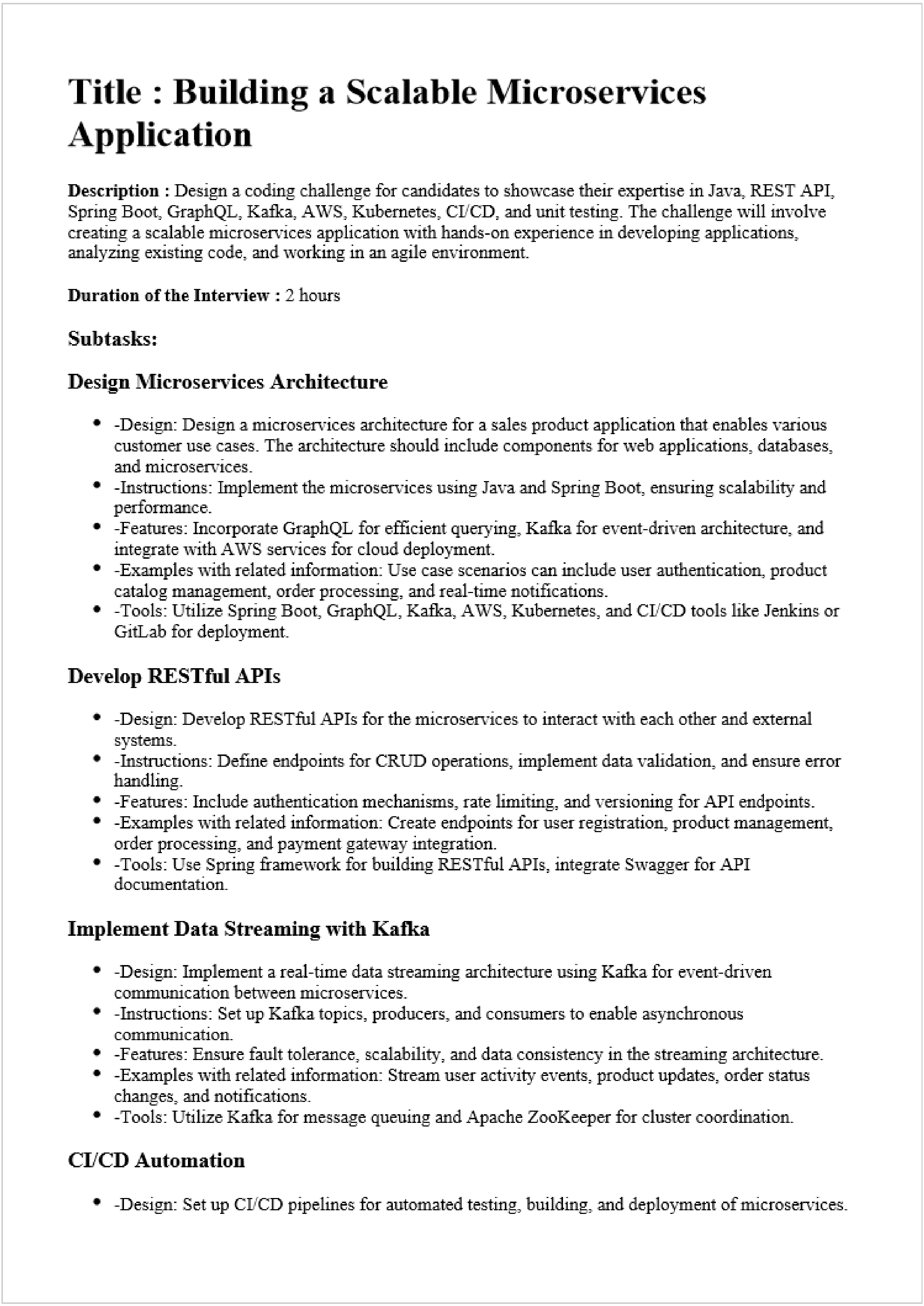
Coding Test

29 April 2024 09:54

UseCase



Title : Building a Scalable Microservices Application

Description : Design a coding challenge for candidates to showcase their expertise in Java, REST API, Spring Boot, GraphQL, Kafka, AWS, Kubernetes, CI/CD, and unit testing. The challenge will involve creating a scalable microservices application with hands-on experience in developing applications, analyzing existing code, and working in an agile environment.

Duration of the Interview : 2 hours

Subtasks:

Design Microservices Architecture

- -Design: Design a microservices architecture for a sales product application that enables various customer use cases. The architecture should include components for web applications, databases, and microservices.
- -Instructions: Implement the microservices using Java and Spring Boot, ensuring scalability and performance.
- -Features: Incorporate GraphQL for efficient querying, Kafka for event-driven architecture, and integrate with AWS services for cloud deployment.
- -Examples with related information: Use case scenarios can include user authentication, product catalog management, order processing, and real-time notifications.
- -Tools: Utilize Spring Boot, GraphQL, Kafka, AWS, Kubernetes, and CI/CD tools like Jenkins or GitLab for deployment.

Develop RESTful APIs

- -Design: Develop RESTful APIs for the microservices to interact with each other and external systems.
- -Instructions: Define endpoints for CRUD operations, implement data validation, and ensure error handling.
- -Features: Include authentication mechanisms, rate limiting, and versioning for API endpoints.
- -Examples with related information: Create endpoints for user registration, product management, order processing, and payment gateway integration.
- -Tools: Use Spring framework for building RESTful APIs, integrate Swagger for API documentation.

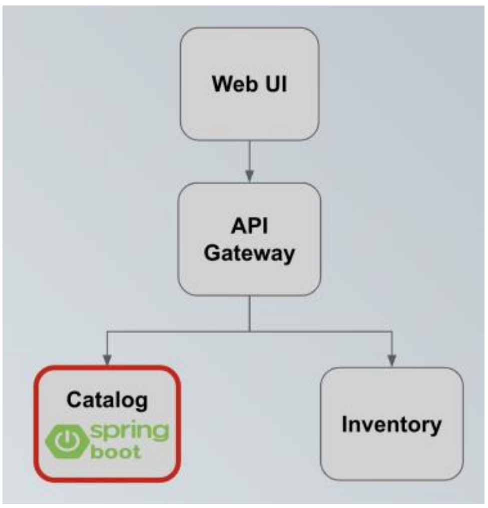
Implement Data Streaming with Kafka

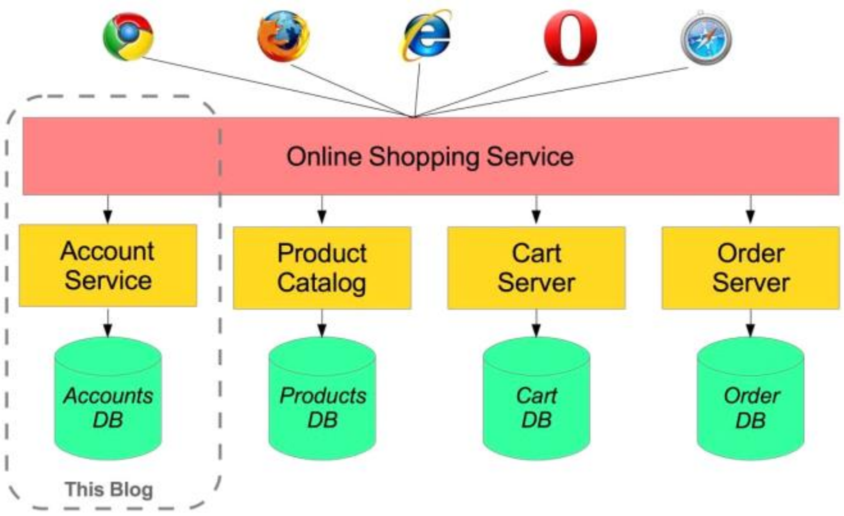
- -Design: Implement a real-time data streaming architecture using Kafka for event-driven communication between microservices.
- -Instructions: Set up Kafka topics, producers, and consumers to enable asynchronous communication.
- -Features: Ensure fault tolerance, scalability, and data consistency in the streaming architecture.
- -Examples with related information: Stream user activity events, product updates, order status changes, and notifications.
- -Tools: Utilize Kafka for message queuing and Apache ZooKeeper for cluster coordination.

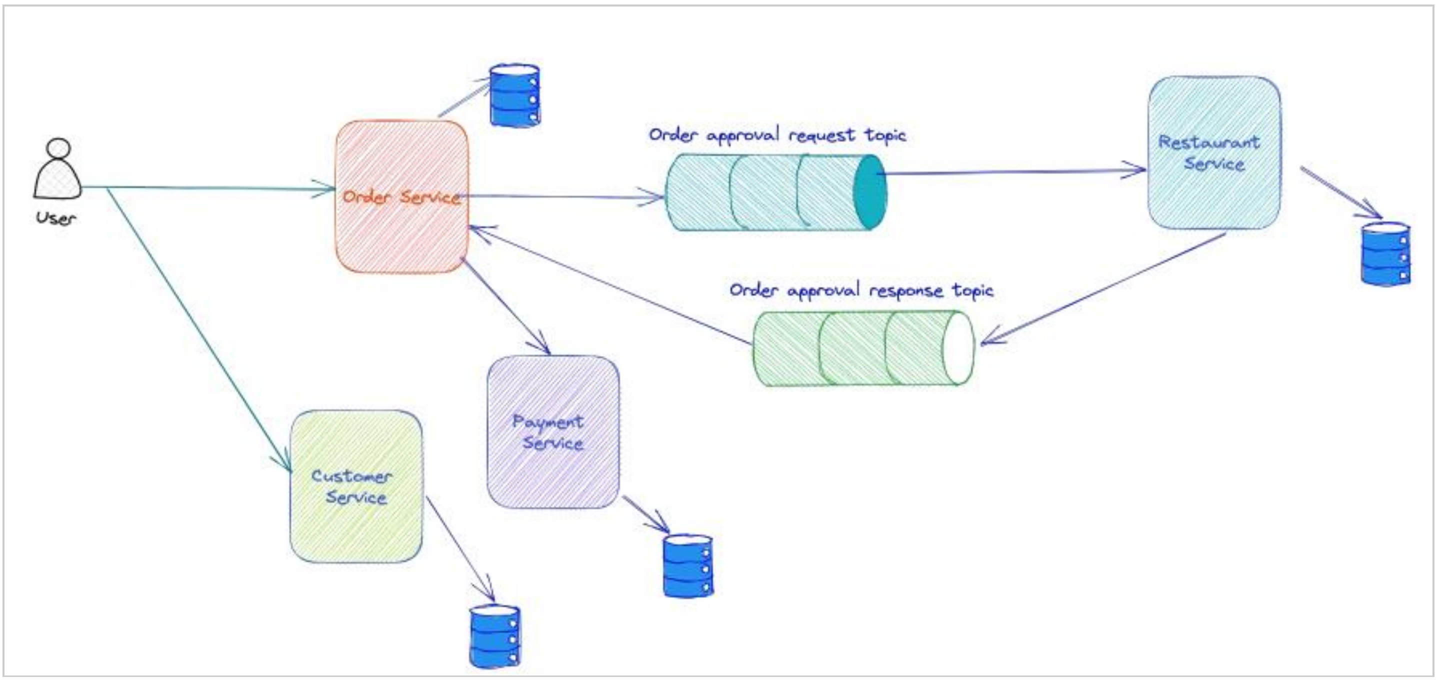
CI/CD Automation

- -Design: Set up CI/CD pipelines for automated testing, building, and deployment of microservices.

- -Instructions: Create Jenkins pipelines to trigger unit tests, integration tests, and deployment to Kubernetes clusters.
- -Features: Implement smoke tests, performance tests, and security scans in the CI/CD process.
- -Examples with related information: Configure webhook triggers, deployment scripts, and environment configurations.
- -Tools: Leverage Jenkins for CI/CD automation, Docker for containerization, and Kubernetes for container orchestration.







E-Commerce

Relational database schema design

UserService
    UserAccount
        UserId [Auto generated, Primary key]
        UserName

```
                UserType [customer/vendor/admin]
                UserPhone
                CartId [Foreign Key to CartService CartId]
                AddrId List [Foreign Key to UserAddress ArrdId]
                OrderId List [Foreign Key to OrderService OrderId]
        UserAddress
                AddrId [Auto generated, PrimaryKey]
                IsAddrPrimmry [boolean Type]
                AddrLine1
                AddrLine2
                LandMark
                City
                State
                Counttry
                Pincode
                PhoneNumber

OrderService
        OrderId [Auto generated, PrimaryKey]
        OrderStatus
        OrderDate
        PaymentId [Foreign Key to PaymentService PaymentId]
        ProductList[]

ProductService
        ProductId [Auto generated, PrimaryKey]
        ProductType [Scale of 1 to 10]
        ProductCategory Solid/Liquid/Gas
        ProductMRP
        ProductMargine
        ProductStatus
        ProductImagesList[]

CartService
        CartId [Auto generated, PrimaryKey]
        UserId
        ProductList[]

PaymentService [Auto generated, PrimaryKey]
        PaymentId
        TransactionId
        PaymentMode
        PaymentStatus
        PaymentCurrency
        PaymentCountry
        PaymentDate
```

Create Query

```sql
CREATE TABLE UserAccount (
   UserId INT PRIMARY KEY AUTO_INCREMENT,
   UserName VARCHAR(255),
   UserType ENUM('customer', 'vendor', 'admin'),
   UserPhone VARCHAR(20),
   CartId INT,
   AddrId INT,
   OrderId INT,
   FOREIGN KEY (CartId) REFERENCES CartService(CartId),
   FOREIGN KEY (AddrId) REFERENCES UserAddress(AddrId),
   FOREIGN KEY (OrderId) REFERENCES OrderService(OrderId)
);

CREATE TABLE UserAddress (
   AddrId INT PRIMARY KEY AUTO_INCREMENT,
   UserId INT,
   IsAddrPrimary BOOLEAN,
   AddrLine1 VARCHAR(255),
   AddrLine2 VARCHAR(255),
   Landmark VARCHAR(255),
   City VARCHAR(100),
   State VARCHAR(100),
   Country VARCHAR(100),
   Pincode VARCHAR(20),
   PhoneNumber VARCHAR(20),
   FOREIGN KEY (UserId) REFERENCES UserAccount(UserId)
);

CREATE TABLE OrderService (
   OrderId INT PRIMARY KEY AUTO_INCREMENT,
   OrderStatus VARCHAR(50),
   OrderDate DATE,
   PaymentId INT,
   FOREIGN KEY (PaymentId) REFERENCES PaymentService(PaymentId)
);

CREATE TABLE ProductService (
   ProductId INT PRIMARY KEY AUTO_INCREMENT,
   ProductType INT,
   ProductCategory ENUM('Solid', 'Liquid', 'Gas'),
   ProductMRP DECIMAL(10, 2),
   ProductMargin DECIMAL(10, 2),
   ProductStatus VARCHAR(50)
);

CREATE TABLE CartService (
   CartId INT PRIMARY KEY AUTO_INCREMENT,
   UserId INT,
   FOREIGN KEY (UserId) REFERENCES UserAccount(UserId)
);


CREATE TABLE PaymentService (
   PaymentId INT PRIMARY KEY AUTO_INCREMENT,
   TransactionId VARCHAR(100),
   PaymentMode VARCHAR(50),
   PaymentStatus VARCHAR(50),
   PaymentCurrency VARCHAR(10),
   PaymentCountry VARCHAR(100),
   PaymentDate DATE
);
```

Insert Query

```
-- Insert queries for UserAccount table
INSERT INTO UserAccount (UserName, UserType, UserPhone, CartId, AddrId, OrderId)
VALUES ('John Doe', 'customer', '1234567890', 1, 1, 1),
       ('Jane Smith', 'vendor', '9876543210', 2, 2, 2);

-- Insert queries for UserAddress table
INSERT INTO UserAddress (UserId, IsAddrPrimary, AddrLine1, AddrLine2, Landmark, City, State, Country, Pincode, PhoneNumber)
VALUES (1, true, '123 Main St', 'Apt 101', 'Near Park', 'New York', 'NY', 'USA', '10001', '1234567890'),
       (2, true, '456 Oak Ave', 'Suite 202', 'Next to Mall', 'Los Angeles', 'CA', 'USA', '90001', '9876543210');

-- Insert queries for OrderService table
INSERT INTO OrderService (OrderStatus, OrderDate, PaymentId)
VALUES ('pending', '2022-04-12', 1),
       ('shipped', '2022-04-13', 2);

-- Insert queries for ProductService table
INSERT INTO ProductService (ProductType, ProductCategory, ProductMRP, ProductMargin, ProductStatus)
VALUES (8, 'Solid', 50.0, 10.0, 'available'),
       (6, 'Liquid', 30.0, 5.0, 'available');

-- Insert queries for CartService table
INSERT INTO CartService (UserId)
VALUES (1),
       (2);

-- Insert queries for PaymentService table
```

```
INSERT INTO PaymentService (TransactionId, PaymentMode, PaymentStatus, PaymentCurrency, PaymentCountry, PaymentDate)
VALUES ('TXN123456', 'credit card', 'success', 'USD', 'USA', '2022-04-12'),
       ('TXN789012', 'paypal', 'pending', 'USD', 'USA', '2022-04-13');
```

Micro Services

```
WebClient
UserService [com.ecom.user, SpringController, RDBMS]
        - addUser
        - editUser
        - deleteUser
        - getUserById
        - getUserByCondition
OrderService [com.ecom.order, KafkaListener, MongoDB]
        - addOrder
        - editOrder [patch call]
        - deleteOrder
        - getOrderById
        - getUserByDateRange
ProductService [com.ecom.product, KafkaListener, MngoDB]
        - addProduct
        - editProduct
        - deleteProduct
        - getProductById
        - getproductByCategory
CartService [com.ecom.cart, SpringController, MongoDB]
        - addProduct
        - deleteProduct
PaymentService [com.ecom.payment, RDBMS]
        - addPayment
```

Sample Request and Response

…