, statistics, and user settings.

Pages and Key Components:

1. **Dashboard Page (Overview):**
    - **My Cards Section:**
        - Display multiple card details with card balance, cardholder name, and card number (partially masked).
        - Include a "See All" button for navigating to a full list of cards.
    - **Recent Transactions:**
        - Display a list of recent transactions with an icon, transaction description, date, and amount (both positive and negative amounts).
    - **Weekly Activity Chart:**
        - A bar chart showing daily deposit and withdrawal activities for the week.
    - **Expense Statistics:**
        - A pie chart showing a breakdown of expenses by category (e.g., Entertainment, Bill Expenses, Investments, Others).

- **Quick Transfer:**
    - Display a list of frequent transfer contacts with profile pictures, names, and roles.
    - Allow input of transfer amount and a "Send" button for initiating transfers. (UI Only)
- **Balance History Chart:**
    - A line chart representing the balance trend over the past few months.

2. **Settings Page:**
    - **Tabs:**
        - "Edit Profile," "Preference," and "Security" tabs should be present.
    - **Edit Profile Section:**
        - Allow users to edit fields such as Name, Username, Email, Password, Date of Birth, Present Address, Permanent Address, City, Postal Code, and Country.
    - Provide a profile picture upload/edit feature.
    - Include a "Save" button to apply changes.

Functional Requirements:

1. **Responsive Design:**
    - The application should be fully responsive and adapt to different screen sizes (mobile, tablet, and desktop).
2. **Data Visualization:**
    - Implement the charts using a library like Chart.js or D3.js for dynamic and interactive charting.
    - The charts should dynamically update based on user data.
3. **Interactive Elements:**
    - Ensure that buttons (e.g., "See All," "Send," "Save") have hover effects and appropriate feedback when clicked.
    - The card and transaction lists should be scrollable if there are multiple entries.

4. **Form Validations:**
    - Implement form validation for the settings page, ensuring fields like email and password adhere to standard input formats.
5. **User Experience:**
    - Provide smooth transitions between different sections and tabs.
    - Use icons consistently across the application to enhance the visual hierarchy.

Technical Requirements:

1. **Frameworks/Libraries:**
    - Use React.js as the primary framework for the front end.
    - Utilize state management tools like Redux or Context API for handling state across the application.
    - Use a CSS framework like TailwindCSS or styled-components for styling.
2. **API Integration:**
    - Assume dummy API endpoints to fetch user information, card details, transaction history, and chart data. Mock these endpoints if necessary.
3. **Routing:**
    - Implement React Router for navigation between the "Dashboard" and "Settings" pages.
4. **Charting Library:**
    - Use Chart.js, D3.js, or a similar library for rendering the charts shown in the screenshots.
5. **Version Control:**
    - Maintain the project in a Git repository with a clean branching strategy.

Non-Functional Requirements:

1. **Performance:**
    - Ensure the application is optimized for performance, with lazy loading for components where applicable.

2. **Accessibility:**
    - Make sure the application adheres to basic accessibility standards (e.g., ARIA labels, keyboard navigation).
3. **Browser Compatibility:**
    - Ensure the application works seamlessly across major browsers (Chrome, Firefox, Safari, Edge).

Submission Guidelines:

- Provide a link to the Git repository with the source code.
- Include a README file with setup instructions and any assumptions made during development.
- The application should be deployable locally with minimal setup steps.
- Provide a Vercel live demo link for the website