# API Testing Strategy

### 1. Overview

API testing ensures the reliability, security, and performance of the Payment Processing Module at Tanami Capital. This strategy focuses on testing RESTful API endpoints using Postman, covering endpoint validation, error handling, security measures, and automation integration.

### 2. Testing Approach

We will use Postman for manual API testing and Newman CLI for automated execution in CI/CD pipelines. The strategy covers:

* **Functional Testing**: Ensuring API requests return expected responses.
* **Security Testing**: Validating authentication, authorization, and SQL injection protection.
* **Error Handling**: Testing system behavior for invalid inputs, missing parameters, and network failures.
* **Performance Testing**: Evaluating API response times and concurrent load handling.

### 3. API Test Cases

3.1 Positive Test Cases

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Scenario** | **Method** | **Steps** | **Expected Result** | **Severity** |
| **API-001** | Valid Payment Request | POST /payment | 1. Send valid payment details 2. Submit request | API returns **200 OK** with transaction details | High |
| **API-002** | Fetch Valid Transaction | GET /transaction/{id} | 1. Use valid transaction ID 2. Submit request | API returns **200 OK** with correct transaction data | High |
| **API-003** | Currency Conversion Validation | GET /exchange-rate | 1. Request conversion rate 2. Validate rate format | API returns **200 OK** with accurate exchange rate | Medium |

3.2 Negative Test Cases

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Scenario** | **Method** | **Steps** | **Expected Result** | **Severity** |
| **API-004** | Invalid Payment Details | POST /payment | 1. Enter incorrect card details 2. Submit request | API returns **400 Bad Request** with error message | High |
| **API-005** | Missing Required Fields | POST /payment | 1. Send request without card\_number 2. Submit request | API returns **400 Bad Request** with validation error | High |
| **API-006** | Invalid Authentication Token | POST /payment | 1. Use expired/fake token 2. Submit request | API returns **401 Unauthorized** | High |
| **API-007** | SQL Injection Attempt | POST /payment | 1. Inject SQL code into card\_number field 2. Submit request | API **sanitizes input** and returns **400 Bad Request** | Critical |
| **API-008** | High Request Rate (Rate Limiting) | POST /payment | 1. Send 100+ requests in a minute | API returns **429 Too Many Requests** | Medium |
| **API-009** | Unauthorized Role Access | GET /admin/payments | 1. Log in as a regular user 2. Attempt to fetch admin-only payments | API returns **403 Forbidden** | High |

### 4. Security Testing

Security testing ensures only authorized users access API endpoints.

4.1 Authentication & Authorization Checks

* **Verify API rejects invalid/expired tokens**.
* **Ensure role-based access control** restricts user permissions.
* **Confirm session expiration** prevents unauthorized actions.

4.2 Data Tampering Protection

* Modify API response payload to check if system detects unauthorized changes.
* Test for API parameter manipulation to validate security controls.

4.3 SQL Injection & Input Validation

* Inject SQL commands into request parameters.
* Validate system prevents unauthorized database access.

### 5. API Automation with Postman & Newman

To ensure **continuous API validation**, tests will be automated using **Newman CLI** in CI/CD pipelines.

5.1 Running Automated API Tests

* Use Postman collections for structured API test cases.
* Execute tests using Newman in CI/CD:

newman run PaymentAPITests.postman\_collection.json --reporters cli,junit

5.2 Integration into CI/CD

* **Trigger API tests on every commit.**
* **Generate test reports** and log failures.
* **Halt deployment if critical API tests fail.**

### 6. Next Steps

* Validate all API test scenarios.
* Implement API automation using Postman & Newman.
* Ensure security measures like SQL injection prevention & authentication failures are covered.