## **Recommended Task Distribution for Library Management Project Testing (Evenly Distributed)**

This document outlines a recommended approach to distribute testing tasks for the Library Management System project among a group of four students. The team consists of two students strong in coding (Coders) and two less proficient in coding who excel in specification analysis and user-centric evaluation (Non-Coders/Specification Experts). This plan aims for an **even distribution of workload and responsibility** across these two sub-groups, leveraging their respective strengths while fostering deep collaboration. This plan is based on the "Summary of Tests Needed for Library Management Project" document.

**Team Composition:**

* **2 Students Strong in Coding (Coders)**
* **2 Students Less Proficient in Coding (Non-Coders/Specification Experts)**

**General Principles for Task Distribution:**

* **Balanced Workload:** While tasks will align with skill sets, the overall effort, ownership, and volume of work should be comparable between the Coder and Non-Coder groups.
* **Leverage Strengths:**
  + **Coders:** Primarily handle tasks requiring deep code understanding, script development, automation, technical analysis (white-box, API automation, complex integration, technical security/performance).
  + **Non-Coders/Specification Experts:** Primarily handle tasks driven by requirements, user perspective, UI interaction, documentation, and black-box techniques (functional design & execution, usability, manual regression, documentation reviews, state transition design).
* **Deep Collaboration:** Continuous communication, knowledge sharing, and peer review of all testing artifacts (test cases, scripts, reports) are essential.
* **Test Manager Role:** One student (Coder or Non-Coder with strong organizational and communication skills) should act as Test Manager. This role is crucial for overseeing the even distribution of tasks, tracking progress, managing documentation, facilitating communication, and re-balancing workloads if necessary.

### **Recommended Task Breakdown by Testing Category:**

**(Referencing sections from "Summary of Tests Needed for Library Management Project")**

**I. Functional Testing (Black-Box - SRS based)**

* **Balanced Responsibility:** While Non-Coders lead design and manual execution, Coders actively contribute to complex scenario identification and troubleshooting.
* **Non-Coders/Specification Experts - Tasks:**
  + **Lead Test Case Design (Sections I.A - I.D):** Thoroughly analyze SRS; design detailed test cases using Equivalence Partitioning, Boundary Value Analysis, Error Guessing. Document with clear steps, expected results, test data.
  + **Lead State Transition Testing Design (Section I.E):** Identify entities with states; design and document test cases for valid/invalid transitions and sequences.
  + **Lead Manual Test Execution:** Execute all designed functional and state transition test cases via the UI.
  + **Lead Defect Reporting:** Meticulously document defects with reproduction steps, actual vs. expected results, severity.
  + **Lead Test Data Preparation:** Design, create, and manage comprehensive test data sets.
* **Coders - Tasks:**
  + **Review & Enhance Test Cases:** Review functional test cases for technical feasibility, edge cases missed, and alignment with backend logic. Suggest additions or modifications.
  + **Troubleshoot & Verify Defects:** Assist in diagnosing the root cause of defects found during functional testing, particularly those that might indicate backend issues. Verify fixes.
  + **Identify Complex Scenarios:** Based on code understanding, identify complex interaction scenarios or potential failure points that Non-Coders might not infer from specifications alone, and help design tests for them.
  + **Support Test Data Generation:** Assist with generating or scripting bulk test data if needed for specific functional tests.

**II. API Testing**

* **Balanced Responsibility:** Coders lead automation and technical execution; Non-Coders lead scenario design from a user/requirements perspective and participate in execution.
* **Coders - Tasks:**
  + **Lead API Test Automation:** Develop and maintain automated API test scripts (e.g., Jest/Supertest, Python) for core API functionalities, focusing on regression and complex validation.
  + **Technical API Analysis:** Review backend code to identify all API endpoints, their detailed specifications, and potential security vulnerabilities at the API level.
  + **Execute & Debug Automated API Tests:** Run automated suites, analyze failures, and debug scripts or underlying API issues.
  + **Advanced Postman Usage:** Handle complex Postman scenarios, scripting, and environment setups.
* **Non-Coders/Specification Experts - Tasks:**
  + **Lead API Test Scenario Design (User-Centric):** Based on SRS and user workflows, design test scenarios for API interactions (e.g., what API calls support "borrowing a book" use case). Document these scenarios.
  + **Manual API Testing (Postman):** Utilize tools like Postman to execute manual API tests for designed scenarios, focusing on request/response validation for specific user workflows.
  + **Define API Test Data:** Provide realistic and comprehensive test data for API payloads based on SRS analysis and functional test scenarios.
  + **Review API Test Results:** Review API test reports (both manual and automated) to ensure they align with expected functional outcomes.

**III. White-Box Testing (Logic Coverage - Based on Source Code)**

* **Primary Responsibility:** Coders (due to the nature of the task)
* **Coders - Tasks:**
  + **Source Code Analysis & Coverage Planning:** Analyze internal structure of critical modules; decide on target coverage levels (Decision/Condition, etc.).
  + **Unit Test Development & Execution:** Write, execute, and maintain unit tests (e.g., Jest) for frontend and backend components, covering paths, conditions, loops, error handling. Include Data Flow testing considerations.
* **Non-Coders/Specification Experts - Contribution (to ensure engagement):**
  + **Understand Unit Test Goals:** Participate in discussions to understand what functionalities are being covered by unit tests.
  + **Review Unit Test Summaries:** Review reports or summaries from unit tests to understand the health of components related to features they are functionally testing. This helps bridge the gap.

**IV. Non-Functional Testing**

* **Balanced Responsibility across sub-categories.**
* **A. Security Testing:**
  + **Coders:** Lead technical security testing: code reviews for vulnerabilities, server-side validation testing via API, penetration testing basics if skills allow, verifying security configurations.
  + **Non-Coders/Specification Experts:** 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.
* **B. Usability Testing:**
  + **Primary Responsibility:** Non-Coders/Specification Experts
  + **Tasks:** Design usability scenarios, conduct heuristic evaluations or informal tests, evaluate UI/UX, document issues and suggestions.
  + **Coders - Support:** Participate as "users" in informal usability tests if needed; provide technical feedback on feasibility of usability suggestions.
* **C. Performance Testing (Basic):**
  + **Non-Coders/Specification Experts:** 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).
  + **Coders:** Conduct basic API response time checks for critical endpoints (Postman/scripts). Investigate significant slowdowns reported by Non-Coders.
* **D. Maintainability:** Primarily assessed via Static Testing (see Section VIII), with balanced contributions.
* **E. Browser Compatibility Testing:**
  + **Primary Responsibility:** Non-Coders/Specification Experts
  + **Tasks:** Lead manual testing of core functionalities and UI rendering across specified browsers. Document inconsistencies.
  + **Coders - Support:** Assist in debugging browser-specific CSS/JS issues identified by Non-Coders.

**V. Integration Testing**

* **Balanced Responsibility:** Coders lead technical setup and backend integration; Non-Coders lead design of end-to-end user scenarios and their execution.
* **Coders - Tasks:**
  + **Lead Technical Setup & Debugging:** Set up environments for integration tests. Debug issues related to client-server communication, backend module interactions, and database integration.
  + **Test Backend Module & Database Integration:** Focus on ensuring backend components and database interactions work correctly together.
* **Non-Coders/Specification Experts - Tasks:**
  + **Lead Design of End-to-End Scenarios:** Define comprehensive integration test scenarios from a user journey perspective, spanning multiple features or modules (e.g., "Full user lifecycle from registration to borrowing multiple books and returning them").
  + **Execute User-Centric Integration Tests:** Perform end-to-end tests based on the designed scenarios, verifying data consistency and workflow integrity across the application.

**VI. Regression Testing**

* **Shared and Balanced Responsibility:** This is critical and ongoing.
  + **Non-Coders/Specification Experts:**
    - **Lead Maintenance & Execution of Manual Regression Suite:** Maintain and execute a comprehensive suite of manual regression tests covering critical user workflows and previously found defects.
    - **Prioritize Manual Regression Tests:** Work with the team to identify which manual tests are most critical for each release/change.
  + **Coders:**
    - **Lead Development & Maintenance of Automated Regression Suite:** Develop, maintain, and execute automated regression tests (unit, API, potentially basic UI if feasible).
    - **Analyze Automated Test Failures:** Investigate and address failures in automated regression suites.
  + **Whole Team:**
    - Collaboratively define the scope of the overall regression strategy.
    - Ensure test cases for new/changed features are added to both manual and automated regression suites appropriately.
    - Review regression results together.

**VII. Installation/Deployment Testing (If applicable)**

* **Balanced Responsibility:**
  + **Coders:** Lead verification of Docker setup, build processes, and any deployment scripts. Troubleshoot technical deployment issues.
  + **Non-Coders/Specification Experts:** Lead the execution of comprehensive smoke tests and basic acceptance tests from a user perspective immediately after each deployment to ensure application stability and core functionality.

**VIII. Static Testing (Non-Execution Based)**

* **Shared and Balanced Responsibility:**
  + **Code Reviews/Inspections:**
    - **Coders:** Lead technical reviews of code (logic, security, efficiency, standards). Prepare code for review.
    - **Non-Coders/Specification Experts:** Actively participate in code reviews by focusing on readability, alignment with SRS, understandability of comments, and whether the code seems to address user stories. Can use checklists focused on these aspects.
  + **Walkthroughs:** Coders lead walkthroughs of their code; Non-Coders actively question and seek clarification to ensure alignment with requirements.
  + **Documentation Review (SRS, Test Plan, Test Cases, Final Report):**
    - **Non-Coders/Specification Experts:** Lead the review of all project documentation for clarity, consistency, completeness, and accuracy. Drive the compilation of test-related sections of the final report.
    - **Coders:** Review documentation for technical accuracy and contribute to sections detailing technical testing efforts (e.g., unit testing approach, API automation).

### **Practical Steps for Your Team to Ensure Even Distribution:**

1. **Assign Test Manager:** Crucial for overseeing this balanced distribution and making adjustments.
2. **Team Kick-off & Role Clarification:** Review this refined plan. Discuss and confirm that the distribution of effort feels equitable.
3. **Develop a Detailed Test Plan Document:** Led by the Test Manager, this document should explicitly assign these balanced responsibilities and establish clear deliverables for each sub-group within each testing type. Use a task board (even a simple one) to visualize workload.
4. **Cross-Training & Tool Familiarization:**
   * Coders to mentor Non-Coders on relevant aspects of Postman for more independent API scenario execution.
   * Non-Coders to explain functional complexities and user expectations clearly to Coders to aid in their technical test design.
5. **Regular Workload Review in Sync-ups:** During regular meetings, specifically discuss workload. Is one group feeling overloaded or underutilized? The Test Manager should facilitate re-balancing if needed.
6. **Joint Ownership of Documentation:** While Non-Coders might lead compilation, Coders are equally responsible for contributing content related to their technical testing efforts, ensuring the final report reflects a balanced team contribution.
7. **Structured Pairing:** Implement structured pairing sessions where a Coder and Non-Coder work together on specific tasks (e.g., a Non-Coder designs API scenarios, then pairs with a Coder to implement/execute them in Postman; a Coder explains a complex module, then pairs with a Non-Coder to review its alignment with requirements).

By actively managing and balancing the workload as described, your team can ensure that all members contribute significantly and effectively, leading to a more robust testing process and a stronger final project.