CRYPTOVERSE

(CryptocurrencyDashboard)

- **TeamDetails:**

TEAMLEADER	EMAILID
VENKATESANB	venkatesanbabu2005@gmail.com

TEAMMEMBERS	EMAILID
PANNEERSELVAMT	panneerselvam2708@gmail.com
DHARMAB	beastdharma475@gmail.com
SUDEEPS	vbsudeep1@gmail.com

CRYPTOVERSE:

Introduction:

The cryptoverse refers to the expansive digital ecosystem surrounding cryptocurrenciesandblockchaintechnology.Itencompassesavariousapplications, includingdecentralizedfinance,virtualworlds,anddigitalassets,enablingusers to engage in innovative financial and creative activities.

Cryptocurrencyisatypeofdigitalorvirtualcurrencythatusescryptographyfor security. It operates on decentralized networks based on blockchain technology, whichis adistributedledgerenforcedbyanetworkofcomputers (oftenreferredto as nodes).

CurrentTrendsintheCryptoverse:

Mass Adoption: Cryptocurrencies are moving beyond rich markets, attracting interest from a broader audience, including everyday consumers and institutional investors.

Environmental Concerns: The energy consumption associated with mining cryptocurrencies has raised concerns, prompting a shift towards more sustainable practices like Proof of Stake.

Educational Initiatives: There is a growing need for educational resources tohelpthegeneralpublicunderstandthecomplexitiesofcryptocurrency.

Pre-requisites:

Node.jsandnpm:

Node.jsisapowerfulJavaScriptruntimeenvironmentthat allowsyoutorunJavaScriptcodeonthelocalenvironment.Itprovidesa scalableand efficient platform for building network application.

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

React.js: It's 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.

- Startthedevelopmentserver:

  npm start

  Thiscommand launches the developmentserver, and access our React app at http://localhost:3000in our web browser.

✔ HTML, CSS, and JavaScript: Basic knowledge of HTML for creating the structureofyourapp,CSSforstyling,andJavaScriptforclient-sideinteractivityis essential.

✔**Version Control:**Use GIT for version control, that enabling collaboration and tracking changes throughout the development process. We use the Platform GitHub an host your repository.

GIT: Download and installation instructions can be found at: https://git-scm.com/downloads

✔ **Development Environment**: Choose a code editor or Integrated Development Environment (IDE) that suits for our preferences, we use Visual Studio Code.

Visualstudiocode:https://code.visualstudio.com/download.

InstallDependencies:

- Navigateintotheclonedrepositoryandinstalllibraries:cd
  crypto
  npminstall

✔StarttheDevelopmentServer:

Tostartthedevelopmentserver,executethefollowingcommand:

npmrundev (vite)ornpmstart.

OURGITHUBREPOSITORYLINK:

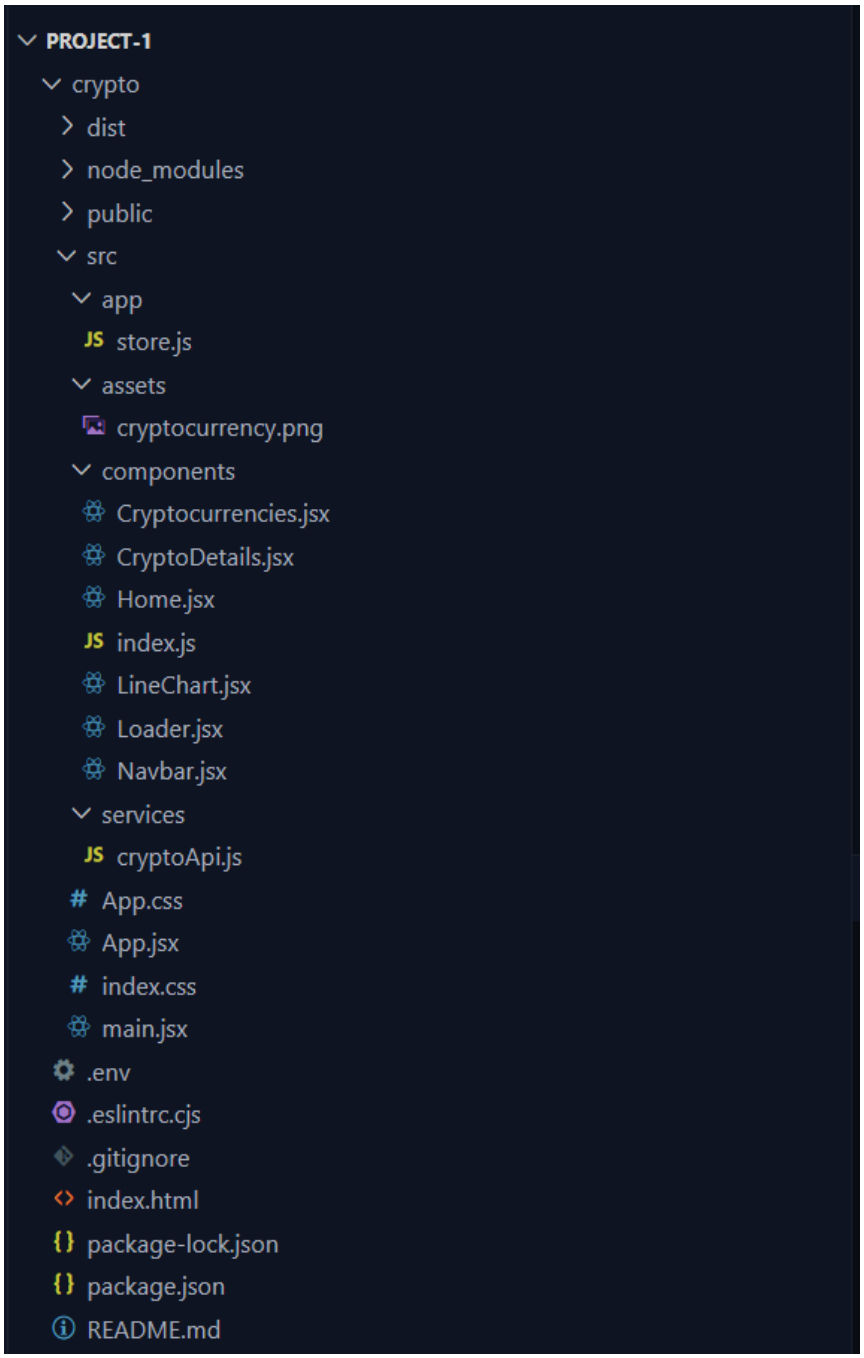
https://github.com/venkatesan-2808/cryptocurrency1

AccesstheApp:

- Openyourwebbrowserandnavigatetohttp://localhost:5173/
- YoushouldseetheourCryptoverseapp'shomepage, indicatingthattheinstallationandsetupweresuccessful.

Nowwehavesuccessfullyinstalledandsetuptheapplicationonourlocal machine. We can now proceed with further customization, development, and testing as needed.

ProjectStructure:



ProjectFlow:

- Projectsetupandconfiguration:

1. **SetupReactApplication:**
   - CreateaReactappintheclientfolder.
   - Installrequiredlibraries
   - Createrequiredpagesandcomponentsandaddroutes.

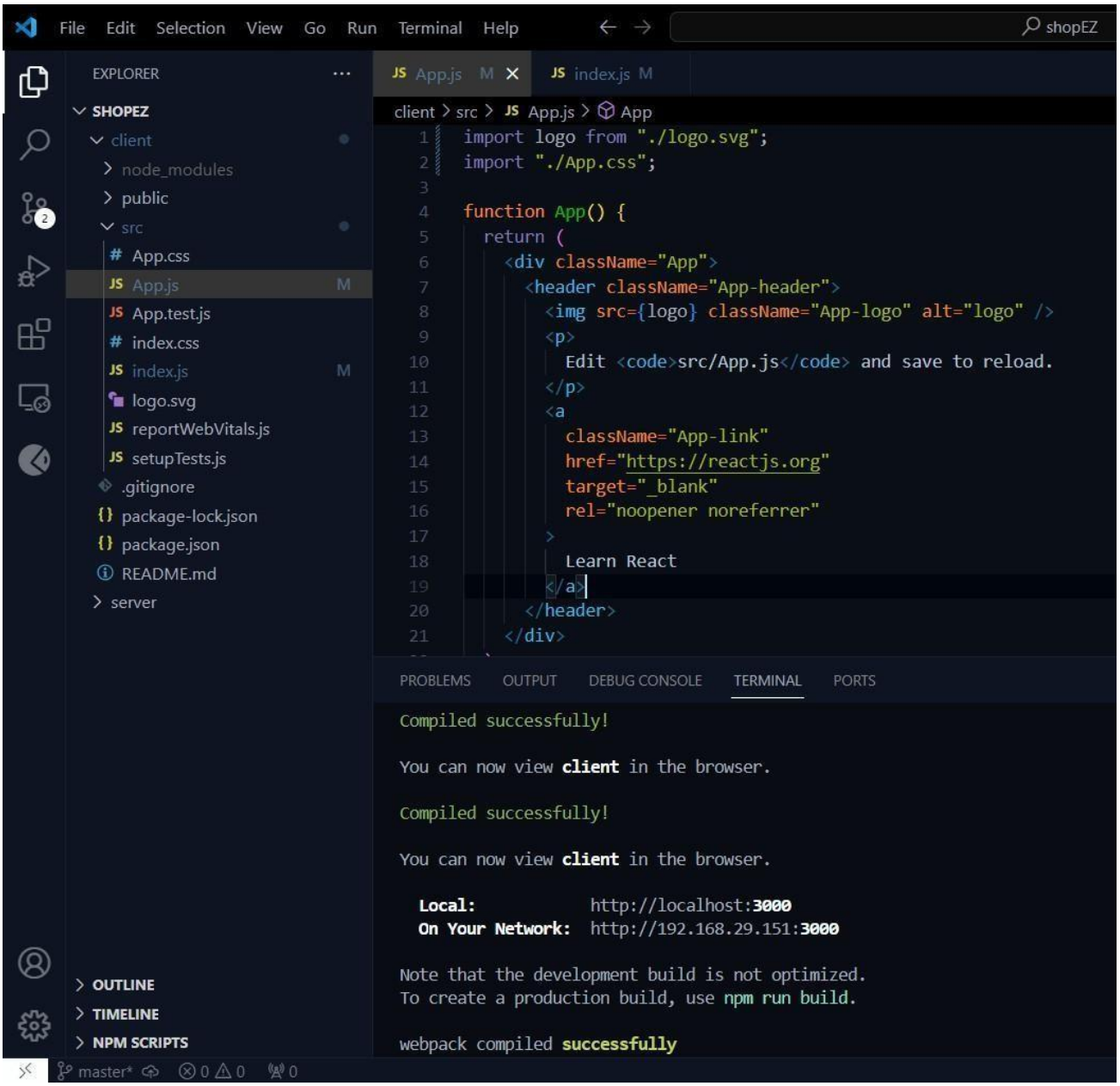
2. **DesignUIcomponents:**
   - CreateComponents.
   - Implementlayoutandstyling.
   - Addnavigation.

3. **Implementfrontendlogic:**
   - IntegrationwithAPIendpoints.
   - Implementdatabinding.

Reference Video Link:

https://drive.google.com/file/d/1-Nqs7mVH_PX5XmJR101Rl13fkWKgjuyv/view?usp=drivesdk

ReferenceImage:



ProjectDevelopment:

- **Createareduxstore:**

1. import{configureStore}from"@reduxjs/toolkit";`:Thislineimportsthe `configureStore`functionfromRedux Toolkit.ReduxToolkitis apackage that provides utilities to simplify Redux development, making it easier to write Redux logic with lessboilerplatecode.

2. `import{cryptoApi}from"../services/cryptoApi";`:Thislineimportsthe `cryptoApi` object from the `cryptoApi.js` file located in the `../services` directory.

Thisobjectlikelycontainsconfigurationsandfunctionsrelatedtomaking API requests for cryptocurrency data.

3. 'exportdefaultconfigureStore({...});`:ThislineexportstheReduxstore configuration created by the `configureStore' function as the default export of this module.

4. `reducer:{[cryptoApi.reducerPath]:cryptoApi.reducer}`:Thispart ofthe configuration specifies the rootreducer for the Redux store. In this case, it sets the `cryptoApi.reducer`asthereducerforthesliceofstatemanagedbythe `cryptoApi` API slice. The `cryptoApi.reducerPath` likely refers to the slice'suniqueidentifier,whichisusedinternallybyReduxToolkit.

5. `middleware: (getDefaultMiddleware) => getDefaultMiddleware().concat(cryptoApi.middleware),`:Thispartofthe configuration specifies middleware for the Redux store. Middleware intercepts actions before they reach the reducers and can be used for variouspurposes,suchaslogging,asynchronousactions,orhandlingAPI requests. Here, it uses the `getDefaultMiddleware`functionprovidedbyReduxToolkittogetthe defaultmiddlewarestackandappendsthe`cryptoApi.middleware`. ThismiddlewarelikelyhandlesasynchronousAPIrequestsanddispatches corresponding actions based on the API response.

this configuration sets up a Redux store with a specific reducer and middlewareprovided by the `cryptoApi` object, which presumablymanagesstaterelated to cryptocurrency data fetched from an externalAPI.Thissetupallowsyoutomanageandinteractwiththisdata 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),
});
```

,andqueryfunctionsrequiredformakingrequeststothecryptocurrencyAPI.

- **CreateaAPIsliceusingReduxtoolkit's:**

1. ImportStatements:

-`import { createApi, fetchBaseQuery } from "@reduxjs/toolkit/query/react";`:Thislineimportsthenecessaryfunctions from Redux Toolkit's query-related module.
`createApi` is used to create an API slice, while `fetchBaseQuery` is a utility function provided by Redux Toolkit for making network requests using `fetch`.

2. HeaderandBaseURLConfiguration:

`constcryptoApiHeaders={...}`:Thisobjectcontainsheadersrequired for makingrequeststothecryptocurrency API.Thevaluesfor`X-RapidAPI- Key"` and

`"X-RapidAPI-Host"`areretrievedfromenvironmentvariablesusing `import.meta.env`.

-`constbaseUrl=https://coinranking1.p.rapidapi.com`:Thisvariable holdsthebaseURLforthecryptocurrencyAPI,whichisalsoretrieved from environment variables.

3. RequestCreationFunction:

'constcreateRequest=(url)=>({url,headers:cryptoApiHeaders});`:This function.

`createRequest` takes a URL and returns an object with the URL and headersrequiredformakingarequest.Itutilizes`cryptoApiHeaders`to include necessary headers intherequest.

4. CreateAPISlice:

`exportconstcryptoApi=createApi({...})`:Thispartusesthe `createApi`functiontocreateanAPIslicenamed`cryptoApi`.Ittakes an object with several properties:

- `reducerPath`:Specifiesthepathunderwhichtheslice'sreducerwill be mounted in the Redux store.

- `baseQuery`:ConfiguresthebasequeryfunctionusedbytheAPIslice. Inthiscase,ituses`fetchBaseQuery`withthebaseURLspecified.

- `endpoints`:DefinestheAPIendpointsavailableintheslice.It'sanobject with keys corresponding to endpoint names and values being endpoint definitions.

5. APIEndpoints:

- `getCryptos`, `getCryptoDetails`, `getCryptoHistory`: These areendpointsdefinedusingthe`builder.query`method.Eachendpointis configuredwitha`query`functionthatreturnstherequestconfiguration object created by `createRequest`.

6. ExportingHooks:

- `exportconst{...}`:Thislineexportshooksgeneratedbythe `createApi`function,allowingcomponentstoeasilyfetchdatafromthe API slice. Each hook corresponds to an endpoint defined in the `endpoints`object.

Overall,thiscodesetsupanAPIslicenamed`cryptoApi`usingRedux Toolkit's query functionality. It defines endpoints for fetching cryptocurrencies, cryptocurrency details, and cryptocurrency history.

App.jsx | Loader.jsx | cryptoApi.js | Cryptocurrencies.jsx | LineChart.jsx

code > src > services > cryptoApi.js > ...

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

const cryptoApiHeaders = {
  "X-RapidAPI-Key": 'd9ccbe2e00mshe785fadd99ee82cp143ca5jsn2b972854cbd8',
  "X-RapidAPI-Host": 'coinranking1.p.rapidapi.com',
};

const baseUrl = 'https://coinranking1.p.rapidapi.com';

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;
```

- **AddingProvidersinthemainfunction:**

ReactRouterwithBrowserRouter:

- `<BrowserRouter>`:Thiscomponentisprovidedby`react-router-dom`andenablesclient-sideroutingusingtheHTML5historyAPI.It wraps the application, allowing it to use routing features.

ReduxProvider:

-

- `<Providerstore={store}>`:Thiscomponentisprovidedby`react-redux` and is used to provide the Redux store to the entire application.Itwrapstheapplication,allowingallcomponentstoaccess the Redux store.

Overall, thiscodeinitializestheReactapplicationbyrenderingtheroot component(`<App/>`)intotheDOM,whilealsoprovidingrouting capabilities through

`BrowserRouter`andstatemanagementwithReduxthrough`Provider`. Additionally, it ensures stricter development mode checks with '<React.StrictMode>'.

```jsx
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>
);

```

- **CreatingaLinechartcomponent:**

  ThiscodedefinesaReactcomponentcalled`LineChart`whichrendersaline chart usingthe`react-chartjs-2` library.

  1. Imports:importReactfrom"react":Importsthe`React`

     module.

     import{Line}from"react-chartjs-2";`:Importsthe`Line`componentfrom the`react-chartjs-2` library, which is used to render line charts.

import{Col,Row,Typography}from"antd";`:Importsspecific components fromtheAntDesignlibrary,including`Col`,`Row`,and `Typography.const{Title}=Typography;`:Destructuresthe`Title`component from the`Typography` module.

2. ComponentDefinition:

   constLineChart=({coinHistory,currentPrice,coinName})=>{...}`: Definesafunctionalcomponentcalled`LineChart`.Ittakesthree props: `coinHistory`,currentPrice`, and `coinName`.

3. DataPreparation:

   - Insidethecomponent,itloopsthroughthe`coinHistory`datatoextract `coinPrice`and`coinTimestamp`arrays.Thesearrayswillbeusedasdata points for the line chart.

4. ChartData:

   `constdata={...}`:Definesthedataobjectforthelinechart.Itincludes labels (timestamps) and datasets (coin prices).

5. Rendering:

   - Insidethereturnstatement,itrendersthechartheader,includingthecoin name, price change, and current price.

   - `Row`and`Col`fromAntDesignareusedtostructurethelayout.

   - The`Line`componentrenderstheactuallinechartusingthedataobject defined earlier.

6. Export:

   - `exportdefaultLineChart;`:Exportsthe`LineChart`componentasthe default export.

Overall,thiscomponentreceiveshistoricaldata(`coinHistory`), current price(`currentPrice'),andthenameofthecryptocurrency(`coinName') as props, and renders a line chart 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;
```

- **Creatingcryptocurrenciescomponent:**

1. ComponentDefinition:

`constCryptocurrencies=({simplified})=>{...}`:Definesafunctional component named `Cryptocurrencies`. It accepts a prop named `simplified`,whichdetermineswhethertodisplayasimplifiedversionof the list.

2. Initialization:

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

3. FetchingCryptocurrencyData:

`const{data:cryptosList,isFetching}=useGetCryptosQuery(count);`: Uses the `useGetCryptosQuery` hook provided by the `cryptoApi' service tofetch cryptocurrency data. It retrieves the list of cryptocurrencies (`cryptosList`)andabooleanflag(`isFetching`)indicatingwhetherthe data is being fetched.

4. FilteringCryptocurrencyData:

- The`useEffect`hookisusedtofilterthecryptocurrencydatabasedonthe `searchTerm`statevariable.Itupdatesthe`cryptos`statewith filtereddatawhenever`cryptosList`or`searchTerm`changes.

5. RenderingLoader:

- `if(isFetching) return <Loader />;`: Ifdatais still being fetched (`isFetching`istrue),itreturnsa`Loader`componenttoindicate that the data is loading.

6. RenderingSearchInput:

- `!simplified&&(...)`:If`simplified`isfalse,itrendersasearchinput fieldallowinguserstosearchforspecificcryptocurrenciesbyname.

7. RenderingCryptocurrencyCards:

- The`Row`and`Col`componentsfromAntDesignareusedtocreateagrid layout for displayingcryptocurrency 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. ReturnStatement:

- `return(...)`:ReturnsJSXrepresentingthecomponent'sstructureand content.

Overall, this component fetches cryptocurrency data, filters it based on a searchterm,andrendersthedatainavisuallyappealingformatwithcard-basedUI.Italso providesasearchfunctionalityforuserstofindspecific 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;
```

- **Createacomponenttoshowthedetailsofcryptocurrency:**

ThiscodedefinesaReactfunctionalcomponentcalled`CryptoDetails’ responsible fordisplayingdetailedinformationaboutaspecificcryptocurrency.Let'sbreak down the code:

1. ComponentDefinition:

   constCryptoDetails=()=>{...}`:Definesafunctionalcomponentnamed CryptoDetails`.Itdoesn'tacceptanypropsdirectlybututilizesReact Router'suseParams`hooktogetthe`coinId`parameterfromtheURL.

2. StateInitialization:

   - Initializesstatevariables`timePeriod`and`coinHistory`. `timePeriod`represents the selected timeperiod for displaying cryptocurrencyhistory,and`coinHistory`storeshistoricaldataoftheselectedcrypto currency.

3. FetchingData:

   Utilizes`useGetCryptoDetailsQuery’and‘usGetCryptoHistoryQuery’ hooksprovided bythe`cryptoApi` serviceto fetchdetailsand historicaldataofthe cryptocurrencyspecifiedby`coinId`.Ituses‘coinId`obtainedfrom`useP arams`tofetch dataforthespecific cryptocurrency.

4. SettingCoinHistory:

   Utilizes `useEffect` hook to update the `coinHistory` state when coinHistoryData`changes.Thisensuresthatthecomponentre-renderswith updated historical data.

5. RenderingLoader:

   - Displaysaloadingindicator(`<Loader/>`)whiledataisbeingfetched (`isFetching` is true).

6. TimePeriodSelection:

   Renders a `Select` component allowing users to choose the time periodfor displayinghistorical data. It triggers the `setTimePeriod` function when the selection changes.

7. RenderingLineChart:

   - Utilizesthe`LineChart`componenttodisplaythehistoricalprice trend ofthecryptocurrencyovertheselectedtimeperiod.

8. RenderingStatistics:

   Displaysvariousstatisticsrelatedtothecryptocurrency,suchasprice,rank, volume,market cap, etc. Thesestatistics are displayedintwo sections: `stats`and‘genericStats`.

9. RenderingDescriptionandLinks:

   - Parsesanddisplaysthedescriptionofthecryptocurrencyusing `HTMLReactParser`.

   - Renderslinksrelatedtothecryptocurrency,suchasofficialwebsites,social media, etc.

10. ReturnStatement:

    - ReturnsJSXrepresentingthestructureandcontentofthecomponent.

Overall, this component fetches and displays detailed informationaboutaspecificcryptocurrency,includinghistorical price data, key statistics, description, and related links.

- **CreateaHomepage:**

  This component, named `Home`, is a React functional componentresponsibleforrenderingthehomepageofthecryptocurrency dashboard. Let's break down the code:

  1. ComponentDefinition:

     `constHome=()=>{...}`:Definesafunctionalcomponentnamed `Home`.

  2. DataFetching:

     - Usesthe`useGetCryptosQuery`hookprovidedbythe`cryptoApi` servicetofetchdataforthetop10cryptocurrencies.Itretrievesdata and a boolean flag indicatingwhetherdataisbeingfetched.

  3. RenderingLoader:

     - Displaysaloadingindicator(`<Loader/>`)whiledataisbeingfetched (`isFetching` is true).

  4. GlobalCryptoStats:

     Renders statistics about the global cryptocurrency market,includingtotalcryptocurrencies,totalexchanges,totalmarket cap, total 24-hour volume, and total markets. These statistics are displayedusingthe `Statistic`componentfromAntDesign.

5. Top10Cryptocurrencies:

- Rendersasectiondisplayingthetop10cryptocurrenciesintheworld.

- Utilizesthe`Cryptocurrencies`componentwiththe`simplified`prop settotruetodisplayasimplifiedversionofthelist.

- Providesalinktoviewmorecryptocurrenciesusingthe`Link` component from React Router.

6. ReturnStatement:

- ReturnsJSXrepresentingthestructureandcontentofthecomponent.

Overall,thiscomponentfetchesanddisplaysglobalcryptocu
rrency
statisticsandthe top 10 cryptocurrencies on the homepage of the dashboard. It provides links for 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;

```

ProjectExecution:

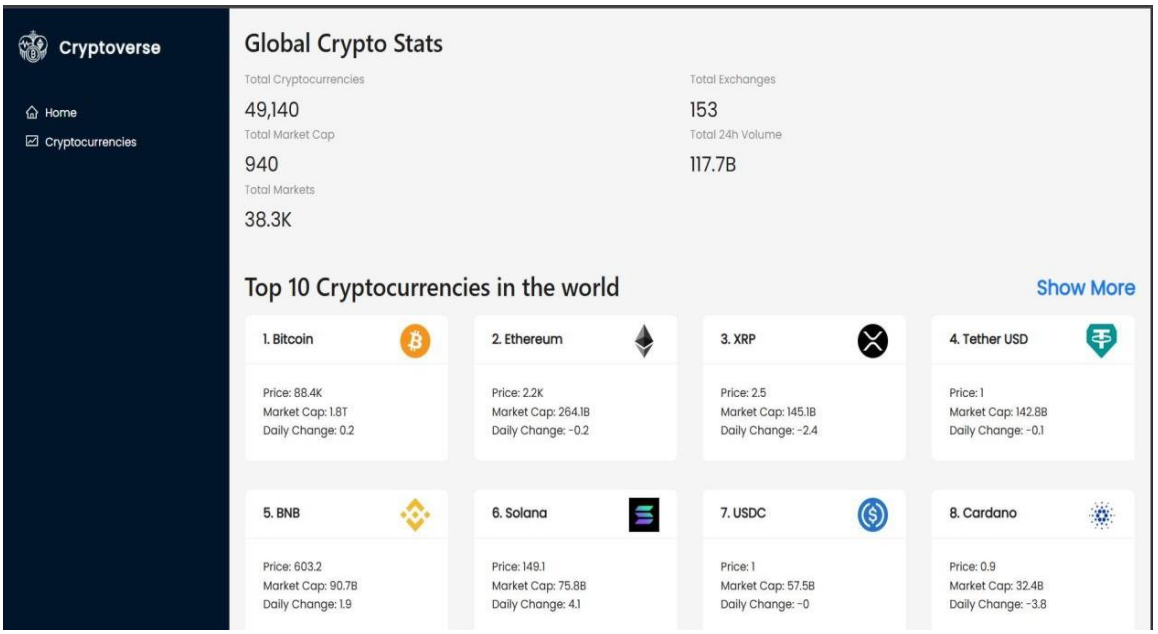
Hereisthevideolinkofreactapplicationexecution:
https://drive.google.com/file/d/1hbDljCv0R_a_lLrRCh-8yMea-WCvKP1e/view?usp=sharing

Projectdemo:

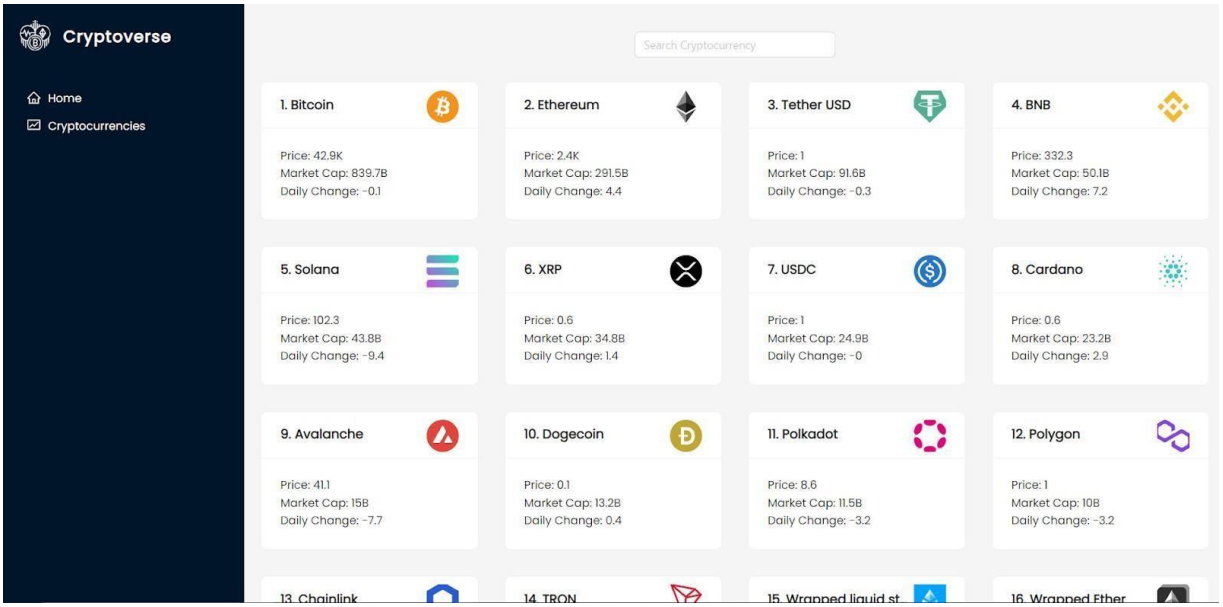
Demo link:
https://drive.google.com/file/d/1FheBywfSJu2XgUM1p13K4G7uo7RU5wE1/view?usp=sharing

UserInterfacesnips:

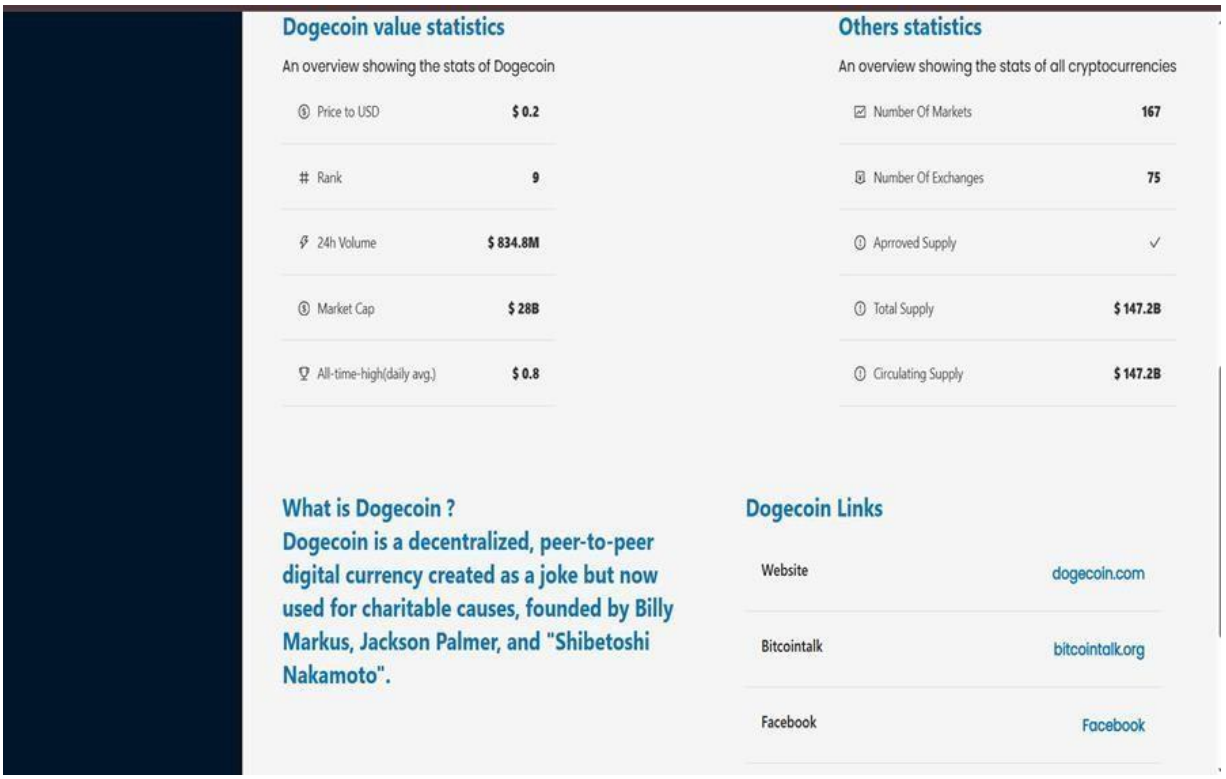
Homepage:Thispagesconsistsofstatsofglobalcryptoliketotalcryptocurrencies,total exchanges,marketcapetc.Alsoconsistoftop10cryptocurrenciesintheworld.



Crypto currencies page: This pages contains all cryptocurrencies which are currently inflow in the world. There is also a search feature where users can searchand findout about their desiredcryptocurrency.



This page contains the line chart with data representation of price of cryptocurrencies.Alsocontainsstatisticsandwebsitelinksofcryptocurrencies.


Dogecoin value statistics
An overview showing the stats of Dogecoin
Price to USD $ 0.2
Rank 9
24h Volume $ 834.8M
Market Cap $ 28B
All-time-high(daily avg.) $ 0.8
Others statistics
An overview showing the stats of all cryptocurrencies
Number Of Markets 167
Number Of Exchanges 75
Aprroved Supply ✓
Total Supply $ 147.2B
Circulating Supply $ 147.2B
What is Dogecoin ?
Dogecoin is a decentralized, peer-to-peer digital currency created as a joke but now used for charitable causes, founded by Billy Markus, Jackson Palmer, and "Shibetoshi Nakamoto".
Dogecoin Links
Website dogecoin.com
Bitcointalk bitcointalk.org
Facebook Facebook

THANKYOU