Edgify

Edgify Python Take Home Assignment

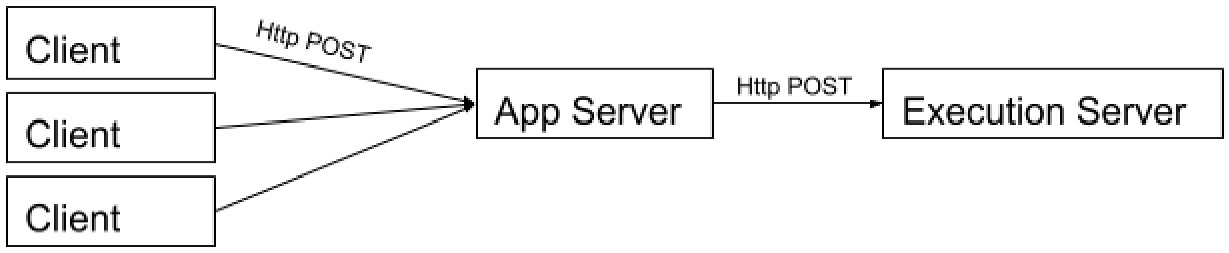
Due to the Coronavirus situations we are not able to conduct in person professional tests- so we'd appreciate your cooperation in completing this home assignment.

- Guidelines:
  - Please conduct this within two days- if you need an extension please let us know.
  - Once completed please return either via e-mail to: asaf.carmel@edgify.ai or through Github

Setup:

The assignment setup is composed from three elements: multiple clients, single application server and single execution server. The focus of this assignment is the application server, and it's the only element you will need to write code for.

Schematic drawing:



Explanation:

The client is a simple web client application that sends a request to buy or sell USD through HTTP POST request. The application server waits for enough requests (default 10) and sends them to the execution server (the execution server is already mocked). The execution server decides what buy/sell orders are approved (or rejected) and returns the response. The application server then responds to the clients if their buy/sell orders succeeded.

The Flow:

- Each client calls the application server with buy/sell order (HTTP POST).
- The application server waits until enough requests arrive (default 10).
- The application server calls the execution server to execute the orders (all the 10 in the same call).
- The execution server responds with approval/rejection to each order (in the same response).
- The application server response and closes the original client requests.
- The application server should be available for new calls during the process

Emphasis:

- For convince the calls to the Execution server are mocked in the ExecutionSdk class so you only need to write the application server code
- Write the server code with http.server.HTTPServer (or asyncio equivalent)
- Please use python 3.7 and up
- You may change the mocks if you like

Please use these classes to mock the execution server SDK:

```
class ExecutionSdk:
  @staticmethod
  def execute_orders(orders: list):
     orders = orders.copy()
     for order in orders:
        order.status = 'approved'

     return orders
```

```python
class Order:
  def __init__(self, price, order):
     self.price = price
     self.order = order
     self.status = None
```