Multi-Tier Online Book Store Program Design

Overview

The Multi-Tier Online Book Store is designed to provide users with an online platform for purchasing, get information, and searching for books. The program consists of three main components: the Catalog Server, the Order Server, and the Frontend Server. These components work together to create a functional online bookstore.

How It Works

1. Catalog Server

The Catalog Server is responsible for managing book information and providing it to the system. Key functionalities include:

- Query Book Information: Users can request book details by book ID. The Catalog Server retrieves this information from a SQLite database and returns it as a JSON response.

- Search Books by Topic: Users can search for books by specifying a topic. The Catalog Server queries the database for books with matching topics and returns the results.

- Purchase Books: When users decide to purchase a book, the Frontend Server communicates with the Order Server, which then communicates with the Catalog API to check book availability and update the database accordingly.

2. Order Server

The Order Server handles book purchases and coordinates with the Catalog Server to ensure that books are available. Its primary functionality is:

- Purchase Books: The Order Server receives purchase requests from the Frontend Server. It verifies book availability, updates the database, save order in Orders database in Order Server, and returns the purchase status to the Frontend.

3. Frontend Server

The Frontend Server serves as the user interface for the online bookstore. Users can interact with the system through a web-based application that offers the following:

- Query Book Information: Users can look up book details by book ID, which triggers a GET request to the Catalog API.

- Search Books by Topic: Users can search for books by topic, and the Frontend API communicates with the Catalog API to retrieve matching book data.

- Purchase Books: When users choose to purchase a book, the Frontend API communicates with the Order API to initiate the purchase.

Design Tradeoffs

The program's design involved several tradeoffs:

1. Modularity vs. Complexity: The decision to split the program into three services (Catalog, Order, and Frontend) enhances modularity and scalability. However, it introduces complexity in terms of inter-service communication.

2. Synchronization and Consistency: Inter-service communication relies on HTTP requests. While this simplifies integration, it introduces the need for synchronization and transaction management to ensure data consistency.

3. Database Choice: The program currently uses SQLite for the Catalog Server. While it simplifies setup, it may not be suitable for large-scale applications.

4. Error Handling: Error handling and recovery mechanisms are minimal. Additional work is needed for comprehensive error handling and logging.

Possible Improvements and Extensions

1. User Authentication: Implement user authentication to enable user-specific actions such as order tracking and book collection management.

2. Payment Integration: Add payment gateway integration for handling real financial transactions securely.

3. Inventory Management: Develop an inventory management system to keep track of book quantities and ensure they are correctly updated when a purchase is made.

4. Caching: Introduce caching mechanisms to reduce the load on the database, particularly for frequently requested book details.

Running the Program

To run the program, follow these steps:

1. Ensure you have Docker and Docker Compose installed on your system.

2. Create Docker images for the Catalog, Order, and Frontend APIs using the provided Dockerfiles.

3. Define the services and network configuration in a Docker Compose file, as you've done in your setup.

4. Run the application using the `docker-compose up` command in the directory containing the Docker Compose file.

5. Access the Frontend API in a web browser by visiting `http://localhost:5000` to interact with the online bookstore.

6. Test various functionalities, including querying book information, searching for books, and making purchases to ensure the system works as expected.

By following these steps, you can run the Multi-Tier Online Book Store program in a development or local environment.