Absolutely—here's a **complete API contract** for the **Payment Gateway module**. This covers merchant registration, authentication, payment lifecycle management, and optional webhook simulation. It's structured so you can implement it cleanly in Spring Boot.

📜 Payment Gateway API Contract

1. Merchant Management

POST /merchants

- **Description:** Register a new merchant.
- **Request:**

```
{
  "name": "Tulip Store"
}
```

- **Response:**

```
{
  "id": 1,
  "name": "Tulip Store",
  "apiKey": "abc123xyz"
}
```

- **Notes:** API key is generated and returned once. Merchant must store it securely.

---

2. Authentication

Header Requirement

- All merchant requests must include:

```
Authorization: ApiKey <merchant-api-key>
```

---

3. Payment Lifecycle

POST /payments

- **Description:** Create a new payment request.
- **Headers:** `Authorization: ApiKey abc123xyz`
- **Request:**

```
{
  "amount": 49.99,
  "currency": "USD",
  "description": "Order #1234"
}
```

- **Response:**

```
{
  "paymentId": 101,
  "status": "PENDING",
  "createdAt": "2025-11-09T12:00:00Z"
}
```

GET /payments/{id}

- **Description:** Retrieve payment details and status.
- **Response:**

```
{
  "paymentId": 101,
  "amount": 49.99,
  "currency": "USD",
  "status": "PENDING",
  "description": "Order #1234",
  "createdAt": "2025-11-09T12:00:00Z"
}
```

POST /payments/{id}/complete

- **Description:** Simulate payment completion (customer pays).
- **Request:**

```
{
  "status": "SUCCESS"
}
```

- **Response:**

```
{
  "paymentId": 101,
  "status": "SUCCESS"
}
```

- **Notes:** Status can be `SUCCESS` or `FAILED`.

POST /payments/{id}/refund

- **Description:** Issue a refund for a completed payment.
- **Request:**

```
{
  "reason": "Customer returned item"
}
```

- **Response:**

```
{
  "paymentId": 101,
  "status": "REFUNDED",
  "refundedAt": "2025-11-09T14:00:00Z"
}
```

---

4. Webhook Simulation (Optional)

POST /webhooks/payments

- **Description:** Simulate sending a webhook notification to merchant's system.
- **Request:**

```
{
  "paymentId": 101,
  "status": "SUCCESS",
  "timestamp": "2025-11-09T12:30:00Z"
}
```

- **Response:**

```
{
  "message": "Webhook delivered successfully"
}
```

---

5. Error Handling

Error Response Format

```
{
  "error": "Invalid API key",
  "code": 401,
  "timestamp": "2025-11-09T12:05:00Z"
}
```

⚖️ Summary of Endpoints

Endpoint	Method	Purpose	Auth Required
`/merchants`	POST	Register merchant, issue API key	No
`/payments`	POST	Create payment request	Yes
`/payments/{id}`	GET	Retrieve payment details	Yes
`/payments/{id}/complete`	POST	Complete payment (simulate)	Yes
`/payments/{id}/refund`	POST	Refund payment	Yes
`/webhooks/payments`	POST	Simulate webhook notification	Yes

✅ This contract gives you a **minimal viable gateway** with lifecycle management, merchant onboarding, and extensibility for features like Pay by Link or fraud detection.

Would you like me to also draft the **database schema (tables + relationships)** that supports these endpoints (Merchant, Payment, Transaction, WebhookLog)? That way you'll have the backend persistence model aligned with the API.