Multi-Tenant eCommerce Platform with Custom Payment Gateway

Table of Contents

---

1. Introduction

The Multi-Tenant eCommerce Platform with a Custom Payment Gateway is a modern web application designed to facilitate online shopping. It supports multiple tenants, integrates with Stripe for payment processing, and provides a secure and scalable solution for eCommerce needs.

2. Project Objectives

- **Multi-Tenancy Support:** Enable multiple tenants to operate within a single application instance.
- **Custom Payment Gateway:** Integrate with Stripe to handle payments.
- **Secure Authentication:** Use OAuth2 for secure access and user management.
- **Product Management:** Provide CRUD operations for managing products.
- **Scalability and Performance:** Optimize for performance and scalability.

3. Technologies Used

- **Java 17:** Latest version for modern Java features.
- **Spring Boot 3.1.1:** Framework for building the application.
- **Spring Security:** Security and authentication.
- **Spring Data JPA:** ORM for database interaction.
- **MySQL:** Relational database management system.
- **Stripe API:** Payment gateway integration.
- **HikariCP:** High-performance JDBC connection pool.
- **Maven:** Build and dependency management tool.

4. System Architecture

4.1 High-Level Architecture

The system follows a layered architecture pattern:

- **Presentation Layer:** Handles user interface and API requests.
- **Service Layer:** Contains business logic.
- **Data Access Layer:** Manages database interactions.

4.2 Component Diagram

Include a diagram showing the interaction between components such as the frontend, backend services, and third-party APIs.

5. Database Design

5.1 Schema Overview

- **Products Table:** Stores product details.
- **Payments Table:** Tracks payment transactions.

5.2 Entity Relationships

Provide an Entity-Relationship Diagram (ERD) showing the relationships between `Product` and `Payment` entities.

6. API Endpoints

6.1 Product Management

- **Get All Products:**

```
http
Copy code
GET /api/products
```

- **Add a Product:**

```
http
Copy code
POST /api/products
Content-Type: application/json

{
    "name": "Sample Product",
    "price": 49.99
}
```

6.2 Payment Processing

- **Process a Payment:**

```
http
Copy code
POST /api/payment
Content-Type: application/json

{
    "amount": 100.0,
    "currency": "USD"
}
```

7. Security

7.1 OAuth2 Authentication

- **Client Registration:** Details for client setup in `application.properties`.
- **Security Configuration:** Uses Spring Security to secure endpoints.

7.2 Data Protection

- **Encryption:** Sensitive data is encrypted using industry-standard techniques.
- **Access Control:** Role-based access control is implemented.

8. Stripe Integration

8.1 API Keys

- **Setup:** Configuration in `application.properties`.

8.2 Payment Flow

- **Create Payment Intent:** Description of how payments are created and processed using the Stripe API.

8.3 Error Handling

- **Handling Failures:** Strategies for managing payment failures and exceptions.

9. Setup and Installation

9.1 Prerequisites

- **Java 17**
- **Maven**
- **MySQL**
- **Stripe Account**

9.2 Installation Steps

1. **Clone the Repository**
2. **Configure `application.properties`**
3. **Build the Project**
4. **Start MySQL**

10. Running the Application

10.1 Start the Application

Use the Maven command to run the application:

```
bash
Copy code
mvn spring-boot:run
```

10.2 Access the Application

The application will be available at `http://localhost:8080`.

11. Testing

11.1 Unit Tests

- **PaymentServiceTest:** Tests for payment processing.
- **ProductServiceTest:** Tests for product management.

11.2 Integration Tests

- **End-to-End Testing:** Testing of the complete flow from product creation to payment.

12. User Guide

12.1 Accessing APIs

Instructions on using Postman or similar tools to interact with the APIs.

12.2 Sample Requests

Provide example requests and responses for different endpoints.

13. Future Enhancements

- **Multi-Currency Support:** Expand payment gateway to handle multiple currencies.
- **Advanced Analytics:** Implement reporting and analytics features.
- **User Profiles:** Extend authentication to include user profiles and roles.

14. Challenges and Solutions

14.1 Multi-Tenancy Implementation

- **Challenge:** Managing multiple tenants in a single database.
- **Solution:** Implemented schema-based multi-tenancy.

14.2 Stripe Integration Issues

- **Challenge:** Handling Stripe API changes and errors.
- **Solution:** Implemented robust error handling and logging.

15. Conclusion

The Multi-Tenant eCommerce Platform is a comprehensive solution that addresses modern eCommerce needs with support for multiple tenants and secure payment processing. The project is designed for scalability and performance, making it suitable for various business applications.

16. References

- Spring Boot Documentation
- Stripe API Documentation
- MySQL Documentation
- OAuth2 Documentation