Stripe Integration for Customer Creation using Temporal Workflow

# Overview

This documentation outlines the tasks involved in integrating Stripe for customer creation within a Spring Boot application that leverages the Temporal Workflow Engine. The project is set up with Java 21, the provided boilerplate code and the Temporal Workflow Engine to orchestrate the necessary business logic.

## Prerequisites

- Java Development Kit (JDK) 21

- [Gradle](https://gradle.org/) for managing dependencies

- [Stripe SDK](https://stripe.com/docs/libraries/java) added to Gradle dependencies

-[GitHub Repository](https://github.com/Midas-Labs/backend-engineer-assessment) cloned locally

### **Task 1: Stripe Integration for Customer Creation**

### 

**1.1 Implementation Overview**

Upon a new user signup, the goal is to integrate the Stripe Create Customer API to create a new customer in Stripe. The Stripe SDK is already added to the project's Gradle dependencies, and boilerplate code is provided. Task is to implement the required workflow using Temporal.

**API secret key :** sk\_test\_51OgXycSDITZLM05NldBuORn03XkMEd7tsESrRIVld4pipM8WNMnFgkBedByterKI4SwMPOVrms1v6ow5t54jqfKG00PTlsSaqK

**1.2 Temporal Workflow Implementation**

**1. Workflow Interface:**

- Define a Temporal workflow interface (`Workflow`) with a method to create a new customer in Stripe (`createStripeCustomer`).

**2. Workflow Implementation:**

- Implement the `createStripeCustomer` method to interact with the Stripe SDK and create a new customer.

- Handle any necessary error scenarios and retries.

**3. Invoke Workflow:**

- Integrate the workflow into the user signup process to trigger the creation of a Stripe customer upon signup. (application.properties)

### **Task 2: Add Fields in the APIs**

**2.1 Update User Model**

- Add a new field `providerType` to the user model with an enum type having values: `stripe`.

- Add a `providerId` field to store the generated Stripe customer ID.

**2.2 Update Application Controller**

- Update the application controller to handle the new fields during the user signup process.

- Ensure that the `providerId` is generated and stored appropriately.

### **Task 3: API Implementation**

**3.1 Verify Integration**

- A `GET /accounts` endpoint is already implemented within the codebase

## **Project Setup**

**1. Clone the GitHub repository:** [backend-engineer-assessment](<https://github.com/Midas-Labs/backend-engineer-assessment>)

2. Open the project in your preferred IDE (Eclipse, IntelliJ, etc.).

3. Ensure that Java 21 and Gradle are properly configured.

4. Verify that the Stripe SDK is added to the Gradle dependencies.

**Running the Application**

- Build and run the Spring Boot application using the provided Gradle wrapper:

[ ./gradlew bootRun ]

- Access the application at [http://localhost:8080](http://localhost:8080).