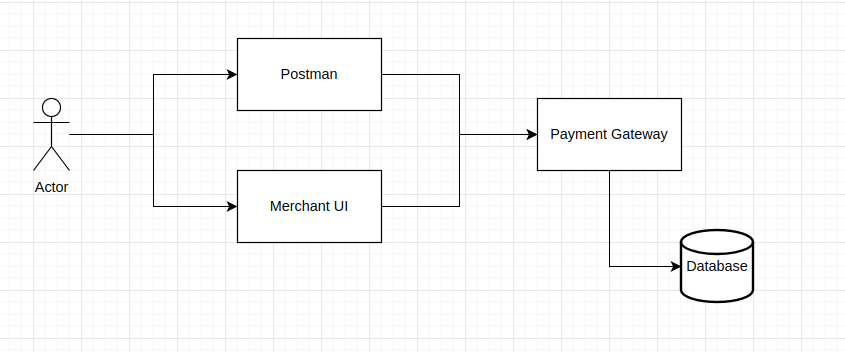
Exercise: Payment Gateway Integration

# Scenario

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:

* Create a REST based Spring Boot application for **merchants** with these 3 features:

1. **Registration:** Make an API call so the application can sign up with the Payment Gateway. Imagine there could be lots of these merchant apps.
2. **Payment:** Use an API call to send a payment request to the Payment Gateway. It will then reply with a random message saying if the payment was successful or not.
3. **Check Payment Status:** Make an API call to ask about payments that were done before. Use a unique identifier(provided by the Gateway) for each payment request.

This Merchant UI will allow the user to verify the three above functionalities.

* Create a REST based Spring Boot app that simulates a **Payment Gateway**. It offers 3 APIs for merchant apps:

1. **Register**: When a Merchant App asks to register, check if they're already registered and reply accordingly.
2. **Payment**: Keep a record of payment requests and reply with a random message that says if the payment worked or not. Attach a special code to each payment request.
3. **Check Payment Status**: If a payment request with a special code is in the database, reply with its payment status.

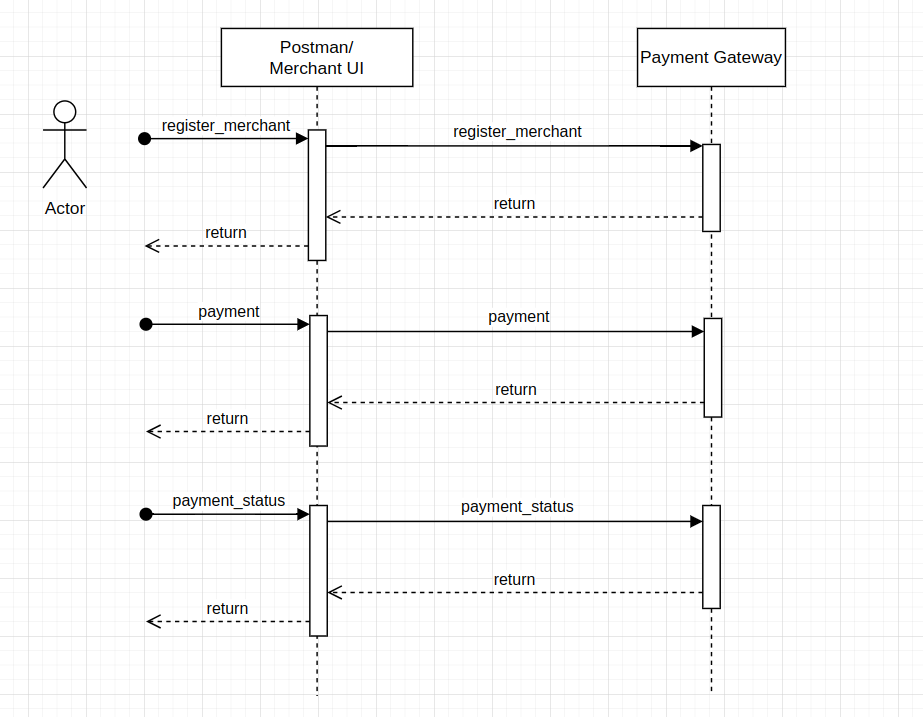
You could use Postman to verify if the above 3 APIs are functioning properly.

## 

## Project Setup:

1. Create two Spring Boot project with Maven (one for “***Merchant UI***” and another for “***Payment Gateway Simulator***”
2. Configure the project to use appropriate dependencies for web development and integration between both the 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 "Register Merchant" API request:**

API Endpoint: POST /register\_merchant

Request Body:

{

"merchantId": "your-merchant-id",

"name": "Merchant Name",

"email": "merchant@example.com",

"businessType": "Online Retail",

"address": "123 Main St, City",

"phone": "123-456-7890"

}

**Here's a sample interface for the “Payment Request” API request:**

API Endpoint: POST /payment

Request Body:

{

"merchantId": "your-merchant-id",

"amount": 50.00,

"currency": "USD",

"orderId": "order123"

}

**Here's a sample for payment status check request:**

API Endpoint: GET /payment\_status/{paymentId}

In this example, {paymentId} 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**