Financial Dashboard

A responsive financial dashboard application built with Next.js, featuring an overview of financial activities, card details, transactions, statistics, and user settings.

Table of Contents

Project Overview

This project is a front-end financial dashboard application designed as part of a Front-End Developer Assessment. The application provides users with a comprehensive view of their financial data, including card details, recent transactions, weekly activity charts, expense statistics, quick transfer functionality, and balance history. It also includes a settings page with profile editing capabilities. The design is responsive and adheres to the specifications provided in the Figma design link.

Features

- **Dashboard Page:**
  - **My Cards:** Displays card details (balance, holder name, partially masked number) with a "See All" navigation button.
  - **Recent Transactions:** Lists transactions with icons, descriptions, dates, and amounts (positive/negative).
  - **Weekly Activity Chart:** Bar chart showing daily deposits and withdrawals.
  - **Expense Statistics:** Pie chart breaking down expenses by category.
  - **Quick Transfer:** UI for selecting frequent contacts and entering transfer amounts.
  - **Balance History Chart:** Line chart showing balance trends over months.
- **Settings Page:**
  - Tabs for "Edit Profile," "Preference," and "Security."
  - Profile editing with form validation and profile picture upload.
- **Responsive Design:** Adapts to mobile, tablet, and desktop screens.
- **Interactive Elements:** Hover effects, scrollable lists, and smooth transitions.
- **Data Visualization:** Dynamic charts using Chart.js.

Technologies Used

- **Framework:** React.js with Next.js (for "use client" directive support)
- **State Management:** Redux
- **Styling:** TailwindCSS

- **Routing:** React Router / navigation
- **Charts:** Chart.js, d3, recharts
- **Form Handling:** React Hook Form with Yup validation
- **Animations:** Framer Motion
- **Version Control:** Git
- **Deployment:** Vercel (live demo)

Setup Instructions

To set up the project locally, follow these steps:

1. **Clone the Repository:**

   ```
   git clone <https://github.com/uche7/soar-assessment.git>
   cd soar-assessment
   ```

Install Dependencies:

Ensure you have Node.js (v16 or higher) installed, then run: bash

- npm install

This will install all required packages, including React, Redux, TailwindCSS, Chart.js, React Hook Form, Yup, Framer Motion, and others listed in package.json.

Environment Setup:

- No additional environment variables are required for local development as mock data is used instead of real API calls.

Running the Application

Start the Development Server:

- npm run dev The application will be available at http://localhost:3000.

Build for Production:

- npm run build This creates an optimized production build and serves it.

Assumptions

During development, the following assumptions were made:

1. Mock Data: Since no real API was provided, dummy data is used for cards, transactions, and chart visualizations. This data is hardcoded in the components or mock files.

2. API Integration: Mock endpoints are assumed (/api/cards, /api/transactions, etc.) implemented with redux. In a real scenario, these would be replaced with actual API calls.
3. Chart Data: Static data is used for charts. In a production environment, this would be fetched dynamically from an API.
4. Security and Preference Tabs: Basic UI placeholders are implemented for these tabs as the focus was on the Edit Profile section per the provided code. Full functionality would require additional specifications.
5. Accessibility: Basic ARIA labels and keyboard navigation are included with full WCAG compliance.
6. Browser Compatibility: Tested primarily in Chrome, Firefox, Safari, and Edge with minor adjustments.
7. Image Assets: Placeholder SVG icons and images are assumed to exist in /assets/icons/ and /assets/images/. Replaced with actual assets from the Figma design.

Deployment

Live Demo: [Live Demo]

Deployment Steps:

- Push the repository to GitHub.
- Connect to Vercel via the Vercel dashboard.
- Deploy with default settings (Next.js project detected automatically).
- Access the live URL provided by Vercel.