1. INTRODUCTION

PROJECT TITLE: CRYPTOVERSE

TEAM MEMBERS:

S. ROHITH VIGNESH (TEAM LEADER)
K. ARASU
H. SELVAKUMAR
K. HARIHARAN
(TEAM ID: SWTID1741172294155363)

Cryptoverse is a powerful yet user-friendly cryptocurrency dashboard designed to help investors analyze market trends over the past five years. By providing easy-to-read charts, historical data, and interactive tools, the dashboard allows users to gain valuable insights into cryptocurrency performance and make informed investment decisions.

With customizable timeframes, users can track price movements, assess risks, and compare multiple cryptocurrencies effortlessly. The dashboard highlights top-performing assets, helping investors identify potential opportunities and market trends with clarity.

Beyond investment insights, Cryptoverse serves as an educational resource, empowering users to understand the evolving nature of cryptocurrency markets and the key factors influencing price changes. Whether for beginners or experienced traders, the platform offers a comprehensive view of the crypto landscape in a simple and intuitive way.

2. PROJECT OVERVIEW

CRYPTOVERSE: A CRYPTOCURRENCY DASHBOARD

PURPOSE:

Cryptoverse is a powerful cryptocurrency analysis platform designed to provide users with real-time market data, historical trends, and in-depth insights into the world of digital assets. It empowers investors and traders with interactive tools to track price movements, assess market fluctuations, and make informed financial decisions.

Key Features

- **Comprehensive Market Data:** Provides real-time updates on cryptocurrency prices, market cap, and trading volume.
- **Historical Data & Trend Analysis:** Allows users to explore price trends over five years to identify market patterns.
- **Interactive Charts & Comparisons:** Enables seamless comparison between different cryptocurrencies using visual analytics.
- **User-Friendly Interface:** Simplifies research and decision-making with an intuitive layout and powerful search functionality.
- **Educational Resource:** Offers insights into market behavior, helping beginners and experts alike understand crypto trends.

Target Users

- **Investors & Traders:** To monitor market movements and optimize portfolio decisions.
- **Researchers & Analysts:** To study historical trends and predict future performance.
- **Crypto Enthusiasts:** To stay updated with real-time data and market insights.

3. ARCHITECTURE

COMPONENT STRUCTURE

Cryptoverse follows a **modular component-based architecture**, making it scalable and easy to maintain. The major components and their roles are outlined below:

Main Components

These components form the core of the application.

Component	Description
Navbar.jsx	The navigation bar that provides links to different pages.
Home.jsx	The homepage displaying global crypto statistics and top cryptocurrencies.

Cryptocurrencies.jsx	Lists all cryptocurrencies with price and market details.
CryptoDetails.jsx	Displays in-depth details of a selected cryptocurrency, including price charts and statistics.
LineChart.jsx	Renders the price history of a cryptocurrency in a chart format.
Loader.jsx	A loading spinner displayed while fetching data from the API.

Component Breakdown

Each component plays a specific role in rendering the application.

1. Navbar.jsx

- Contains navigation links.
- Uses **React Router** for page navigation.
- Provides easy access to Home, Cryptocurrencies, and Crypto Details.

2. Home.jsx

- Displays **global cryptocurrency stats** such as total market cap, total coins, and exchanges.
- Lists the **top 10 cryptocurrencies** fetched from the API.

3. Cryptocurrencies.jsx

- Fetches **all available cryptocurrencies** from the API.
- Displays **price, market cap, and daily change**.
- Includes a **search bar** to filter cryptocurrencies.

4. CryptoDetails.jsx

- Fetches and displays **detailed information** about a single cryptocurrency.
- Provides **historical price data** and **trading statistics**.

- Includes an **interactive chart** showing price trends.

5. LineChart.jsx

- Uses a **charting library** (like Chart.js or Recharts) to display **historical price movements**.
- Updates dynamically based on selected time periods.

6. Loader.jsx

- Displays a **loading spinner** while API data is being fetched.
- Enhances **user experience** by providing feedback.

4. SETUP INSTRUCTIONS

This section provides the necessary steps to set up the Cryptoverse project on your local system.

Prerequisites:

Before setting up the project, ensure you have the following dependencies installed:

Required Software:

- **Node.js** (LTS version recommended) - Download Here
- **npm** (Comes with Node.js) or **yarn** for package management
- **Git** for cloning the repository - Download Here
- **Code Editor** (VS Code recommended) - Download Here

Installation Steps:

Follow these steps to set up Cryptoverse on your local machine:

1. Clone the Repository

Open a terminal or command prompt and run:

git clone https://github.com/your-username/cryptoverse.git

Replace your-username with the actual GitHub repository owner if needed.

Navigate to the project folder:

cd cryptoverse

2. Install Dependencies

To install all required packages, run:

npm install

or if using **yarn**:

yarn install

3. Set Up Environment Variables

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

VITE_CRYPTO_RAPIDAPI_KEY=your-api-key
VITE_CRYPTO_RAPIDAPI_HOST=coinranking1.p.rapidapi.com
VITE_CRYPTO_BASE_URL=https://api.coinranking.com/v2

Note: Replace your-api-key with your actual RapidAPI key.

4. Start the Application

Run the following command to start the development server:

npm run dev

or if using **yarn**:

yarn dev

After running the command, the app should be accessible at:
👉 **http://localhost:5173/**

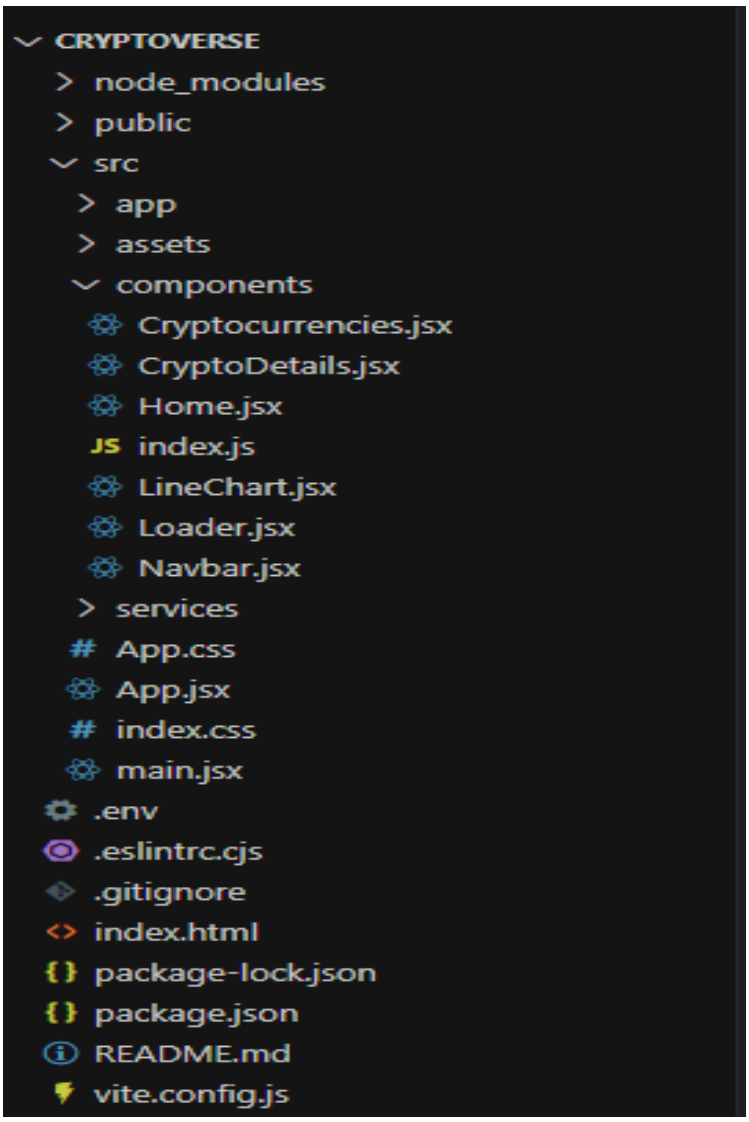
Troubleshooting:

If you encounter errors:

1. Ensure **Node.js** and **npm** are installed correctly.
2. Run npm install again if dependencies are missing.
3. Double-check your **API keys** in the .env file.
4. Restart the development server with npm run dev.

5. FOLDER STRUCTURE FOLDER STRUCTURE

To maintain a clean architecture, the project follows this folder structure:



This structured approach ensures a clean, scalable, and maintainable React application.

This section provides a detailed overview of the **Cryptoverse** project's folder structure, including key directories, their purpose, and utility functions.

Client: Organization of the React Application

The **frontend (React.js)** follows a well-structured and modular design to improve maintainability and scalability.

🗂️ **Root Directory Structure:**

```
cryptoverse/
├── node_modules/          # Installed dependencies (auto-generated)
├── public/                # Static assets
│   ├── index.html         # Root HTML file
│   ├── manifest.json      # PWA configuration (if applicable)
├── src/                   # Main source code directory
│   ├── assets/            # Images, icons, styles, fonts, etc.
│   ├── components/        # Reusable UI components
│   │   ├── Navbar.jsx     # Navigation bar
│   │   ├── CryptoCard.jsx # Displays cryptocurrency cards
│   │   ├── Loader.jsx     # Loading spinner for API requests
│   ├── pages/             # Complete page views
│   │   ├── Home.jsx       # Homepage with crypto stats
│   │   ├── CryptoDetails.jsx# Detailed view of a cryptocurrency
│   │   ├── Cryptocurrencies.jsx # List of cryptocurrencies
│   │   ├── NotFound.jsx   # 404 error page
│   ├── services/          # API services
│   │   ├── cryptoApi.js   # Handles API calls for cryptocurrency data
│   ├── store/             # Global state management (Redux Toolkit)
│   │   ├── store.js       # Redux store configuration
│   ├── App.jsx            # Root React component
│   ├── index.jsx          # Entry point of the application
│   ├── App.css            # Global styles
├── .env                   # Environment variables
├── package.json           # Dependencies and scripts
├── README.md              # Project documentation
```

Breakdown of Key Folders:

📁 **assets/**

Contains **static files** such as:

✅ **Images** (e.g., cryptocurrency icons, logos)

✅ **Fonts** (if custom typography is used)

✅ **Stylesheets** (CSS files for global styles)

📁 **components/**

Stores **reusable UI components** used across multiple pages.

- **Navbar.jsx** – Provides site-wide navigation.
- **CryptoCard.jsx** – Displays a single cryptocurrency's details in a card format.
- **Loader.jsx** – A loading animation shown when data is being fetched.

📁 pages/

Contains **full-page views** that define the application's core functionalities.

- **Home.jsx** – Displays an overview of the cryptocurrency market.
- **CryptoDetails.jsx** – Provides detailed statistics for a selected cryptocurrency.
- **Cryptocurrencies.jsx** – Lists all available cryptocurrencies.
- **NotFound.jsx** – A fallback **404 error page** for invalid routes.

📁 services/

Manages **API requests and data fetching** using **Redux Toolkit Query**.

- **cryptoApi.js** – Fetches real-time cryptocurrency data from the **CoinRanking API**.

📁 store/

Handles **global state management** with **Redux Toolkit**.=

- **store.js** – Configures the Redux store and integrates API slices.

Utilities: Helper Functions & Custom Hooks:

API Service Helper (services/cryptoApi.js)

- **Encapsulates API logic** to maintain cleaner code in components.
- Uses **Redux Toolkit Query** for efficient state management.

Error Handling Functions

- Provides a **centralized approach** to handling API response errors.

Data Formatting Functions (millify for numbers)

- **Converts large numbers** into readable formats (e.g., 1,000,000 → **1M**).

6. RUNNING THE APPLICATION

This section explains how to start the **Cryptoverse** frontend locally after installation.

Steps to Run the Application

1. Navigate to the Project Directory

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

cd cryptoverse

2. Start the Frontend Server

Run the following command inside the project folder:

npm start

This will:

- Start the **React development server**.
- Open the app in your default **web browser** (usually at **http://localhost:5173/** for Vite).
- Watch for changes and **auto-refresh** the page when you save files.

Common Issues & Fixes

If you see an error like "Module Not Found"

Run:

npm install

This installs all missing dependencies.

If the server doesn’t start or crashes

Try:

npm run dev

This is an alternative command for Vite-based projects.

If you made changes and want to restart the server

Press **Ctrl + C** in the terminal to stop the running server, then restart it with:

npm start

7. COMPONENT DOCUMENTATION

This section provides detailed documentation of the key and reusable components used in the **Cryptoverse** application.

KEY COMPONENTS:

These are the major components that define the core functionality of the application.

1. Navbar

Purpose: Displays the navigation menu for the application.
Props: None
Description:

- Contains links to **Home, Cryptocurrencies, Crypto Details** and other pages.
- Uses **React Router's <Link>** for seamless navigation.

2. Home

Purpose: Displays the main dashboard with global cryptocurrency statistics and top 10 cryptos.

Props: None

Description:

- Fetches **global crypto statistics** (total market cap, total coins, etc.).
- Shows a preview of the **top 10 cryptocurrencies** with a link to view more.

3. Cryptocurrencies

Purpose: Displays a list of all available cryptocurrencies with search functionality.

Props:

Prop Name	Type	Description
simplified	Boolean	If true, shows only the top 10 cryptos (used in Home page).

Description:

- Fetches **cryptocurrency data** from the API.
- Displays **name, price, market cap, daily change, and an image** of each coin.
- Includes **a search bar** to filter cryptocurrencies.

4. CryptoDetails

Purpose: Displays detailed information about a selected cryptocurrency.

Props:

Prop Name	Type	Description
coinId	String	The ID of the selected cryptocurrency (retrieved from the URL).

Description:

- Fetches **detailed stats, charts, and historical data** for the selected crypto.
- Shows **price trends** with a **line chart**.
- Displays **links to official sites and exchanges** for further research.

5. Loader

Purpose: Displays a loading spinner when data is being fetched.
Props: None
Description:

- Used in pages like **Cryptocurrencies, CryptoDetails, and Home** to indicate **API loading states**.

REUSABLE COMPONENTS

These are components used multiple times throughout the application.

1. StatisticCard

Purpose: Displays individual **crypto statistics** in a card format.
Props:

Prop Name	Type	Description

title	String	The name of the statistic (e.g., Market Cap).
value	String/Number	The actual value (e.g., $1.5 Trillion).

Description:

- Used in **Home** to display global crypto statistics.
- Styled with **Ant Design's <Statistic> component**.

2. SearchBar

Purpose: A **search input field** for filtering cryptocurrencies.

Props:

Prop Name	Type	Description
onSearch	Function	A callback function triggered when the user types.

Description:

- Used in **Cryptocurrencies** to filter coins.
- Uses Ant Design's <Input> component.

8. STATE MANAGEMENT

Global State Management

Approach Used: Redux Toolkit with @reduxjs/toolkit/query/react

Why Redux Toolkit?

- Manages **global data** across multiple components.

- Simplifies API calls using **RTK Query**.
- Reduces unnecessary re-renders, improving performance.

How Global State Works?

1. **API Calls Using RTK Query**

- API requests are handled inside cryptoApi.js using createApi().
- Example:
- export const cryptoApi = createApi({
- reducerPath: "cryptoApi",
- baseQuery: fetchBaseQuery({ baseUrl: import.meta.env.VITE_CRYPTO_BASE_URL }),
- endpoints: (builder) => ({
- getCryptos: builder.query({
- query: (count) => `/coins?limit=${count}`,
- }),
- }),
- });
- This allows components to **fetch data from the global store** instead of making separate API calls.

2. **Global State Access Using Hooks**

- Components retrieve data using RTK Query's hooks like useGetCryptosQuery().
- Example in **Cryptocurrencies.jsx**:
- const { data: cryptosList, isFetching } = useGetCryptosQuery(100);
- **Benefits:**
  ✅ No need for extra state variables.
  ✅ Automatically updates when data changes.

Local State Management

Why Local State?

- Used for **component-specific UI interactions** like search inputs, user selections, and toggles.

- Managed with **React's useState & useEffect hooks**.

Examples of Local State Usage

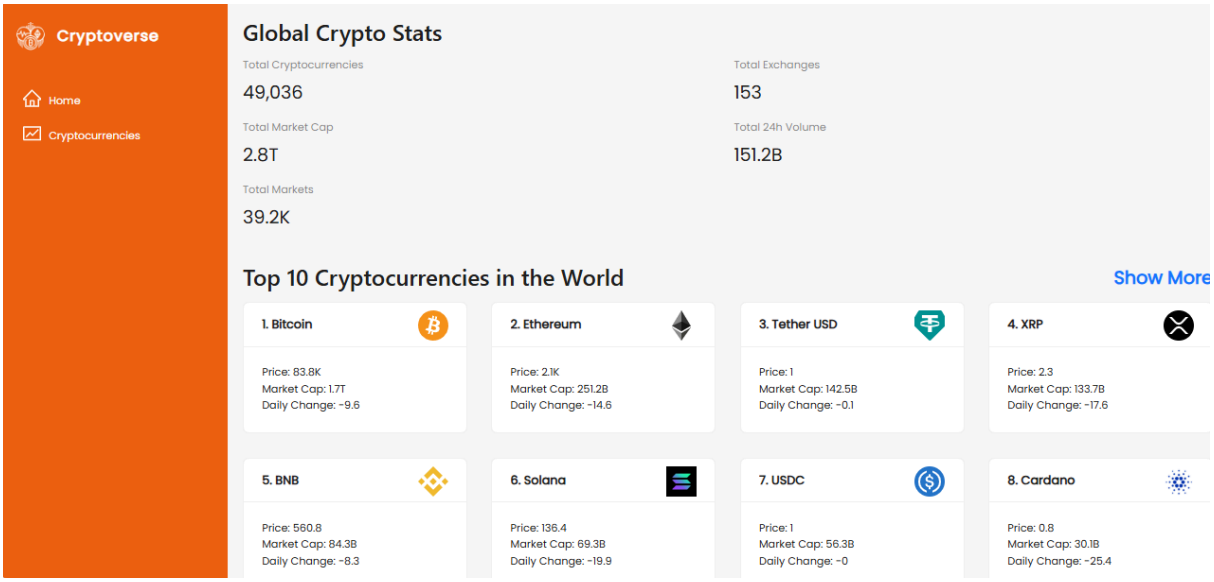
1. **Search Input in Cryptocurrencies Page**

- **Purpose:** Filters cryptocurrencies based on user input.
- **Implementation:**
- const [searchTerm, setSearchTerm] = useState("");
-
- useEffect(() => {
- const filteredData = cryptosList?.data?.coins.filter((coin) =>
- coin.name.toLowerCase().includes(searchTerm.toLowerCase())
-);
- setCryptos(filteredData);
- }, [cryptosList, searchTerm]);
- Updates the displayed list **without affecting global state**.

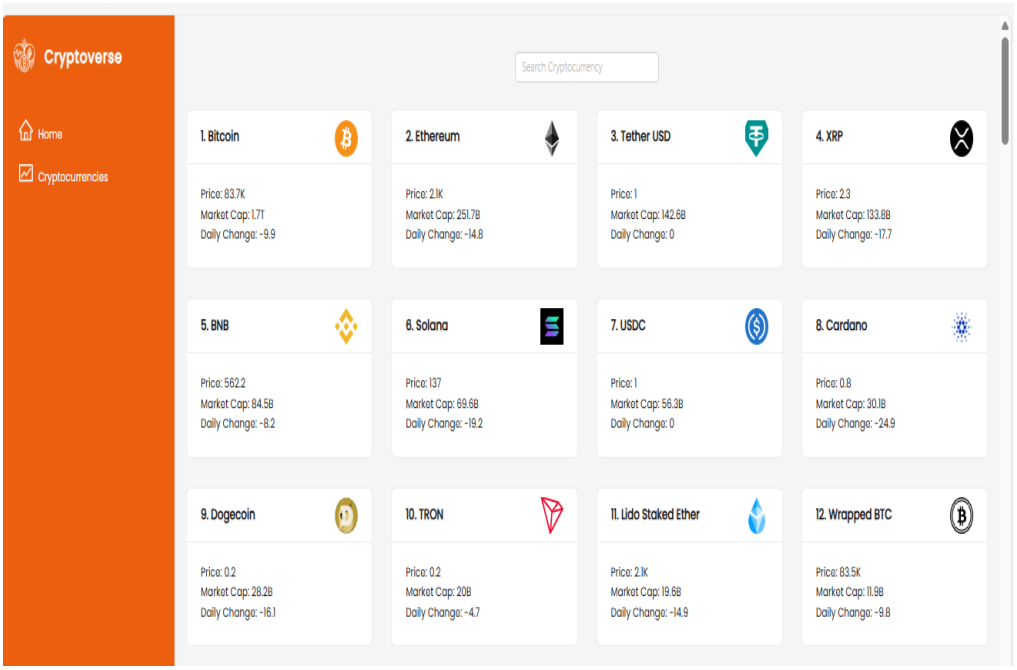
9. USER INTERFACE SNIPS:

Home Page:

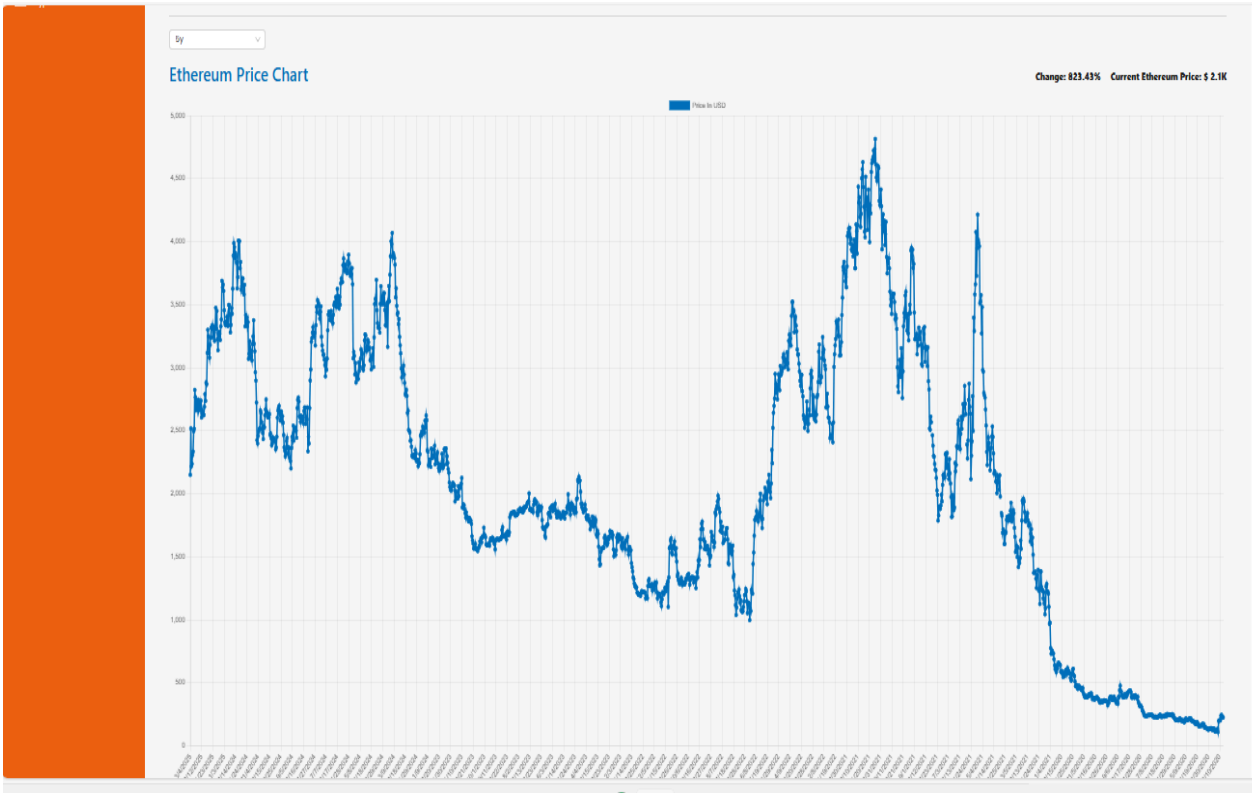
Displays global cryptocurrency statistics, including total cryptocurrencies, total exchanges, market capitalization, and 24-hour trading volume.
Showcases the top 10 cryptocurrencies with key performance indicators.
Provides an intuitive interface for users to navigate and explore further details about individual cryptocurrencies.



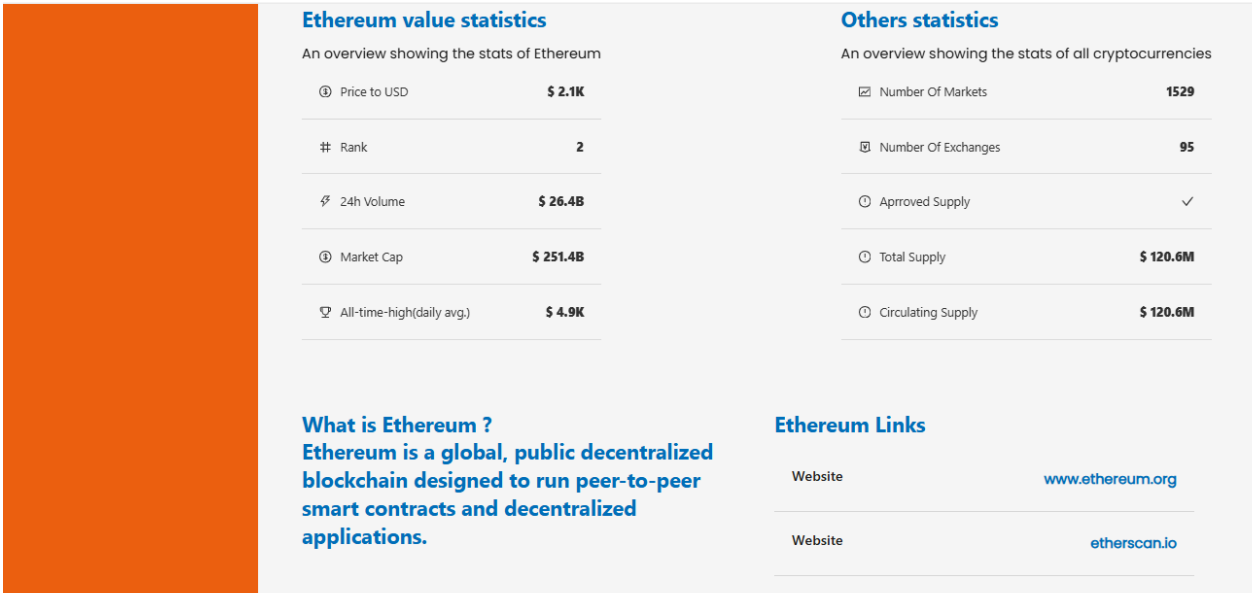
Crypto currencies page : This pages contains all cryptocurrencies which are current flow in the world. There is also a search feature where users can search and find out about their desired cryptocurrency



<u>Line Charts:</u> The **LineChart** component in **Cryptoverse** visualizes historical cryptocurrency price trends, allowing users to track market fluctuations over selectable timeframes using interactive charts.



Crypto currency details page : This page contains the line chart with data representation of price of cryptocurrencies. Also contains statistics and website links of cryptocurrencies.



10. STYLING

Styling in **Cryptoverse** ensures a visually appealing and responsive user experience. It utilizes **CSS frameworks, libraries, and custom theming** to maintain consistency and improve maintainability.

CSS Frameworks/Libraries Used

Ant Design (antd)

- Provides pre-styled UI components like buttons, cards, tables, and forms.
- Enhances UI consistency and responsiveness.
- Example:
- import { Button } from "antd";
- <Button type="primary">Click Me</Button>;
- Used for **layouts, grids, typography, and form elements**.

Custom CSS (App.css)

- Contains **global styles**, utility classes, and layout adjustments.
- Ensures custom branding and design flexibility.
- Example:
- .crypto-card {
- background-color: #1e1e2f;
- border-radius: 10px;
- padding: 15px;
- }

CSS Modules

- Scoped styling to avoid conflicts across components.
- Used for styling individual components efficiently.
- Example:
- /* Navbar.module.css */
- .navbar {
- background-color: #141414;
- padding: 10px;
- }
- import styles from "./Navbar.module.css";
- Navigation ;

Theming & Custom Design System:

Custom Colors & Fonts

- Ensures a **consistent theme across all pages**.
- Defined in **global CSS**.
- Example:
- :root {
- --primary-color: #1890ff;
- --background-dark: #1e1e2f;
- --text-light: #ffffff;
- }

Responsive Design

- Uses **CSS flexbox & media queries** for mobile-friendly layouts.
- Example:
- @media (max-width: 768px) {
- .crypto-card {
- width: 100%;
- padding: 10px;
- }
- }

11. TESTING

Testing is crucial for ensuring **the reliability, performance, and correctness** of the Cryptoverse application. The testing strategy follows a structured approach covering **unit tests, integration tests, and end-to-end (E2E) tests** using modern testing tools.

Testing Strategy:

1. **Unit Testing** (Component-Level)

- Ensures **individual React components work as expected**.
- Tested using **Jest & React Testing Library**.
- Focuses on testing UI elements, props, and behavior.
- Example: Testing the Navbar component:
- import { render, screen } from "@testing-library/react";
- import Navbar from "../components/Navbar";
- test("renders navbar component", () => {
- render(<Navbar />);
- expect(screen.getByText(/Cryptoverse/i)).toBeInTheDocument();
- });

2. **Integration Testing**

- Tests **data fetching, API calls, and interactions** between components.

- Ensures components work correctly **with Redux store and API services**.
- Example: Testing API call in cryptoApi.js:
- import { useGetCryptosQuery } from "../services/cryptoApi";
- test("fetches cryptocurrency data", async () => {
- const { data } = useGetCryptosQuery(10);
- expect(data).toBeDefined();
- expect(data.data.coins.length).toBeGreaterThan(0);
- });

3. End-to-End (E2E) Testing

- Simulates real user actions **(clicks, form inputs, navigation, etc.)**.
- Uses **Cypress** to test **full application workflows**.
- Example: Testing navigation from **Home** to **Cryptocurrencies** page:
- describe("Navigation Test", () => {
- it("should navigate to Cryptocurrencies page", () => {
- cy.visit("/");
- cy.contains("Top 10 Cryptocurrencies in the World").click();
- cy.url().should("include", "/cryptocurrencies");
- });
- });

Code Coverage:

Jest & React Testing Library

- Used for measuring **test coverage for components and functions**.
- Reports **uncovered lines, functions, branches, and statements**.
- Example command to generate coverage report:
- npm test -- --coverage

Cypress for E2E Tests

- Provides **visual test reports & debugging tools**.
- Runs tests in a **real browser environment**.

CI/CD Integration

- **GitHub Actions** or **Jenkins** can automate test execution before deployment.

This structured testing approach **ensures a bug-free, stable, and high-quality application**!

12. PROJECT DEMO

The **Cryptoverse** application is a **feature-rich cryptocurrency dashboard** that allows users to track real-time market data, analyze trends, and explore details about individual cryptocurrencies. This demo provides a **step-by-step walkthrough** of the platform, showcasing its **key features**, including:

✔ **Home Page** – Displays global market stats and trending cryptocurrencies.
✔ **Cryptocurrencies List** – Search and filter from a list of top-performing coins.
✔ **Crypto Details Page** – View price charts, historical data, and key insights.
✔ **Navigation & User Experience** – Smooth page transitions and intuitive design.
✔ **Mobile Responsiveness** – Works seamlessly on different screen sizes.

Demo link:
https://drive.google.com/file/d/1U7yQ_tEPbKMn0rPmGlxGuZF1ptSnxLaN/view?usp=drive_link

13. KNOWN ISSUES

Below are some known issues in the **Cryptoverse** application that developers should be aware of:

1. API Rate Limits & Restrictions

- Some **API calls may fail** due to **RapidAPI's rate limits** on free-tier usage.
- Possible solution: Upgrade to a **higher API tier** or implement **caching mechanisms**.

2. Page Load Performance

- Some pages **take longer to load**, especially the **Cryptocurrency Details** page when fetching large datasets.
- Possible solution: Implement **lazy loading** and optimize **API request sizes**.

3. Data Formatting Issues

- millify.js **sometimes fails** if the API returns null or undefined values.
- Possible solution: Add **null-checking conditions** before formatting numbers.
- value ? millify(value) : "N/A";

4. Inconsistent UI Styling on Dark Mode

- Some text and icons **are not visible** properly in **dark mode**.
- Possible solution: Ensure **proper contrast** and use a **consistent color theme** across all components.

5. Lack of Mobile Responsiveness

- Some UI elements **do not scale well** on smaller screens.
- Possible solution: Improve **CSS media queries** and **flexbox/grid usage**.

14. FUTURE ENHANCEMENTS

The following features are planned to **enhance the Cryptoverse platform** in future updates:

1. Advanced Filtering & Sorting

- Users will be able to **sort cryptocurrencies** based on **market cap, price change, volume, etc.**
- Filters for **specific categories (DeFi, Metaverse, Stablecoins, etc.)**

2. Multi-Currency Support

- Option to view prices in **multiple fiat currencies (USD, EUR, GBP, INR, etc.)**.

3. Live Price Alerts

- Implement **real-time notifications** when a cryptocurrency reaches a **target price**.

4. Interactive Charts & Historical Data

- More **customizable charting options** for deeper market analysis.
- View **historical trends** in different timeframes **(hourly, daily, weekly, yearly)**.

5. Improved UI/UX with Animations

- Smooth **loading animations** and **better transitions**.
- More **interactive elements** for enhanced **user experience**.

6. Mobile App Version

- A **React Native app** for a **seamless experience on mobile devices**.

By implementing these **enhancements**, Cryptoverse aims to **offer a more robust, user-friendly, and feature-rich cryptocurrency tracking experience**!