# CRYPTOVERS

## ****Introduction****

### ****Project Title:**** CRYPTOTRACKER

### ****Team Members:****

* **Team Leader :** T. PUGAZHENTHI
* **Team Members :**
  + S. NAGAPOOSHNAM
  + K.NAVEEN KUMAR
  + M.PHILLIP SANTHOSH
  + D.RAJKUMAR

## ****2. Project Overview****

### ****2.1 Purpose****

This project aims to create a fully functional cryptocurrency web application using React.js. The frontend will provide users with real-time market data, trading functionalities, and an intuitive user interface for portfolio management.

### ****2.2 Features****

* **Real-time Cryptocurrency Data**: Display updated cryptocurrency prices, market trends, and historical data.
* **User Authentication**: Secure login and signup functionality.
* **Portfolio Management**: Users can track their holdings and view real-time updates.
* **Trading Dashboard**: Buy and sell cryptocurrency within the application.
* **Interactive Charts**: Visualization of market trends using graphs.
* **Responsive Design**: Mobile-friendly layout for enhanced user experience.
* **Dark Mode**: Toggle between light and dark themes.

## ****3. Architecture****

Here’s a detailed breakdown of the **Architecture** for the Cryptocurrency frontend project using React.js.

## ****Architecture of Cryptocurrency Frontend****

### ****1. Overview****

The frontend architecture follows a modular and scalable approach using React.js. The application is designed with a **component-based structure**, ensuring code reusability, maintainability, and performance efficiency.

### ****2. Key Architectural Components****

The architecture includes the following key areas:

* **Component Structure**
* **State Management**
* **Routing**
* **API Integration**
* **Security Considerations**

### ****3. Component Structure****

The application is built using reusable React components, following a hierarchical structure:

/src

│── /components

│ │── Navbar.js

│ │── CryptoList.js

│ │── TradeForm.js

│ │── PortfolioCard.js

│ │── Button.js

│ │── Modal.js

│── /pages

│ │── Home.js

│ │── Market.js

│ │── Portfolio.js

│ │── Trade.js

│ │── Settings.js

│── /assets

│── /utils

│── App.js

│── index.js

* **Pages (**/pages**)**: Represents full pages like Home, Market, Portfolio, and Trade.
* **Components (**/components**)**: Reusable UI elements such as buttons, modals, and trading forms.
* **Utilities (**/utils**)**: Contains helper functions, API handlers, and configuration settings.

### ****4. State Management****

The application uses **Redux Toolkit** for global state management and **React’s useState/useContext** for local states.

#### ****4.1 Global State (Redux)****

* Stores application-wide data such as user authentication, cryptocurrency prices, and portfolio details.
* Example slice:

import { createSlice } from '@reduxjs/toolkit';

const cryptoSlice = createSlice({

name: 'crypto',

initialState: { prices: [], portfolio: [] },

reducers: {

setPrices: (state, action) => { state.prices = action.payload; },

updatePortfolio: (state, action) => { state.portfolio = action.payload; },

},

});

export const { setPrices, updatePortfolio } = cryptoSlice.actions;

export default cryptoSlice.reducer;

#### ****4.2 Local State (useState, useContext)****

* Used for managing UI interactions like modals, form inputs, and toggle states.

### ****5. Routing****

* The application uses **React Router** for navigation between pages.
* Example:

import { BrowserRouter as Router, Route, Routes } from 'react-router-dom';

import Home from './pages/Home';

import Market from './pages/Market';

import Portfolio from './pages/Portfolio';

function App() {

return (

<Router>

<Routes>

<Route path="/" element={<Home />} />

<Route path="/market" element={<Market />} />

<Route path="/portfolio" element={<Portfolio />} />

</Routes>

</Router>

);

}

export default App;

### ****6. API Integration****

* Fetches real-time cryptocurrency data from an external API (e.g., CoinGecko, Binance API).
* Uses axios for API calls.
* Example:

import axios from 'axios';

export const fetchCryptoData = async () => {

const response = await axios.get('https://api.coingecko.com/api/v3/coins/markets', {

params: { vs\_currency: 'usd', order: 'market\_cap\_desc' }

});

return response.data;

};

### ****7. Security Considerations****

* **Authentication**: Uses JWT tokens to secure user logins.
* **Data Protection**: Ensures API keys are stored securely using .env files.
* **CORS Handling**: Implements proper CORS policies for API calls.
* **Input Validation**: Prevents XSS and SQL Injection attacks.

## ****Setup Instructions****

## ****Setup Instructions for Cryptocurrency Frontend Application****

This guide provides a step-by-step process to set up and run the **Cryptocurrency Frontend** project built with **React.js**.

## ****1. Prerequisites****

Before setting up the project, ensure that you have the following installed on your system:

### ****1.1 Required Software****

* **Node.js** (LTS version recommended) – [Download Here](https://nodejs.org/)
* **npm** (comes with Node.js) or **yarn**
* **Git** – [Download Here](https://git-scm.com/)
* **Code Editor** (VS Code recommended) – [Download Here](https://code.visualstudio.com/)

## ****2. Installation Steps****

### ****2.1 Clone the Repository****

Open the terminal or command prompt and run:

git clone https://github.com/your-repository/cryptocurrency-app.git

Navigate into the project directory:

cd cryptocurrency-app

### ****2.2 Install Dependencies****

Run the following command to install all required npm packages:

npm install

If using **yarn**, run:

yarn install

### ****2.3 Configure Environment Variables****

Create a .env file in the root directory and add the following (example):

REACT\_APP\_API\_BASE\_URL=https://api.coingecko.com/api/v3

REACT\_APP\_API\_KEY=your\_api\_key\_here

Replace your\_api\_key\_here with a valid API key if required.

### ****2.4 Run the Application****

Start the React development server:

npm start

or

yarn start

The app should now be running on <http://localhost:3000/>.

### ****2.5 Build for Production****

To create an optimized build for deployment:

npm run build

This will generate a build/ folder containing the production-ready files.

## ****3. Folder Structure****

/cryptocurrency-app

│── /public

│── /src

│ │── /components

│ │── /pages

│ │── /assets

│ │── /utils

│── package.json

│── README.md

│── .env

* /src – Contains all React components and pages.
* /public – Static files like the favicon and index.html.
* package.json – Manages dependencies and scripts.
* .env – Stores environment variables securely.

## ****4. Troubleshooting****

### ****4.1 Common Issues & Fixes****

**Issue:** npm start fails  
**Fix:** Run npm install again to ensure all dependencies are installed.

**Issue:** API calls are not working  
**Fix:** Ensure .env variables are correctly set and restart the app.

**Issue:** Port 3000 is already in use  
**Fix:** Run:

npx kill-port 3000

npm start

## ****5. Next Steps****

Once the setup is complete, you can start modifying components and testing the application using Jest or Cypress.

This setup ensures you can **run, develop, and deploy** the Cryptocurrency frontend application smoothly. 🚀

## ****5. Folder Structure****

/cryptocurrency-app

│── /public

│── /src

│ │── /components

│ │── /pages

│ │── /assets

│ │── /utils

│── package.json

│── README.md

* **components/** - Contains reusable UI components.
* **pages/** - Contains main pages like Home, Portfolio, and Trade.
* **assets/** - Images, icons, and styles.
* **utils/** - Helper functions and API calls.

## ****6. Running the Application****

To start the application, run:

npm start

## ****7. Component Documentation****

### ****7.1 Key Components****

* **Navbar** - Main navigation bar.
* **CryptoList** - Displays the list of cryptocurrencies.
* **TradeForm** - Handles buy/sell transactions.
* **PortfolioCard** - Shows the user’s holdings.

### ****7.2 Reusable Components****

* **Button**: Customizable buttons for interactions.
* **Modal**: Pop-up windows for user confirmation.

## ****8. State Management****

### ****8.1 Global State****

Managed using Redux Toolkit, ensuring consistency across the app.

### ****8.2 Local State****

Handled using React’s useState and useContext hooks for component-specific logic.

## ****9. User Interface****

The UI follows modern web standards with interactive elements. Screenshots and UI previews are provided in the documentation folder.

## ****10. Styling****

### ****10.1 CSS Frameworks/Libraries****

* Tailwind CSS for quick styling.
* Styled Components for component-level styling.

### ****10.2 Theming****

A theme provider manages dark and light modes dynamically.

## ****11. Testing****

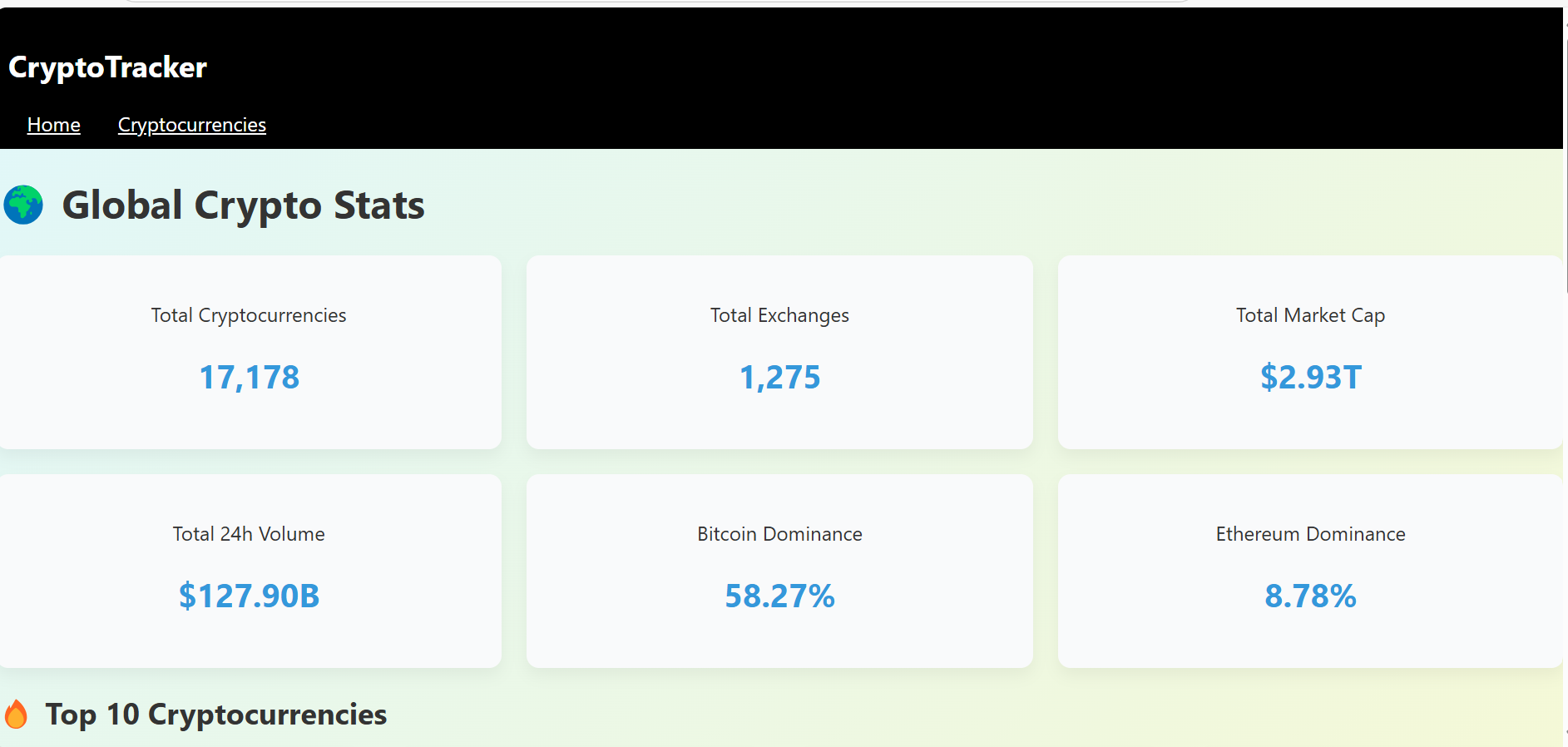
### ****11.1 Testing Strategy****

* **Unit Testing**: Jest and React Testing Library.
* **Integration Testing**: Ensures component interactions work as expected.
* **End-to-End Testing**: Cypress for testing the entire user journey.

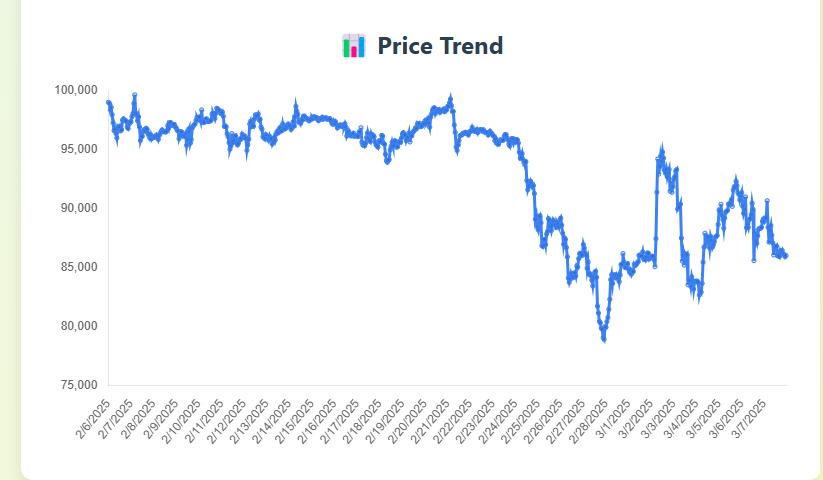
### ****11.2 Code Coverage****

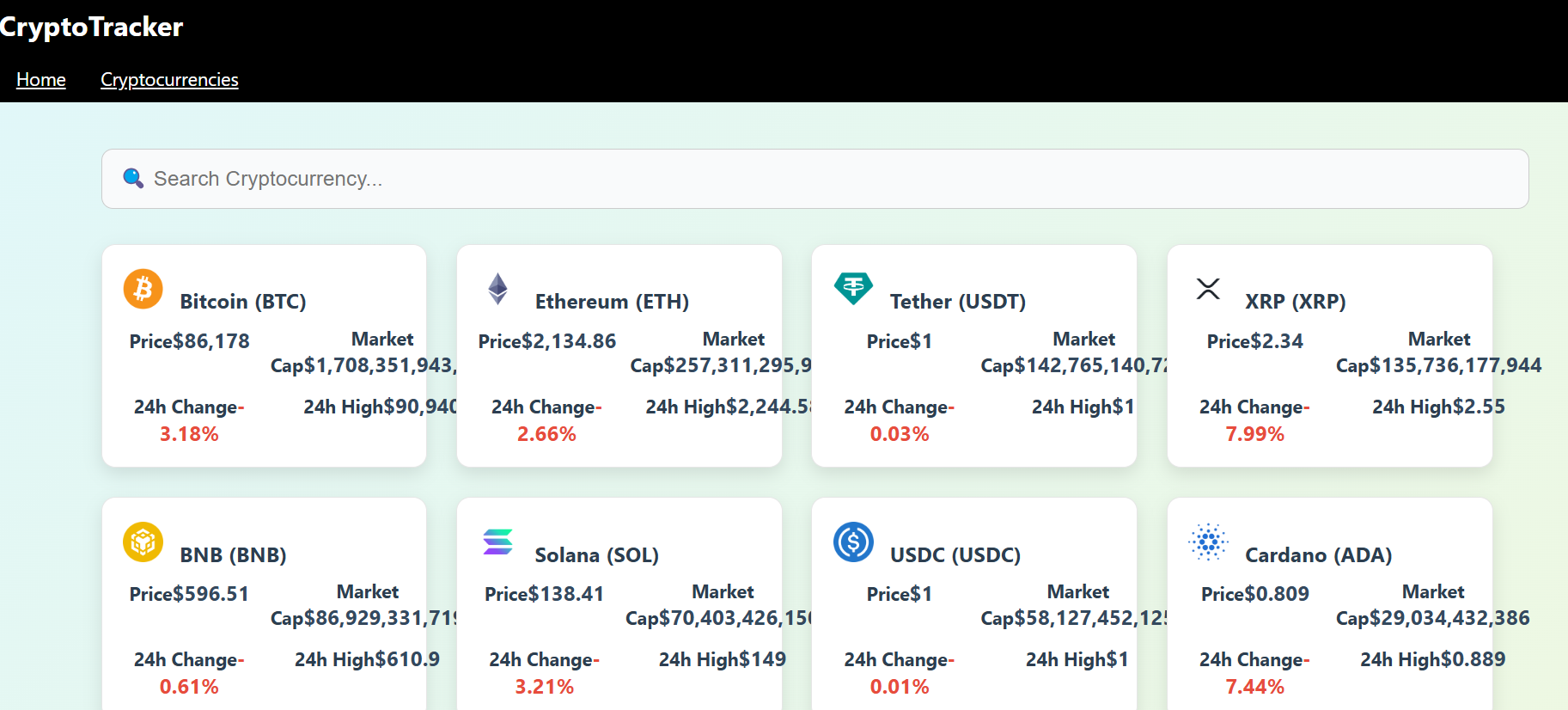
Reports generated using Jest ensure high test coverage.

## ****12. Screenshots or Demo****









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

* Minor layout bugs on certain screen sizes.
* Some API calls may experience latency issues.

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

* Integration with multiple exchanges.
* Advanced charting tools.
* Mobile application version.
* AI-driven trading insights.

This document provides a comprehensive report on the Cryptocurrency Frontend Development with React.js project, detailing its structure, implementation, and future roadmap.