Library Management System - Software Testing Plan and Timeline

SE507 - Group Project - Spring 2025

Student Group (Replace with Your Team Name/Members)

May 24, 2025 (Adjust as needed)

Note: This document provides an illustrative plan. Actual dates and durations should be adjusted by the team. The project timeline assumes a start in early March 2025 and submission by June 15, 2025.

Work Breakdown Structure - Phases

- **Phase 1: Planning & Initial Static Testing**
  1.1. Understand SRS & Coursework Requirements
  1.2. Develop Detailed Test Plan (incl. techniques, criteria) & Task Distribution
  1.3. VIII. Static Testing: Documentation Review (SRS, Test Plan)
  1.4. VIII. Static Testing: Initial Code Review Setup & Process Definition
  1.5. **Milestone:** P1 Complete: Test Plan Finalized
- **Phase 2: Core Functional & API Test Design & Execution**
  2.1. I.A Functional: Authentication Test Design (Non-Coders)
  2.2. I.B Functional: Entity Mgt. (incl. Data Integrity & Constraint) Test Design (Non-Coders)
  2.3. I.C-D Functional: Feature & Use Case (incl. Error Handling) Test Design (Non-Coders)
  2.4. I.E Functional: State Transition Test Design (Non-Coders)
  2.5. II. API Test Scenario Design (Non-Coders)

2.6. II. API Test Automation Setup & Initial Scripts (Coders)
2.7. I. Functional Test Execution & Defect Reporting (Non-Coders)
2.8. II. API Test Execution (Manual & Automated) (Team)
2.9. **Milestone:** P2 Complete: Core Functional/API Tests Executed

- **Phase 3: White-Box & Non-Functional Testing**

3.1. III. White-Box Test Planning & Unit Test Development (Coders)
3.2. III. White-Box Test Execution & Code Coverage Analysis (e.g., Decision, Path) (Coders)
3.3. IV.A Security Test Design & Execution (Team)
3.4. IV.B Usability Test Design & Execution (Non-Coders)
3.5. IV.C Performance (incl. Basic Network Conditions) Test Design & Execution (Team)
3.6. IV.E Browser Compatibility Testing (Non-Coders)
3.7. VIII. Static Testing: Ongoing Code Reviews (Team)
3.8. **Milestone:** P3 Complete: White-Box/NFR Tests Executed

- **Phase 4: Integration & Early Regression Testing**

4.1. V. Integration Test Scenario Design (Non-Coders)
4.2. V. Integration Test Execution (Team)
4.3. VI. Regression Test Suite Development (Manual - Non-Coders)
4.4. VI. Regression Test Suite Development (Automated - Coders)
4.5. VI. Early Regression Test Cycles (Team)
4.6. **Milestone:** P4 Complete: Integration & Initial Regression Done

- **Phase 5: System Changes, Final Testing & Report**

5.1. Suggest, Implement & Test System Changes (Team)
5.2. VI. Final Regression Testing Cycles (Team)
5.3. VII. Installation/Deployment Testing (Team)
5.4. VIII. Static Testing: Final Documentation Review (Team)
5.5. Prepare Final Report & Presentation (Team)
5.6. **Milestone:** Project Submission

Detailed Task Assignments Overview

The following table provides a detailed breakdown of tasks. Refer to the Gantt chart for timeline allocation. The ”Primary Responsibility (Group)” indicates the lead, but collaboration is expected.

Table 1: Detailed Task Distribution

Main Testing Category	Sub-Category / Specific Focus	Primary Resp. (Group)	Tasks for Coders	Tasks for Non-Coders/Spec. Experts	Notes/Collaboration
I. Functional Testing (Black-Box - SRS based)	Overall Lead & Manual Execution	Non-Coders/ Specification Experts	”Review & Enhance Test Cases; Troubleshoot & Verify Defects; Identify Complex Scenarios; Support Test Data Generation”	”Lead Test Case Design (Sections I.A - I.D); Lead State Transition Testing Design (Section I.E); Lead Manual Test Execution; Lead Defect Reporting; Lead Test Data Preparation”	”Coders actively contribute to complex scenario identification and troubleshooting. Techniques: Equivalence Partitioning, Boundary Value Analysis, Error Guessing, State Transition Testing.”
I. Functional Testing (Black-Box - SRS based)	A. Authentication & Authorization	Non-Coders/ Specification Experts	”Review test cases for technical edge cases (e.g. session handling details).”	”Design and execute tests for User Registration (Librarian creating users - uniqueness, required fields, invalid formats); User Login (valid/invalid credentials, redirection); User Logout; Role-Based Access Control (Librarian vs. Member access to features).”	
I. Functional Testing (Black-Box - SRS based)	B. Entity Management (CRUD Operations - Books, Authors, Genres, Users, Borrowals, Reviews)	Non-Coders/ Specification Experts	”Review CRUD test cases for data integrity implications from backend perspective.”	”Design and execute tests for Create (valid, invalid, constraints, defaults); Read (list all, specific item, empty/non-existent); Update (valid, invalid, non-existent, constraints); Delete (valid, non-existent, referential integrity considerations).”	
I. Functional Testing (Black-Box - SRS based)	C. Specific Feature Testing (Dashboard, Book Mgt, Borrowal Mgt, User Mgt, Review Mgt)	Non-Coders/ Specification Experts	”Provide insights if specific feature behavior is tied to complex backend logic.”	”Design and execute tests for Librarian Dashboard data display; Book Management (linking authors/genres, availability); Borrowal Management (status updates, due dates, history filtering); User Management (role updates); Review Management (ratings, CRUD).”	
I. Functional Testing (Black-Box - SRS based)	D. Use Case Testing (SRS Section 7 and implicit use cases)	Non-Coders/ Specification Experts	”Assist in understanding backend responses for alternative/exception flows.”	”Design and execute tests for Main Flows and all specified Alternative/Exception Flows for each use case, ensuring graceful error handling and informative messages.”	
I. Functional Testing (Black-Box - SRS based)	E. State Transition Testing	Non-Coders/ Specification Experts	”Review state models for technical correctness if backend implements explicit state machines.”	”Identify entities/features with states (Borrowal Status, Book Availability, User Session); Design and execute tests for every valid/invalid transition and sequences of transitions.”	
II. API Testing	Overall	Balanced (Coders lead automation, Non-Coders lead user-centric scenario design)	”Lead API Test Automation (scripts for core functionalities, regression, complex validation); Technical API Analysis (endpoints, specs, security); Execute & Debug Automated API Tests; Advanced Postman Usage.”	”Lead API Test Scenario Design (User-Centric - based on SRS & user workflows); Manual API Testing (Postman for designed scenarios, request/response validation); Define API Test Data; Review API Test Results.”	”Collaboration on test data definition. Non-Coders can execute pre-written Postman collections.”
III. White-Box Testing (Logic Coverage - Based on Source Code)	Overall	Coders	”Source Code Analysis & Coverage Planning (Decision/Condition, Path, Data Flow); Unit Test Development & Execution (Jest for frontend/backend); Unit Test Maintenance.”	”Participate in discussions to understand unit test goals; Review unit test summaries/reports to understand component health related to features they are functionally testing.”	”Focus on critical modules, controllers, models, complex logic.”
IV. Non-Functional Testing	A. Security Testing	Balanced	”Lead technical security testing: code reviews for vulnerabilities, server-side validation testing via API, penetration testing basics, verifying security configurations.”	”Lead user-facing & black-box security testing: attempt XSS in UI, test access controls from user roles, check for sensitive data exposure in UI, verify password policy enforcement from UI.”	
IV. Non-Functional Testing	B. Usability Testing	Non-Coders/ Specification Experts	”Participate as ’users’ in informal usability tests if needed; provide technical feedback on feasibility of usability suggestions.”	”Design usability scenarios; conduct heuristic evaluations or informal tests; evaluate UI/UX (intuitiveness, consistency, messages, navigation, responsiveness, learnability, efficiency); document issues and suggestions.”	
IV. Non-Functional Testing	C. Performance Testing (Basic)	Balanced	”Conduct basic API response time checks for critical endpoints (Postman/scripts); Investigate significant slowdowns reported by Non-Coders.”	”Systematically observe, document, and report on perceived page load times and UI responsiveness for a predefined set of common user actions under varied conditions (e.g., few vs. many records).”	
IV. Non-Functional Testing	D. Maintainability	Shared	”Primarily assessed via Static Testing (Code Reviews - see Section VIII).”	”Primarily assessed via Static Testing (Documentation Reviews - see Section VIII).”	
IV. Non-Functional Testing	E. Browser Compatibility Testing	Non-Coders/ Specification Experts	”Assist in debugging browser-specific CSS/JS issues identified.”	”Lead manual testing of core functionalities and UI rendering consistency across specified browsers (Chrome, Firefox, Edge, Safari); Document inconsistencies.”	
V. Integration Testing	Overall	Balanced (Coders lead technical setup; Non-Coders lead end-to-end scenario design & execution)	”Lead Technical Setup & Debugging for integration tests; Test Backend Module & Database Integration.”	”Lead Design of End-to-End User Scenarios spanning multiple features; Execute User-Centric Integration Tests verifying data consistency and workflow integrity.”	
VI. Regression Testing	Overall	Shared and Balanced	”Lead Development & Maintenance of Automated Regression Suite (unit, API); Analyze Automated Test Failures.”	”Lead Maintenance & Execution of Manual Regression Suite (critical user workflows, previously found defects); Prioritize Manual Regression Tests.”	”Whole team collaborates on regression strategy, scope, and review of results. New tests for changes are added to both manual/automated suites.”
VII. Installation/Deployment Testing	Overall	Balanced	”Lead verification of Docker setup, build processes, deployment scripts; Troubleshoot technical deployment issues.”	”Lead execution of comprehensive smoke tests and basic acceptance tests from a user perspective after each deployment.”	

Continued on next page

Table 1 – continued from previous page

Main Testing Category	Sub-Category / Specific Focus	Primary Resp. (Group)	Tasks for Coders	Tasks for Non-Coders/Spec. Experts	Notes/Collaboration
VIII. Static Testing (Non-Execution Based)	Code Reviews/Inspections	Coders	"Lead technical reviews of code (logic, security, efficiency, standards); Prepare code for review."	"Actively participate in code reviews focusing on readability, alignment with SRS, understandability of comments, user story alignment. Can use checklists."	
VIII. Static Testing (Non-Execution Based)	Walkthroughs	Coders	"Lead walkthroughs of their code."	"Actively question and seek clarification during walkthroughs to ensure alignment with requirements."	
VIII. Static Testing (Non-Execution Based)	Documentation Review (SRS, Test Plan, Test Cases, Final Report)	Non-Coders/ Specification Experts	"Review documentation for technical accuracy; Contribute to sections detailing technical testing efforts."	"Lead the review of all project documentation for clarity, consistency, completeness, accuracy; Drive compilation of test-related sections of the final report."	