Library Management System

****A Desktop Application using F# & WinForms**

1. Project Overview

The **F# Library Management System** is a lightweight desktop application developed in **F#** on the **.NET 8** platform using **Windows Forms** as the graphical user interface. The system manages a small library by allowing two distinct user roles:

- **Regular User** – can search, borrow, and return books.
- **Administrator** – can add, edit, delete, and search books.

All data is persisted in a JSON file (library.json) using System.Text.Json. The application follows clean separation of concerns: business logic is isolated in LibraryManager.fs while the UI is handled by separate forms.

2. Objectives

- Implement CRUD operations in a functional-first language (F#).
- Provide role-based access (User vs Admin).
- Ensure data integrity (prevent edit/delete of borrowed books).
- Demonstrate serialization/deserialization with JSON.
- Write comprehensive unit tests using xUnit.

3. Main Features

Common Features (both panels)

- Search books by title or author (case-insensitive)
- View all books with status (Available / Borrowed)
- Automatic save/load from library.json

User Panel

- Borrow a book by ID
- Return a book by ID
- Visual indication (red background for borrowed books)

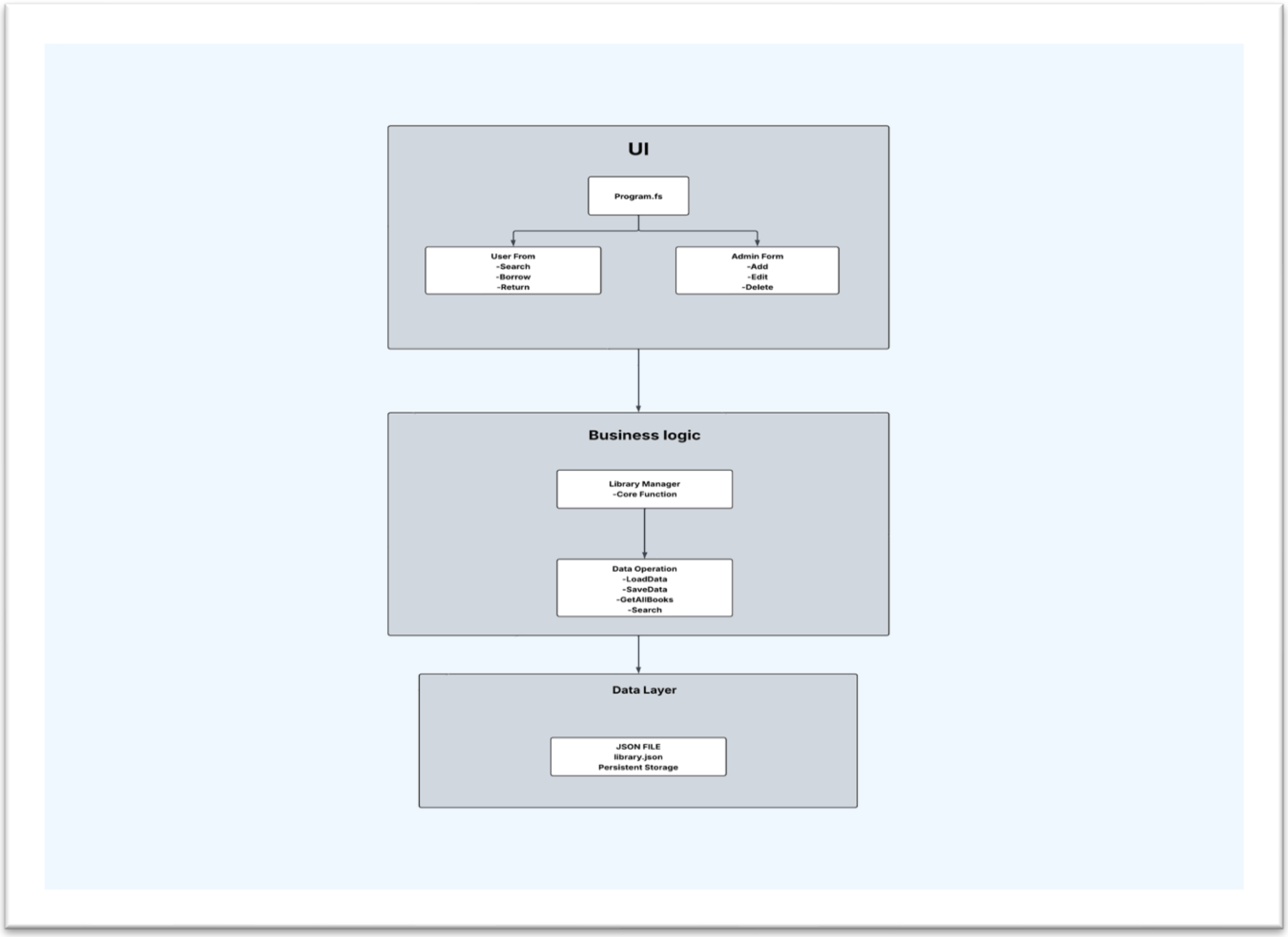
Admin Panel

- Add new book
- Edit existing book (blocked if borrowed)
- Delete book (blocked if borrowed)
- Re-numbering of IDs after deletion

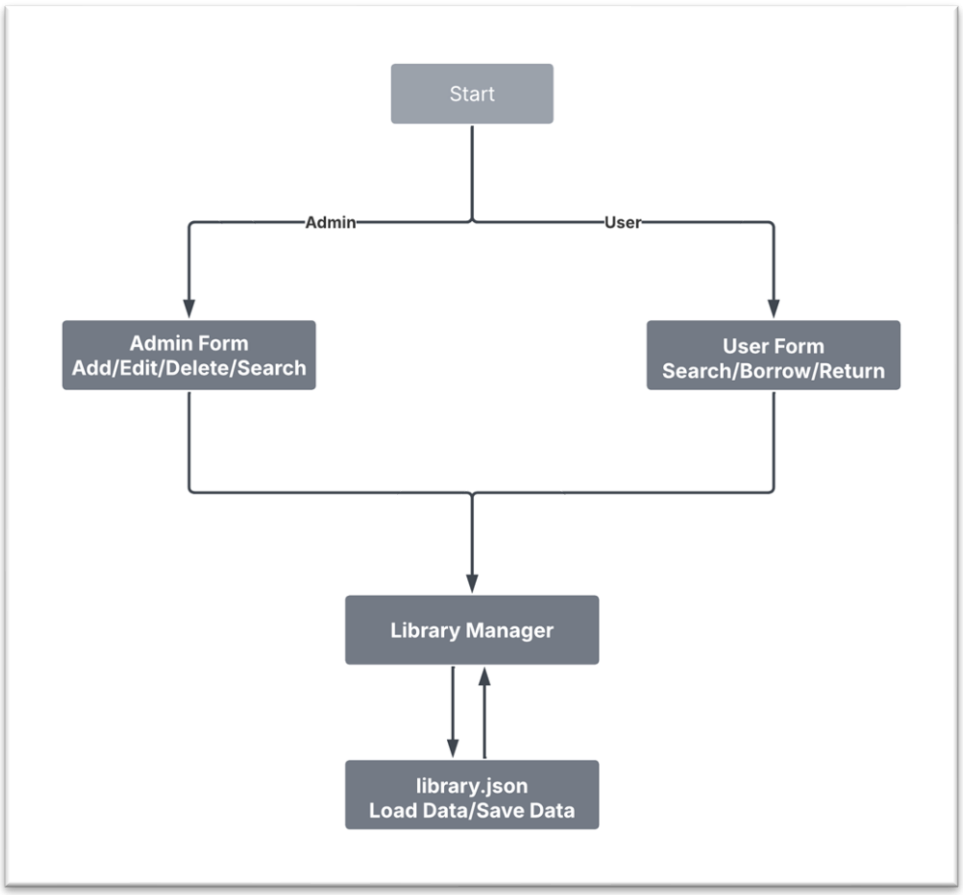
4. Technologies Used

- **Language:** F# 8.0 (.NET 8)
- **UI Framework:** Windows Forms (WinForms)
- **Data Persistence:** System.Text.Json
- **Testing:** xUnit
- **IDE:** Visual Studio

5. System Architecture



6. Block Diagram



7. User Interface Description

- **Welcome Form** – Simple role selection screen.
- **User Form** – Clean layout with search bar, book list, and Borrow/Return buttons.
- **Admin Form** – Additional input fields and Add/Edit/Delete controls. All forms use the same color scheme and consistent font (Segoe UI

8. Testing Strategy

Framework: xUnit **Number of Tests:** 5 unit tests covering all core functions.

Test Name	Purpose	Expected Result
Add book increases count	Verify a new book is added correctly	Count = 1
Search works	Case-insensitive search returns correct book	1 result
Borrow and Return works	Toggle availability correctly	Status changes
Edit book updates data	Edit fails when book is borrowed	Update blocked if borrowed
Delete book removes it and renumbers	Delete fails when borrowed, IDs renumbered	Correct count & IDs

All tests run on an isolated test_library.json file to ensure independence.

9. How to Run the Project

1. Open the solution in Visual Studio 2022.
2. Set LibraryManagementSystem (not the Tests project) as Startup Project.
3. Press F5.
4. Welcome screen appears → choose User or Admin.
5. Data is automatically saved to bin\Debug\net8.0-windows\library.json.

10. Conclusion & Future Enhancements

The project successfully demonstrates the power of F# in building maintainable desktop applications with a clean functional core. Possible future improvements:

- SQLite or SQL Server database
- User authentication with login screen
- Borrowing history and due dates
- Reporting module (overdue books, most borrowed, etc.)