# Test Plan: Payment Module

## Overview

The test plan outlines the strategy and scope for testing the Payment Module in the UNITY Payment system, which integrates with Biller Aggregator. The testing approach follows a hybrid Agile-Waterfall methodology to ensure thorough and iterative validation across all levels of the software development lifecycle.

## Scope

- Standing Order Registration and Execution

- Biller Aggregator Integration

- Account Validation and Fund Handling

- Fee and Charge Calculations

- Notification System (ENS)

- Reporting and Reconciliation

## Test Types & Strategy

1. Unit Test

- Objective: Validate individual components and functions.

- Responsibility: Developers

- Tools: JUnit/TestNG, Postman (unit level API validation), Mock services

- Coverage:

- Account format validation

- Charge fee calculation

- Date logic handling (Start/Due Date)

- Utility methods (e.g., currency conversions)

## Integration Test - By Sprint

- Objective: Validate integrated features developed per sprint.

- Responsibility: QA team

- Approach:

- Story-by-story validation

- Manual + automation regression

- Functional + boundary validation

## Integration Test A (Standalone)

- Objective: Validate the Payment Module in isolation using mock/stubbed external systems.

- Tools: SoapUI/Postman, Mock Server

- Coverage:

- Inquiry Billing with simulated Biller response

- Submit Standing Order to local DB

- Account status validation (inactive, frozen, etc.)

- Retry mechanism without full system

## Integration Test B (End-to-End)

- Objective: Validate Payment Module working with actual upstream/downstream systems.

- Environment: Staging / UAT

- Coverage:

- Full flow from SO registration to transaction creation

- API to Biller Aggregator communication

- Charge and principal debit from respective accounts

- Reconciliation data match

- MT940 / CAMT053 narrative validation

- OVS interaction (hold/booking/release)

## Non-Functional Tests

- Performance Testing

- Load test for Submit SO, Inquiry Billing APIs

- Concurrent SO submissions

- Bulk Export and Report generation

- Security Testing

- Role-based access control (Maker/Checker)

- Data masking/encryption (if applicable)

- Input sanitization (prevent injections)

- System Recovery

- BCP test: cut connectivity mid-process

- Auto retry validation

- Error logging and alerting

## Entry & Exit Criteria

Entry Criteria:

- Requirements are signed off

- Test environment ready

- Unit tests completed

Exit Criteria:

- All critical test cases passed

- No high-severity defects open

- Test report reviewed and signed off

## Tools & Environment

- Test Management: JIRA, XRay / Zephyr

- Automation: Selenium, JMeter, Postman

- Reporting: Excel, Allure Reports

- Test Data: SQL scripts + synthetic data loader

## Deliverables

- Test Plan Document

- Test Scenarios & Cases (Excel)

- Test Summary Report

- Defect Log

- Performance Report

## Risk & Mitigation

| Risk | Mitigation |

|------|------------|

| API changes from Biller Aggregator | Close coordination with vendor, contract-based schemas |

| Late environment readiness | Use of mocks for early testing |

| Volume spikes | Load test with production-like data |

| Data loss during retries | Implement idempotency and transaction logs |

## Timeline & Schedule

- Unit Testing: Sprint-based

- Integration A: Sprint-based after unit

- Integration B: Post-feature freeze (2 weeks)

- Non-Functional: Post-stable functional sign-off