Scalable API for Clinical Recommendations with Event-Driven Notifications

Objective

Develop a backend service using Python and FastAPI that exposes a RESTful API to process patient health data and return a mock clinical recommendation based on predefined business rules. The service must be scalable, secure, event-driven, and fully containerized using Docker.

The candidate will implement:

- A REST API to receive patient data and return a recommendation.
- A rule-based engine that simulates a clinical recommendation system.
- Event-driven processing, where recommendations trigger background events.
- Scalability & performance optimizations (e.g., caching, async processing).
- A Dockerized setup to ensure the service runs out-of-the-box.
- Testing & documentation.

Mandatory Tech Stack

- Python (FastAPI)
- Docker for containerization
- OpenAPI for API documentation

Other choices (database, message broker, caching) are flexible.

Requirements

1. API Development

- Implement a REST API with the following endpoints:
- POST /evaluate → Accepts patient data and returns a mock clinical recommendation.
- GET /recommendation/{id} → Retrieves a previously generated recommendation by ID.
- The API must be documented in OpenAPI format, auto-generated via FastAPI.

2. Clinical Recommendation Logic (Mock AI Model)

- The recommendation should be based on predefined rule-based logic, such as:
- If age > 65 and has chronic pain, recommend “Physical Therapy”.
- If BMI > 30, recommend “Weight Management Program”.
- If recent surgery, recommend “Post-Op Rehabilitation Plan”.
- Future extensibility: The logic should be easy to replace with a real AI model.

3. Event-Driven Processing (Asynchronous Notifications)

Event Emission (Producer)

- When a recommendation is generated, an event should be published to a message broker (Kafka, RabbitMQ, or Redis Pub/Sub).
- The event should contain:

```
{
   "patient_id": 123,
   "recommendation_id": "abc-456",
   "recommendation": "Physical Therapy",
   "timestamp": "2025-02-17T10:00:00Z"
}
```

Event Processing (Consumer - Background Worker)

- A separate process/service should listen for events and perform a task, such as:
- Logging the recommendation to a file or database.
- Simulating sending an email/SMS notification to the patient.
- Preparing a report for analytics purposes.

4. Scalability & Performance

- Implement caching (e.g., Redis) to optimize repeated queries.
- Ensure concurrent request handling using FastAPI's async capabilities.

5. Security & Authentication

- Secure the API using JWT-based authentication.

6. Docker & Deployment

- The service must run out-of-the-box using Docker.
- Provide a Dockerfile and a docker-compose.yml to spin up:
  - The API service (FastAPI);
  - The database;
  - The message broker (Kafka, RabbitMQ, or Redis Pub/Sub).
  - The background worker.

Submission Guidelines

- The GitHub repository must be private and shared with the reviewer joaoppereira
- Ensure that the entire system runs successfully with Docker.
- Include documentation and tests to support your implementation.