# Cryptocurrency Dashboard - Frontend Documentation

## 1. Introduction

* **Project Title:** Cryptocurrency Dashboard
* **Team Leader -** A.Abarnath
* **Team Members:** Y.AbuShayeeth,P.Babu ganesh,A.Adhithiya,Bharath Raj.S

## 2. Project Overview

* **Purpose:**
  + The Cryptocurrency Dashboard provides real-time data, trends, and insights into various cryptocurrencies.
  + Users can view live prices, historical trends, and market statistics in an intuitive interface.
* **Features:**
  + Real-time cryptocurrency prices and market cap
  + Interactive charts for price trends
  + Search and filter options for various cryptocurrencies
  + User authentication (if applicable)
  + Dark/light mode support (if implemented)

## Architecture

The architecture of the **Cryptocurrency Dashboard** frontend is structured to ensure modularity, scalability, and maintainability. Below are the key aspects of the architecture:

#### ****Component Structure****

The application follows a **component-based architecture** in React.js, where UI elements are broken down into reusable components. The key components include:

* **Dashboard**: Displays real-time cryptocurrency data and trends.
* **Navbar**: Provides navigation between different sections of the app.
* **SearchBar**: Allows users to search for specific cryptocurrencies.
* **CryptoCard**: A reusable component displaying cryptocurrency details.
* **Chart**: Displays graphical representations of market trends.
* **ErrorBoundary**: Handles unexpected UI errors gracefully.

#### ****State Management****

State management is handled using **React's Context API** to manage global state, such as user preferences and fetched cryptocurrency data. Local state within individual components is managed using the **useState** and **useEffect** hooks for efficient rendering and updates.

#### ****Routing****

The application uses **React Router** to enable seamless navigation between different pages, such as:

* / - Home page with an overview of cryptocurrencies
* /coin/:id - Detailed view of a specific cryptocurrency
* /about - Information about the application

## 4. Setup Instructions

**Prerequisites:**

Node.js and npm:

Node.js is a powerful JavaScript runtime environment that allows you to run

JavaScript code on the local environment. It provides a scalable and efficient

platform for building network applications.

Install Node.js and npm on your development machine, as they are required to

run JavaScript on the server-side.

● Download: https://nodejs.org/en/download/

● Installation instructions: https://nodejs.org/en/download/package-manager/

React.js:

React.js is a popular JavaScript library for building user interfaces. It enables

developers to create interactive and reusable UI components, making it easier to

build dynamic and responsive web applications.

Install React.js, a JavaScript library for building user interfaces.

**Installation:**

1. Clone the repository:

git clone https://github.com/Abarnath/cryptocurrency.git

1. Navigate to the project folder:

Cd cryptocurrency

1. Install dependencies:

npm install

1. Start the development server:

npm run dev

**Development Environment:**

Choose a code editor or Integrated Development Environment (IDE) that suits your preferences, such as Visual Studio Code, Sublime Text, or WebStorm.

• Visual Studio Code: Download from https://code.visualstudio.com/download

• Sublime Text: Download from https://www.sublimetext.com/download

•WebStorm:Download from https://www.jetbrains.com/webstorm/download

## 5. Folder Structure

cryptoverse/

├── src/

│ ├── components/ (Reusable UI components)

│ ├── pages/ (Main pages/views)

│ ├── services/ (API functions)

│ ├── hooks/ (Custom hooks)

│ ├── styles/ (CSS or styled-components)

│ ├── App.js (Main application component)

│ ├── index.js (Entry point)

│ ├── routes.js (React Router setup)

├── public/ (Static assets)

├── .env (Environment variables)

├── package.json

└── README.md

## 6. Running the Application

* **Start the development server:**

npm start

## Component Documentation

**Key Components:**

Navbar: Provides navigation between different sections of the dashboard.

CryptoList: Displays a list of cryptocurrencies along with key market data.

CryptoDetails: Provides in-depth information about a selected cryptocurrency, including historical data.

Chart: Visualizes price trends and market fluctuations using a charting library.

**Reusable Components:**

Button: A custom button component used across the application.

Card: A flexible card component for displaying cryptocurrency information.

Loader: A loading indicator displayed while fetching data from APIs.

SearchBar: Allows users to search for specific cryptocurrencies.

## State Management

**Global State:**

React Context API is used to manage global state for theme and user preferences.

The Context API enables state sharing across multiple components without prop drilling.

**Local State:**

Local component state is used for managing UI interactions such as search filters, modals, and dropdowns.

useState and useEffect hooks handle dynamic data updates and API fetching.

## User Interface

Dashboard Layout:

The dashboard presents global cryptocurrency statistics in a structured card layout.

The top section provides key metrics such as total cryptocurrencies, exchanges, market cap, 24-hour volume, and Bitcoin/Ethereum dominance.

Navigation:

A navbar allows users to switch between the home page and the cryptocurrency list.

Internal routing ensures smooth transitions between different sections.

Visual Elements:

Each cryptocurrency is displayed with its logo, ranking, and key details.

The UI is clean and minimal, ensuring readability and easy access to critical data.

Responsiveness:

The UI adapts to various screen sizes, making it accessible on both desktop and mobile devices.

## 10. Styling

### ****CSS Frameworks/Libraries:****

* **Styled-Components**: The project uses Styled-Components for writing modular and reusable CSS-in-JS styles. This helps in maintaining scoped styles and dynamic theming.
* **Tailwind CSS (if applicable)**: Tailwind CSS provides utility-first styling, making it easier to apply consistent spacing, colors, and responsive designs without writing custom CSS.
* **Custom Styles**: Additional custom CSS files are used for fine-tuned styling where necessary, ensuring a clean and modern user interface.

#### ****Theming:****

* **Light and Dark Mode**: The application supports a light and dark mode, allowing users to toggle between themes for better usability.
* **Context API for Theme Management**: The theme state is managed globally using React Context API, ensuring a seamless theme switch across all components.
* **Dynamic Styling with Styled-Components**: The theme colors, backgrounds, and typography adapt dynamically based on the selected theme using Styled-Components.
* **Local Storage for Theme Persistence**: The user's theme preference is stored in local storage, ensuring that the selected theme persists even after page reloads.

## 11 .Testing

### ****Testing Strategy****

To ensure the reliability and robustness of the Cryptocurrency Dashboard, the following testing approaches are implemented

**Unit Testing**:

* + Uses **Jest** for testing individual functions and React components.
  + **React Testing Library (RTL)** for testing component rendering and user interactions.

**Integration Testing**:

* + Tests interactions between components, such as API calls and state updates.
  + Uses **Mock Service Worker (MSW)** to mock API responses.

**End-to-End (E2E) Testing**:

* + Tests the complete user journey, including navigation and data fetching.
  + Uses **Cypress** for browser automation testing.

### ****Code Coverage****

**Jest Code Coverage**:

* + Jest generates a coverage report showing how much of the code is tested.
  + Coverage includes **statements, branches, functions, and lines**.
  + T

npm test -- --coverage

* + The results are displayed in the terminal and a report is generated in coverage/.

**Threshold Enforcement**:

module.exports = {

collectCoverage: true,

coverageThreshold: {

global: {

branches: 80,

functions: 80,

lines: 80,

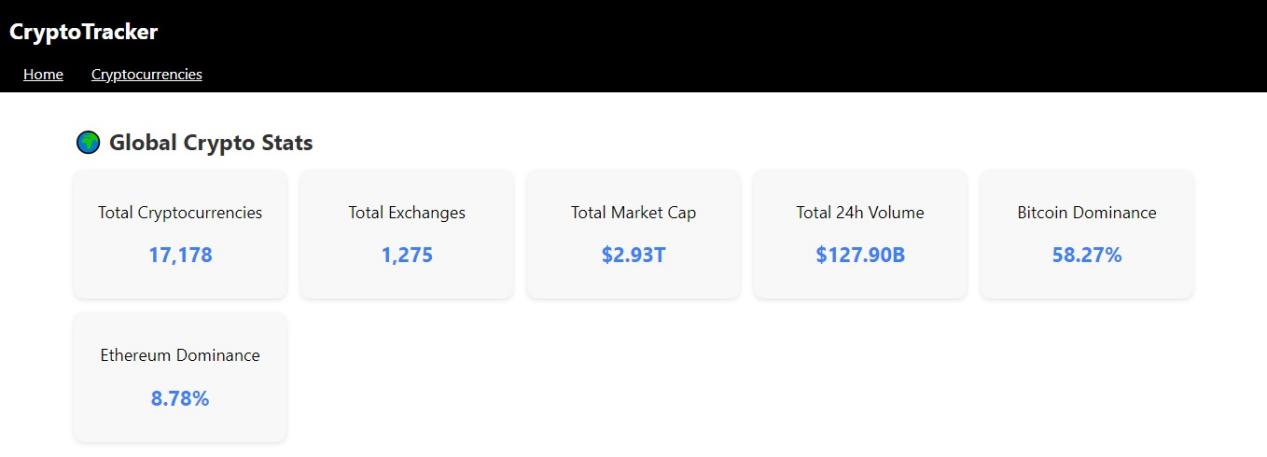
statements: 80

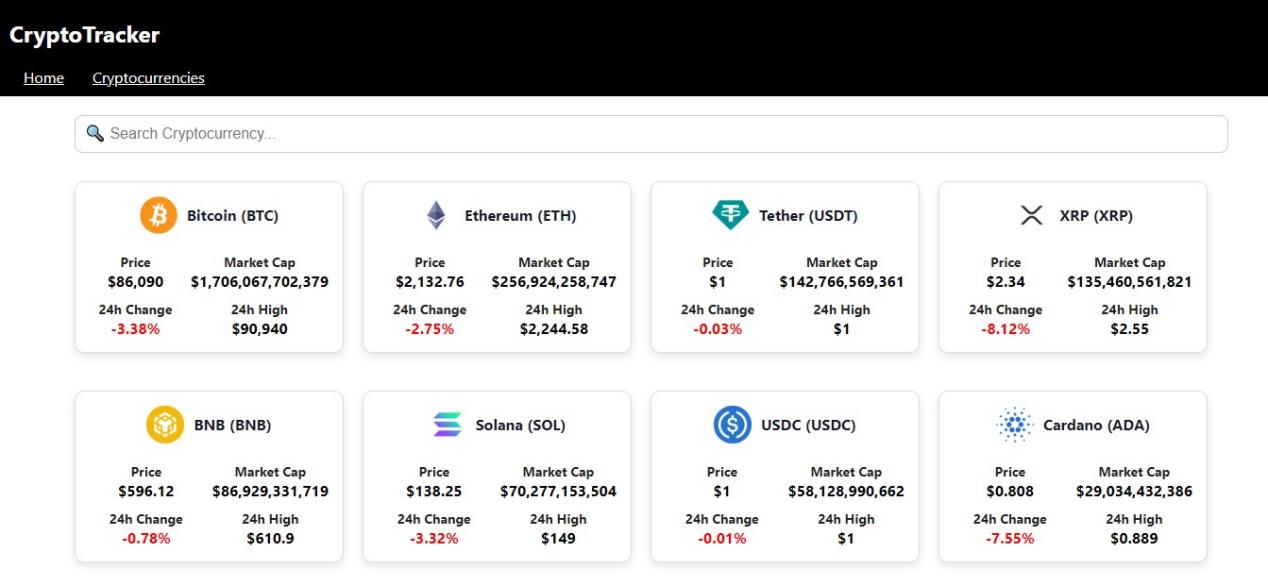
}

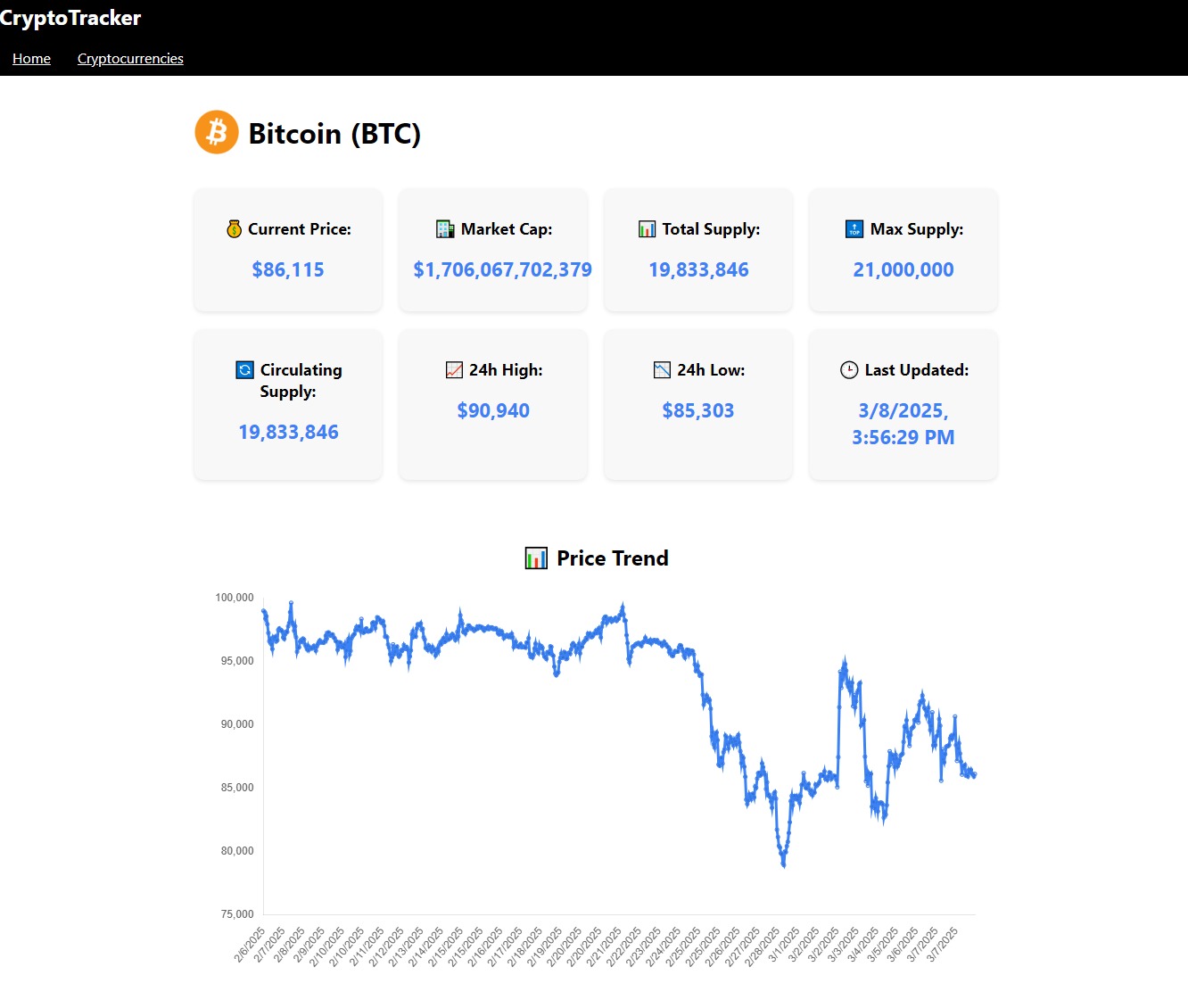
}

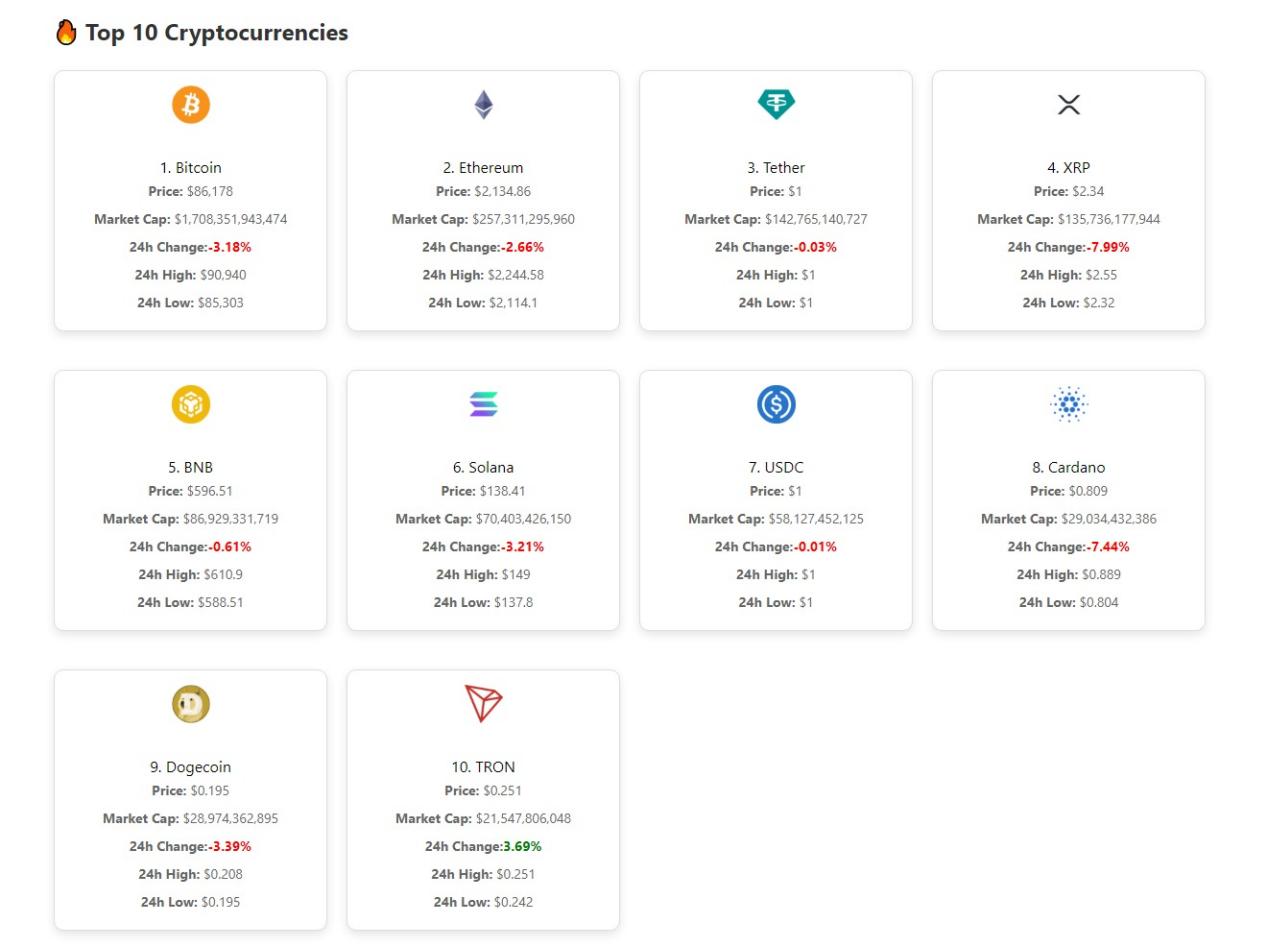
};

## Screenshots









**Demo video**

https://drive.google.com/file/d/1snohBHxv720pksXdmwvN0kicFizvk6No/view?usp=drivesdk

## ****13. Known Issues****

### ****1. API Rate Limits****

* The application relies on third-party APIs like CoinGecko, which may impose rate limits on free-tier users.
* **Issue:** If too many requests are sent in a short period, the API may reject further requests, causing missing or delayed data.
* **Workaround:** Implement caching and reduce unnecessary API calls by storing data temporarily in local storage or state.

### ****2. Performance Issues on Large Data Sets****

* Rendering large lists of cryptocurrencies can cause slow performance, especially on lower-end devices.
* **Issue:** The UI may lag when handling thousands of cryptocurrencies at once.
* **Workaround:** Implement pagination or infinite scrolling to load data in smaller chunks.

### ****3. Inconsistent Dark Mode Behavior****

* Some UI components may not fully adapt to dark mode due to missing styles.
* **Issue:** Certain elements (charts, buttons, or modals) may remain bright in dark mode.
* **Workaround:** Ensure all components inherit theme properties and update styles accordingly.

### ****4. Mobile Responsiveness****

* Some elements may not display correctly on smaller screens.
* **Issue:** Cards and tables may overflow or require excessive horizontal scrolling.
* **Workaround:** Improve responsive design by using flexbox, media queries, and grid layouts.

### ****5. Delayed Data Updates****

* The dashboard may not refresh instantly when new data is available.
* **Issue:** Users may see outdated cryptocurrency prices until they manually refresh.
* **Workaround:** Implement real-time updates using WebSockets or periodic polling

## 14. Future Enhancements

## ****14. Future Enhancements****

### ****1. User Authentication & Personalization****

* Implement user authentication using Firebase or OAuth to allow personalized features.
* Users can create accounts, save favorite cryptocurrencies, and set up watchlists.

### ****2. Real-Time Price Alerts****

* Add push notifications or email alerts for significant price changes in selected cryptocurrencies.
* Users can configure alerts for specific coins when they cross a defined price threshold.

### ****3. Advanced Data Visualization****

* Enhance charts with interactive features like zooming, historical data comparisons, and custom time range selection.
* Integrate more sophisticated libraries like D3.js or Chart.js for better data representation.

### ****4. Multi-Language Support****

* Expand the application’s accessibility by adding localization for multiple languages.
* Allow users to switch between different languages dynamically.

### ****5. AI-Based Market Predictions****

* Implement machine learning models to analyze historical trends and predict future cryptocurrency prices.
* Provide insights and recommendations for potential investment opportunities.

### ****6. Portfolio Management****

* Enable users to track their cryptocurrency holdings and portfolio performance.
* Include profit/loss calculations and historical transaction records.

### ****7. Dark Mode & UI Customization****

* Expand the theme customization options to allow users to select custom color schemes.
* Improve dark mode with better contrast and dynamic adaptation.

### ****8. Mobile App Development****

* Convert the web application into a mobile app using React Native.
* Provide a seamless experience across both desktop and mobile platforms