1. INTRODUCTION

1.1 Project Title: Cryptoverse

1.2 Team ID: SWTID1741256183147251

1.3 Team Member:

1. **YOGESWARAN S** (yogeshsridhar391@gmail.com)
2. **MADESH N** (2004madesh@gmail.com)
3. **SAMRIN FATHIMA S** (samrinshahina@gmail.com)
4. **YOGALAKSHMI S** (yogalaksmi2k20@gmail.com)

2. PROJECT OVERVIEW

2.1 Purpose:

The Cryptocurrency Dashboard project aims to provide users with a real-time, interactive, and user-friendly platform to track and analyze cryptocurrency market trends. This dashboard will display essential data such as live prices, market capitalization, trading volume, historical trends, and other key metrics. By integrating API data from leading crypto exchanges, the system ensures accurate and up-to-date information. The project is designed to help traders, investors, and enthusiasts make informed decisions by offering visual analytics, price alerts, and portfolio tracking. With an intuitive interface and insightful data representation, the dashboard enhances cryptocurrency market accessibility and understanding.

2.2 Features:

The Cryptocurrency Dashboard project is a powerful and interactive platform designed to provide users with real-time insights into the ever-changing cryptocurrency market. It integrates live price updates, market capitalization, trading volume, and historical trends, ensuring that users can make informed investment decisions. With dynamic and interactive charts, including candlestick and line graphs, users can analyze market trends with ease. The dashboard also features a robust search and filtering system, allowing users to quickly find specific cryptocurrencies and sort them based on key performance metrics. A built-in portfolio tracking feature enables users to monitor their holdings, view total investment value, and track gains and losses in real-time. To enhance user convenience, the platform includes customizable price alerts and notifications, helping investors stay informed about significant price movements. Additionally, a dedicated news section aggregates the latest updates from the cryptocurrency world, providing valuable insights into market trends, regulations, and technological advancements. The dashboard also includes a currency conversion tool, allowing users to compare prices across different fiat currencies effortlessly. Designed with a user-friendly interface, intuitive navigation, and visually appealing analytics, this cryptocurrency dashboard serves as a valuable resource for traders, investors, and enthusiasts who seek to stay ahead in the fast-paced world of digital assets.

3. ARCHITECTURE

3.1 Component Structure:

Here’s a more detailed component structure of your React-based crtyptocurrency dashboard with subheadings and expanded explanations.

1. Root Structure:

The Project follows a modular and scalable architecture.

```
/Code
│── /code
│   │── /public
│   │── /src
│   │   ├── /assets
│   │   ├── /components
│   │   ├── /services
│   │   ├── /app
│   │   ├──App.jsx
│   │   ├── main.jsx
│   │── package.json
│   │── README.md
│   │── index.html
│   │── vite.config.js
```

/Code/code:

This is the main directory of your project, containing configuration files, source code, and public assets.

Folders:

1. Public(/public):

- Stores static assets like images, icons, and other files that don’t change dynamically.
- **Files Inside:**
    - vite.svg → A default Vite logo (can be removed or replaced).
    - _redirects → Used for handling redirects (typically for Netlify deployments).

2.Source(/src):

- The main source code for the application.

3. Assets(/assets):

- Stores images, logos, and other static resources.
- **Files Inside:**
    - cryptocurrency.png → Likely an image used in the UI.

4. Components(/components):

- Houses reusable UI components.
- **Files Inside:**
    - Loader.jsx → A loading animation component.
    - Navbar.jsx → The navigation bar.
    - CryptoDetails.jsx → A component to show details of a specific cryptocurrency.
    - Cryptocurrencies.jsx → A component that lists multiple cryptocurrencies.
    - LineChart.jsx → A chart component to display cryptocurrency trends.
    - Home.jsx → The home page of the app.

    - index.js → Likely an export file for organizing components.

5. Services(/services):

- Handles API requests and external data fetching.
- **Files Inside:**
    - cryptoApi.js → Manages API calls related to cryptocurrency data.

6. App(/app):

- Manages the global state (Redux store or context).
- **Files Inside:**
    - store.js → Redux store configuration.

Files in /src:

- index.css → Global CSS styles for the app.
- App.css → Additional styles for App.jsx.
- App.jsx → The main React component that wraps the entire application.
- main.jsx → The entry point that renders the App component into the DOM.

Configuration and Metadata Files:

- .gitignore → Specifies files that should be ignored by Git.
- .eslintrc.cjs → Configuration file for ESLint (used for enforcing code quality).
- index.html → The main HTML file that loads the React app.
- README.md → Documentation file for the project.
- package.json → Defines project dependencies and scripts.
- package-lock.json → Locks dependency versions for consistency.
- vite.config.js → Configuration file for Vite (used for fast builds and development).

3.2 State Management:

1. State Management in the Cryptocurrency Dashboard:

Effective state management is crucial for ensuring smooth data flow and real-time updates in the cryptocurrency dashboard. This project utilizes **Context API** along with the **React useReducer hook** to manage global state efficiently.

1. Why Context API?

Context API is chosen for its lightweight nature, making it ideal for managing global state without the complexity of external libraries like Redux. It allows seamless state sharing across components without excessive prop drilling.

2. Handling Global State with Context API

The application wraps key components inside a CryptoContextProvider, which stores and provides access to essential cryptocurrency data such as real-time prices, market trends, and portfolio tracking. This ensures that multiple components can access and update state efficiently.

3. useReducer for State Updates

To handle complex state updates, the **useReducer hook** is integrated within the Context API. This helps in managing API calls, filtering cryptocurrencies, and updating portfolio data while maintaining performance and scalability.

4. API Data Management

The dashboard fetches cryptocurrency data from external APIs and stores it in the global state. Context API ensures that the latest prices and trends are available to all components, reducing redundant API calls and improving efficiency.

5. Alternative Approaches

While Redux could also be used for state management, Context API with useReducer is preferred due to its simplicity, reduced boilerplate code, and built-in React support. If the application scales further, Redux Toolkit or Zustand can be considered for more advanced state management.

By using **Context API and useReducer**, the cryptocurrency dashboard maintains a responsive, scalable, and efficient state management system, ensuring a seamless user experience.

2. Redux (For Large-Scale Applications):

1. Why Redux for State Management?

Redux is chosen for its ability to handle complex state updates, maintain a single source of truth, and efficiently manage asynchronous data fetching using middleware like **Redux Thunk** or **Redux Saga**. This approach ensures that cryptocurrency data updates consistently across all components.

2. Centralized Global Store

The application maintains a **Redux store** that holds key state variables, including real-time price data, market trends, user portfolio details, and filtering preferences. By structuring the state in a predictable manner, Redux enhances maintainability and scalability.

3. Actions & Reducers for State Updates

Redux **actions** define the types of state changes, such as fetching cryptocurrency data, updating portfolio holdings, or filtering search results. These actions are handled by **reducers**, which update the state immutably and efficiently based on dispatched actions.

3.3 Routing:

Routing in this project is managed using **React Router**, enabling smooth navigation between different sections without reloading the page. Users can navigate between home, cryptocurrency details, and other pages efficiently.

1. Setting Up Routing:

The app uses **React Router** to define routes for different pages. The main routes include:

- **Public Routes:** Home, Cryptocurrencies, and NotFound (404).
- **Protected Routes:** CryptoDetails (requires authentication to view specific cryptocurrency details).

2. Protected Routes:

Some routes, such as detailed cryptocurrency information, require authentication. A **PrivateRoute** component ensures that only logged-in users can access them, redirecting unauthorized users to the home page.

3. Dynamic Routing:

The app handles cryptocurrency details dynamically. For example, `/cryptocurrencies/:id` loads details for a specific cryptocurrency based on its ID.

4. Navigation with Links:

Instead of traditional `<a>` tags, the app uses React Router's **`Link`** and **`NavLink`** for navigation, ensuring smooth page transitions without full reloads.

5. Redirecting Users:

After certain actions, such as login, users are redirected using **`useNavigate()`**. For example, after selecting a cryptocurrency, users are directed to its detailed view.

6. 404 Handling:

If users enter an invalid URL, they are redirected to a custom **404 Not Found** page, improving user experience.

7. Performance Optimization with Code Splitting:

To enhance performance, **lazy loading** is implemented with **React's `lazy()` and `Suspense`**, ensuring that pages load only when needed, reducing initial load time.

Let me know if you need a code implementation for this!

4. SETUP INSTRUCTIONS

4.1 Prerequisites:

Before setting up and running the React Cryptocurrency App, ensure that your system meets the following requirements:

1. Install Node.js and npm

This app is built using React.js with Vite, which requires **Node.js** and **npm** to run.

- **Download Node.js (LTS version):** Node.js Official Website
- npm comes bundled with Node.js, so installing Node.js will also install npm.

Verify installation:

```
node -v
npm -v
```

You should see version numbers displayed for both.

2. Install Git (Optional, if cloning the repository)

If you need to clone the project from a repository, install Git:

- **Download Git:** Git Official Website

Verify installation:

```
git --version
```

If installed, this will return the Git version.

3. Install a Package Manager (npm or yarn)

The project uses **npm** by default, but you can also use **yarn**.
To install yarn (optional), run:

```
npm install -g yarn
```

Check the version with:

```
yarn -v
```

4. Install a Code Editor (Recommended: VS Code)

A good text editor improves the development experience.

- **Download VS Code:** VS Code Official Website
- Install recommended **React** and **ESLint** extensions for better coding support.

5. Install a Web Browser (Recommended: Google Chrome)

A modern browser is needed for testing the app. Google Chrome is recommended, as it offers great developer tools.

- **Download Chrome:** Google Chrome Official Website

6. Install Node.js Dependencies

Once inside the project folder, install all necessary dependencies by running:

```
npm install
```

or

```
yarn install
```

4.2 Installation:

1. Clone or Download the Project

If the project is hosted on a Git repository, use the following command to clone it:

```
bash
CopyEdit
git clone https://github.com/your-repo/crypto-app.git
cd crypto-app
```

If you have received the project as a ZIP file, extract it and navigate to the project folder.

2. Install Dependencies

Once inside the project folder, install all required packages by running:

```
npm install
```

or, if using yarn:

```
yarn install
```

This will download and set up all dependencies listed in **package.json**.

3. Set Up Environment Variables (If Required)

If the project includes a **.env.example** file, create a **.env** file by running:

```
cp .env.example .env
```

Then, open **.env** and configure necessary values, such as API keys or database URLs.

4. Start the Development Server

Run the following command to launch the app:

```
npm run dev
```

or

```
yarn dev
```

This will start a local development server, and the app will be accessible in your browser at:http://localhost:5173/

5. FOLDER STRUCTURE

5.1 Client:

The client-side of the Cryptocurrency Dashboard follows a structured approach to keep the code base organized, maintainable, and scalable. Below is the typical React project structure used for the frontend.

```
/Code
|── /code
|   ├── .gitignore
|   ├── .eslintrc.cjs
|   ├── index.html
|   ├── README.md
|   ├── package.json
|   ├── package-lock.json
|   ├── vite.config.js
|   |── /src
|   |   ├── index.css
|   |   ├── App.css
|   |   ├── App.jsx
|   |   ├── main.jsx
|   |   |── /assets
|   |   |   ├── cryptocurrency.png
|   |   |── /components
|   |   |   ├── Loader.jsx
```

```
│   │   │   ├── Navbar.jsx
│   │   │   ├── CryptoDetails.jsx
│   │   │   ├── Cryptocurrencies.jsx
│   │   │   ├── LineChart.jsx
│   │   │   ├── Home.jsx
│   │   │   ├── index.js
│   │   │── /services
│   │   │   ├── cryptoApi.js
│   │   │── /app
│   │   │   ├── store.js
│   │── /public
│   │   ├── vite.svg
│   │   ├── _redirects
```

Folder Breakdown:

1. /code (Root Folder):

This is the main project directory that contains all the files and folders necessary for the React application. It includes configuration files, dependencies, and source code.

2. /public:

This folder contains static assets that don’t go through Webpack or Vite processing.

- **vite.svg** - A Vite logo used in the project.
- **_redirects** - Configuration for handling routing when deploying the app.

3. /src (Main Application Source Code):

This folder contains all the core React files and business logic of the application.

a) /assets

- Stores static files like images, icons, and logos used in the application.
- **cryptocurrency.png** - An image related to cryptocurrencies, likely used in the UI.

b) /components

- Contains reusable React components for different parts of the UI.
    - **Loader.jsx** - A component that displays a loading spinner.
    - **Navbar.jsx** - The navigation bar component.
    - **CryptoDetails.jsx** - A page that displays details of a specific cryptocurrency.
    - **Cryptocurrencies.jsx** - A page that lists all cryptocurrencies.
    - **LineChart.jsx** - A component that renders a line chart for price trends.
    - **Home.jsx** - The homepage component of the app.
    - **index.js** - Likely an entry point for exporting components.

c) /services

- Contains API service files to fetch cryptocurrency data.
    - **cryptoApi.js** - A service file that handles API requests for cryptocurrency data.

d) /app

- Manages global state using Redux Toolkit or another state management approach.
    - **store.js** - The main Redux store configuration file.

4. Root-Level Files:

- **index.html** - The main HTML file where the React app is injected.
- **App.jsx** - The root React component that renders the entire application.
- **main.jsx** - The entry point that renders <App /> into the DOM.
- **index.css & App.css** - Contains global and component-specific styles.
- **.gitignore** - Specifies which files should be ignored by Git.
- **.eslintrc.cjs** - Configuration for ESLint, used to enforce coding standards.
- **package.json & package-lock.json** - Contains project dependencies and scripts.
- **vite.config.js** - Configuration file for Vite, the build tool used for the project.
- **README.md** - Documentation about the project, including setup instructions.

6. RUN THE APPLICATION

Follow these steps to start the Crypto currency Dashboard on your local machine.

1. Navigate to the Project Directory:

If you haven’t already, open a terminal and go to the project folder:

```
cd path/to/cryptocurrency-dashboard
```

If the project has separate client and server folders, navigate to the client directory:

```
cd client
```

2. Install Dependencies:

Ensure all required packages are installed before running the app:

```
npm install
```

or

```
yarn install
```

3. Start the Development Server:

Run the following command to launch the React application:

```
npm run dev
```

or

```
yarn dev
```

This will start a development server, and the app will open automatically in your default web browser at:

http://localhost:5173

4. Running with Backend (If Applicable):

If the project has a backend, make sure to start the backend server before running the frontend. Navigate to the backend directory and start the server:

```
cd server
npm start
```

Then, in a separate terminal, start the frontend as described in step 3.

5. Troubleshooting Issues:

Port Already in Use (if something else is running on port 5173):

```
npx kill-port 5173
```

Then restart the app.

Dependency Issues:

Clear the cache and reinstall dependencies:

```
rm -rf node_modules package-lock.json && npm install
```

Backend Not Connecting:

Ensure that the API URL in .env is correctly set, e.g.:

```
VITE_API_URL=http://localhost:5000
```

Restart the frontend after making changes.

6. Running in Production Mode (Optional):

For optimized performance, you can build the app and serve it:

```
npm run build
```

Then, serve the build using a static server:

```
npx serve -s dist
```

7. COMPONENT DOCUMENTATION

7.1 Key Components:

The Cryptocurrency Dashboard is built using React with a modular component structure for reusability and maintainability. Below is a categorized breakdown of the key components.

1. Layout Components (Shared across multiple pages):

These components help structure the app and provide navigation.

- **Navbar.jsx** – Contains links to different sections such as Home and Cryptocurrencies.

2. Cryptocurrency Data Components (Handles data display & visualization):

These components display cryptocurrency-related data dynamically.

- **Cryptocurrencies.jsx** – Lists various cryptocurrencies with price, market cap, and ranking.
- **CryptoDetails.jsx** – Displays detailed information about a selected cryptocurrency.
- **LineChart.jsx** – Provides a graphical representation of cryptocurrency price trends.

3. Dashboard & Home Components (Main sections of the application):

These components provide an overview and main functionalities of the dashboard.

- **Home.jsx** – The landing page displaying market stats and featured cryptocurrencies.

4. Utility Components (Enhance UI/UX with reusable elements):

These components help in maintaining UI consistency and improving user experience.

- **Loader.jsx** – Displays a loading animation while fetching cryptocurrency data.

5. API & Data Handling Components (Manages API calls & global state):

These components handle communication with external APIs and manage data.

- **cryptoApi.js** (inside /services/) – Fetches real-time cryptocurrency data from an external API.
- **store.js** (inside /app/) – Manages global state using Redux or Context API (if applicable).

8. STATE MANAGEMENT

8.1 Global States:

The Cryptocurrency Dashboard uses Redux Toolkit for managing global state efficiently. This ensures that data such as cryptocurrency details, market trends, and user preferences are accessible across different components without excessive prop drilling.

Cryptocurrency Data State:

The cryptocurrency state stores real-time market data, including coin prices, rankings, and historical trends. The Redux store fetches data from an external API and keeps it updated. This allows components like **Cryptocurrencies.jsx**, **CryptoDetails.jsx**, and **LineChart.jsx** to access the latest market information seamlessly.

State Integration in the App:

To make the global state accessible across all components, the Redux store is configured in **store.js** and wrapped around the main application using the Redux Provider. This ensures that all components requiring cryptocurrency data can retrieve it efficiently from the global state.

Benefits of Global State Management:

- **Efficient Data Sharing** – Ensures real-time cryptocurrency data is available across multiple components.
- **Improved Scalability** – The app can expand without increasing state complexity.
- **Better Performance** – Reduces unnecessary re-renders by managing state efficiently.

This structured state management approach enhances maintainability, making the application robust and easy to extend.

8.2 Local State Management:

In addition to global state management using Redux Toolkit, the Cryptocurrency Dashboard also utilizes **local state** within individual components using React's useState and useEffect hooks. Local state is used to manage UI-specific data that does not need to be shared across the application.

1. UI State (Component-Level Data Management):

Local state is used for managing UI interactions, such as:

- **Search Input:** The search functionality in Cryptocurrencies.jsx maintains a local state (searchTerm) to filter cryptocurrencies dynamically.
- **Loading State:** Components like CryptoDetails.jsx and LineChart.jsx use local state (isLoading) to track API data fetching and prevent UI flickering.
- **Dropdowns & Toggles:** UI elements such as currency selectors or theme toggles are managed locally using useState.

2. Handling Asynchronous Data with useEffect:

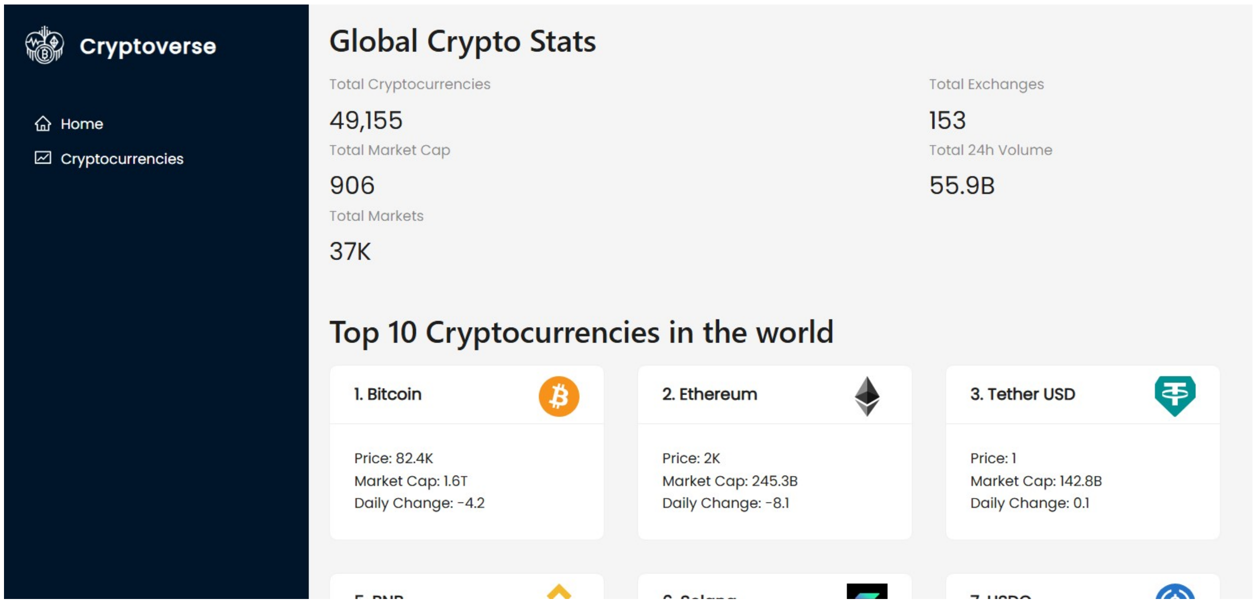
Local state is combined with useEffect to handle API calls and manage side effects. For example:

- In **CryptoDetails.jsx**, useEffect triggers API requests when the selected cryptocurrency changes, updating the local state with new data.
- In **LineChart.jsx**, useEffect listens for changes in selected timeframes and updates the chart accordingly.

3. Benefits of Local State Management:

- **Better Performance** – Prevents unnecessary global re-renders by keeping transient UI data local.
- **Simplifies Component Logic** – Keeps state encapsulated within components that actually need it.
- **Optimized API Calls** – Reduces redundant API requests by updating state only when necessary.

9. USER INTERFACE



Global Crypto Stats

Total Cryptocurrencies
49,155
Total Market Cap
906
Total Markets
37K
Total Exchanges
153
Total 24h Volume
55.9B

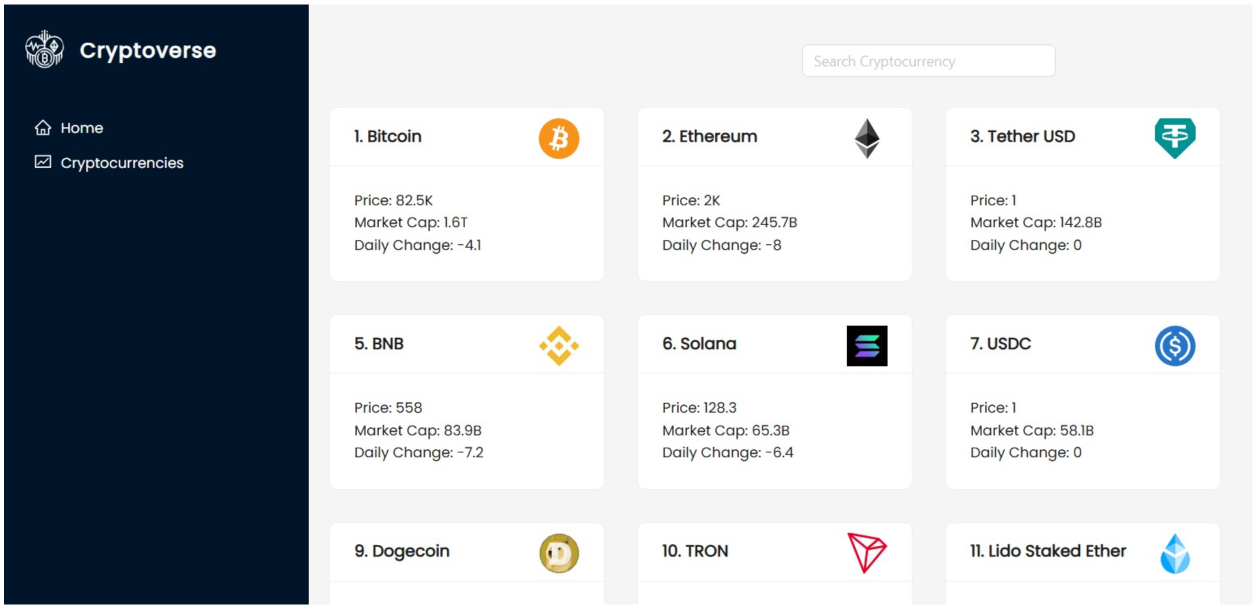
Top 10 Cryptocurrencies in the world

1. Bitcoin
Price: 82.4K
Market Cap: 1.6T
Daily Change: -4.2

2. Ethereum
Price: 2K
Market Cap: 245.3B
Daily Change: -8.1

3. Tether USD
Price: 1
Market Cap: 142.8B
Daily Change: 0.1

Figure 9.1 Screensot – 1



1. Bitcoin
Price: 82.5K
Market Cap: 1.6T
Daily Change: -4.1

2. Ethereum
Price: 2K
Market Cap: 245.7B
Daily Change: -8

3. Tether USD
Price: 1
Market Cap: 142.8B
Daily Change: 0

5. BNB
Price: 558
Market Cap: 83.9B
Daily Change: -7.2

6. Solana
Price: 128.3
Market Cap: 65.3B
Daily Change: -6.4

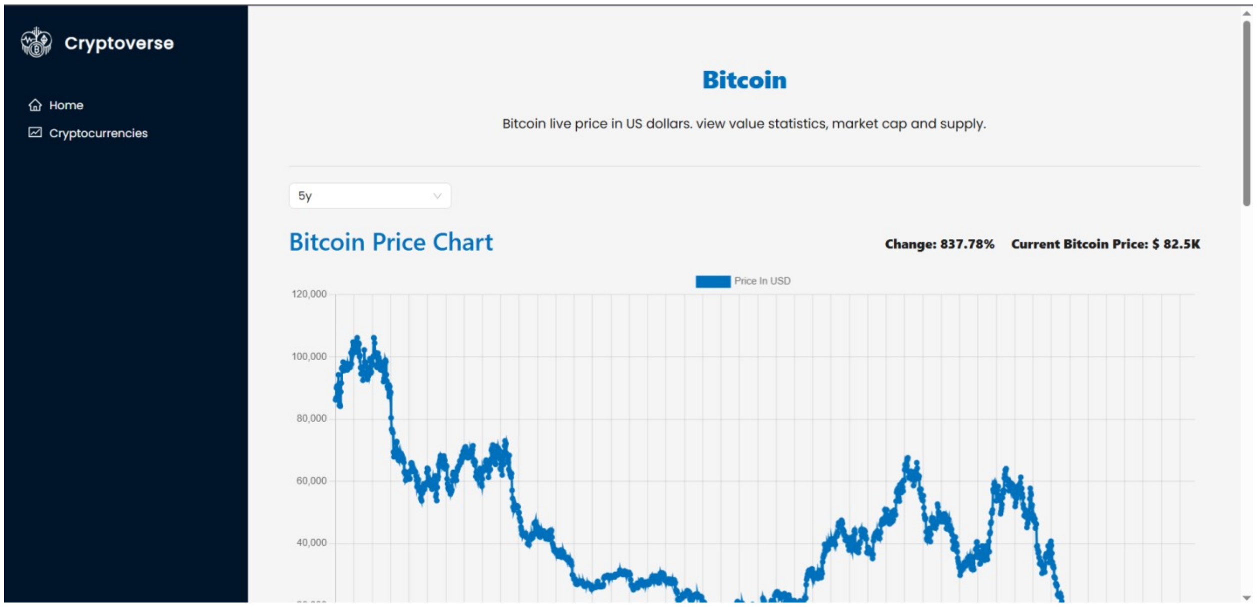
7. USDC
Price: 1
Market Cap: 58.1B
Daily Change: 0

9. Dogecoin

10. TRON

11. Lido Staked Ether

Figur 9.2 Screenshot – 2



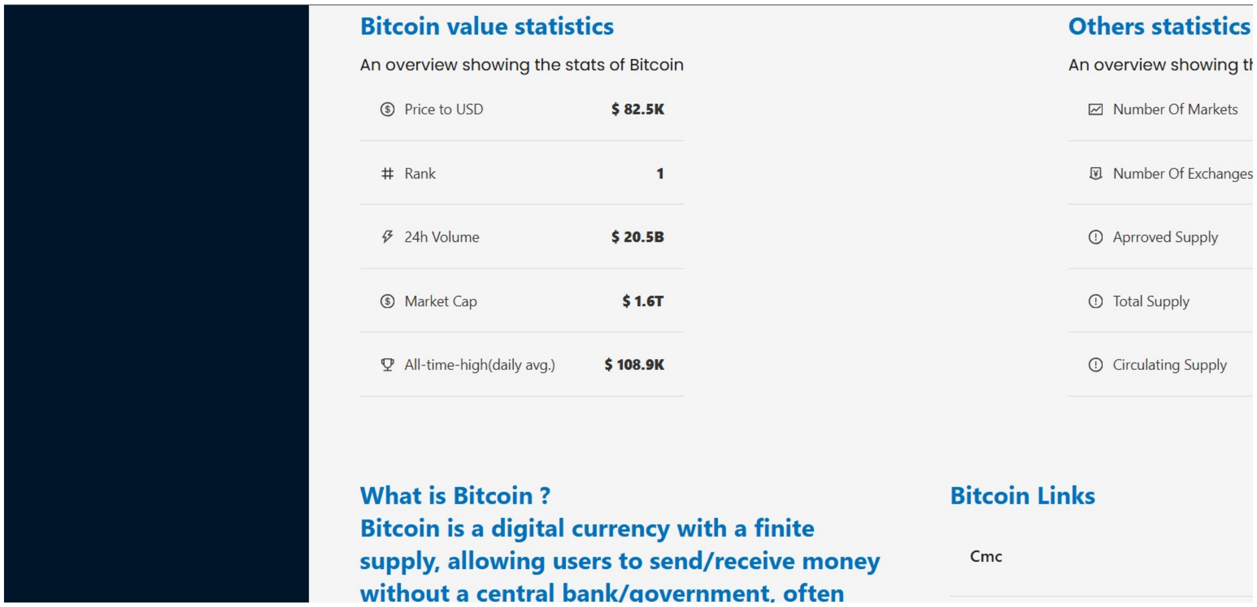
Bitcoin

Bitcoin live price in US dollars. view value statistics, market cap and supply.

Bitcoin Price Chart

Change: 837.78%   Current Bitcoin Price: $ 82.5K

Figure 9.3 Screenshot – 3



Bitcoin value statistics

An overview showing the stats of Bitcoin

Price to USD	$ 82.5K
Rank	1
24h Volume	$ 20.5B
Market Cap	$ 1.6T
All-time-high(daily avg.)	$ 108.9K

Others statistics

An overview showing th

Number Of Markets

Number Of Exchanges

Aprroved Supply

Total Supply

Circulating Supply

What is Bitcoin ?

Bitcoin is a digital currency with a finite supply, allowing users to send/receive money without a central bank/government, often

Bitcoin Links

Cmc

Figure 9.3 Screenshot – 4

10. STYLING

10.1 CSS Framework/Libraries:

The cryptocurrency dashboard follows a structured styling approach to maintain a consistent and responsive user experience. The project primarily relies on **CSS** for styling, with potential support for external libraries if configured in the project dependencies.

1. Styling Approaches:

The project includes different styling techniques:

- **Global Styles**: Defined in index.css, setting universal styles such as fonts, colors, and default spacing.
- **Component-Level Styles**: Managed within App.css, ensuring that specific components have their own scoped styles.
- **Vite Configuration**: The project uses **Vite** as the build tool, which supports modern CSS handling for optimized performance.

2. Responsive Design and Layouts:

The app is designed to be fully responsive, adapting to various screen sizes using:

- **CSS Flexbox & Grid**: Used for dynamic layout structuring.
- **Media Queries**: Implements breakpoints to optimize UI across different devices.
- **Container-Based Adjustments**: Ensures that components adapt dynamically based on available space.

Example: Responsive Grid Layout:

```
.grid-container {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(250px, 1fr));
  gap: 20px;
}

@media (max-width: 768px) {
  .grid-container {
    grid-template-columns: 1fr;
  }
}
```

3. Theming and Customization:

The project may support theme customization using:

- **CSS Variables**: For defining colors, typography, and spacing.
- **Custom Fonts and Icons**: Ensures branding consistency.
- **Dynamic Theme Switching**: If implemented, it allows users to toggle between light and dark modes.

Example: Theme Toggle Using CSS Variables:

```
:root {
  --background-color: #ffffff;
  --text-color: #333;
}
.dark-mode {
  --background-color: #1e1e1e;
  --text-color: #fff;
}
js
const toggleTheme = () => {
  document.body.classList.toggle("dark-mode");
};
```

4. Interactive UI and Animations:

The UI experience is enhanced with smooth transitions and animations:

- **Hover Effects**: Provides a responsive feel for interactive elements.
- **CSS Transitions**: Adds subtle animations for elements like buttons and modals.
- **Framer Motion** (if used): Enables advanced animations for page transitions and UI elements.

Example: Animated Button:

```
.button {
  background-color: #007bff;
  color: white;
  padding: 10px 20px;
  border-radius: 5px;
  transition: background-color 0.3s ease-in-out;
}

.button:hover {
  background-color: #0056b3;
}
```

This styling structure ensures a clean, scalable, and responsive design for the cryptocurrency dashboard.

11. TESTING

Code Coverage Implementation:

1. **Using Jest for Unit Test Coverage**
   - If your project includes Jest for testing, you can generate a code coverage report using:

     ```
     npm test -- --coverage
     ```

   - This will create a coverage/ folder with a detailed report on which parts of the code are covered.

2. **Using Cypress for End-to-End (E2E) Test Coverage:**
   - If Cypress is used, you can integrate code coverage with the @cypress/code-coverage plugin.
   - Install the plugin:

     ```
     npm install --save-dev @cypress/code-coverage istanbul
     ```

   - Add this to your cypress/support/index.js:

     ```
     import '@cypress/code-coverage/support';
     ```

   - Then, in cypress/plugins/index.js, add:

     ```
     js
     module.exports = (on, config) => {
       require('@cypress/code-coverage/task')(on, config);
       return config;
     };
     ```

- Run Cypress tests and generate a coverage report:

```
npx cypress run
```

3. **Using Istanbul (nyc) for Backend Coverage (if applicable)**
    - If your project includes a backend (Node.js/Express), you can use nyc:

    ```
    npm install --save-dev nyc
    ```

    - Modify the test script in package.json:

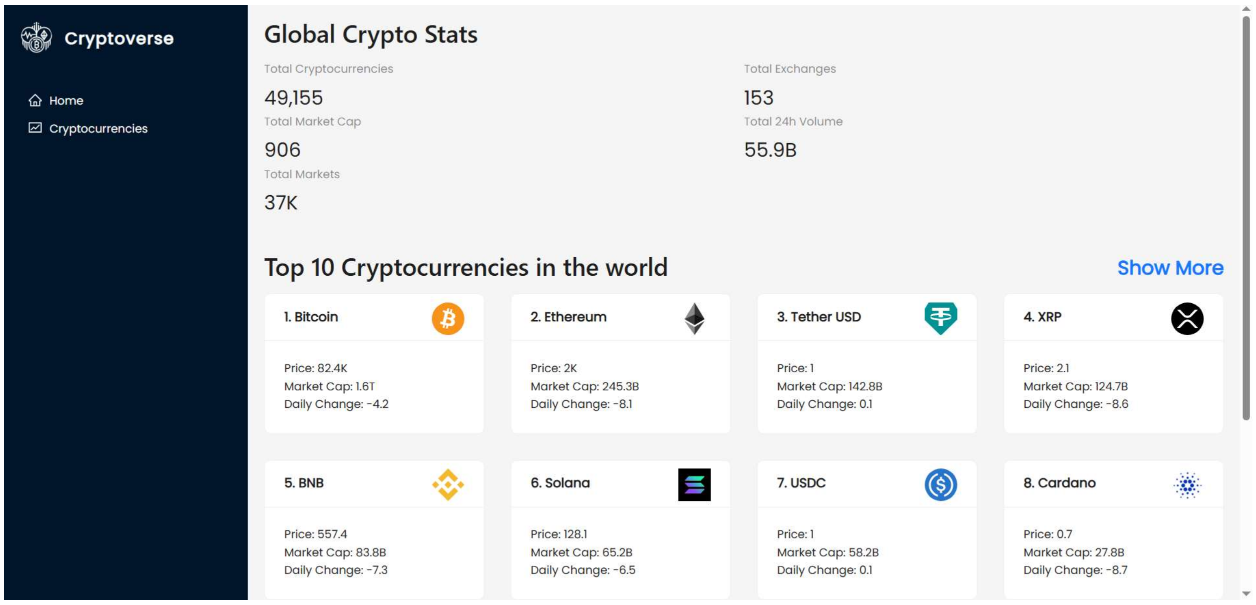
    ```
    "scripts": {
      "test": "nyc --reporter=text mocha"
    }
    ```

    - Run backend tests with:

    ```
    npm test
    ```

4. **Viewing Coverage Reports**
    - After running tests, a /coverage folder is created with detailed reports.
    - Open coverage/lcov-report/index.html in a browser to view graphical reports.

12. SCREENSHOT OR DEMO


Cryptoverse

Home

Cryptocurrencies

Global Crypto Stats

Total Cryptocurrencies

49,155

Total Exchanges

153

Total Market Cap

906

Total 24h Volume

55.9B

Total Markets

37K

Top 10 Cryptocurrencies in the world

Show More

1. Bitcoin
Price: 82.4K
Market Cap: 1.6T
Daily Change: -4.2

2. Ethereum
Price: 2K
Market Cap: 245.3B
Daily Change: -8.1

3. Tether USD
Price: 1
Market Cap: 142.8B
Daily Change: 0.1

4. XRP
Price: 2.1
Market Cap: 124.7B
Daily Change: -8.6

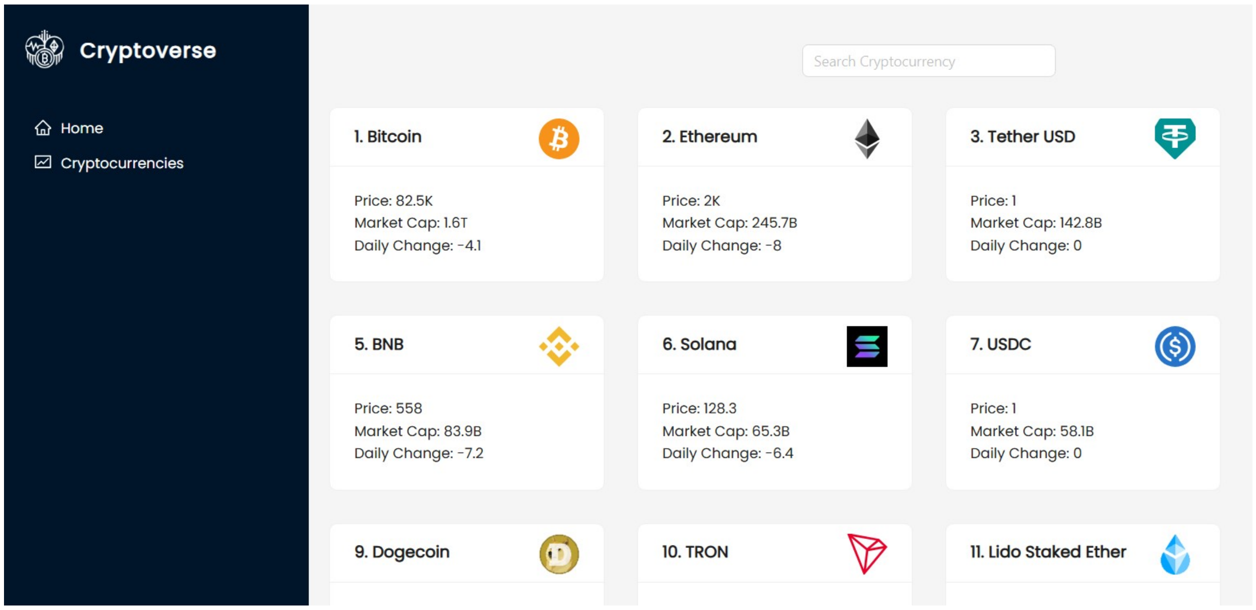
5. BNB
Price: 557.4
Market Cap: 83.8B
Daily Change: -7.3

6. Solana
Price: 128.1
Market Cap: 65.2B
Daily Change: -6.5

7. USDC
Price: 1
Market Cap: 58.2B
Daily Change: 0.1

8. Cardano
Price: 0.7
Market Cap: 27.8B
Daily Change: -8.7

Figure 12.1 Screenshot 1


Cryptoverse

Home

Cryptocurrencies

Search Cryptocurrency

1. Bitcoin
Price: 82.5K
Market Cap: 1.6T
Daily Change: -4.1

2. Ethereum
Price: 2K
Market Cap: 245.7B
Daily Change: -8

3. Tether USD
Price: 1
Market Cap: 142.8B
Daily Change: 0

5. BNB
Price: 558
Market Cap: 83.9B
Daily Change: -7.2

6. Solana
Price: 128.3
Market Cap: 65.3B
Daily Change: -6.4

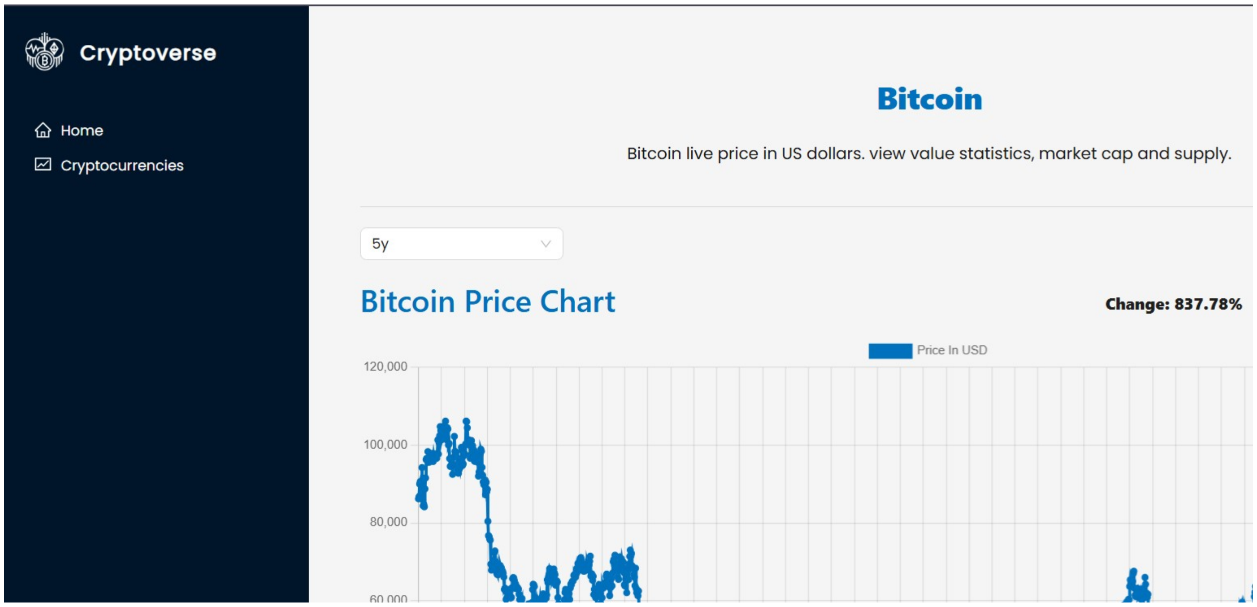
7. USDC
Price: 1
Market Cap: 58.1B
Daily Change: 0

9. Dogecoin

10. TRON

11. Lido Staked Ether

Figure 12.2 Screenshot 2



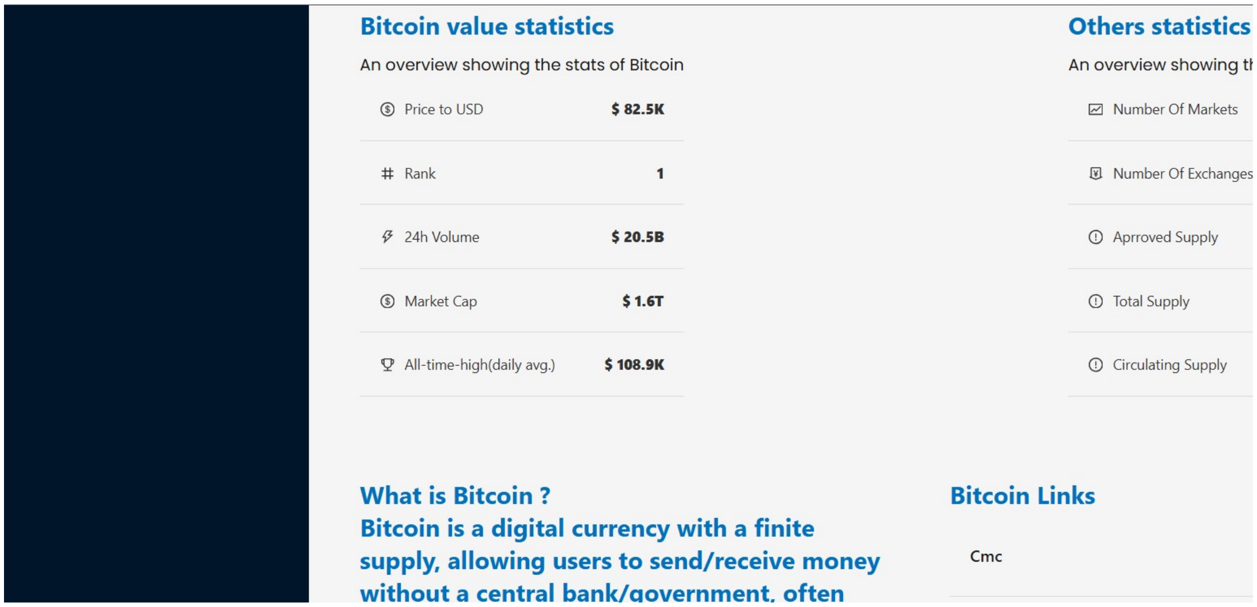
Bitcoin

Bitcoin live price in US dollars. view value statistics, market cap and supply.

Bitcoin Price Chart

Change: 837.78%

Figure 12.3 Screenshot 3



Bitcoin value statistics

An overview showing the stats of Bitcoin

Price to USD	$ 82.5K
Rank	1
24h Volume	$ 20.5B
Market Cap	$ 1.6T
All-time-high(daily avg.)	$ 108.9K

Others statistics

An overview showing th

Number Of Markets

Number Of Exchanges

Aprroved Supply

Total Supply

Circulating Supply

What is Bitcoin ?

Bitcoin is a digital currency with a finite supply, allowing users to send/receive money without a central bank/government, often

Bitcoin Links

Cmc

Figure 12.4 Screenshot 4

13. KNOWN ISSUES

1. API and Data Handling Issues:

- **Rate Limiting by API Providers:**
  Some cryptocurrency APIs might have request limits, leading to failed data fetches or incomplete information display.
- **Error Handling Gaps:**
  Certain API responses (like 500 Internal Server Error or 404 Not Found) might not be gracefully handled, resulting in empty or broken UI components.

2. UI and UX Limitations:

- **Non-Responsive Components:**
  Some UI components, especially tables or charts, may not render well on smaller devices due to limited responsive design considerations.
- **Incomplete Dark Mode Implementation:**
  If a theme switcher is present, not all components may correctly switch styles, leading to inconsistent visuals.

3. Performance Constraints:

- **Heavy Initial Load Time:**
  Large data payloads or unoptimized assets (like images or large CSS files) may slow down the initial page load.
- **Memory Usage in Charts:**
  If the dashboard displays a large dataset in charts, memory usage may spike, leading to slow rendering or browser performance issues.

4. State Management Concerns:

- **State Persistence Across Sessions:**
  The dashboard may lack proper state persistence, resulting in a reset of filters or preferences upon page reloads.
- **Redundant Re-Renders:**
  Inefficient state management may cause unnecessary component re-renders, impacting overall app performance.

5. Security Vulnerabilities:

- **Exposure of API Keys:**
  API keys might be hardcoded or not securely managed, risking exposure in public repositories.
- **Lack of Input Validation:**
  Forms or input fields may lack validation, opening the risk of invalid data submissions or XSS vulnerabilities.

6. Testing and Code Coverage Gaps:

- **Low Test Coverage:**
  Some core components, like data fetch services or critical UI components, may lack unit or integration tests.
- **Uncaught Edge Cases:**
  Certain scenarios, like network failures or unexpected API structures, may not be adequately tested, risking app stability.

14. FUTURE ENHANCEMENTS

1. Advanced Data Visualization:

- **Interactive Charts:**
  Implement more interactive charts with zooming, panning, and real-time data updates to enhance the data analysis experience.
- **Comparative Analysis:**
  Allow users to compare the performance of multiple cryptocurrencies side by side within charts.
- **Customizable Dashboards:**
  Enable users to customize their dashboard layout by selecting which widgets or data points to display.

2. Enhanced User Experience (UX):

- **Dark/Light Mode Toggle:**
  Provide users with an easy option to switch between light and dark themes for better accessibility and comfort.
- **Advanced Filtering Options:**
  Introduce more advanced filters for sorting coins by market cap, price changes, volume, or other custom criteria.
- **User Notifications:**
  Allow users to set price alerts or receive notifications when significant market events occur.

3. Performance Optimizations:

- **Lazy Loading Components:**
  Implement lazy loading for non-critical components to improve initial load time and performance.
- **Optimized Data Fetching:**
  Introduce data caching strategies and optimize API calls to reduce unnecessary network requests.
- **Reduced Bundle Size:**
  Utilize techniques like tree-shaking and code splitting to minimize the final JavaScript bundle size.

4. Security Enhancements:

- **Secure API Integration:**
  Secure sensitive data like API keys by integrating with a secure backend service or using environment variables effectively.
- **User Authentication (If Applicable):**
  Add a secure authentication system to allow personalized experiences, such as tracking personal watchlists.
- **Input Validation:**
  Ensure strict input validation on forms to prevent potential security vulnerabilities like XSS or SQL injection.