# Cryptoverse: A Cryptocurrency Dashboard

**DR.MGR JANAKI COLLEGE OF ARTS AND SCIENCE FOR WOMEN**
**(B.Sc., Computer Science- Final Year)**

## Presented By:

**Team Leader:** Joycilin.V**(**asunm1423222207998)

Mail-id : joycilindass566@gmail.com

**Team Members:**

1) Karunya. K.G.(asunm1423222208000)

Mail-id: rajeekarunya@gmail.com

2) Mangaleshwari. A (asunm1423222208003)

Mail-id: mangaleshwari0180@gmail.com

3) Marish. G (asunm1423222208004)

Mail-id: taervragavi1995@gmail.com

# Introduction

A cryptocurrency dashboard showcasing historical price trends over the past five years is a valuable tool for investors seeking deeper market insights. This interactive platform provides a comprehensive view of cryptocurrency performance, enabling thorough analysis and informed decision-making. With intuitive charts and graphs, users can compare multiple digital assets, identify topper formers, and track overall market trends. Customizable time frames allow for precise examination of price fluctuations,supporting volatility assessments and risk evaluations. By leveraging historical data, investors can recognize recurring patterns and market cycles, enhancing their strategic decisions. Additionally, the dashboard serves as an educational resource, helping users understand the evolving cryptocurrency landscape and the key factors influencing price movements.

# Description

Cryptoverse is an advanced cryptocurrency dashboard that offers investors in-depth market insights through a five-year historical price analysis. Equipped with interactive charts, user-friendly tools, and smooth navigation, the platform enables users to track top-performing assets and make well-informed investment choices. Its powerful search functionality allows easy exploration and comparison of various cryptocurrencies over time. Beyond portfolio optimization, Cryptoverse serves as an educational resource, helping users grasp the ever-changing cryptocurrency landscape.

## Scenario

Riya, a passionate trader, wants to analyze historical cryptocurrency price data to make informed investment decisions.

1. Goal: She aims to identify cryptoassets with steady growth over the past five years to diversify her portfolio.

2. Exploring Cryptoverse: Riya launches the Cryptoverse platform on her computer.

3. Seamless Navigation: With the help of react-router-dom, she easily navigates to the “Cryptocurrencies” section to explore various digital assets.

4. Browsing Assets: Riya browses a diverse selection of cryptocurrencies, from popular options like Bitcoin and Ethereum to lesser-known altcoins.

5. Visual Insights: Each cryptocurrency is presented with intuitive charts displaying historical price trends, allowing Riya to quickly assess past performance.

6. Interactive Analysis: She clicks on Bitcoin's chart to access detailed historical data, usingreact-chartjs-2 and Chart.js to adjust timeframes and analyze price movements over five years.

7. Comparing Trends: Observing Bitcoin's volatility, she compares its performance with other cryptocurrencies to identify promising investment opportunities.

8. Search Functionality :Using the search tool, Riya quickly locates specific assets like Bitcoin and Ripple, reviewing their historical trends for better comparison.

9. Investment Decisions: After analyzing multiple assets, Riya identifies cryptocurrencies with consistent growth and decides to add them to her portfolio.

10. Educational Value: Beyond data analysis, she finds Cryptoverse to be an informative resource, deepening her understanding of market trends and price influencers.

11. Feedback &Community: Impressed by Cryptoverse's capabilities, Riya shares her feedback for future enhancements and introduces the platform to fellow traders.

## Pre-requisites:

Here are the key prerequisites for developing a frontend application using React.js:✔
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. Can you rewrite this paragraph using different new words and remove the unwanted lines

- Download:https://nodejs.org/en/download/
- Installationinstructions: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.

- Create a new React app:

npx create-react-app my-react-app

Replace my-react-app with your preferred project name.

- Navigate to the project directory: cd my-react-app
- Running the React App:

With the React app created, you can now start the development server and see your React application in action.

- Start the development server:

npm start

This command launches the development server, and you can access your React app at http://localhost:3000 in your web browser.

✔HTML, CSS, and JavaScript : Basic knowledge of HTML for creating the structure of your app, CSS for styling, and JavaScript for client-side interactivity is essential.

✔Version Control: Use Git for version control, enabling collaboration and tracking changes throughout the development process. Platforms like GitHub or Bitbucket can host your repository.

- Git: Download and installation instructions can be found at: https://gitscm.com/downloads

✔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✔Getthe code from google drive:
- Download the code from the drive link given below:

Drive_link:

https://drive.google.com/drive/folders/1p_YleWU4shUBALbnM_HGED5bA2HWbYT5?usp=drive_link

✔Clone the code from github repository:

Follow below steps:

Git repository: https://github.com/SSC369/cryptoverse

Git clone command: git clonehttps://github.com/SSC369/cryptoverse Use this command to clone code into your project folder.

Install Dependencies:

• Navigate into the cloned repository directory and install

libraries: cd crypto npm

install

✔Start the Development Server:

• To start the development server, execute the following command: npm run dev(vite)

or npm start Access the App:

• Open your web browser and navigate to http://localhost:3000.

• You should see the Cryptoverse app's homepage, indicating that the installation and setup were successful.

You have successfully installed and setup the application on your local machine.You can now proceed with further customization, development, and testing as needed.

## Project Structure:


## Project Flow:

- Project setup and configuration:

1. Setup React Application:

• Create a React app in the client folder.
• Install required libraries

• Create required pages and components and add routes

2. Design UI components:

• Create Components.

• Implement layout and styling.

• Add navigation.

3. Implement frontend logic:

• Integration with API endpoints.

• Implement data binding.

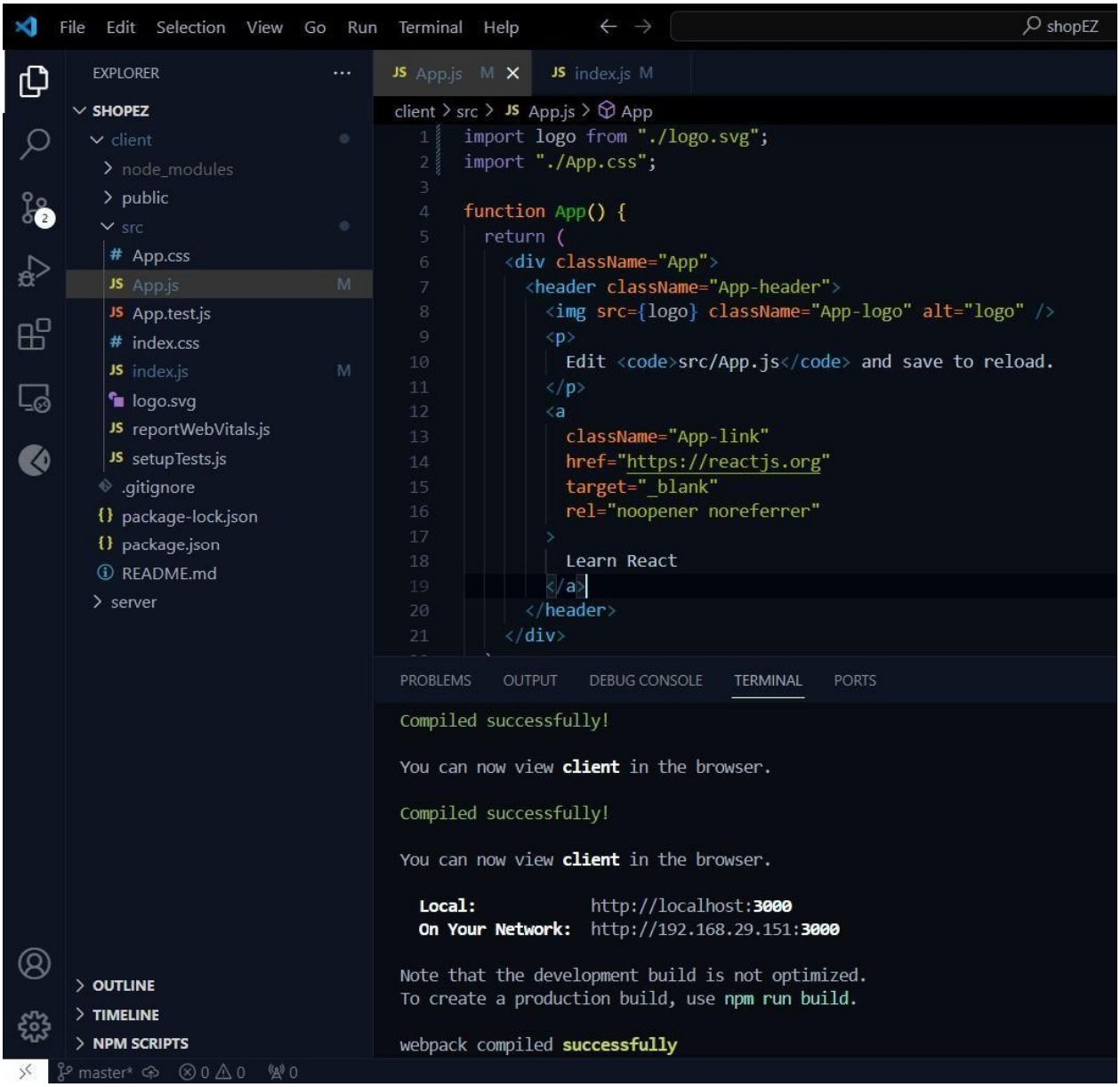
Reference Video Link:

https://drive.google.com/file/d/1EokogagcLMUGiIluwHGYQo65x8GRpDcP/view?usp=sharing

Reference Article Link:

https://www.w3schools.com/react/react_getstarted.asp

Reference Image:




## Project Development:

- Create a redux store:

1. Import {configureStore} from"@reduxjs/toolkit";: This line imports the configureStore function from Redux Toolkit. Redux Toolkit is a package that provides utilities to simplify Redux development, making it easier to write Redux logic with less boilerplate code.

2. Import {configureStore} from"@reduxjs/toolkit";: This line imports the configureStore function from Redux Toolkit. Redux Toolkit is a package that provides utilities to simplify Redux development, making it easier to write Redux logic with less boilerplate code.

3. Import {cryptoApi} from"../services/cryptoApi";: This line imports the cryptoApi object from the cryptoApi.js file located in the../services directory. This object likely contains configurations and functions related to making API requests for cryptocurrency data.

4. export default configureStore({...});: This line exports the Redux store configuration created by the configureStore function as the default export of this module.

5. reducer:{[cryptoApi.reducerPath]:cryptoApi.reducer}: This part of the configuration specifies the root reducer for the Redux store. In this case, it sets the cryptoApi. reducer as the reducer for the slice of state managed by the cryptoApi API slice.ThecryptoApi.reducer Path likely refers to the slice's unique identifier,which is used internally by Redux Toolkit.

6. middleware:(getDefaultMiddleware)=>

getDefaultMiddleware().concat(cryptoApi.middleware),: This part of the configuration specifies middleware for the Redux store. Middleware intercepts actions before they reach the reducers and can be used for various purposes, such as logging, asynchronous actions, or handling API requests. Here, it uses the getDefaultMiddleware function provided by Redux Toolkit to get the default middleware stack and appends the cryptoApi.middleware. This middleware likely handles asynchronous API requests and dispatches corresponding actions based on the API response.

this configuration sets up a Redux store with a specific reducer and middleware provided by the cryptoApi object, which presumably manages state related to cryptocurrency data fetched from an external API.This set up allows you to manage and interact with this data using Redux within your React application.

```
import { configureStore } from "@reduxjs/toolkit";
import { cryptoApi } from "../services/cryptoApi";

export default configureStore({
  reducer: {
    [cryptoApi.reducerPath]: cryptoApi.reducer,
  },
  middleware: (getDefaultMiddleware) =>
    getDefaultMiddleware().concat(cryptoApi.middleware),
});
```

- Create a API slice using Redux toolkit's:

1. Import Statements:
- import { createApi, fetchBaseQuery } from "@reduxjs/toolkit/query/react";: This line imports the necessary functions from Redux Toolkit's query-related module.create Api is used to create an API slice, while fetch Base Query is a utility function provided by Redux Toolkit for making network requests using fetch.

2. Header and Base URL Configuration:
- const cryptoApi Headers ={ ... }: This object contains headers required for making requests to the cryptocurrencyAPI. The values for "X-RapidAPI-Key"and "X-RapidAPI-Host" are retrieved from environment variables using import.meta.env.
- const base Url= import.meta.env.VITE_BASE_URL;: This variable holds the base URL for the cryptocurrency API, which is also retrieved from environment variables.

3. Request Creation Function:
- Const create Request=(url)=>({url,headers:cryptoApiHeaders});:This function create Request takes a URL and returns an object with the URL and headers required for making a request. It utilizes cryptoApi Headers to include necessary headers in the request.

4. Create API Slice:
- export const cryptoApi = createApi({ ... }): This part uses the create Api function to create an API slice named cryptoApi. It takes an object with several properties:

- reducer Path:Specifies the path under which the slice's reducer will be mounted in the Redux store.

- Base Query:Configures the base query function used by the API slice .In this case, it uses fetch Base Query with the base URL specified.
- End points: Defines the API endpoints available in the slice.It's an object with keys corresponding to endpoint names and values being endpoint definitions.

5. API Endpoints:
- getCryptos, getCryptoDetails, getCryptoHistory: These are endpoints defined using the builder.querymethod. Each endpoint is configured with a query function that returns the request configuration object created by createRequest.

6. Exporting Hooks:

-`export const { ... }`: This line exports hooks generated by the `createApi` function, allowing components to easily fetch data from the API slice. Each hook corresponds to an endpoint defined in the `endpoints` object.

Overall, this code sets up an API slice named`cryptoApi`using ReduxToolkit's query functionality. It defines endpoints for fetching cryptocurrencies, cryptocurrency details, and cryptocurrency history. The slice is configured with base URL, headers, and query functions required for making requests to the cryptocurrency API.

```
import { createApi, fetchBaseQuery } from "@reduxjs/toolkit/query/react";

const cryptoApiHeaders = {
  "X-RapidAPI-Key": import.meta.env.VITE_RAPID_API_KEY,
  "X-RapidAPI-Host": import.meta.env.VITE_RAPID_API_HOST,
};

const baseUrl = import.meta.env.VITE_BASE_URL;

const createRequest = (url) => ({ url, headers: cryptoApiHeaders });

export const cryptoApi = createApi({
  reducerPath: "cryptoApi",
  baseQuery: fetchBaseQuery({ baseUrl }),
  endpoints: (builder) => ({
    getCryptos: builder.query({
      query: (count) => createRequest(`/coins?limit=${count}`),
    }),

    getCryptoDetails: builder.query({
      query: (coinId) => createRequest(`/coin/${coinId}`),
    }),
    getCryptoHistory: builder.query({
      query: ({ coinId, timePeriod }) =>
        createRequest(`coin/${coinId}/history?timePeriod=${timePeriod}`),
    }),
  }),
});

export const {
  useGetCryptosQuery,
  useGetCryptoDetailsQuery,
  useGetCryptoHistoryQuery,
} = cryptoApi;

```

- **Adding Providers in the main function:**

React Router with `BrowserRouter`:

- `<BrowserRouter>`:This component is provided by `react-router-dom` and enables client-side routing using the HTML5 history API. It wraps the application, allowing it to use routing features.

Redux Provider:

- `<Provider store={store}>`: This component is provided by `react-redux` and is used to provide the Redux store to the entire application. It wraps the application, allowing all components to access the Redux store.

Overall, this code initializes the React application by rendering the root component (`<App />`) into the DOM, while also providing routing capabilities through`BrowserRouter`and state management with Redux through`Provider`. Additionally, it ensures stricter development mode checks with `<React.StrictMode>`.

```
import React from "react";
import ReactDOM from "react-dom/client";
import App from "./App.jsx";
import "./index.css";
import { BrowserRouter } from "react-router-dom";
import { Provider } from "react-redux";
import store from "./app/store.js";

ReactDOM.createRoot(document.getElementById(
"root")).render(
  <React.StrictMode>
    <BrowserRouter>
      <Provider store={store}>
        <App />
      </Provider>
    </BrowserRouter>
  </React.StrictMode>
);

```

- **Creating a Linechart component:**

This code defines a React component called `LineChart` which renders a line chart using the `react-chartjs-2` library.

1. Imports:

- `import React from "react";`: Imports the `React` module.

- `import{Line}from"react-chartjs-2";`:Imports the `Line` component from the `react-chartjs-2`library, which is used to render linecharts.

- `import {Col, Row, Typography} from"antd";`: Imports specific components from the Ant Design library, including `Col`, `Row`, and `Typography`.

- `const{Title}=Typography;`: Destructures the `Title` component from the `Typography` module.

2. Component Definition:

- `const LineChart= ({coinHistory,currentPrice,coinName})=> {...}`: Defines a functional component called `LineChart`. It takes three props: `coinHistory`, `currentPrice`,and`coinName`.

3. Data Preparation:

- Inside the component, it loops through the `coinHistory` data to extract `coinPrice `and` coinTimestamp` arrays. These arrays will be used as data points for the line chart.

4. Chart Data:

- `const data= {...}`: Defines the data object for the linechart. It includes labels (timestamps) and datasets (coin prices).

5. Rendering:

- Inside the return statement, it renders the chart header, including the coin name, price change, and current price.

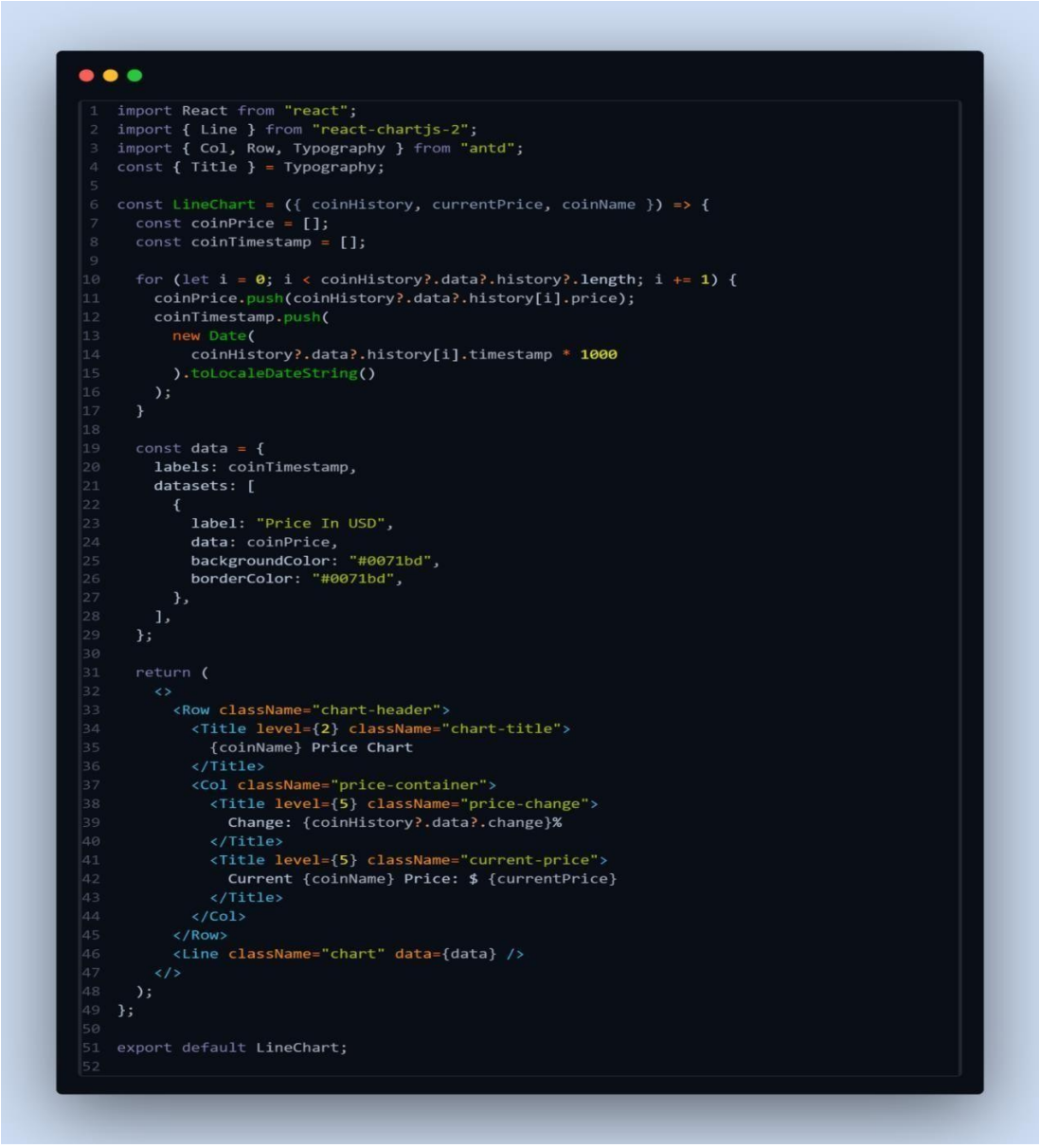
- `Row` and `Col` from Ant Design are used to structure the layout.

- The `Line `component renders the actual linechart using the data object defined earlier.

6. Export:

- `export default LineChart;`: Exports the `LineChart` component as the default export.

Overall, this component receives historical data (`coinHistory`), current price (`currentPrice`), and the name of the cryptocurrency (`coinName`) as props, and renders a linechart displaying the historical price data. It also includes additional information such as the price change and current price displayed above the chart



```jsx
import React from "react";
import { Line } from "react-chartjs-2";
import { Col, Row, Typography } from "antd";
const { Title } = Typography;

const LineChart = ({ coinHistory, currentPrice, coinName }) => {
  const coinPrice = [];
  const coinTimestamp = [];

  for (let i = 0; i < coinHistory?.data?.history?.length; i += 1) {
    coinPrice.push(coinHistory?.data?.history[i].price);
    coinTimestamp.push(
      new Date(
        coinHistory?.data?.history[i].timestamp * 1000
      ).toLocaleDateString()
    );
  }

  const data = {
    labels: coinTimestamp,
    datasets: [
      {
        label: "Price In USD",
        data: coinPrice,
        backgroundColor: "#0071bd",
        borderColor: "#0071bd",
      },
    ],
  };

  return (
    <>
      <Row className="chart-header">
        <Title level={2} className="chart-title">
          {coinName} Price Chart
        </Title>
        <Col className="price-container">
          <Title level={5} className="price-change">
            Change: {coinHistory?.data?.change}%
          </Title>
          <Title level={5} className="current-price">
            Current {coinName} Price: $ {currentPrice}
          </Title>
        </Col>
      </Row>
      <Line className="chart" data={data} />
    </>
  );
};

export default LineChart;
```

- ## Creating cryptocurrencies component:

1. Component Definition:

- `const Cryptocurrencies = ({ simplified }) => { ... }`: Defines a functional component named`Cryptocurrencies`. It accepts a prop named `simplified`, which determines whether to display a simplified version of the list.

2. Initialization:

- `const count=simplified?10:100;`: Initializes the `count` variable based on the value of the `simplified` prop. If `simplified` is true, `count` is set to 10; otherwise, it's set to 100.

3. Fetching Cryptocurrency Data:

- `const { data: cryptosList, isFetching } = useGetCryptosQuery(count);`: Uses the `useGetCryptosQuery` hook provided by the`cryptoApi`service to fetch cryptocurrency data. It retrieves the list of cryptocurrencies (`cryptosList`) and a boolean flag (`isFetching`) indicating whether the data is being fetched.

4. Filtering Cryptocurrency Data:

- The `useEffect` hook is used to filter the cryptocurrency data based on the `searchTerm`statevariable.Itupdatesthe`cryptos`state with filtered data whenever `cryptosList` or `searchTerm` changes.

5. Rendering Loader:

- `if (isFetching) return <Loader />;`: If data is still being fetched (`isFetching`istrue), it returns a `Loader` component to indicate that the data is loading.

6. Rendering Search Input:

- `!simplified&&(...)`:If`simplified` is false, it renders a search input field allowing users to search for specific cryptocurrencies by name.

7. Rendering Cryptocurrency Cards:

- The `Row` and `Col` components from Ant Design are used to create a grid layout for displaying cryptocurrency cards.

- For each cryptocurrency in the `cryptos` array, it renders a `Card` component containing details such as name, price, market cap, and daily

change. Each card is wrapped in a `Link` component, allowing users to navigate to the details page of a specific cryptocurrency.

8. Return Statement:

- `return(...)`:Returns JSX representing the component's structure and content.

Overall, this component fetches cryptocurrency data ,filters it based on a search term, and renders the data in a visually appealing format with card-based UI. It also provides a search functionality for users to find specific cryptocurrencies.

```jsx
import React, { useEffect, useState } from "react";
import millify from "millify";
import { Link } from "react-router-dom";
import { Card, Row, Col, Input } from "antd";
import { useGetCryptosQuery } from "../services/cryptoApi";
import Loader from "./Loader";

const Cryptocurrencies = ({ simplified }) => {
  const count = simplified ? 10 : 100;
  const { data: cryptosList, isFetching } = useGetCryptosQuery(count);
  const [cryptos, setcryptos] = useState([]);
  const [searchTerm, setsearchTerm] = useState("");

  useEffect(() => {
    const filteredData = cryptosList?.data?.coins.filter((item) =>
      item.name.toLowerCase().includes(searchTerm.toLowerCase())
    );
    setcryptos(filteredData);
  }, [cryptosList, searchTerm]);

  if (isFetching) return <Loader />;

  return (
    <>
      {!simplified && (
        <div className="search-crypto">
          <Input
            placeholder="Search Cryptocurrency"
            onChange={(e) => setsearchTerm(e.target.value)}
          />
        </div>
      )}
      <Row gutter={[32, 32]} className="crypto-card-container">
        {cryptos?.map((currency) => {
          return (
            <Col
              xs={24}
              sm={12}
              lg={6}
              className="crypto-card"
              key={currency.uuid}
            >
              <Link key={currency.uuid} to={`/crypto/${currency.uuid}`}>
                <Card
                  extra={
                    <img classname="crypto-image" src={currency.iconUrl} />
                  }
                  title={`${currency.rank}. ${currency.name}`}
                >
                  <p>Price: {millify(currency.price)}</p>
                  <p>Market Cap: {millify(currency.marketCap)}</p>
                  <p>Daily Change: {millify(currency.change)}</p>
                </Card>
              </Link>
            </Col>
          );
        })}
      </Row>
    </>
  );
};

export default Cryptocurrencies;
```

## Create a component to show the details of cryptocurrency:

This code defines a React functional component called`CryptoDetails`responsible for displaying detailed information about a specific cryptocurrency. Let's break down the code:

1. Component Definition:

- `const CryptoDetails=()=>{...}`: Defines a functional component named `CryptoDetails`. It doesn't accept any props directly but utilizes React Router's `useParams` hook to get the `coinId` parameter from the URL.

2. State Initialization:

- Initializes state variables `timePeriod` and `coinHistory`. `timePeriod` represents the selected timeperiod for displaying cryptocurrency history, and `coinHistory` stores historical data of the selected cryptocurrency.

3. Fetching Data:

- Utilizes`useGetCryptoDetailsQuery`and`useGetCryptoHistoryQuery`hooks provided by the `cryptoApi` service to fetch details and historical data of the cryptocurrency specified by `coinId`. It uses `coinId` obtained from `useParams` to fetch data for the specific cryptocurrency.

4. Setting Coin History:

- Utilizes`useEffect` hook to update the`coinHistory` state when `coinHistoryData` changes. This ensures that the component re-renders with updated historical data.

5. Rendering Loader:

- Displays a loading indicator (`<Loader/>`) while data is being fetched (`isFetching` is true).

6. Time Period Selection:

- Renders a `Select` component allowing users to choose the time period for displaying historical data. It triggers the `setTimePeriod` function when the selection changes.

7. Rendering Line Chart:

- Utilizes the `LineChart` component to display the historical price trend of the cryptocurrency over the selected time period.

8. Rendering Statistics:

- Displays various statistics related to the cryptocurrency, such as price, rank, volume, market cap, etc. The statistics are displayed in two sections: `stats` and `genericStats`.

9. Rendering Description and Links:

- Parses and displays the description of the cryptocurrency using `HTMLReactParser`.

- Renders links related to the cryptocurrency, such as official websites, social media, etc.

10. Return Statement:

- Returns JSX representing the structure and content of the component.

Overall, this component fetches and displays detailed information about a specific cryptocurrency, including historical price data, key statistics, description, and related links.

**Reference Image Link :**details.png

- **Create a Homepage:**

This component, named `Home`, is a React functional component responsible for rendering the home page of the cryptocurrency dashboard. Let's break down the code:

1. Component Definition:

- `const Home=()=>{...}`: Defines a functional component named `Home`.

2. Data Fetching:

- Uses the `useGetCryptosQuery `hook provided by the`cryptoApi` service to fetch data for the top 10 cryptocurrencies. It retrieves data and a boolean flag indicating whether data is being fetched.

3. Rendering Loader:

- Displays a loading indicator (`<Loader/>`) while data is being fetched (`isFetching` is true).

4. Global Crypto Stats:

- Renders statistics about the global cryptocurrency market, including total cryptocurrencies, total exchanges, total market cap, total 24-hour volume, and total markets. These statistics are displayed using the `Statistic` component from Ant Design.

5. Top 10 Cryptocurrencies:

- Renders a section displaying the top 10 cryptocurrencies in the world.

- Utilizes the `Cryptocurrencies` component with the `simplified` prop set to true to display a simplified version of the list.

- Provides a link to view more cryptocurrencies using the `Link` component from React Router.

6. Return Statement:

- Returns JSX representing the structure and content of the component.

Overall, this component fetches and displays global cryptocurrency statistics and thetop10cryptocurrenciesonthehomepageofthedashboard.Itprovideslinksfor users to navigate to the full list of cryptocurrencies.

```jsx
import React from "react";
import milify from "millify";
import { Typography, Row, Col, Statistic } from "antd";
import { Link } from "react-router-dom";
const { Title } = Typography;
import { useGetCryptosQuery } from "../services/cryptoApi";
import Cryptocurrencies from "./Cryptocurrencies";
import Loader from "./Loader";

const Home = () => {
  const { data, isFetching } = useGetCryptosQuery(10);
  if (isFetching) return <Loader />;
  const globalStats = data?.data?.stats;

  return (
    <>
      <Title level={2} className="heading">
        Global Crypto Stats
      </Title>
      <Row>
        <Col span={12}>
          <Statistic title="Total Cryptocurrencies" value={globalStats.total} />
        </Col>
        <Col span={12}>
          <Statistic
            title="Total Exchanges"
            value={milify(globalStats.totalExchanges)}
          />
        </Col>
        <Col span={12}>
          <Statistic
            title="Total Market Cap"
            value={milify(globalStats.totalMarketCap)}
          />
        </Col>
        <Col span={12}>
          <Statistic
            title="Total 24h Volume"
            value={milify(globalStats.total24hVolume)}
          />
        </Col>
        <Col span={12}>
          <Statistic
            title="Total Markets"
            value={milify(globalStats.totalMarkets)}
          />
        </Col>
      </Row>
      <div className="home-heading-container">
        <Title level={2} className="home-title">
          Top 10 Cryptocurrencies in the world
        </Title>
        <Title level={3} className="show-more">
          <Link to="/cryptocurrencies">Show More</Link>
        </Title>
      </div>
      <Cryptocurrencies simplified />
    </>
  );
};

export default Home;

```

## Project Execution:

Here is the reference video link of react application execution
https://drive.google.com/file/d/1ZVccy5-RPXbYthwZK__PinMUYyWAlb4/view?usp=sharing

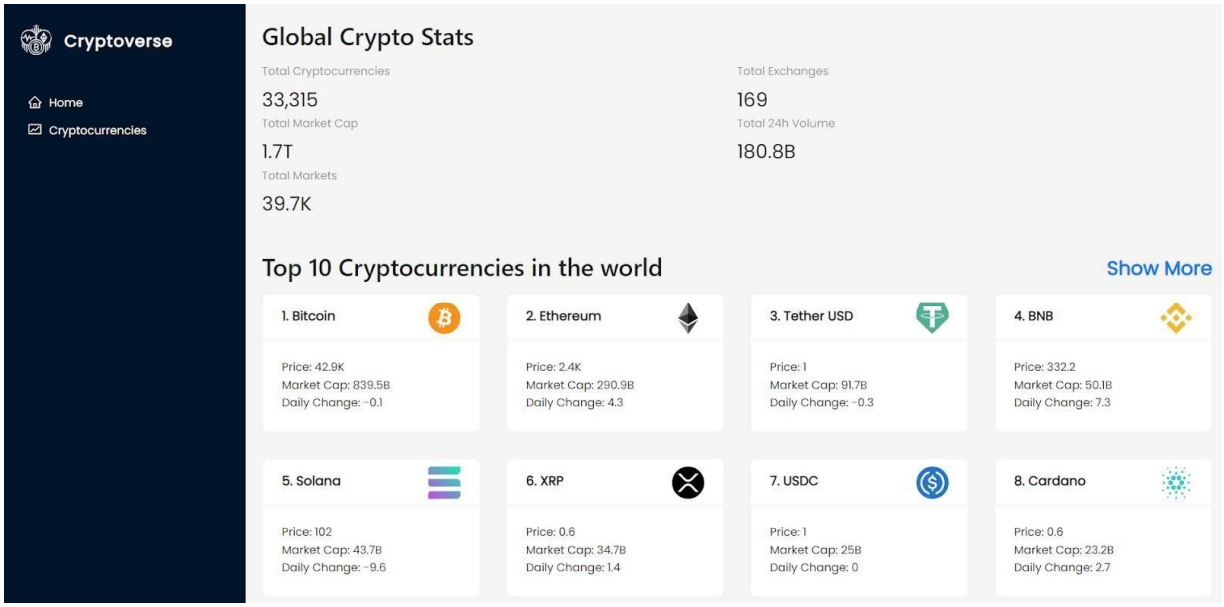
## Project demo:

Demo Link:

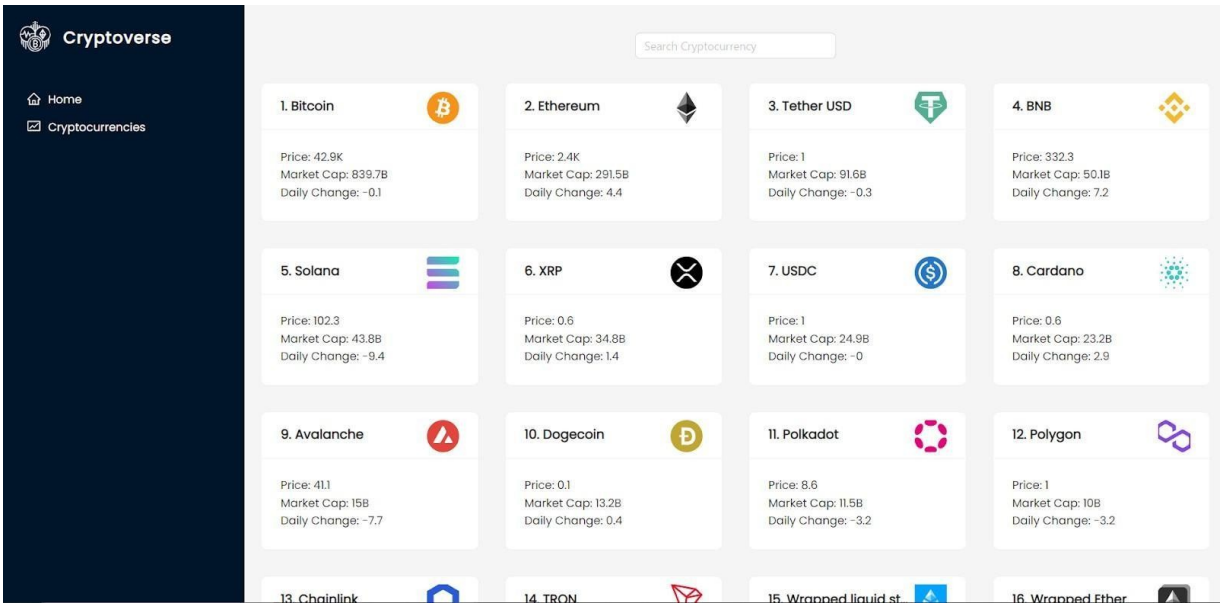
https://drive.google.com/drive/folders/1aNcgUKfY78gljNAMVe45loZaJMOgvQLH

User Interface snips:

➢ Home page : This pages consists of stats of global crypto like total cryptocurrencies, total exchanges, market cap etc. Also consist of top 10 cryptocurrencies in the world.



➢ Cryptocurrencies page: This pages contains all cryptocurrencies which are currently inflow in the world. There is also a search feature where users can search and find out about their desired cryptocurrency.

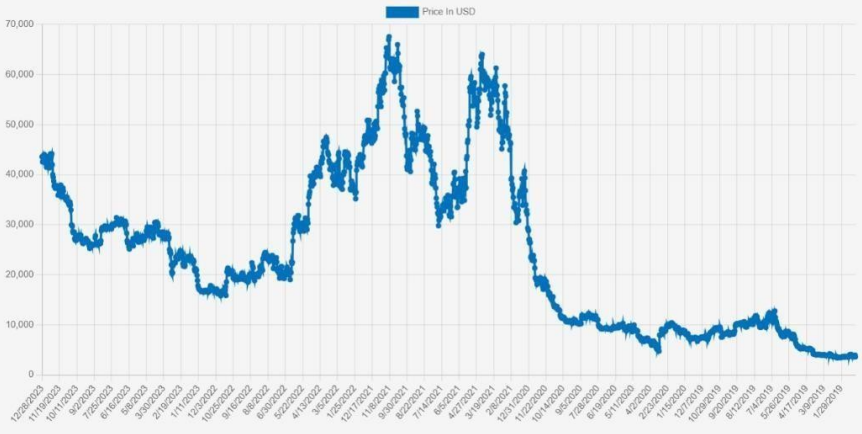



➢ Cryptocurrency details page: This page contains the line chart with data representation of price of cryptocurrencies. Also contains statistics and website links of cryptocurrencies.

5y

## Bitcoin Price Chart

**Change: 1084.17%** **Current Bitcoin Price: $ 42.9K**




### Bitcoin value statistics

An overview showing the stats of Bitcoin

| | |
|---|---|
| Price to USD | $ 42.9K |
| Rank | 1 |
| 24h Volume | $ 22.8B |
| Market Cap | $ 840.4B |
| All-time-high(daily avg.) | $ 68.8K |

### Others statistics

An overview showing the stats of all cryptocurrencies

| | |
|---|---|
| Number Of Markets | 3212 |
| Number Of Exchanges | 121 |
| Approved Supply | ✓ |
| Total Supply | $ 19.6M |
| Circulating Supply | $ 19.6M |

### What is Bitcoin ?

**Bitcoin is a digital currency with a finite supply, allowing users to send/receive money without a central bank/government, often nicknamed "Digital Gold".**

### Bitcoin Links

| | |
|---|---|
| Website | bitcoin.org |
| Website | bitcoinmagazine.com |
| Bitcointalk | bitcointalk.org |