LMS – TEST STRATEGY & RELEASE PLAN

Product Line:	[redacted]
Release:	[redacted]
Product Manager:	[redacted]
Author:	Sanyogita Herwathe

Table of Contents

Revision History

Revision

Date (yyyy-mm-dd)	Author	Version	Description
2025-06-21	Sanyogita H	1.0	First Draft
2025-06-23	Sanyogita H	1.1	Revised Draft Post Review

Review

Date (yyyy-mm-dd)	Author	Version	Reviewed By
2025-06-23	Sanyogita H	1.1	Edmond Right

Document Scope

The scope of this document is to provide testing framework to fully test Loan Management System software. The test cases will be created and used to test an application as a part of a smoke, system and regression test.

LMS – TEST STRATEGY & RELEASE PLAN

1. Introduction

This Test Strategy defines the overall testing approach for the Loan Management System (LMS).

The objective is to validate functional accuracy, data integrity, security controls, and performance reliability across all modules while aligning with Agile sprint-based delivery.

Testing is structured to ensure production readiness through layered validation:

- Manual Functional Testing
- API Testing
- UI Automation
- Regression Testing
- Risk-Based Validation

2. Scope of Testing

The following functional areas are in scope:

Core Modules

- Authentication (Login / Logout)
- User Management
- Customer Management
- Loan Processing & Calculations
- Prequalification Letter (PDF generation)
- Dashboard
- Navigation & Link Validation
- Event Management
- System Policies & Configuration

Non-Functional Areas

- Security Testing (RBAC, session handling)
- Performance & Load validation
- Compatibility Testing (browser & OS coverage)
- Error Handling & Warning validations
- Usability validation

3. Testing Approach

3.1. Risk-Based Testing

Testing prioritization is driven by:

- Financial calculation accuracy
- Role-based access controls
- Data integrity across modules
- Authentication security
- PDF generation correctness

High-risk modules (Loans, Interest Rate Calculations, Central Bank Rate retrieval) receive deeper regression coverage and automation support.

3.2. Manual Testing

Manual testing includes:

- Functional validation
- Negative testing
- Boundary value analysis
- Equivalence partitioning
- UI validation
- Data validation
- End-to-end scenario testing

Manual testing is executed per sprint and validated against acceptance criteria.

3.3. API Testing

API testing includes:

- Endpoint validation
- Status code verification
- Schema validation
- Negative request testing
- Authentication token validation
- Data integrity across requests

Tools:

- Postman
- Newman (CLI execution)

3.4. UI Automation Strategy

Automation is implemented for:

- Authentication workflows
- High-risk regression flows
- Role-based navigation
- Core Loan lifecycle scenarios

Framework:

- Java
- Selenium WebDriver
- TestNG
- Maven
- Page Object Model (POM)
- Automation supports regression validation and future CI/CD integration.

4. Test Levels

- Smoke Testing – Executed on every new build
- System Testing – Full workflow validation
- Integration Testing – Cross-module validation
- Regression Testing – After bug fixes and feature updates
- Security Testing – Role & access validation
- Performance Testing – Load validation (JMeter optional)

5. Entry Criteria

Testing begins when:

- Functional requirements are finalized
- User stories are groomed and approved
- Test cases are prepared
- Test environment is stable
- Build is deployed successfully
- Unit testing is completed

6. Exit Criteria

Testing concludes when:

- All planned test cases are executed
- Critical, High, and Medium defects are resolved and retested
- Regression testing is completed
- Test pass rate ≥ 95%

- No open defects remain in non-terminal states
- Sprint acceptance criteria are satisfied

7. Defect Management

Defects are tracked using GitHub Issues with the following lifecycle:
Open → Assigned → In Progress → Fixed → Retest → Closed
Severity Classification:

- Critical – System failure / Data corruption
- High – Major functionality broken
- Medium – Functional issue with workaround
- Low – Cosmetic / UI issue

Daily triage is conducted during active sprint testing.

8. Test Environment

Environment includes:

- Windows 10 / 11
- Linux (Ubuntu)
- MacOS
- Chrome, Edge, Firefox

Database: MySQL
Backend: Laravel (PHP)

9. Estimation Model

Based on historical metrics:

- Test case design: 0.10 hr per test case
- Execution per build: 0.10 hr per test case
- Defect handling buffer applied
- Regression multiplier included

Estimation is adjusted per sprint capacity.

10. Agile Alignment

Testing follows sprint-based delivery:

- Sprint Planning → Story review
- Test Design → Linked to user stories
- Execution → Within sprint window
- Daily Standups → Defect tracking
- Sprint Review → Quality summary
- Retrospective → Improvement actions

Traceability is maintained:
Requirement → Test Case → Defect → Automation Coverage

11. Continuous Improvement

Future Enhancements:

- CI/CD automation pipeline
- Automated regression suite
- Code coverage reporting
- Performance baseline tracking
- Security automation integration