Description

I have created the application using React, Express and node.js.
Frontend using React
Backend using Express and Node.js

I have used Coinbase and Gemini exchanges API and showing the prices from both the APIs.
Application will refresh every second to show the new prices from exchanges.

Link to Github account:

https://github.com/SmitaRath/Cryptocurrency-API

Instructions for running the application

Kindly download the node.js if it is not installed in the system from the below link:
https://nodejs.org/en/download/

These are the steps which needs to be followed to run the application.

Open 2 terminals in the project folder

Terminal 1-

- cd server
- npm install
- node app.js

Terminal 2-

- cd client
- npm install
- npm start

Backend

The server will start running at localhost:4000

Frontend

The React client will start running at localhost:3000 automatically.

Questionnaire:

1. Are there any sub-optimal choices(or short cuts taken due to limited time) in your implementation?

   Ans) No

2. Is any part of it over-designed? (It is fine to over-design to showcase your skills as long as you are clear about it)

   Ans) No

3. If you have to scale your solution to 100 users/second traffic what changes would you make, if any?

   Ans)

   1. For this application I am using free API, if my users are more then I will go for premium API where I will have unlimited requests.
   2. I will deploy a proxy cache server to which my users will request and get data directly. The proxy cache server will fetch data from the API and make it available in the cache persistently. In that case for every user request, my application will not hit the exchange server, but it can get the data from the proxy cache server.

4. What are some other enhancements you would have made, if you had more time to do this implementation?

   Ans)

   Three enhancements I could have done.

   1. First, I would have also used Redis to store the data on temporary cache, so if the application has recommended to buy from some exchange, then that price can be saved in the temporary cache, for the selling recommendation.
   2. Second, the historical prices can also be shown to the user for the better recommendation of buying and selling.
   3. For the previous week I can store the exchange prices, profit and timestamp, whenever there is a profit and can be shown to the user for last week which will give a better picture to user about the market trend.