Java/ Spring Boot Code Challenge

Requirements:

For this code challenge, you are required to create a web application using Spring Boot that integrates with both Salesforce and SQL Server databases. You will also need to implement Swagger for API documentation, secure the endpoints using Spring Security, and leverage Spring WebFlux for reactive programming. We would like you to showcase your skills in cloud implementation and the best practices that can be followed along the way.

Requirements:

Task 1: Setup

Create a new Spring Boot project using the Spring Initializr.

Add the necessary dependencies for Web, Spring Security, WebFlux, Salesforce integration, and SQL Server integration in your pom.xml or build.gradle.

Configure the Salesforce connection with the necessary credentials. (You can create a test account).

Task 2: Model

Create a Customer model class with the following attributes:

id (String)

name (String)

email (String)

phone (String)

Task 3: API Endpoints

Implement RESTful API endpoints for customer management:

POST /api/customers: Create a new customer.

GET /api/customers/{id}: Get a customer by ID.

GET /api/customers: Get all customers.

PUT /api/customers/{id}: Update a customer by ID.

DELETE /api/customers/{id}: Delete a customer by ID.

"Tips: Showcase your talent by securely exposing the APIs to the customer. You will be assessed on your ability to set up and manage the infrastructure."

Task 4: Swagger Integration

- ➢ Integrate Swagger into your project for API documentation.
- ➢ Configure Swagger to generate documentation for your customer management API.

Task 5: Spring Security

- ➢ Secure the API endpoints using Spring Security.
- ➢ Implement authentication and authorization for the endpoints using JWT (JSON Web Tokens).
- ➢ Users with the role "ADMIN" should be able to access all endpoints, while users with the role "USER" should have limited access.

Task 6: Salesforce Integration

- ➢ Implement methods to synchronize customer data between the SQL Server database and Salesforce. Ensure that a secure connection is created between the systems to keep the data in transit and at rest secure.
- ➢ Create a scheduler or a manual trigger to initiate the synchronization process.

Task 7: Reactive Programming

- ➢ Modify your API implementation to use Spring WebFlux for reactive programming.
- ➢ Ensure the API performs non-blocking operations when interacting with the SQL Server database and Salesforce.

Task 8: Testing

- ➢ Write unit tests for the API endpoints and the data synchronization process.
- ➢ Verify that secured endpoints can only be accessed with valid JWT tokens.

Task 9: Documentation

- ➢ Provide clear and concise documentation on how to set up and run the application.
- ➢ Include details on the API endpoints, authentication, data synchronization process, and any additional features you added.

Submission:

Submit your code in a GitHub repository with clear instructions on how to set up and run the application. Also, provide a README.md file documenting your implementation, including details about the API endpoints, authentication, Salesforce integration, SQL Server integration, and any additional features you added.

Note: The challenge is designed to assess your skills in Spring Boot, Swagger, Spring Security, WebFlux, Salesforce integration, and SQL Server integration. Feel free to add your creative touches and optimizations, and make sure your code is clean, well-organized, and follows best practices.

Tools/Framework to be used:

1. Spring Boot / Batch /Apache Camel – Use of JPA is a must
2. Spring Security
3. Reactive Programming
4. AWS instances for infrastructure setup.
5. SQL Server 2017 (Data Modelling to be done)
6. Salesforce Test Account

Directions:

Feel free to make any assumptions where required to complete the exercise unless they contradict the specified business rules and/or requirements.

• We are not aiming for a perfect, production-ready solution. Only spend enough time to produce a testable and maintainable solution for the given requirements.

• Follow best practices while creating the solution. Securing the data is a must, and more weightages will be given to this aspect.

• We will be focusing our attention on your integration skills; therefore, we will be evaluating the quality and structure of your code and programming skills.

• Document the infrastructure setup with screenshots and provide the Postman collections along with build details.

• Define the Restful API schema.

• Feel free to contact us if you need any help with setting up the infrastructure on your local machine or AWS Account.