Subject Name: **Front End Engineering**

Subject Code: CS186

Cluster: iGammaGroup: **G19**

Department: **DCSE**



|  |  |  |
| --- | --- | --- |
| **Submitted By:**  Mohd. Tariq  2110991954  G19 |  | **Submitted To:**  Ms. Pritpal Kaur |

**Cryptocurrency Tracking Application**

**Introduction**

This document provides a detailed explanation of the React application designed for tracking cryptocurrency data. The application fetches cryptocurrency information from the CoinGecko API and allows users to search for specific coins by name.

**Technologies Used:**

* HTML
* CSS
* JavaScript
* React
* API

**Table of Contents**

* Application Structure
* Dependencies
* Component Breakdown
* Data Fetching
* User Search Functionality
* Rendering CoinsUsage

**Flow Chart:**



**Usage and Further Development**

**1. Application Structure**

The application is structured as a React functional component with two state variables and several functions for fetching and displaying cryptocurrency data.

**2. Dependencies**

React: The core library for building the user interface.

useEffect and useState: React hooks used for managing state and side effects.

'./App.css': A CSS file for styling the application.

axios: A popular JavaScript library for making HTTP requests.

Coin: A custom React component used to display individual coin information.

**3. Component Breakdown**

The App component is the main structure of the application. It has two state variables:

coins: Stores cryptocurrency data fetched from the CoinGecko API.

search: Stores the user's search query.

**4. Data Fetching**

Cryptocurrency data is fetched when the component mounts using the useEffect hook. The data is requested from the CoinGecko API with the following parameters:

Base currency: Indian Rupees (INR)

Order: By market capitalization in descending order

Per page: 100 results

Page: 1

Sparkline: Disabled

The fetched data is set in the coins state, and the response is logged to the console for debugging purposes.

**5. User Search Functionality**

The handleChange function is used for searching cryptocurrencies. When a user enters text in the search input field, the search state is updated with the entered value, enabling real-time search functionality. The search is case-insensitive, making it user-friendly.

**6. Rendering Coins**

The application renders a list of coins that match the user's search criteria. The filteredCoins variable is created by filtering the coins based on the user's search query. For each filtered coin, the Coin component is rendered, displaying key information:

**Coin name**

**Coin image**

**Symbol**

**Market capitalization**

**Current price**

**Price change in the last 24 hours**

**7. Usage and Further Development**

* This application provides a foundation for tracking and searching cryptocurrency data. You can expand upon it by:
* Adding more detailed information for each coin.
* Implementing pagination for the coin list.
* Styling the application for a more appealing user interface.
* Handling errors and edge cases.
* Incorporating more advanced features like charts or historical data.
* In summary, this documentation explains how the provided code creates a basic cryptocurrency tracking application. It's a starting point for building a more sophisticated and user-friendly cryptocurrency tracking platform.

**Code for API:**

import React, { useEffect, useState } from 'react';

import './App.css';

import axios from 'axios';

import Coin from './Coin';

Imports: This section imports the necessary modules and components for the application:

React is the core library for building the user interface.

useEffect and useState are hooks provided by React for managing state and side effects.

'./App.css' imports a CSS file for styling the application.

axios is a popular JavaScript library for making HTTP requests.

Coin is a custom component used to display individual coin information.

javascript

function App() {

const [coins, setCoins] = useState([]);

const [search, setSearch] = useState('');

Component Setup: The App function defines the main React functional component. It uses two state variables:

coins: Stores the cryptocurrency data fetched from the API.

search: Stores the user's search query.

javascript.

useEffect(() => {

axios.get('https://api.coingecko.com/api/v3/coins/markets?vs\_currency=INR&order=market\_cap\_desc&per\_page=100&page=1&sparkline=false')

.then(res => {

setCoins(res.data);

console.log(res.data);

}).catch(error => console.log(error));

}, []);

Data Fetching: This useEffect hook is executed when the component is mounted. It fetches cryptocurrency data from the CoinGecko API and updates the coins state with the response data. The API endpoint used provides data in Indian Rupees (INR) for the top 100 coins by market capitalization.

javascript.

const handleChange = e => {

setSearch(e.target.value);

}

Search Functionality: The handleChange function is called when the user enters text in the search input field. It updates the search state with the value entered by the user, enabling dynamic search functionality.

javascript.

const filteredCoins = coins.filter(coin =>

coin.name.toLowerCase().includes(search.toLowerCase())

);

Filtering Coins: The filteredCoins variable is created by filtering the coins state based on the user's search query. It performs a case-insensitive search on the coin names to match the search query.

javascript.

return (

<div className="coin-app">

<div className="coin-search">

<form action="">

<input type="text" className="coin-input" placeholder="Provide the coin name" onChange={handleChange} />

</form>

</div>

{filteredCoins.map(coin => {

return (

<Coin

key={coin.id}

name={coin.name}

image={coin.image}

symbol={coin.symbol}

marketcap={coin.market\_cap}

price={coin.current\_price}

pricechange={coin.price\_change\_percentage\_24h}

/>

);

})}

</div>

);

}

Rendering: The return statement defines the structure of the web page using JSX (JavaScript XML). It includes:

An input field for searching coins.

A list of filteredCoins mapped to the Coin component, with various properties from the cryptocurrency data passed as props for rendering.

This code creates a basic cryptocurrency tracking application that allows users to search for coins and view their details. You can expand upon this foundation to add more features, styling, and functionality as needed for your project.

**Conclusion:**

In conclusion, the provided React application for cryptocurrency tracking is a basic yet functional example of how to fetch data from an external API, filter and display the data based on user input, and create a user-friendly interface. The application demonstrates several key concepts in web development and React:

State Management: The use of React state variables (coins and search) to manage and update the application's data and user input.

API Integration: Utilizing the Axios library to make HTTP requests to the CoinGecko API for cryptocurrency data retrieval.

Dynamic Filtering: Implementing real-time filtering of cryptocurrency data based on user-provided search queries.

Component-Based Architecture: Utilizing React components to modularize the user interface and encapsulate functionality.

User Interaction: Allowing users to search for specific cryptocurrencies and see key information about them in real-time.

Potential for Further Development: The application serves as a starting point for more advanced features, such as pagination, detailed cryptocurrency information, and enhanced user experience.

The flowchart provided in the previous response outlines the primary interactions within the application, helping to visualize its functionality. While this application is relatively simple, it can serve as a foundation for building more sophisticated cryptocurrency tracking platforms or similar web applications. Further development could include enhancing the user interface, integrating additional API endpoints, and implementing advanced features for a more comprehensive user experience.