SETUP: For this Project

Category	Name / Version	Purpose / Notes
.NET Version	.NET 6.0 / .NET 7.0 / .NET Framework 4.8	Required to run/build the application
NuGet Package	Microsoft.EntityFrameworkCore	Entity Framework Core ORM
NuGet Package	Microsoft.EntityFrameworkCore.SqlServer	SQL Server provider for EF Core
NuGet Package	Microsoft.EntityFrameworkCore.Tools	EF Core CLI tools (for migrations, optional)
NuGet Package	System.Data.SqlClient *(if used)*	Direct SQL Server access (optional)
Database Engine	SQL Server LocalDB *(recommended for offline)*	Local, lightweight SQL Server instance
Database Engine	OR SQL Server Express	For multi-user or production setups
VS Extension	WinForms Designer	For designing WinForms UI
VS Extension	Entity Framework Core Tools	For managing migrations (optional)
Database Setup	Your SQL scripts or .mdf file	To create or copy the database
Config File	appsettings.json or App.config (provided in file already)	Update connection string for the new environment

Tool/Library	Version/Name (Example)	How to Check / Notes
SQL Server Management Studio (SSMS)	19.1.56.2	*Help → About* in SSMS
SQL Server Engine	Microsoft SQL Server 2022 (16.0.1000)	Run SELECT @@VERSION; in SSMS
EF Core SQL Server Provider	Microsoft.EntityFrameworkCore.SqlServer	Check NuGet packages in your project
SQL Client Library *(if used)*	System.Data.SqlClient	Check NuGet packages in your project

Main Roles and Their Roles

File	Purpose
Program.cs	Application entry point; starts the main form.
Models/Book.cs	Defines the Book entity (BookId, Title, Author, etc.).
Models/Student.cs	Defines the Student entity (StudentId, Name, ReferenceID, etc.).
Models/Transaction.cs	Defines the Transaction entity (TransactionId, BookId, StudentId, BorrowDate, ReturnDate).
Data/LibraryDbContext.cs	Entity Framework Core context; manages database access and entity sets.
BLL/LibraryManager.cs	Business logic layer; handles CRUD operations and business rules for books, students, transactions.
Forms/MainForm.cs	Main WinForms UI; displays tabs for books, students, transactions, and handles user actions.
Forms/BorrowBookForm.cs	UI for borrowing a book; lets user select student and book, triggers borrow logic.
Forms/ReturnBookForm.cs	UI for returning a book; lets user select student and book, triggers return logic.
Utils/FineCalculator.cs	Utility class for calculating overdue fines.

Key Functions and Their Roles

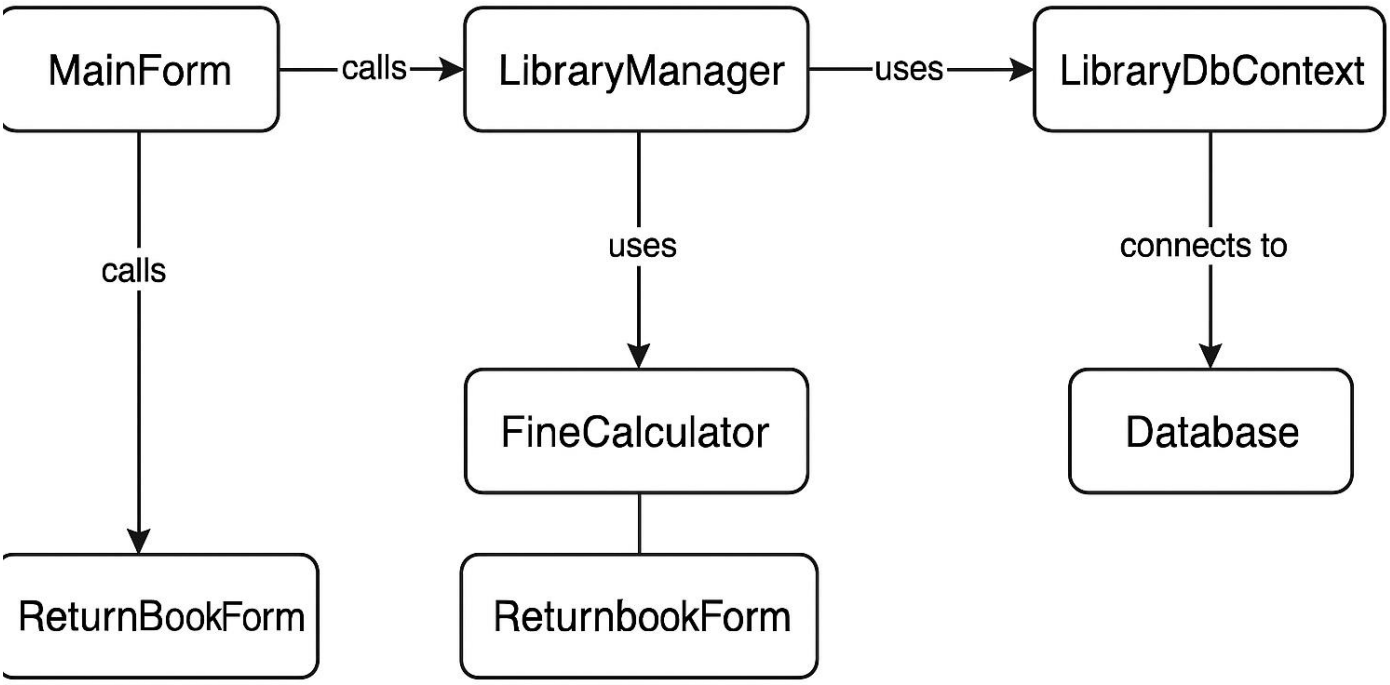
File	Function	Purpose / How It's Used
LibraryManager.cs	GetAllBooks()	Returns a list of all books from the database. Used to populate the books grid in the UI.
	GetAllStudents()	Returns a list of all students. Used to populate the students grid.
	GetAllTransactions()	Returns all transactions, including book and student info. Used for the transactions grid.
	BorrowBook(int bookId, int studentId)	Creates a new transaction, marks book as unavailable. Called from BorrowBookForm.
	ReturnBook(int bookId, int studentId)	Marks a transaction as returned, updates book availability. Called from ReturnBookForm or MainForm.
	DeleteBook(int bookId)	Deletes a book if not currently borrowed.
	DeleteStudent(int studentId)	Deletes a student if they have no borrowed books.
	ClearAllTransactions()	Deletes all transactions and resets book availability.

FineCalculator.cs	CalculateFine(DateTime borrowDate, DateTime? returnDate)	Calculates overdue fine (₱5/day after 7 days). Used when displaying or processing returns.
BorrowBookForm.cs	btnBorrow_Click	Handles the borrow button click: validates input, finds student, calls LibraryManager.BorrowBook.
MainForm.cs	LoadData()	Loads and displays all books, students, and transactions in the UI.
	BtnBorrowBook_Click	Opens BorrowBookForm for the selected book.
	BtnReturnBook_Click	Handles returning a book, calls LibraryManager.ReturnBook.
	ExportGridToCsv	Exports data from grids to CSV files.
	RestartApplication	Restarts the app, with a warning if there are unreturned books.

How the Code Works Together

1. **Startup:**
   - Program.cs launches MainForm.
   - MainForm displays tabs for books, students, and transactions.
2. **Data Loading:**
   - MainForm calls LibraryManager.GetAllBooks(), GetAllStudents(), and GetAllTransactions() to populate grids.
3. **Borrowing a Book:**
   - User clicks "Borrow Book" → BorrowBookForm opens.
   - User enters/selects Reference ID → btnBorrow_Click validates and calls LibraryManager.BorrowBook.
   - Book is marked unavailable, transaction is created.
4. **Returning a Book:**
   - User clicks "Return Book" → BtnReturnBook_Click in MainForm or ReturnBookForm is triggered.
   - Calls LibraryManager.ReturnBook, which sets the return date and marks the book as available.
   - Fine is calculated using FineCalculator if overdue.

5. **Managing Data:**
    - Users can add/edit/delete books and students via forms.
    - Transactions can be cleared, and data can be exported to CSV.
6. **Database Access:**
    - All data operations go through LibraryDbContext (Entity Framework Core).
    - The context manages connections to the SQL Server/LocalDB database.



Architecture Overview

Your project follows a **layered architecture** (sometimes called N-Tier or 3-Tier), which separates concerns into distinct layers:

1. **Presentation Layer (UI)**
2. **Business Logic Layer (BLL)**
3. **Data Access Layer (DAL) / Models**
4. **Database**

1. Presentation Layer (UI)

- **Files:** Forms/MainForm.cs, Forms/BorrowBookForm.cs, Forms/ReturnBookForm.cs, etc.
- **Role:**
  - Handles all user interactions (buttons, forms, grids).
  - Displays data to the user and collects input.
  - Calls the BLL (LibraryManager) to perform actions.
- **Design:**
  - Uses WinForms for a tabbed, user-friendly interface.
  - Each form is responsible for a specific task (borrowing, returning, managing books/students).

---

2. Business Logic Layer (BLL)

- **File:** BLL/LibraryManager.cs
- **Role:**
  - Contains all the business rules and logic (e.g., can't borrow if already borrowed, can't delete a student with borrowed books).
  - Acts as a bridge between the UI and the database.
  - Exposes methods for CRUD operations and higher-level actions (borrow, return, calculate fines).
- **Design:**
  - Centralizes logic so rules are enforced consistently, no matter where the action is triggered from.

---

3. Data Access Layer (DAL) / Models

- **Files:** Models/Book.cs, Models/Student.cs, Models/Transaction.cs, Data/LibraryDbContext.cs
- **Role:**
  - Defines the structure of your data (entities).

- LibraryDbContext manages database connections and entity sets using Entity Framework Core.
- Handles all communication with the SQL Server/LocalDB database.

- **Design:**
  - Uses Entity Framework Core for ORM (Object-Relational Mapping), so you work with C# objects instead of raw SQL.

---

4. Database

- **Technology:** SQL Server LocalDB or SQL Server Express
- **Role:**
  - Stores all persistent data (books, students, transactions).
  - Enforces data integrity with primary keys, foreign keys, and constraints.

Supporting Utilities

- **File:** Utils/FineCalculator.cs
- **Role:**
  - Provides reusable logic for calculating overdue fines.
  - Keeps utility code separate from business logic for maintainability.

Design Principles Used

- **Separation of Concerns:**

Each layer has a clear responsibility, making the code easier to maintain and extend.

- **Reusability:**

Business logic and utility functions are centralized and reusable across the application.

- **Scalability:**

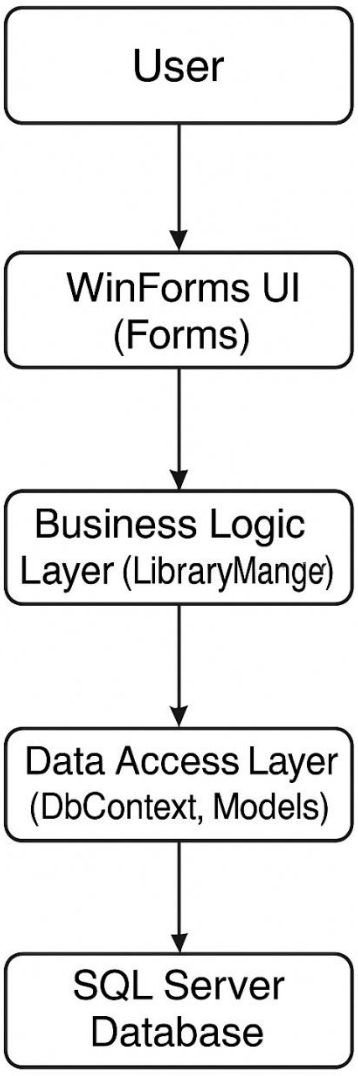
The architecture allows for easy addition of new features (e.g., new forms, new business rules).

- **Maintainability:**

Changes in one layer (e.g., UI) do not require changes in others (e.g., database).

How the Layers Interact

1. **User interacts with the UI** (e.g., clicks “Borrow Book”).
2. **UI calls the BLL** (LibraryManager.BorrowBook).
3. **BLL uses the DAL/DbContext** to read/write data.
4. **DbContext communicates with the database** to persist or retrieve data.
5. **BLL may use utilities** (e.g., FineCalculator) for calculations.
6. **Results/data flow back up** to the UI for display.



References:

Term / Acronym	Full Form / Meaning	Description
.NET	.NET Framework / .NET Core	A software framework developed by Microsoft for building and running applications on Windows.
EF Core	Entity Framework Core	A lightweight, extensible, open-source Object-Relational Mapper (ORM) for .NET applications.
ORM	Object-Relational Mapper	A tool that converts data between incompatible type systems (e.g., between SQL and C# objects).
NuGet	NuGet Package Manager	A package manager for .NET that helps install and manage libraries (packages) in a project.
SQL	Structured Query Language	A language used to interact with databases—querying, inserting, updating, and deleting data.
SQL Server	Microsoft SQL Server	A relational database management system developed by Microsoft.
LocalDB	SQL Server LocalDB	A lightweight version of SQL Server for developers; runs locally and requires minimal setup.
DbContext	Database Context	A class in EF Core that manages entity objects and database communication.
BLL	Business Logic Layer	The layer that contains application logic and rules (e.g., can a book be borrowed?).
DAL	Data Access Layer	The layer that directly interacts with the database through models and EF Core.
UI	User Interface	The visual part of the application the user interacts with, built with WinForms in this case.
WinForms	Windows Forms	A GUI class library in .NET for building desktop applications on Windows.
CRUD	Create, Read, Update, Delete	The four basic functions of persistent storage used in database operations.
Migrations	EF Core Migrations	A feature in EF Core used to update the database schema as the model changes over time.
.mdf	Microsoft Database File	The primary data file for SQL Server databases.
App.config / appsettings.json	Configuration Files	Files where application settings, such as database connection strings, are stored.

CLI	Command Line Interface	A way to interact with tools or programs using typed commands instead of a GUI.
SSMS	SQL Server Management Studio	A GUI tool used to manage and interact with SQL Server databases.
Reference ID	—	A unique identifier assigned to a student in your system.
Overdue Fine	—	A monetary penalty calculated for late return of a book.
Tab (UI)	—	A visual section of a form used to organize content (e.g., tabs for books, students).