Library Management System - Documentation

1. Project Overview

The **Library Management System** is a desktop application built with **F#** and **Avalonia UI**. It is designed to help librarians manage book inventory, track borrowed items, and maintain a database of books and borrowers. The system emphasizes data integrity, validation, and a seamless user experience.

2. Architecture & Design

High-Level Flow

1. **Presentation Layer (UI)**: The LibrarySystemAvalonia project handles all user interactions (clicks, inputs). It collects data and calls the underlying services.
2. **Business Logic Layer (Core)**: The LibrarySystem project contains the rules
3. **Storage Layer**: The data is persisted in a JSON file (library.json).

3. Key Features

Inventory Management

- **Add New Books**: Add books with Title, Author, Year, and Quantity.
- **Input Sanitization**: The system automatically trims extra spaces (e.g., " Harry Potter " becomes "Harry Potter").
- **Validation**: Prevents invalid data such as future years, empty fields, or author names with symbols/digits.
- **Stock Management**: Add copies to existing books or remove copies (unless they are currently borrowed).

Lending System

- **Borrowing**:
  - Validates Borrower Name and Phone (must be 11 digits, starting with '01').
  - **Identity Check**: Prevents one phone number from being used by multiple names.

    - **Borrow Limit**: Users are restricted to borrowing a maximum of **2 books** at a time.
    - **Stock Check**: Prevents borrowing if no copies are available.
- **Returning**: Updates the inventory and removes the borrower record.

Persistence

- Data is saved to library.json.
- **Smart Migration**: The system can detect legacy data formats and automatically upgrade them to the current structure without data loss.

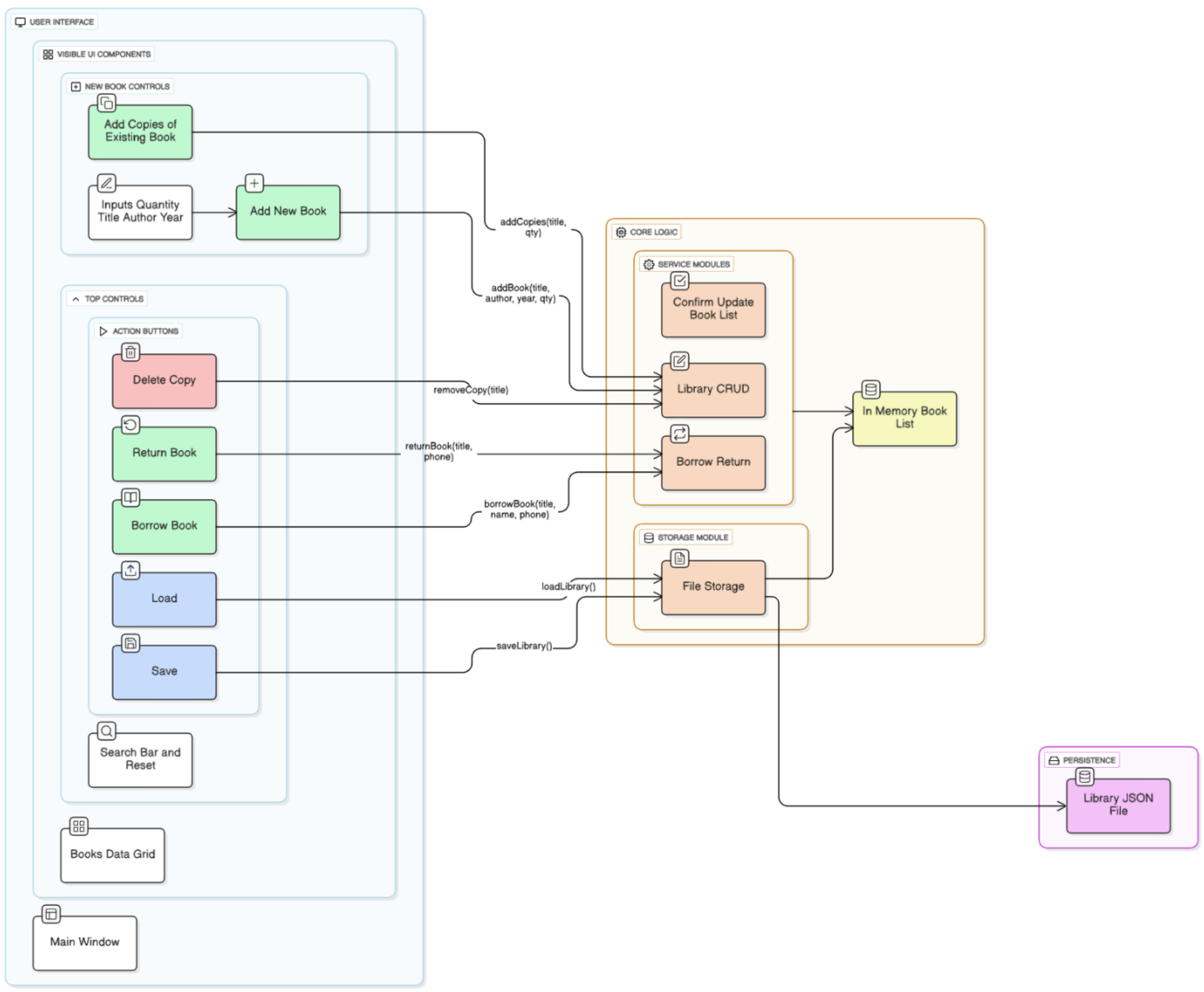
4. Testing Strategy

uses **xUnit**.

Test Coverage

1. **Unit Tests**:
    - **Validation Logic**: Tests that invalid inputs (bad phone numbers, future years, symbols in names) are rejected with clear error messages.
    - **Business Rules**: Verifies that borrow limits (max 2) are enforced and that duplicate books cannot be added.
    - **Data Sanitization**: Ensures that "messy" input strings are cleaned before storage.
2. **State & Lifecycle Tests**:
    - **Stock Decrement**: Verifies that borrowing a book reduces the "Available" count.
    - **Return Logic**: Ensures returning a book restores the stock count.
    - **Removal Constraints**: Tests that a copy cannot be deleted from the database if it is currently in a borrower's hands.
3. **Integration Scenarios**:
    - End-to-end flows are simulated (e.g., "Stock runs out -> User B fails to borrow -> User A returns -> User B succeeds").

5. System Architecture Diagram


USER INTERFACE
VISIBLE UI COMPONENTS
NEW BOOK CONTROLS
Add Copies of Existing Book
Inputs Quantity Title Author Year
Add New Book
TOP CONTROLS
ACTION BUTTONS
Delete Copy
Return Book
Borrow Book
Load
Save
Search Bar and Reset
Books Data Grid
Main Window
addCopies(title, qty)
addBook(title, author, year, qty)
removeCopy(title)
returnBook(title, phone)
borrowBook(title, name, phone)
loadLibrary()
saveLibrary()
CORE LOGIC
SERVICE MODULES
Confirm Update Book List
Library CRUD
Borrow Return
STORAGE MODULE
File Storage
In Memory Book List
PERSISTENCE
Library JSON File