ERİM ERKİN DOĞAN
2019400225

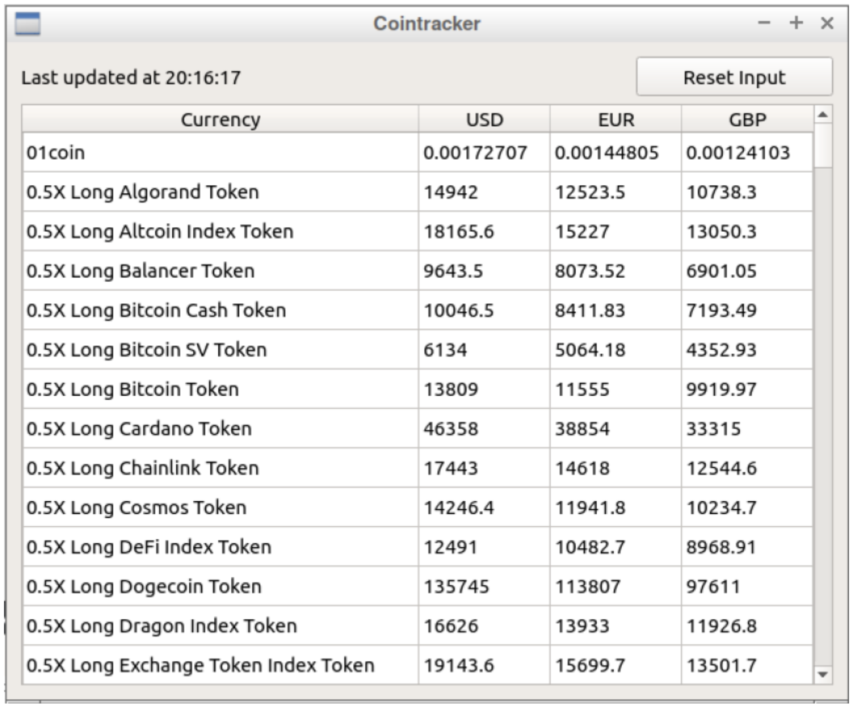
CMPE230 PROJECT 3

Problem Description

In this project we are asked to make a GUI application that tracks prices of cryptocurrencies that are given in a file which is encoded in an environment variable. To find the active prices of currencies in USD, EUR and GBP, we need to use an API that supplies live price and parse the resulting data to our GUI application.

Implementation

To design the project, I used QGridLayout, QPushButton and extended QTableWidget and QLabel classes. So if we look at the UI in the picture, it shows an extended QLabel(UpdateLabel) that says: "Last updated at hh:dd:ss" which is used to show age of data in table. There is also a "Reset Input" QPushButton that is connected to a function that resets the memory of the program and causes it to read again from new or old input file and show the prices for these currencies. Bottom widget is of MyTableWidget class which inherits from QTableWidget class, this widget displays the currencies and updates the prices and label's timestamp every 30 seconds.



Last updated at 20:16:17

Currency	USD	EUR	GBP
01coin	0.00172707	0.00144805	0.00124103
0.5X Long Algorand Token	14942	12523.5	10738.3
0.5X Long Altcoin Index Token	18165.6	15227	13050.3
0.5X Long Balancer Token	9643.5	8073.52	6901.05
0.5X Long Bitcoin Cash Token	10046.5	8411.83	7193.49
0.5X Long Bitcoin SV Token	6134	5064.18	4352.93
0.5X Long Bitcoin Token	13809	11555	9919.97
0.5X Long Cardano Token	46358	38854	33315
0.5X Long Chainlink Token	17443	14618	12544.6
0.5X Long Cosmos Token	14246.4	11941.8	10234.7
0.5X Long DeFi Index Token	12491	10482.7	8968.91
0.5X Long Dogecoin Token	135745	113807	97611
0.5X Long Dragon Index Token	16626	13933	11926.8
0.5X Long Exchange Token Index Token	19143.6	15699.7	13501.7

To understand the program's workings, we can check the classes and events in its order:

- **Coin class:**

  This class is used to store properties of cryptocurrencies such as their names, ids, symbols and prices in GBP, USD, EUR.

- **FileOps class:**

  This class handles inputs by reading environment variable MYCRYPTOCONVERT and then reading the given file line by line. Then the read input is classified with help of retrieved list from API, which is then used to create coin objects that store input coins' information. After this, it signals MyTableWidget. Also the "Reset Input" button shown in UI resets inputs of program, table and coins so repeats steps of reading and parsing input.

- **MyTableWidget:**

  This class extends QTableWidget class and used to display prices of input coins. To get list of coins, it waits for signal from FileOps. As the class is alerted by FileOps, it retrieves price for given coins by sending HTTPS request to server and parsing the resulting JSON into Coin objects. Then the data from Coin objects are displayed on the table. This class also has a Timer that ticks every 30 seconds which refreshes the prices for current displayed coins. So every 30 seconds the table updates and class also signals UpdateLabel to update the "Last updated at hh:mm:ss" label. Also the "Reset Input" button shown in UI resets the table and coins, making this class wait for signal from FileOps.

- **UpdateLabel**:

  This class is just a subclass for QLabel, and it is equipped with QDateTime which is then used to calculate current time and change content of label when a signal from MyTableWidget comes.

Conclusion

In conclusion, I think this project has shown me how real world applications are made, what is the workflow and design choices making these applications are, etc. So even if the UI and project is simple, it touched important details such as API calls, JSON parsing and creating and manipulating UI elements based on asynchronous events. Designing workarounds between these things and learning fundamental operations such as handling async events and JSON parsing are all gains for me in this project.