Assignment 2

Project: Library Management System - Complete Implementation & CI/CD

Project Overview

Building upon Assignment 1, you will now complete the Library Management System implementation, create comprehensive test suites, leverage AI tools for test generation, and set up professional CI/CD workflows using GitHub Actions.

Learning Objectives

By the end of this assignment, you will be able to:

- Implement complete business logic functions following specifications
- Create comprehensive test suites for both existing and new functionality
- Use Large Language Models (LLMs) to generate effective test cases
- Set up Continuous Integration/Continuous Deployment (CI/CD) pipelines
- Implement professional project documentation with status badges

Tasks to Complete

You are required to complete the following 4 main tasks:

1. **Complete Function Implementation** - Implement all remaining TODO functions
2. **Comprehensive Test Suite Development** - Create tests for all functionality
3. **AI-Assisted Test Generation** - Use LLMs to generate additional test cases
4. **CI/CD Pipeline Setup** - Deploy to GitHub with automated testing and status badges

Task 1: Complete Function Implementation (40%)

1.1 Implement `return_book_by_patron(patron_id: str, book_id: int)`

- **Requirements**: Implement R4 (Book Return Processing)
- **Business Logic**: Verify book is borrowed by patron, update return_date, increment available_copies
- **Return**: `Tuple[bool, str]` - (success, message)

1.2 Implement `calculate_late_fee_for_book(book_id: int, patron_id: str)`

- **Requirements**: Implement R5 (Late Fee Calculation API)
- **Business Logic**: Calculate days overdue, apply $1.00 per day late fee rate
- **Return**: `Tuple[bool, str, float]` - (success, message, fee_amount)

1.3 Implement `search_books_in_catalog(query: str, search_type: str)`

- **Requirements**: Implement R6 (Book Search Functionality)
- **Business Logic**: Case-insensitive partial matching for title/author, exact matching for ISBN
- **Return**: `Tuple[bool, str, List[Dict]]` - (success, message, book_list)

1.4 Implement `get_patron_status_report(patron_id: str)`

- **Requirements**: Implement R7 (Patron Status Report)
- **Business Logic**: List currently borrowed books, calculate total late fees, show due dates
- **Return**: `Tuple[bool, str, Dict]` - (success, message, status_report)

1.5 Testing Your Implementation

```
# Test your implementations
python -m pytest tests/ -v -k "not test_unimplemented"
```

Task 2: Comprehensive Test Suite Development (30%)

2.1 Update Assignment 1 Tests

- Fix tests to handle corrected ISBN validation and borrowing limit bugs
- Add more edge cases to existing test files (`test_add_book_to_catalog.py`, `test_borrow_book_by_patron.py`)

2.2 Create Tests for `return_book_by_patron()`

- Create `tests/test_return_book_by_patron.py` with 8-10 test cases
- Cover valid returns, invalid patron ID, book not borrowed, already returned

2.3 Create Tests for `calculate_late_fee_for_book()`

- Create `tests/test_calculate_late_fee_for_book.py` with 8-10 test cases
- Cover on-time returns, overdue scenarios, invalid inputs, calculation edge cases

2.4 Create Tests for `search_books_in_catalog()`

- Create `tests/test_search_books_in_catalog.py` with 10-12 test cases
- Cover title/author/ISBN searches, no matches, invalid search types, special characters

2.5 Create Tests for `get_patron_status_report()`

- Create `tests/test_get_patron_status_report.py` with 8-10 test cases
- Cover active borrowers, no books borrowed, invalid patron ID, overdue scenarios

Task 3: AI-Assisted Test Generation (15%)

3.1 Generate Edge Cases Using LLMs

- Use ChatGPT, Claude, or Copilot to generate additional test cases for your functions
- Focus on edge cases that might break your implementations

3.2 Document LLM Usage

- Create `docs/llm_test_generation.md` with your prompts and LLM responses

- Include which LLM platform you used and effectiveness analysis

3.3 Example Prompt Template

```
I have a Python function that [describe function]. The function should
handle [describe requirements].
Generate 5 comprehensive test cases including edge cases that might break
this function.
Format as pytest test functions with clear test names and assertions.
```

3.4 Implement Generated Tests

- Add at least 10 LLM-generated test cases to your test suite
- Validate and adapt generated tests to ensure they're accurate and valuable

3.5 Analysis Report

- Document which generated tests were most valuable
- Include lessons learned about effective prompt engineering

Task 4: CI/CD Pipeline Setup (15%)

4.1 Create GitHub Repository

- Create public repository named `cisc327-library-management-a2-[your-student-id]`
- Initialize with proper `.gitignore` for Python projects

4.2 Setup GitHub Actions

- Create `.github/workflows/tests.yml` for automated testing
- Configure matrix testing for Python 3.8, 3.9, and 3.10

4.3 Implement Test Coverage

```
- name: Run tests with coverage
  run: |
    pytest tests/ -v --cov=library_service --cov-report=xml
```

4.4 Add Status Badges

- Add test status badge to README.md
- Add coverage badge using Codecov

4.5 Professional Documentation

- Update README.md with project description, installation instructions, usage examples
- Ensure all tests pass in GitHub Actions with green badge status

Deliverables & Submission

Submission Requirements

Create a report `A2_LastName_last4digitID.md` containing:

1. **Implementation Summary** - List of completed functions and key challenges
2. **Test Suite Analysis** - Total test cases and coverage improvements
3. **LLM-Assisted Development Report** - Platforms used and effectiveness
4. **CI/CD Setup Documentation** - Repository URL and workflow status

GitHub Repository Requirements

- All 4 functions implemented and working
- Comprehensive test suite (minimum 40 total test cases)
- Working GitHub Actions workflow
- Green test status badge in README
  -LLM test generation documentation

Grading Criteria

- **Function Implementation (40%)**: Correctness and code quality
- **Test Suite Quality (30%)**: Coverage and edge cases
- **AI-Assisted Development (15%)**: Effective LLM usage and documentation
- **CI/CD Pipeline (15%)**: Working GitHub Actions and status badges