Imagine you are the developer for the backend services of a Pizza restaurant website. The restaurant has started recently and it is estimated that only few people will order via the website. As a backend developer, you are expected to design APIs which will be integrated into the website

After internal discussion in your team it is agreed to use the Flask framework and MongoDB in the backend. You have been assigned the development of the following APIs in Flask -

1. Welcome API

Path - **/welcome**
HTTP method - GET
Return – "Welcome to Pizza House"

2. Accept Order API

This API takes JSON data as input and stores that order in the MongoDB. API will accept JSON input.

Example input to API - {"order": ["Pizza1", "Pizza2"]}

Path - **/order**
HTTP method - POST
Return - Order Id

Please name your MongoDB Database and collection as follows -

- Database - "pizza_house"
- Collection name - "orders"

3. Get order details APIs

1. **/getorders** - Fetch all the orders in the MongoDB at the given instance.
   HTTP method - GET
   Returns - All orders in the database.

2. **/getorders/<order_id>** - Get the order by the order id
   HTTP method - GET
   Returns - Order details of a given order id , 404 Not found if record not present.

4. Can you introduce a message queue ?

The restaurant starts becoming very popular and your server is receiving a huge amount of orders via /order API. Out of the many solutions to tackle this huge load and other anticipated architecture changes, team members have suggested introducing a message queue for the

'/order' API. You start thinking if queuing the orders would be of help and decide to use a message queue which will enqueue the API requests. Your team has left the decision on you regarding which message queue(for eg- rabbitmq) would you use here. Can you refactor the API(or create new, if needed) to introduce a message queue ?

Also, keep your old '/order' API code (Point 2) in the same file and comment it, so that we can compare.

Notes -

1. Wherever possible, responses should be in JSON format.
2. Return the API response with a proper status code.
3. Writing unit test cases is highly recommended.
4. No UI/html page is needed. You can check your APIs by Postman or by curl.

References -

1. Official flask documentation - https://flask.palletsprojects.com/en/2.2.x/
2. Maybe helpful if using rabbitMQ - https://www.rabbitmq.com/tutorials/tutorial-one-python.html

Code sharing:

To share the completed assignment code use GitHub and share the repository link with us.