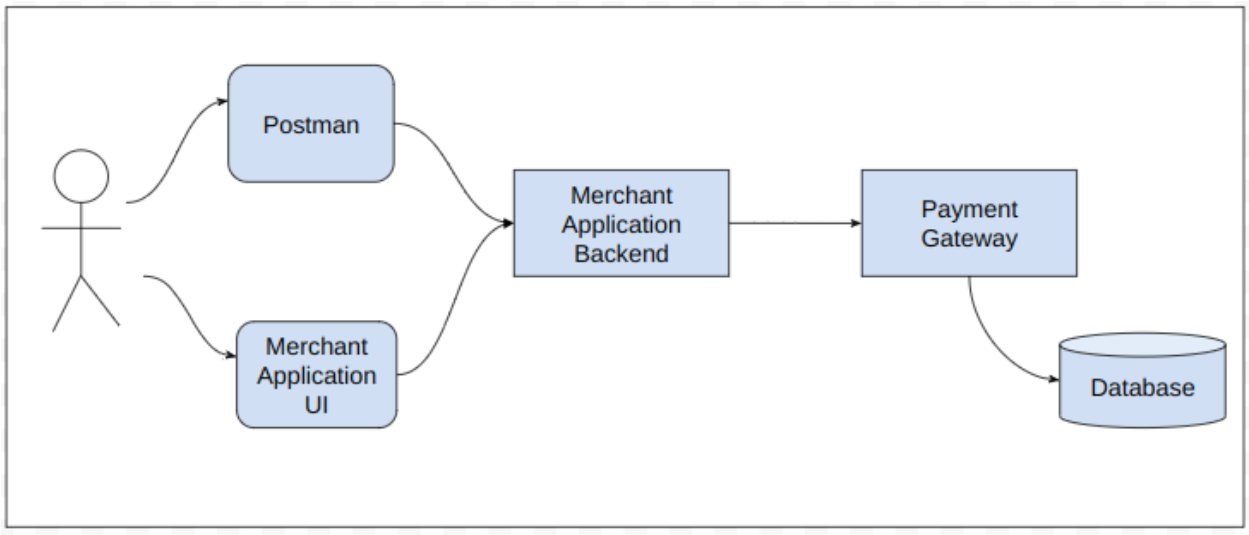
Exercise: Payment Gateway Integration

Scenario

ABC Corporation (merchant) sells furniture. They have created a website for selling the furniture online. To provide online payment functionality they have contracted with one Payment Gateway which processes card payments. Payment Gateway provides APIs related to payments.

You are tasked with building Spring Boot applications that demonstrate the integration of a merchant application with a payment gateway for processing payments.

Think of it as a basic interaction between a client and server application. In this case, the client is the simple Merchant's Spring Boot application. It needs to connect with the server which is a Payment Gateway Spring Boot application.



Requirements:

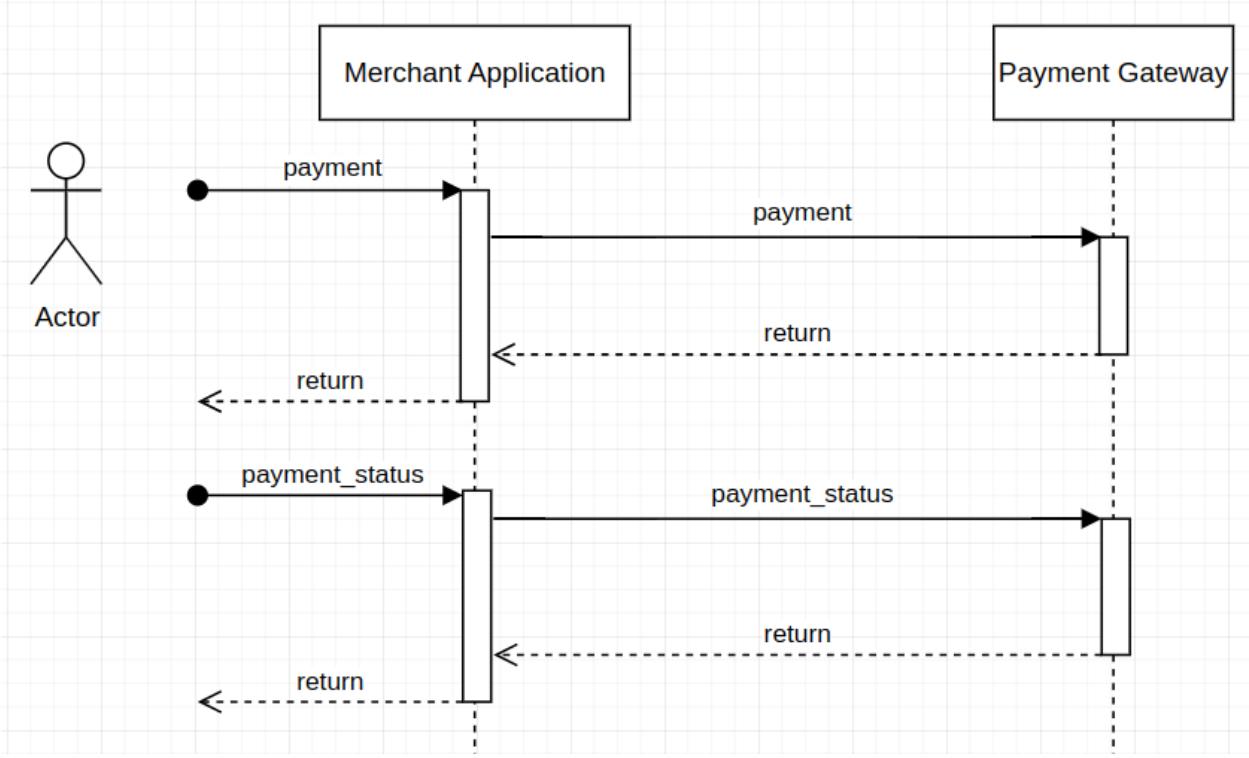
As part of this exercise, you need to create 3 components:

01. **Payment Gateway**: Create a REST-based Spring Boot app that simulates a Payment Gateway. It offers 2 APIs for merchant applications:
    a. **Make Payment**: This API needs to capture the payment details and respond with a message that says if the payment worked or not. It also returns a unique code (payment reference) for each payment request.
    b. **Check Payment Status:** This API takes payment reference as input and returns payment details with the current status (captured, processing, processed, failed, etc.)
02. **Merchant Application Backend:** Create a REST-based Spring Boot app that exposes endpoints for UI and communicates with Payment Gateway to fulfill payments. It offers 2 APIs for UI applications:
    a. **Make Payment:** This API needs to capture the payment details, make use of Payment Gateway's MakePayment API to process the payment, and respond with a message that says if the payment worked or not. It also returns a unique code (payment reference) received from Payment Gateway for each payment request
    b. **Check Payment Status:** This API takes payment reference as input, makes use of Payment Gateway's CheckPaymentStatus API, and returns payment details with the current status received from Payment Gateway.
03. **Merchant Application UI:** Create one UI application that makes use of Merchant-Application-Backend APIs to make payments and check payment status for existing payments. You need to create two pages:
    a. **Make Payment Page**: prepare a form to capture the payment details and show payment status and payment reference after getting a response from the backend.
    b. **Check Payment Status Page**: prepare a form to capture payment reference and show payment details with status after getting a response from the backend.

Project Setup:

1. Create two Spring Boot projects with Maven (one for "***Merchant Application***" and another for "***Payment Gateway Application***"
2. Configure the project to use appropriate dependencies for web development and integration between both applications.

Flow Diagram



Development Approach:

1. You have two options for submitting the source code: either as a zip file or by providing a GitHub link.
2. You can use either a relational or a NoSQL database.
3. For local databases, it's recommended to utilize the MySQL or PostgreSQL Docker Hub images.

Deliverables:

Source Code:

Organize your project following the Maven structure. You have the flexibility to package the source code, SQL scripts, and related files either as a compressed zip archive or upload them to GitHub. Please provide us with the link to access these resources. You can follow the same method for sharing documentation and any recording you might have.

Additionally, you can include SQL scripts within the same source repository. Be sure to outline the specifics of these scripts in the provided documentation.

Documentation:

As part of what we're looking for, we want a simple document.

This document should have easy-to-follow steps for setting up different tools and softwares. It should also cover things like how different parts of the solution are designed, such as components, APIs, and database tables.

The main goal of this document is to provide clear, step-by-step instructions that even someone new to development can follow without knowing much about the software involved. The idea is that they can follow the instructions and recreate the whole solution successfully.

Record yourself:

If you're interested in sharing a brief video where you explain how you created the sample system, including a walkthrough of the steps while recording your screen (and possibly including a video of yourself), you have the option to use the recorder provided below.

This could be really useful for us to grasp how you tackled the problem and any challenges you faced along the way.

You can use chrome extension screen recorder to record your screen
https://chrome.google.com/webstore/detail/screen-recorder/hniebljpgcogalllopnjokppmgbhaden

Sample messages:

Here's a sample interface for the PaymentGateway's "Make Payment Request" API request:

API Endpoint: POST /payment

Request Body:

```
{
 "merchantId": "your-merchant-id",
 "amount": 50.00,
 "currency": "USD",
 "orderId": "order123"
}
```

Response Body:

```
{
 "status": "success",
 "paymentRef": "ref1234"
}
```

Here's a sample interface for the Merchant Application's "Make Payment Request" API request:

API Endpoint: POST /payment

Request Body:

```
{
 "amount": 50.00,
 "currency": "USD",
 "orderId": "order123"
}
```

Response Body:

```
{
 "status": "success",
 "paymentRef": "ref1234"
}
```

Here's a sample for payment status check request (for both applications):

API Endpoint: GET /payment_status/{paymentRef}

In this example, {paymentRef} should be replaced with the unique identifier of the payment you want to check.

No request body is needed for this API as it's a GET request.

ALL THE BEST