**Instructions:**

- Install Flask
  - $ pip install Flask
  - Link to install Flask: https://flask.palletsprojects.com/en/2.0.x/installation/
- Download the source folder https://github.com/meenalsawant017/Chainalysis_Assignement/tree/main/source
- Run python code provided on github.
- Open the webpage using the URL shown in the output.
- To get updated prices, refresh the webpage.

**Steps Followed**:

1. Run python Code
2. Code Output

```
 * Serving Flask app 'python_code' (lazy loading)
 * Environment: production
[31m   WARNING: This is a development server. Do not use it in a production dep
loyment.[0m
[2m   Use a production WSGI server instead.[0m
 * Debug mode: on
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

3. Copy URL shown in the output http://127.0.0.1:5000/

4. Open the webpage using URL shown in the output Final Output

## Bitcoin and Ethereum Prices

**Recommended prices for buying(Ask) and Selling(Bid) are on the top of the book along with the Exchanges.**

**BTC**

| Bid Exchange | Bid Price | Bid Size | Ask Price | Ask Size | Ask Exchange |
|---|---|---|---|---|---|
| Binance | 65173.26000000 | 0.91568000 | 65173.27000000 | 0.88037000 | Binance |
| Bitstamp | 65137.12 | 0.07666005 | 65200.55 | 0.36916915 | Bitstamp |

**ETH**

| Bid Exchange | Bid Price | Bid Size | Ask Price | Ask Size | Ask Exchange |
|---|---|---|---|---|---|
| Binance | 4699.99000000 | 17.23480000 | 4700.00000000 | 106.91430000 | Binance |
| Bitstamp | 4696.14 | 0.85106498 | 4702.45 | 0.53164817 | Bitstamp |

We are showing BTC and ETH prices from two exchanges(Binance, Bitstamp) and creating a book that has sorted the Bid prices from high to low order price and Ask prices from low to high price. So the best price will always be on the top of the book.

In the above example, we can see that the Ask price for BTC is 65173.27 on Binance Exchange and 65200.55 on Bitstamp exchange. So the recommended exchange to buy is Binance Exchange.

**Questionnaire:**

1. Are there any sub-optimal choices( or short cuts taken due to limited time ) in your implementation?
    - 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)
    - No
3. If you have to scale your solution to 100 users/second traffic what changes would you make, if any?
    - If there are 100 users making API calls, then instead of making a multiple API call to the exchange, we can create a centralized server which is doing the API calls to the exchanges by some configured frequency. So all the users do not have to call exchange instead call this centralized server
4. What are some other enhancements you would have made, if you had more time to do this implementation
    - No