Project 1: Payment API Integration - Step-by-Step Guide

# Project Overview

The goal of this project is to develop a backend API that simulates processing payments, logs transactions to a MySQL database, and deploys using AWS Lambda. This project demonstrates skills in backend development, database integration, and cloud deployment.

# Step-by-Step Process

## Step 1: Set Up the Environment (2-3 hours)

1. Install necessary software: Python, Node.js, MySQL.  
2. Set up a virtual environment and install dependencies (Flask, SQLAlchemy, AWS SDK).  
3. Configure a local MySQL database for storing transaction data.

## Step 2: Design API Structure (1-2 hours)

1. Define API endpoints (e.g., /process-payment, /transaction-history).  
2. Create a UML diagram to visualize the data flow and structure.

## Step 3: Implement Payment Processing Endpoint (3-4 hours)

1. Create the Flask application.  
2. Implement the /process-payment endpoint to receive and log transactions.  
3. Validate transaction data (amount, currency, payment method).

## Step 4: Implement Transaction History Endpoint (2-3 hours)

1. Query the MySQL database to retrieve transaction logs.  
2. Format the response as JSON.  
3. Implement pagination for large datasets.

## Step 5: Connect to AWS Lambda (3-4 hours)

1. Package the API into a zip file for Lambda deployment.  
2. Configure AWS Lambda with API Gateway for public access.  
3. Test the API endpoint through AWS.

## Step 6: Testing and Debugging (2-3 hours)

1. Unit test individual functions (e.g., data validation, database interactions).  
2. Perform integration tests with the AWS-hosted API.  
3. Fix any bugs or inconsistencies.

## Step 7: Documentation and Finalization (1-2 hours)

1. Write README.md explaining setup, usage, and endpoints.  
2. Create API documentation using Swagger or Postman.  
3. Perform final code cleanup and push to GitHub.