

Payment Service - Event Ticketing System

Author - Amitosh Gautam

Overview

This documentation describes the design, development, and deployment of the Payment Service, which is one of the key microservices in the Event Ticketing and Seat Reservation System (ETS). The Payment Service manages financial transactions, including processing user charges, issuing refunds, and maintaining idempotent request handling to prevent duplicate operations. It operates as an independent Spring Boot microservice following the database-per-service pattern for full data isolation and scalability.

Service Responsibilities

The Payment Service is responsible for:

- Processing new payment charges from confirmed orders.
- Recording and updating transaction statuses – PENDING, SUCCESS, FAILED, REFUNDED.
- Handling refund requests for successful payments.
- Enforcing idempotency using an Idempotency-Key to prevent duplicate charges.
- Providing RESTful APIs for order and finance modules to initiate or verify payments.
- Persisting all transactions in a dedicated PostgreSQL database.

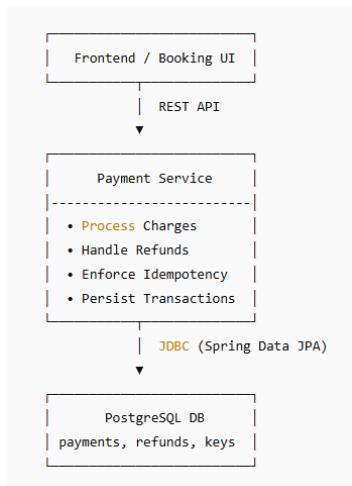
Each payment record is uniquely linked to an Order ID from the Order Service.

Architecture and Design

The Payment Service follows a four-layered architecture:

- Controller Layer → Exposes REST endpoints to clients and other services.
- Service Layer → Implements business logic for charge processing, refunds, and idempotency.
- Repository Layer → Handles database persistence via Spring Data JPA.
- Model Layer → Defines entity classes for Payment, Refund, and Idempotency Key.

Architecture Diagram-



Entity Relationship Diagram (ERD)-

Entities:

1. Payment
 - o Tracks each charge request.
 - o Linked to an orderId.
 - o Has status, amount, method, and reference.
2. Refund
 - o Linked to a paymentId.
 - o Tracks refund amount and reference number.
3. IdempotencyKey
 - o Stores request fingerprints and cached responses to ensure idempotent POST operations.

Relationships:

- One Payment can have multiple Refunds.
- Each IdempotencyKey is unique per API call.

REST API Endpoints -

Method	Endpoint	Description
POST	/v1/payments/charge	Process a payment charge (requires Idempotency-Key header)
POST	/v1/payments/refund	Refund a completed payment
GET	/v1/payments/{id}	Retrieve details of a specific payment
GET	/v1/payments/health	Health check endpoint
GET	/actuator/prometheus	Prometheus metrics for monitoring

Containerization with Docker –

The Payment Service is fully containerized using Docker.

- Dockerfile: Builds a lightweight image from eclipse-temurin:17-jdk-alpine.
- docker-compose.yml: Deploys both the application and a dedicated PostgreSQL database.

Build and Run Commands:

```
mvn clean package -DskipTests  
docker-compose up --build
```

Once running:

- Application: <http://localhost:8080/v1/payments/health>
- Database: localhost:5432 (paymentsdb)

```
C:\Windows\System32\cmd.exe
[WARNING] File encoding has not been set, using platform encoding Cp1252, i.e. build is platform dependent!
[INFO] Compiling 11 source files to C:\Users\user\Desktop\New folder (3)\target\classes
[WARNING] /:/Users/user/Desktop/New folder (3)/src/main/java/com/ticketing/payment/service/PaymentService.java: C:\Users\user\Desktop\New folder (3)\src\main\java\com\ticketing\payment\service\PaymentService.java uses unchecked or unsafe operations.
[WARNING] /:/Users/user/Desktop/New folder (3)/src/main/java/com/ticketing/payment/service/PaymentService.java: Recompile with -Xlint:unchecked for details.
[INFO]
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ payment-service ---
[WARNING] Using platform encoding (Cp1252 actually) to copy filtered resources, i.e. build is platform dependent!
[INFO] Copying 0 resource
[INFO]
[INFO] --- maven-compiler-plugin:3.1:testCompile (default-testCompile) @ payment-service ---
[INFO] Changes detected - recompiling the module!
[WARNING] File encoding has not been set, using platform encoding Cp1252, i.e. build is platform dependent!
[INFO] Compiling 1 source file to C:\Users\user\Desktop\New folder (3)\target\test-classes
[INFO]
[INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ payment-service ---
[INFO] Tests are skipped.
[INFO]
[INFO] --- maven-jar-plugin:2.4:jar (default-jar) @ payment-service ---
[INFO] Building jar: C:\Users\user\Desktop\New folder (3)\target\payment-service-1.0.0.jar
[INFO]
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 01:20 min
[INFO] Finished at: 2025-11-02T21:32:11+05:30
[INFO]
C:\Users\user\Desktop\New folder (3)>
```

```
C:\Windows\System32\cmd.exe
[INFO] --- maven-jar-plugin:2.4:jar (default-jar) @ payment-service ---
[INFO] Building jar: C:\Users\user\Desktop\New folder (3)\target\payment-service-1.0.0.jar
[INFO]
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 01:06 min
[INFO] Finished at: 2025-11-02T21:53:30+05:30
[INFO]

C:\Users\user\Desktop\New folder (3)>dir target
Volume in drive C has no label.
Volume Serial Number is F830-1582

Directory of C:\Users\user\Desktop\New folder (3)\target

11/02/2025  09:53 PM    <DIR>        .
11/02/2025  09:53 PM    <DIR>        ..
11/02/2025  09:53 PM    <DIR>        classes
11/02/2025  09:52 PM    <DIR>        generated-sources
11/02/2025  09:53 PM    <DIR>        generated-test-sources
11/02/2025  09:53 PM    <DIR>        maven-archiver
11/02/2025  09:52 PM    <DIR>        maven-status
11/02/2025  09:53 PM           14,686 payment-service-1.0.0.jar
11/02/2025  09:53 PM    <DIR>        test-classes
11/02/2025   1 File(s)      14,686 bytes
     8 Dir(s)  45,692,452,864 bytes free

C:\Users\user\Desktop\New folder (3)>jar tf target/payment-service-1.0.0.jar | find "MANIFEST.MF"
META-INF/MANIFEST.MF

C:\Users\user\Desktop\New folder (3)>docker build -t payment-service:local .
[+] Building 17.5s (8/8) FINISHED                                            docker:desktop-linux
=> [internal] load build definition from Dockerfile                         1.4s
=> [internal] transfer Dockerfile: 464B                                     0.6s
=> [internal] load metadata for docker.io/library/eclipse-temurin:17-jdk-alpine 4.4s
=> [internal] load .dockerignore                                         0.3s
=> [internal] transfer context: 28                                         0.0s
=> [1/3] FROM docker.io/library/eclipse-temurin:17-jdk-alpine@sha256:eb42bc053cbff0d750d76fa0705b6faec2677131a1358d0bafcc844051b8872c 0.8s
=> [2/3] resolve docker.io/library/eclipse-temurin:17-jdk-alpine@sha256:eb42bc053cbff0d750d76fa0705b6faec2677131a1358d0bafcc844051b8872c 0.75s
=> [internal] load build context                                         0.5s
=> [internal] transfer context: 14.78kB                                    0.0s
=> CACHED [2/3] WORKDIR /app                                           0.0s
=> [3/3] COPY target/payment-service-1.0.0.jar app.jar                  1.3s
```

```

C:\Windows\System32\cmd.exe
=> [internal] transferring dockerfile: 464B
=> [internal] load metadata for docker.io/library/eclipse-temurin:17-jdk-alpine
=> [internal] load .dockerrigore
=> [internal] transfering context: 2B
=> [1/3] FROM docker.io/library/eclipse-temurin:17-jdk-alpine@sha256:eb42bc053cbff0d750d76fa0705b6faec2677131a1358d0bafcc844051b8872c
=> resolve docker.io/library/eclipse-temurin:17-jdk-alpine@sha256:eb42bc053cbff0d750d76fa0705b6faec2677131a1358d0bafcc844051b8872c
=> [Internal] load build context
=> transferring context: 14.78kB
=> CACHED [2/3] WORKDIR /app
=> [1/3] COPY target/payment-service-1.0.0.jar app.jar
=> exporting to image
=> exporting layers
=> exporting manifest sha256:582c66da05a520ef8055df17a64a0f90dfddff65FSe242cb47950a6b160fda09
=> exporting config sha256:79616b74463e3ce7867a1d4f76534f438dfa887906bb0dc30cfaf9015b965d
=> exporting annotation manifest sha256:556c780dd2a407e5e806b95df7b5d70c1847258c325a576f32ad99ff13be
=> exporting manifest list sha256:4e109378581c7fbabf0310c38b41cfaf83a756dad394d78478e87a003876fe8
=> naming to docker.io/library/payment-service:local
=> unpacking to docker.io/library/payment-service:local
C:\Users\user\Desktop\New folder (3)>docker images
REPOSITORY          TAG           IMAGE ID            CREATED             SIZE
payment-service     local          4e109378581c      22 seconds ago   507MB
postgres            13             9a41ba632f72      2 weeks ago       618MB
C:\Users\user\Desktop\New folder (3)>

```

```

C:\Windows\System32\cmd.exe - docker-compose up
a 17.0.16 with PID 1 (/app/app.jar started by root in /app)
payment-service | 2025-11-02T18:23:38.658Z INFO 1 --- [ payment-service | 2025-11-02T18:23:35.247Z INFO 1 --- [
LT mode.
payment-service | 2025-11-02T18:23:35.460Z INFO 1 --- [
Found 3 JPA repository interfaces.
payment-service | 2025-11-02T18:23:38.466Z INFO 1 --- [
payment-service | 2025-11-02T18:23:38.582Z INFO 1 --- [
payment-service | 2025-11-02T18:23:38.592Z INFO 1 --- [
payment-service | 2025-11-02T18:23:38.732Z INFO 1 --- [
payment-service | 2025-11-02T18:23:38.742Z INFO 1 --- [
d in 7667 ms
payment-service | 2025-11-02T18:23:39.718Z INFO 1 --- [
payment-service | 2025-11-02T18:23:41.408Z INFO 1 --- [
.PgConnection@96abc76
payment-service | 2025-11-02T18:23:41.417Z INFO 1 --- [
payment-service | 2025-11-02T18:23:41.838Z INFO 1 --- [
fault]
payment-service | 2025-11-02T18:23:42.049Z INFO 1 --- [
payment-service | 2025-11-02T18:23:42.285Z INFO 1 --- [
payment-service | 2025-11-02T18:23:43.122Z INFO 1 --- [
mer
payment-service | 2025-11-02T18:23:46.458Z INFO 1 --- [
e.transaction.jta.platform" to enable JTA platform integration)
payment-service | 2025-11-02T18:23:47.763Z WARN 1 --- [
payment-service | 2025-11-02T18:23:47.764Z WARN 1 --- [
on "idempotency_keys" does not exist, skipping
payment-service | 2025-11-02T18:23:47.899Z INFO 1 --- [
e unit 'default'
payment-service | 2025-11-02T18:23:49.937Z WARN 1 --- [
fore, database queries may be performed during view rendering. Explicitly configure spring.jpa.open-in-view to disable this warning
payment-service | 2025-11-02T18:23:53.279Z INFO 1 --- [
;
payment-service | 2025-11-02T18:23:53.711Z INFO 1 --- [
main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port 8080 (http) with context path ''
payment-service | 2025-11-02T18:23:53.770Z INFO 1 --- [
main] c.t.payment.PaymentServiceApplication : Started PaymentServiceApplication in 27.59 seconds
(payment process running for 38.68)
payment-service | 2025-11-02T18:24:25.579Z INFO 1 --- [nio-8080-exec-1] o.a.c.c.C.[tomcat].[localhost].[/] : Initializing Spring DispatcherServlet 'dispatcherServlet'
payment-service | 2025-11-02T18:24:25.583Z INFO 1 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
payment-service | 2025-11-02T18:24:25.588Z INFO 1 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Completed initialization in 5 ms

```

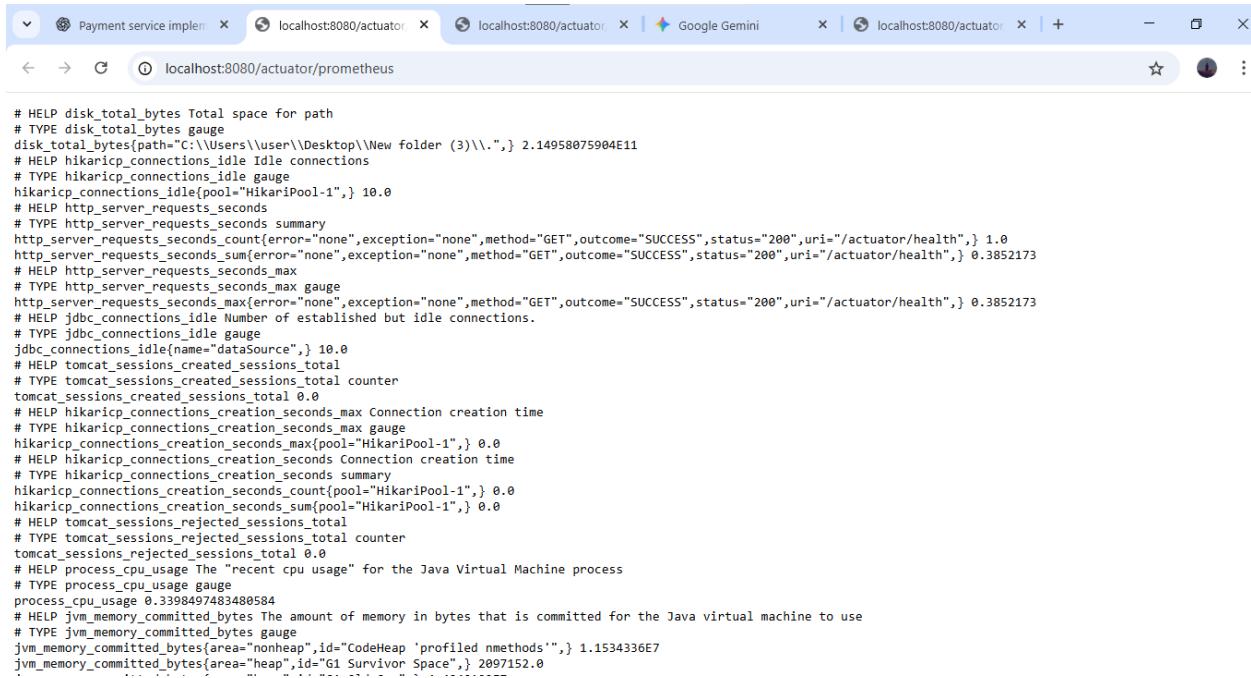
<http://localhost:8080/actuator/health>

localhost:8080/actuator/health

Pretty-print

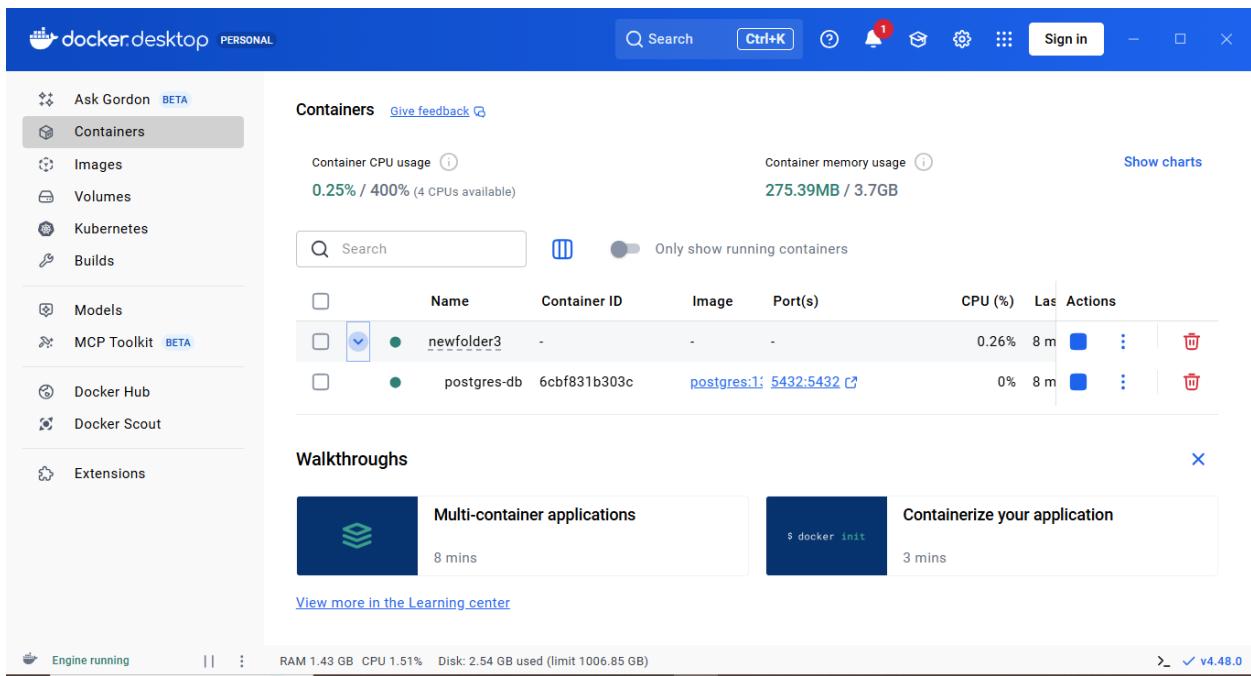
```
{"status": "UP", "components": {"db": {"status": "UP", "details": {"database": "PostgreSQL", "validationQuery": "isValid()", "diskSpace": {"status": "UP", "details": {"total": 1081101176832, "free": 1023381925888, "threshold": 10485760, "path": "/app/.", "exists": true}, "ping": {"status": "UP"}}}}}
```

<http://localhost:8080/actuator/prometheus> -



```
# HELP disk_total_bytes Total space for path
# TYPE disk_total_bytes gauge
disk_total_bytes{path="C:\\Users\\user\\Desktop\\New folder (3)\\.."} 2.14958075904E11
# HELP hikaricp_connections_idle Idle connections
# TYPE hikaricp_connections_idle gauge
hikaricp_connections_idle{pool="HikariPool-1"} 10.0
# HELP http_server_requests_seconds
# TYPE http_server_requests_seconds summary
http_server_requests_seconds_count{error="none",exception="none",method="GET",outcome="SUCCESS",status="200",uri="/actuator/health"} 1.0
http_server_requests_seconds_sum{error="none",exception="none",method="GET",outcome="SUCCESS",status="200",uri="/actuator/health"} 0.3852173
# HELP http_server_requests_seconds_max
# TYPE http_server_requests_seconds_max gauge
http_server_requests_seconds_max{error="none",exception="none",method="GET",outcome="SUCCESS",status="200",uri="/actuator/health"} 0.3852173
# HELP jdbc_connections_idle Number of established but idle connections.
# TYPE jdbc_connections_idle gauge
jdbc_connections_idle{name="dataSource"} 10.0
# HELP tomcat_sessions_created_sessions_total
# TYPE tomcat_sessions_created_sessions_total counter
tomcat_sessions_created_sessions_total 0.0
# HELP hikaricp_connections_creation_seconds_max Connection creation time
# TYPE hikaricp_connections_creation_seconds_max gauge
hikaricp_connections_creation_seconds_max{pool="HikariPool-1"} 0.0
# HELP hikaricp_connections_creation_seconds Connection creation time
# TYPE hikaricp_connections_creation_seconds summary
hikaricp_connections_creation_seconds_count{pool="HikariPool-1"} 0.0
hikaricp_connections_creation_seconds_sum{pool="HikariPool-1"} 0.0
# HELP tomcat_sessions_rejected_sessions_total
# TYPE tomcat_sessions_rejected_sessions_total counter
tomcat_sessions_rejected_sessions_total 0.0
# HELP process_cpu_usage The "recent cpu usage" for the Java Virtual Machine process
# TYPE process_cpu_usage gauge
process_cpu_usage 0.339849748348054
# HELP jvm_memory_committed_bytes The amount of memory in bytes that is committed for the Java virtual machine to use
# TYPE jvm_memory_committed_bytes gauge
jvm_memory_committed_bytes{area="nonheap",id="CodeHeap 'profiled nmethods'"} 1.1534336E7
jvm_memory_committed_bytes{area="heap",id="G1 Survivor Space"} 2097152.0
```

[Docker](#) -



Containers [Give feedback](#)

Name	Container ID	Image	Port(s)	CPU (%)	Last Action
newfolder3	-	-	-	0.26%	8 m
postgres-db	6cbf831b303c	postgres:12	5432:5432	0%	8 m

Walkthroughs

Multi-container applications 8 mins

\$ docker init 3 mins

The screenshot shows the Eclipse IDE interface with the following components:

- Project Explorer**: Shows the project structure with packages like payment-service, com.ticketing.payment, and com.ticketing.payment.model.
- Editor**: Displays the `PaymentService.java` file containing Java code for handling payment requests.
- Outline**: Shows the class hierarchy and methods for `PaymentService`.
- Console**: Displays the application logs from the command-line interface, indicating the application has started successfully.

Sample Request and Response -

POST – Charge Payment

Request:

```
{
  "orderId": "ORD1001",
  "amount": 500.0,
  "currency": "INR",
  "method": "CARD"
}
```

Headers:

Idempotency-Key: charge-101

Response (Success):

```
{
  "paymentId": 1,
  "status": "SUCCESS",
  "orderId": "ORD1001",
  "amount": 500.0,
  "reference": "ETP-a1b2c3d4"
}
```

Response (Duplicate Key):

```
{
  "error": "Idempotency conflict"
}
```

POST – Refund Payment

Request:

```
{  
  "paymentId": 1,  
  "amount": 500.0  
}
```

Response:

```
{  
  "refundId": 1,  
  "status": "SUCCESS",  
  "paymentId": 1,  
  "amount": 500.0  
}
```

GET – Retrieve Payment

Request:

GET /v1/payments/1

Response:

```
{  
  "paymentId": 1,  
  "orderId": "ORD1001",  
  "amount": 500.0,  
  "currency": "INR",  
  "status": "SUCCESS",  
  "method": "CARD",  
  "reference": "ETP-a1b2c3d4"  
}
```

Idempotency-

The screenshot shows the Postman application interface. On the left, there's a sidebar with 'Collections' (Amitosh's Workspace), 'Environments', 'Flows', and 'History'. The main area shows a 'My Collection' section with a 'GET Get data' item. Below it, a 'POST Post data' item is expanded, showing a sub-item 'POST http://localhost:8080/v1/payments/...'. A 'Test' item is also listed under this. The central part of the screen displays a POST request to 'http://localhost:8080/v1/payments/charge'. The 'Headers' tab is selected, showing two header entries: 'Idempotency-Key' with value '5678-1234' and 'Content-Type' with value 'application/json'. The 'Body' tab shows a JSON response with a timestamp, status, error message, and path. The status bar at the bottom indicates a 500 Internal Server Error with a response time of 34 ms.

```
1 {  
2   "timestamp": "2025-11-03T18:55:42.813+00:00",  
3   "status": 500,  
4   "error": "Internal Server Error",  
5   "path": "/v1/payments/charge"  
6 }
```

eclipse-workspace-newVersion - org.apache.coyote.AbstractProcessorLight - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Project Explorer X payment-service payment-service application.yml AbstractProcessorLight

```

59 } else if (status == SocketEvent.OPEN_WRITE) {
60     // Extra write event likely after async, ignore
61     state = SocketState.LONG;
62 } else if (status == SocketEvent.OPEN_READ) {
63     state = service(socketWrapper);
64 } else if (status == SocketEvent.CONNECT_FAIL) {
65     logAccess(socketWrapper);
66 } else {
67     // Default to closing the socket if the SocketEvent passed in
68     // is not consistent with the current state of the Processor
69     state = SocketState.CLOSED;
70 }
71
72 if (getLog().isDebugEnabled()) {
73     getLog().debug(
74         "Socket: [" + socketWrapper + "]. Status in: [" + status + "]. State out: [" + state + "]"
75     );
76 }
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174

```

Problems Servers Data Source Explorer Properties Console X Progress Install Java 25 Support Eclipse IDE for Enterprise Java... X

PaymentServiceApplication [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe [Nov 3, 2025, 11:51:46 PM elapsed: 0:16:26] [pid: 8180]

```

at org.apache.coyote.AbstractProtocol$ConnectionHandler.process(AbstractProtocol.java:896) ~[tomcat-embed-core-10.1.16.jar:10.1.16]
at org.apache.tomcat.util.net.NioEndpoint$SocketProcessor.doRun(NioEndpoint.java:1744) ~[tomcat-embed-core-10.1.16.jar:10.1.16]
at org.apache.tomcat.util.net.SocketProcessorBase.run(SocketProcessorBase.java:52) ~[tomcat-embed-core-10.1.16.jar:10.1.16]
at org.apache.tomcat.util.threads.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1191) ~[tomcat-embed-core-10.1.16.jar:10.1.16]
at org.apache.tomcat.util.threads.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:659) ~[tomcat-embed-core-10.1.16.jar:10.1.16]
at org.apache.tomcat.util.threads.TaskThreadWrappingRunnable.run(TaskThread.java:61) ~[tomcat-embed-core-10.1.16.jar:10.1.16]
at java.base/java.lang.Thread.run(Thread.java:842) ~[na:na]

2025-11-04T00:06:13.451+05:30 ERROR 8180 --- [nio-8080-exec-8] o.a.c.c.C.[.][/dispatcherServlet] : Servlet.service() for servlet [dispatcherServlet]
java.lang.RuntimeException: Idempotency conflict: same key, different payload
at com.ticketing.payment.service.PaymentService.charge(PaymentService.java:52) ~[classes:na]
at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke0(Native Method) ~[na:na]
at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:77) ~[na:na]
at java.base/jdk.internal.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43) ~[na:na]
at java.base/java.lang.reflect.Method.invoke(Method.java:568) ~[na:na]
at org.springframework.aop.support.AopUtils.invokeJoinpointUsingReflection(AopUtils.java:352) ~[spring-aop-6.1.1.jar:6.1.1]
at org.springframework.aop.framework.AopProxy.invoke(AopProxy.java:309) ~[spring-aop-6.1.1.jar:6.1.1]

```

Read-Only Smart Insert 67 : 78 : 3125

<http://localhost:8080/v1/payments/charge> -

Home Workspaces API Network

Amitosh's Workspace New Import

My Collection / Get data

POST http://localhost:8080/v1/payments/charge

Params Authorization Headers (10) Body Scripts Settings Cookies

Key	Value	Description	Bulk Edit	Presets
<input checked="" type="checkbox"/> Idempotency-Key	5678-1234			
<input checked="" type="checkbox"/> Content-Type	application/json			
Key	Value	Description		

Body 200 OK 233 ms 256 B Save Response

```
{
  "reference": "ETP-c32da8ca",
  "amount": 1300.0,
  "orderId": 1003,
  "paymentId": 3,
  "status": "SUCCESS"
}
```

Online Find and replace Console Runner Start Proxy Cookies Vault Trash ?

The screenshot shows the API Network tool interface. On the left, there's a sidebar with 'Collections' (selected), 'Environments', 'Flows', and 'History'. The main area shows a collection named 'My Collection' with a single item 'Get data' (POST). The request URL is `http://localhost:8080/v1/payments/charge`. The 'Body' tab is selected, showing the following JSON payload:

```
1 {
2   "orderId": 1003,
3   "amount": 1300,
4   "currency": "INR",
5   "method": "CARD"
6 }
```

The response section shows a **200 OK** status with a response time of 233 ms and a response size of 256 B. The response body is also JSON:

```
1 {
2   "reference": "ETP-c32da8ca",
3   "amount": 1300.0,
4   "orderId": 1003,
5   "paymentId": 3,
6   "status": "SUCCESS"
7 }
```

Retry and same result –

This screenshot is identical to the one above, showing a second successful POST request to `http://localhost:8080/v1/payments/charge`. The payload and response are the same as the first one, indicating a repeatable and successful operation.

Failed scenario when amount odd –

The screenshot shows the Postman interface. On the left, the sidebar has 'Collections' selected. In the main area, there's a collection named 'My Collection' with a 'Get data' endpoint. The 'Body' tab is selected, showing a JSON payload:

```
1 {  
2   "orderId": 1003,  
3   "amount": 1301,  
4   "currency": "INR",  
5   "method": "CARD"  
6 }
```

Below the body, the response status is 200 OK with a response time of 46 ms and a size of 228 B. The response body is:

```
1 {  
2   "amount": 1301.0,  
3   "orderId": 1003,  
4   "paymentId": 4,  
5   "status": "FAILED"  
6 }
```

The screenshot shows a PostgreSQL command prompt window titled 'Command Prompt - psql -U postgres -d paymentsdb'. The user runs several SQL queries to inspect the database:

```
paymentsdb=# SELECT * FROM idempotency_keys;  
id | idempotency_key | request_fingerprint | response_code  
---+-----+-----+-----+  
1 | 5678-1234 | c46708cdcd28d36f751645caed72a7e079aded1ba37e3d987dbd66ee89b47f94 | {"reference":"ETP-c32da8ca", "amount":1300.0,"orderId":1003,"paymentId":3,"status":"SUCCESS"} | 200  
(1 row)  
  
paymentsdb=# SELECT * FROM refunds;  
id | amount | created_at | payment_id | provider_ref | status  
---+-----+-----+-----+-----+  
1 | 1200 | | 1 | REF-7c4c7f4b | SUCCESS  
(1 row)  
  
paymentsdb=# SELECT * FROM payments;  
payment_id | amount | created_at | method | order_id | reference | status  
---+-----+-----+-----+-----+-----+  
1 | 1200 | | CARD | 1001 | ETP-e2ab4afb | REFUNDED  
2 | 1201 | | CARD | 1002 | | FAILED  
3 | 1300 | | CARD | 1003 | ETP-c32da8ca | SUCCESS  
4 | 1301 | | CARD | 1003 | | FAILED  
(4 rows)
```

Failed status should not refund –

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the project structure for "PaymentService".
- Code Editor:** Displays a portion of the `AbstractProcessorLight.java` file, specifically the logic for handling socket events.
- Outline View:** Shows the class hierarchy and methods for `AbstractProcessorLight`.
- Terminal:** Shows the command-line output of the application, indicating it was run with Java 17 and PID 8180. It includes stack traces for two errors: one from the payment service and one from the AOP framework.

```
else if (status == SocketEvent.OPEN_WRITE) {
    // Extra write event likely after async, ignore
    state = SocketState.LONG;
} else if (status == SocketEvent.OPEN_READ) {
    state = service(socketWrapper);
} else if (status == SocketEvent.CONNECT_FAIL) {
    logAccess(socketWrapper);
} else {
    // Default to closing the socket if the SocketEvent passed in
    // is not consistent with the current state of the Processor
    state = SocketState.CLOSED;
}

if (getLog().isDebugEnabled()) {
    getLog().debug(
        "Nrkrkt: [" + <socketWrapper + "]. Status in: [" + status + "]. State out: [" + state + "]"
    );
}
```

```
PaymentServiceApplication [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe (Nov 3, 2025, 11:51:46 PM elapsed: 0:43:39) [pid: 8180]
    at org.apache.tomcat.util.threads.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1191) ~[tomcat-embed-core-10.1.16.jar:10.1.16]
    at org.apache.tomcat.util.threads.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:859) ~[tomcat-embed-core-10.1.16.jar:10.1.16]
    at org.apache.tomcat.util.threads.TaskThreadWrappingRunnable.run(TaskThread.java:61) ~[tomcat-embed-core-10.1.16.jar:10.1.16]
    at java.base/java.lang.Thread.run(Thread.java:842) ~[na:na]

2025-11-04T00:33:38.173+05:30 ERROR 8180 --- [nio-8080-exec-2] o.a.c.c.C.[.[].[dispatcherServlet] : Servlet.service() for servlet [dispatcherServlet]
java.lang.RuntimeException: only successful payments can be refunded
    at com.ticketing.payment.service.PaymentService.refund(PaymentService.java:113) ~[classes/:na]
    at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke0(Native Method) ~[na:na]
    at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:77) ~[na:na]
    at java.base/jdk.internal.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43) ~[na:na]
    at java.base/java.lang.reflect.Method.invoke(Method.java:568) ~[na:na]
    at org.springframework.aop.support.AopUtils.invokeJoinpointUsingReflection(AopUtils.java:352) ~[spring-aop-6.1.1.jar:6.1.1]
    at org.springframework.aop.framework.ReflectiveMethodInvocation.invokeJoinpoint(ReflectiveMethodInvocation.java:196) ~[spring-aop-6.1.1.jar:6.1.1]
    at org.springframework.aop.framework.ReflectiveMethodInvocation.proceed(ReflectiveMethodInvocation.java:163) ~[spring-aop-6.1.1.jar:6.1.1]
    at org.springframework.aop.framework.CglibAopProxy$CglibMethodInvocation.proceed(CglibAopProxy.java:765) ~[spring-aop-6.1.1.jar:6.1.1]
```

Refund - <http://localhost:8080/v1/payments/refund> -

The screenshot shows the API Platform interface. On the left, there's a sidebar with 'Amitosh's Workspace' selected, showing 'Collections', 'Environments', 'Flows', and 'History'. The main area has a 'My Collection / Get data' tab open. A POST request is being made to `http://localhost:8080/v1/payments/refund`. The 'Body' tab is selected, showing the following JSON:

```

1 {
2   "paymentId": 3,
3   "amount": 1300
4 }

```

The response is a `200 OK` with a response time of 80 ms and a size of 227 B. The response body is:

```

1 {
2   "amount": 1300.0,
3   "paymentId": 3,
4   "refundId": 2,
5   "status": "SUCCESS"
6 }

```

Below this, there's a terminal window titled 'Command Prompt - psql -U postgres -d paymentsdb' showing database queries:

```

paymentsdb=# SELECT * FROM payments;
payment_id | amount | created_at | method | order_id | reference | status
-----+-----+-----+-----+-----+-----+-----+
1 | 1200 |           | CARD | 1001 | ETP-e2ab4afb | REFUNDED
2 | 1201 |           | CARD | 1002 | ETP-c32da8ca | FAILED
3 | 1300 |           | CARD | 1003 | ETP-c32da8ca | SUCCESS
4 | 1301 |           | CARD | 1003 | ETP-c32da8ca | FAILED
(4 rows)

paymentsdb=# SELECT * FROM refunds;
id | amount | created_at | payment_id | provider_ref | status
-----+-----+-----+-----+-----+-----+
1 | 1200 |           | 1 | REF-7c4c7f4b | SUCCESS
(1 row)

paymentsdb=# SELECT * FROM refunds;
id | amount | created_at | payment_id | provider_ref | status
-----+-----+-----+-----+-----+-----+
1 | 1200 |           | 1 | REF-7c4c7f4b | SUCCESS
2 | 1300 |           | 3 | REF-75b28e2b | SUCCESS
(2 rows)

paymentsdb=#

```

Payment ID – 1 ---

Testing and Validation -

Testing was performed using Postman and Swagger UI.

Verified endpoints:

- Charge request with and without Idempotency-Key.
- Refund of successful transactions.
- Retrieval of payment details.
- Health and Prometheus monitoring endpoints.

Confirmed:

- Database persistence in PostgreSQL.
- Accurate status transitions (PENDING → SUCCESS/FAILED → REFUNDED).
- Consistent JSON responses.
- Duplicate prevention via Idempotency-Key logic.

Database – PostgreSQL -

Tables

- **payments**: Stores charge details and transaction statuses.
- **refunds**: Tracks refund operations.
- **idempotency_keys**: Manages request fingerprints and cached responses.

SELECT * FROM payments;

```
Command Prompt - psql -U postgres -d paymentsdb
DROP TABLE idempotency_keys;
paymentsdb=# CREATE TABLE idempotency_keys (
paymentsdb(#   id SERIAL PRIMARY KEY,
paymentsdb(#   idempotency_key VARCHAR(255) UNIQUE,
paymentsdb(#   request_fingerprint TEXT,
paymentsdb(#   response_body JSONB,
paymentsdb(#   response_code INTEGER
paymentsdb(# );
CREATE TABLE
paymentsdb=# SELECT * FROM payments;
payment_id | amount | created_at | method | order_id | reference | status
-----+-----+-----+-----+-----+-----+-----+
  1 | 1200 | CARD | 1001 | ETP-e2ab4afb | REFUNDED
  2 | 1201 | CARD | 1002 | ETP-e2ab4afb | FAILED
(2 rows)

paymentsdb=# ALTER TABLE idempotency_keys
paymentsdb=# ALTER COLUMN response_body TYPE text USING response_body::text;
ALTER TABLE
paymentsdb=# SELECT * FROM payments;
payment_id | amount | created_at | method | order_id | reference | status
-----+-----+-----+-----+-----+-----+-----+
  1 | 1200 | CARD | 1001 | ETP-e2ab4afb | REFUNDED
  2 | 1201 | CARD | 1002 | ETP-e2ab4afb | FAILED
  3 | 1300 | CARD | 1003 | ETP-c32da8ca | SUCCESS
(3 rows)

paymentsdb=#

```

SELECT * FROM idempotency_keys;

```
Command Prompt - psql -U postgres -d paymentsdb
paymentsdb=# SELECT * FROM idempotency_keys;
 id | idempotency_key | request_fingerprint | response_code
----+-----+-----+-----+
  1 | 5678-1234 | c46708cdcd28d36f751645caed72a7e079aded1ba37e3d987dbd66ee89b47f94 | {"reference": "ETP-c32da8ca", "amount": 1300.0, "orderId": 1003, "paymentId": 3, "status": "SUCCESS"} | 200
  2 | 5678-5678 | c4ce304c199d2d56f0ce3d26c1d088d2498c0ceb3c3d782836a3e5d314dd8b5b | {"amount": 1301.0, "orderId": 1003, "paymentId": 4, "status": "FAILED"} | 402
(2 rows)

paymentsdb=#

```

SELECT * FROM refunds;

```

Command Prompt - psql -U postgres -d paymentsdb
1 | 1200 |           | 1 | REF-7c4c7f4b | SUCCESS
(1 row)

paymentsdb=# SELECT * FROM payments;
payment_id | amount | created_at | method | order_id | reference | status
-----+-----+-----+-----+-----+-----+-----+
 1 | 1200 |           | CARD | 1001 | ETP-e2ab4afb | REFUNDED
 2 | 1201 |           | CARD | 1002 | ETP-c32da8ca | FAILED
 3 | 1300 |           | CARD | 1003 | ETP-c32da8ca | SUCCESS
 4 | 1301 |           | CARD | 1003 | ETP-c32da8ca | FAILED
(4 rows)

paymentsdb=# SELECT * FROM refunds;
id | amount | created_at | payment_id | provider_ref | status
-----+-----+-----+-----+-----+-----+
 1 | 1200 |           | 1 | REF-7c4c7f4b | SUCCESS
 2 | 1300 |           | 3 | REF-75b28e2b | SUCCESS
(2 rows)

paymentsdb=#

```

MiniKube –

```

C:\Users\user\Desktop\New folder (3)>kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
payment-service-7fdcf58576-k2xps   1/1     Running   2 (34s ago)   6m32s
postgres-84f4c5b4fc-kgs4s        1/1     Running   0          6m30s

```

The screenshot shows the Swagger UI interface for a payment API. At the top, the URL is `localhost:8080/swagger-ui/index.html`. The main content area displays the **OpenAPI definition v0 OAS3**. Below this, there is a dropdown for **Servers** set to `http://localhost:8080 - Generated server url`. The **payment-controller** section lists three endpoints:

- POST /v1/payments/refund**
- POST /v1/payments/charge**
- GET /v1/payments/{id}**

Below the endpoints, there is a **Schemas** section containing a single entry: `RefundRequest >`.

localhost:8080/swagger-ui/index.html#/

payment-controller

POST /v1/payments/refund

POST /v1/payments/charge

GET /v1/payments/{id}

Parameters

Name	Description
id <small>* required</small>	integer(\$int32) 1 (path)

Responses

Curl

```
curl -X 'GET' \
  'http://localhost:8080/v1/payments/1' \
  '-H 'accept: */*'
```

Request URL

```
http://localhost:8080/v1/payments/1
```

Server response

Code Details

Cancel

Execute Clear

docker desktop PERSONAL

Ask Gordon BETA

Containers Give feedback

Container CPU usage i 995.28% / 400% (4 CPUs available) Container memory usage i 346.3MB / 3.7GB Show charts

Name	Container ID	Image	Port(s)	CPU (%)	Last started	Actions
minikube	ad1e1bdacbe3	k8s-minikube	49577:22	0%	1 hour ago	[Edit] [More] [Delete]
newfolder3	-	-	-	0%	19 seconds ago	[Edit] [More] [Delete]

Search Only show running containers

Models MCP Toolkit BETA

Docker Hub Docker Scout

Extensions

Walkthroughs

Multi-container applications 8 mins \$ docker init Containerize your application 3 mins

[View more in the Learning center](#)

Engine running RAM 2.75 GB CPU 94.59% Disk: 6.01 GB used (limit 1006.85 GB) v4.48.0

Swagger UI –

The screenshot shows the Swagger UI interface displaying two API definitions:

- payment-controller**:
 - POST /v1/payments/refund**:
 - Parameters: No parameters.
 - Request body (required): application/json. Example value: { "paymentId": 0, "amount": 0 }.
- payment-controller**:
 - POST /v1/payments/charge**:
 - Parameters:

Name	Description
Idempotency-Key	* required string (header)
X-Correlation-Id	string (header)
 - Request body (required): application/json. Example value: { "orderId": 0, "amount": 0, "currency": "string", "method": "string" }.

localhost:8080/swagger-ui/index.html#/payment-controller/getPayment

GET /v1/payments/{id}

Parameters

Name	Description
id <small>required</small>	integer(\$int32) (path)

Responses

Curl

```
curl -X 'GET' \
'http://localhost:8080/v1/payments/1' \
-H 'accept: */*'
```

Request URL

```
http://localhost:8080/v1/payments/1
```

Server response

Code Details

200 Response body

```
{
  "paymentId": 1,
  "orderId": 1001,
  "amount": 100.0,
  "method": "CARD",
  "status": "REFUNDED",
  "reference": "EFT-e2ab4af5b",
  "createdAt": null
}
```

Download

localhost:8080/swagger-ui/index.html#/payment-controller/getPayment

{ "paymentId": 1, "orderId": 1001, "amount": 100.0, "method": "CARD", "status": "REFUNDED", "reference": "EFT-e2ab4af5b", "createdAt": "2025-11-04T20:57:37.435Z" }

Schemas

RefundRequest ↴ { paymentId integer(\$int32) amount number(\$double) }

PaymentRequest ↴ { orderId integer(\$int32) amount number(\$double) currency string method string }

Payment ↴ { paymentId integer(\$int32) orderId integer(\$int32) amount number(\$double) method string status string reference string createdAt string(\$date-time) }

API docs -



```
{ "openapi": "3.0.1", "info": { "title": "OpenAPI definition", "version": "v0" }, "servers": [ { "url": "http://localhost:8080", "description": "Generated server url" } ], "paths": { "/v1/payments/refund": { "post": { "tags": [ "payment-controller" ], "operationId": "refund", "requestBody": { "content": { "application/json": { "schema": { "type": "object", "properties": { "id": { "type": "string" }, "amount": { "type": "number" }, "method": { "type": "string" }, "currency": { "type": "string" } } } } } }, "responses": { "200": { "description": "OK", "content": { "application/json": { "schema": { "type": "object", "properties": { "id": { "type": "string" }, "amount": { "type": "number" }, "method": { "type": "string" }, "status": { "type": "string" } } } } } } } } }, "/v1/payments/get": { "get": { "tags": [ "payment-controller" ], "operationId": "getPayment", "parameters": { "id": { "in": "path", "required": true, "schema": { "type": "string" } } } }, "responses": { "200": { "description": "OK", "content": { "application/json": { "schema": { "type": "object", "properties": { "id": { "type": "integer" }, "amount": { "type": "double" }, "method": { "type": "string" }, "status": { "type": "string" } } } } } } } } } }
```

Key Learnings –

- Gained hands-on experience with Spring Boot 3, PostgreSQL, and Docker.
- Implemented idempotency for safe and reliable payment operations.
- Understood microservice design with clear service boundaries.
- Achieved successful deployment on Docker and Kubernetes.
- Strengthened understanding of RESTful APIs, error handling, and monitoring.

Conclusion –

The Payment Service successfully fulfills its role in the Event Ticketing System by ensuring secure, consistent, and idempotent transaction management.

Its modular design, containerized deployment, and clear API structure make it a reliable and scalable component in the distributed architecture.