Assignment for Backend Developer

Objective

Develop a microservices-based application that allows users to manage tasks. The application should be designed with scalability, resiliency, and best practices in mind. Candidates should demonstrate their skills in microservices architecture, database design, event-driven architecture, optimized database querying, automated unit testing, authentication & authorization, RESTful APIs, code structure, containerization, and monitoring & alerting. Use Open source technologies wherever possible.

Requirements

Functional Requirements

1. **User Management**
    - Users should be able to sign up, log in, and log out.
    - Implement authentication and authorization.
2. **Task Management**
    - Users can create, update, delete, and retrieve tasks.
    - Tasks should have attributes like title, description, status, and due date.
    - Users can mark tasks as complete.
3. **Event-Driven Notifications**
    - Send notifications when a task is created, updated, or completed.

Non-Functional Requirements

1. **Scalability & Resiliency**
    - Design the system to handle a large number of users and tasks.
    - Ensure the system is resilient to failures.
2. **Optimized DB Querying**
    - Follow best practices for querying the PostgreSQL database.
3. **Containerization**
    - Use Docker to containerize the application.
4. **Monitoring & Alerting**
    - Implement basic monitoring and alerting for the microservices.

Deliverables

1. **Code Solution**
    - Source code for the microservices.
    - Dockerfile for each microservice.
    - Postman collection for API testing.
    - Unit tests for critical parts of the application.
2. **High-Level Diagrams & Documentation**
    - Architecture Diagram: Show the overall microservices architecture.
    - Database Schema: Diagram illustrating the database design.
    - Sequence Diagram: Show the flow of data between microservices.
    - Deployment Diagram: Describe how the application is deployed using Docker.

    - Explanation of Design Choices: Document explaining the choices made for scalability, resiliency, and best practices.

Evaluation Criteria

- **Code Quality:** Clean, readable, and well-documented code.
- **Architecture:** Effective use of microservices architecture.
- **Database Design:** Efficient and scalable database schema.
- **Event-Driven Design:** Proper implementation of event-driven architecture.
- **Testing:** Coverage and quality of unit tests.
- **Security:** Proper implementation of authentication and authorization.
- **API Design:** Adherence to RESTful principles.
- **Scalability & Resiliency:** Design considerations for handling load and failures.
- **Containerization:** Correct usage of Docker.
- **Monitoring & Alerting:** Implementation of basic monitoring and alerting mechanisms.

Submission Guidelines

1. **Repository:** Provide a GitHub repository link with the complete source code.
2. **Documentation:** Include a README file explaining how to set up and run the application, along with the high-level diagrams.
3. **Video Demonstration:** Optionally, provide a short video demonstrating the application's functionality.

High-Level Diagrams & Documentation

1. **Architecture Diagram:**
    - Illustrate the microservices and their interactions.
2. **Database Schema:**
    - Diagram showing tables, relationships, and indexes.
3. **Sequence Diagram:**
    - Show the flow of data and events across services.
4. **Deployment Diagram:**
    - Describe the deployment using Docker and Docker Compose.
5. **Design Choices Documentation:**
    - Explain the choices made for each aspect of the application, focusing on scalability, resiliency, and best practices.