# Library Project

This project has been developed using Ubuntu 2404 LTS, VS Code, Intellij Rider, Docker, and SQL server.

The project should work without any problems using localdb in Visual Studio.

To change connection string settings head to /Library.Backend/Library.Presentation/appsettings.json

There are 3 seeded users when you first start the backend server:

- An admin: super_admin, 12345678@sa
- A client: ma7moud_kanbar, 12345678@mk
- A client: yanar_aldaghestani, 12345678@ya

## 1. Library.Backend

Contains all the backend server code, and it's built upon Clean Architecture philosophy.

The backend consist of 3 layers “C# projects” as follows:

1. **Library.Domain**

   Contains all the system Entities, Manager Interfaces, Manager Implementations and Repository Interfaces.

   Also the layer defines an interface to handle session management scenarios.

   All the other layers depend on this layer.

2. **Library.Infranstructure**

   Contains all Repository Implementations using the Repository Interfaces that are defined in the Library.Domain layer.

   This is the only layer that interact with the SQL Server using ADO .NET.

   Here we define database creation and seeding logic.

   There are Materializer methods that bind SQL query result into the appropriate domain entities using SqlDataReader.

3. **Library.Presentation**

Uses ASP .NET, and it's the starting point of the backend server.

Here we register repositories, services and managers using the Dependency Injection provider in ASP .NET.

Also we read some configuration files like database settings (connection string), and JWT Bearer authentication provider settings, And store the configurations inside C# objects using Options Pattern.

This is the place where all the endpoints live using ASP .NET Controllers.

This layer orchestrate the logic between Library.Domain managers and repositories.

Responsible for authentication and authorization (I didn't use an authorization middleware due to the simplicity of the project).

Responsible for generating the appropriate responses when any error occur in the backend server using the ExceptionHandler middleware.

**NOTES:**

I didn't use any DTOs in the Library.Presentation to speedup the development time, even though it's a bad practice.

The system is susceptible to SQL Injection attacks (I didn't add any regex validation to speed up development time).

## 2. Library.WebApp

This project represent the frontend application that will communicate with the backend server.

I've used Blazor pages to implement the project UI.

There are two actors in the system an admin and a client.

Clients can only view, borrow and release books.

Admins can:

- View users and borrowing list.
- View, add, borrow and release books.

There is no sign-up page, just log-in page, to sign up a new user head to the API: localhost:PORT/Users/SignUp

I didn’t have any prior knowledge in Blazor or any web frontend framework, so I’ve tried to build this app by learning Blazor basics and by the help of some LLMs (Gemini and ChatGPT).