koombea

Technical Test: Event Notification System

Goal

Develop a simple **Event Notification System** composed of two services that interact with each other, showcasing **SOA principles**. One service should use **FastAPI** to expose APIs, while the other processes messages from a **message queue**. The implementation must demonstrate **OOP principles**, include **unit tests**, and be **dockerized**.

Problem Statement

Create two services that work together to log and notify users about events.

Service 1: Event Logger (FastAPI)

- Exposes a REST API:
    - Endpoint to create events
        - Accepts a payload with the following attributes
            - `user_id`:
                - string, required
            - `description`:
                - string, required
        - The payload must be validated.
            - If valid:
                - The event should be recorded in an in-memory store with an automatically generated `event_id` assigned as a UUID.
                - It should then be sent to a message queue for processing by the Notification Processor (the second service).
            - If invalid:
                - Return proper errors.
    - Endpoint to update the event status:

        - Updates the `status` (string) attribute of an existing event.
    - Endpoint to retrieve all the details for a given event from the in-memory store

Service 2: Notification Processor

- Processes events from the message queue.
- Simulates sending notifications, by simply printing them on the screen as follows: `“Event {event_id} has been sent to {user_id}: {description}”`.
- Updates the **Event Logger** service via its API to mark the event as `"processed"`.

Requirements

1. **Technologies**
    - Use **FastAPI** for the Event Logger service. Properly document the API.
    - Use Python and any lightweight queue library or tool (e.g., **Redis**, **RabbitMQ**, etc) for the Notification Processor.
    - Demonstrate **OOP principles**, encapsulating business logic in classes.
2. **Unit Testing**
    - Include unit tests for both services to verify key functionality
3. **Dockerization**
    - Use Docker to containerize all the services
    - Provide a `docker-compose.yml` file to run the system locally.
4. **Documentation**
    - Include a `README.md` with:
        - Steps to set up and run the services.
        - Instructions to run tests.
        - Instructions pointing to the API documentation

Please provide the link to the repository containing your test development. If any previously mentioned points were not addressed, kindly explain why. If the reason was time constraints, please briefly outline how you would address those points with additional time.