Books App — MySQL Mini Report

Course: Object-Oriented Analysis & Design + Advanced Web Programming

Team: Yoganathan Renaud, Dambron Max, Sghaier Souhaib

Goal

Build a simple, normalized MySQL-backed website for discovering books, browsing by genre, and posting user reviews. This version adds a more detailed database structure and an entity-relationship model to illustrate our design choices.

ER Diagram

The ER diagram shows how the main entities are connected:

- Book is the central entity.
- Each Book is written by one Author, published by one Publisher, and written in one Language.
- A Book can belong to several Genres through the link table Book_Genre.
- Users can post Reviews about different books.
- Relationships are mainly one-to-many, except Book ↔ Genre, which is many-to-many.

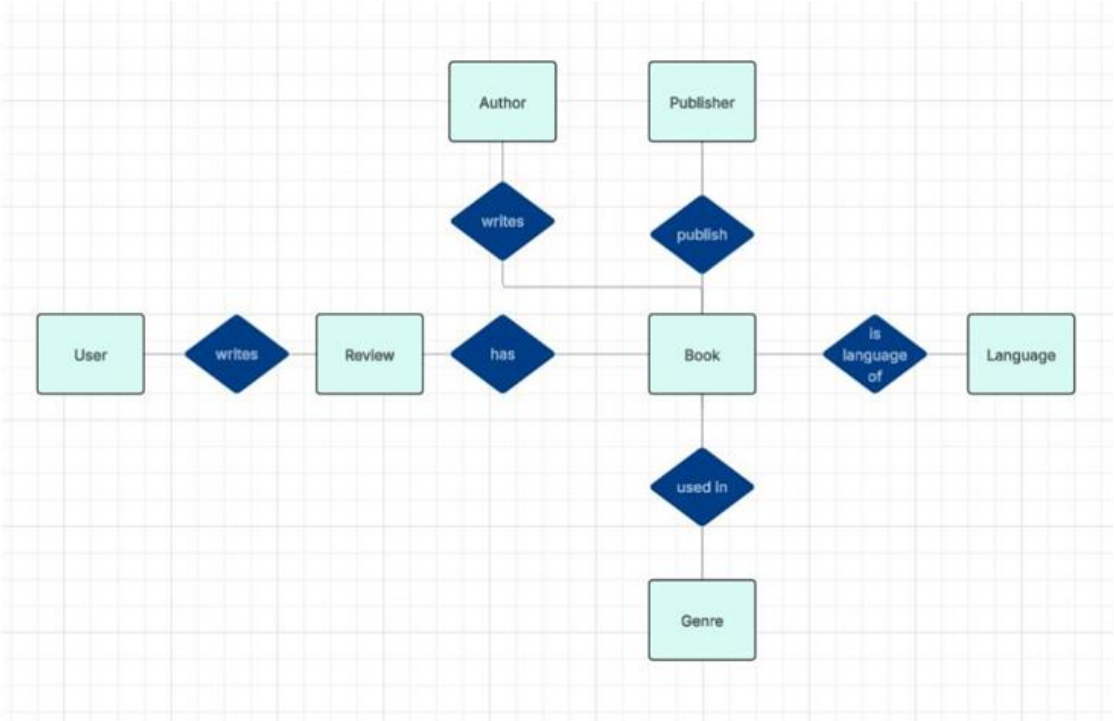


Table Structure Description

The diagram represents the logical structure of the database used for the book application. The goal was to design a simple, normalized model where each entity has a clear and unique purpose.

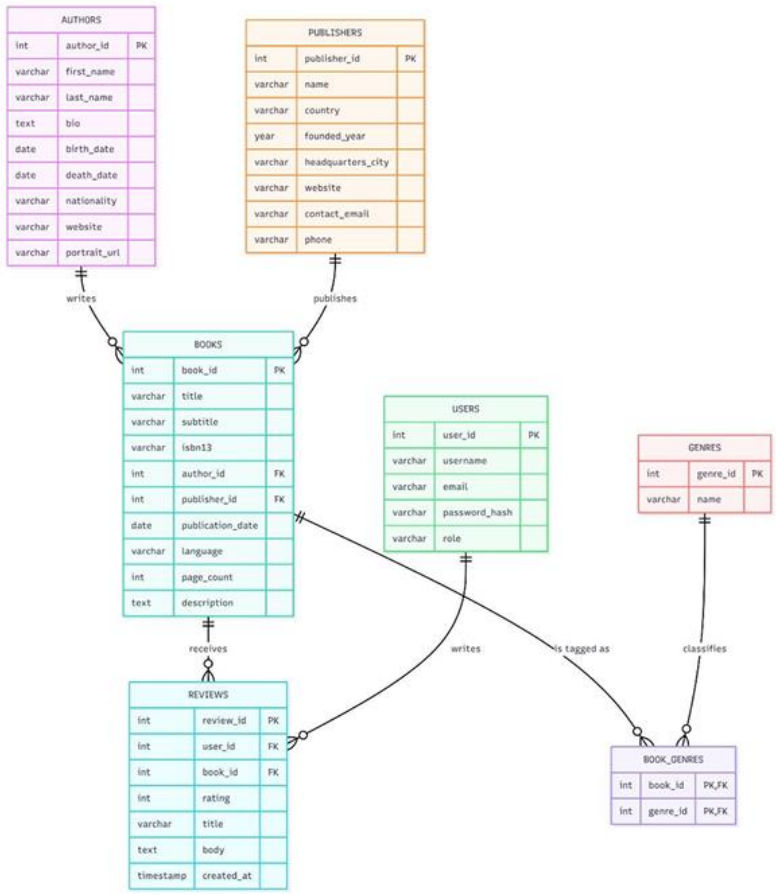
The **Books** table is the central element of the model. Each book is written by one **Author** and published by one **Publisher**, which explains the *one-to-many* relationships between these tables — one author or publisher can be linked to several books.

The **language** field was integrated directly inside the Books table to simplify the design and avoid an unnecessary extra table.

Users are represented in the Users table and can write multiple **Reviews**; each review being related to a single book.

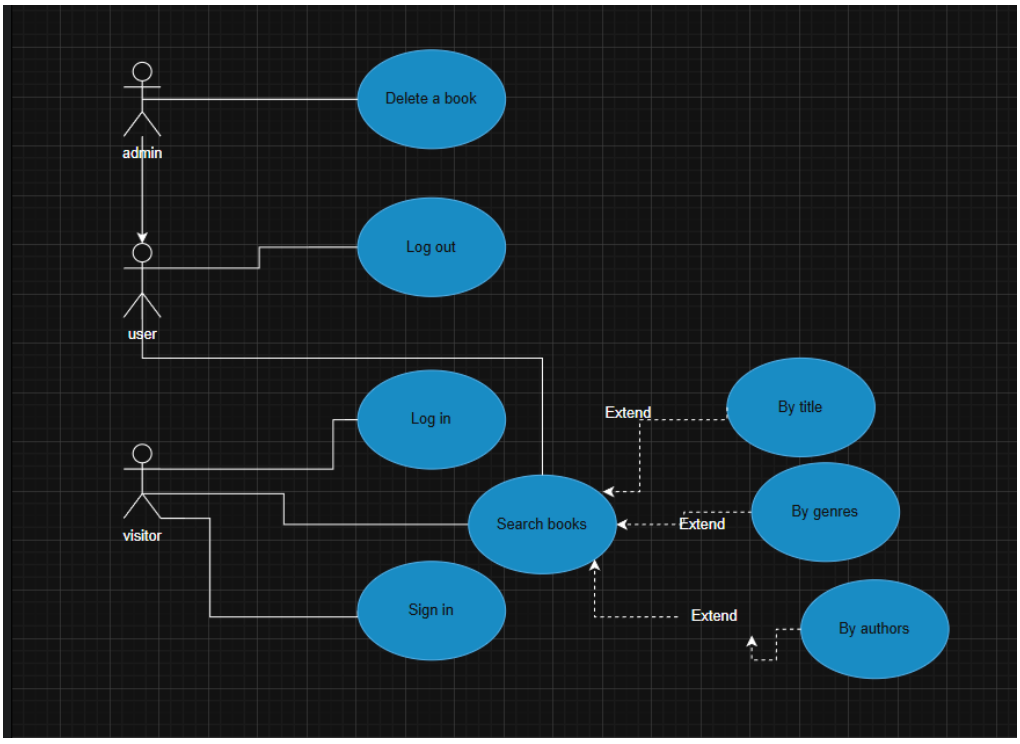
This creates a one-to-many relation between *Users → Reviews* and *Books → Reviews*.

Books can also belong to several **Genres**, and each genre can contain multiple books. This *many-to-many* relation is managed by the associative table **Book_Genres**, ensuring flexibility when classifying books.

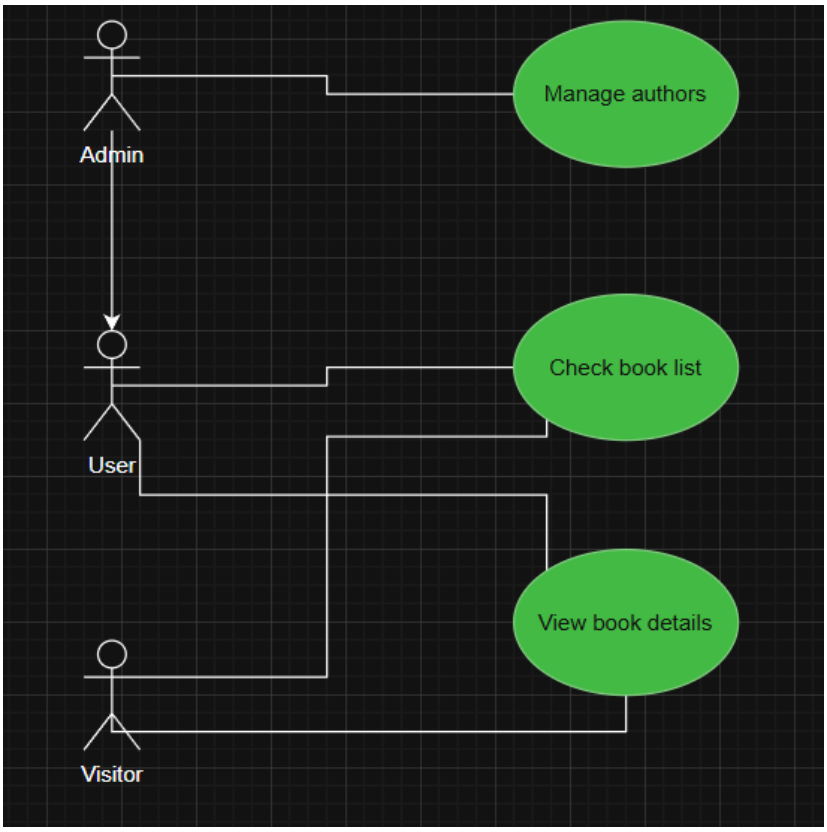


User diagram:

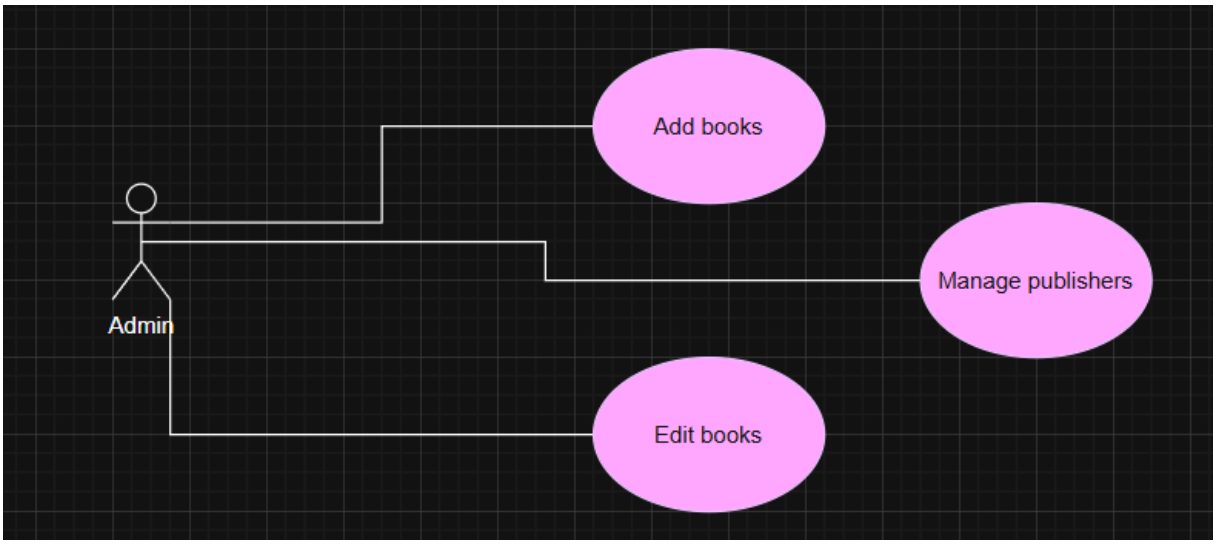
Renaud



Max

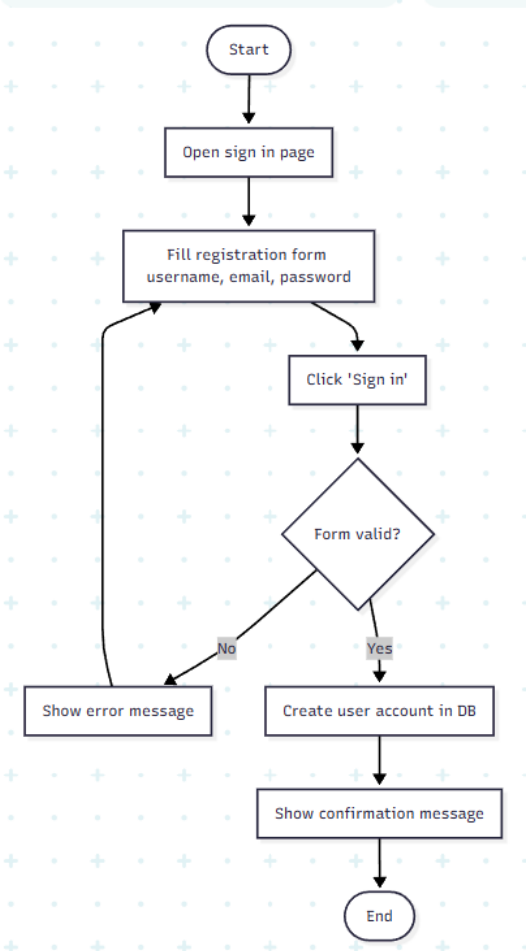


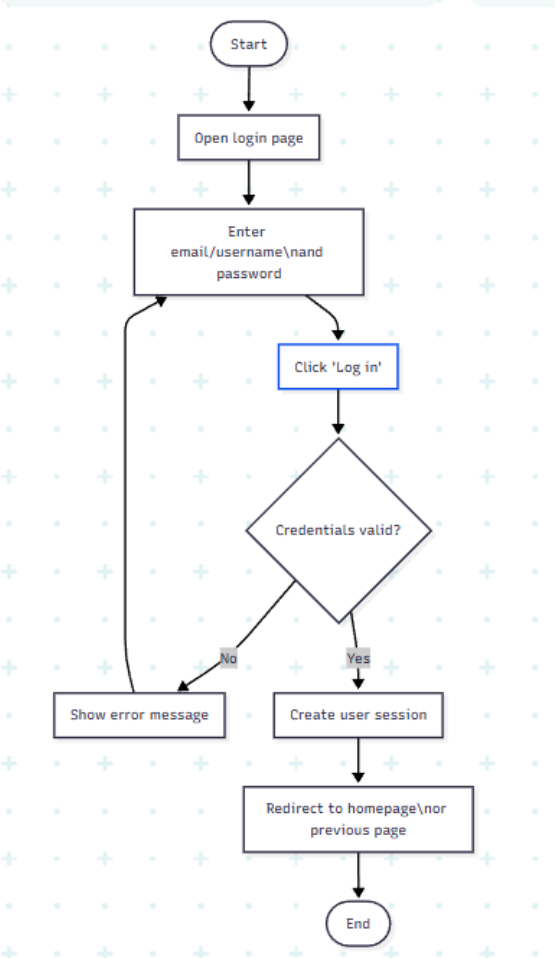
Souhaib

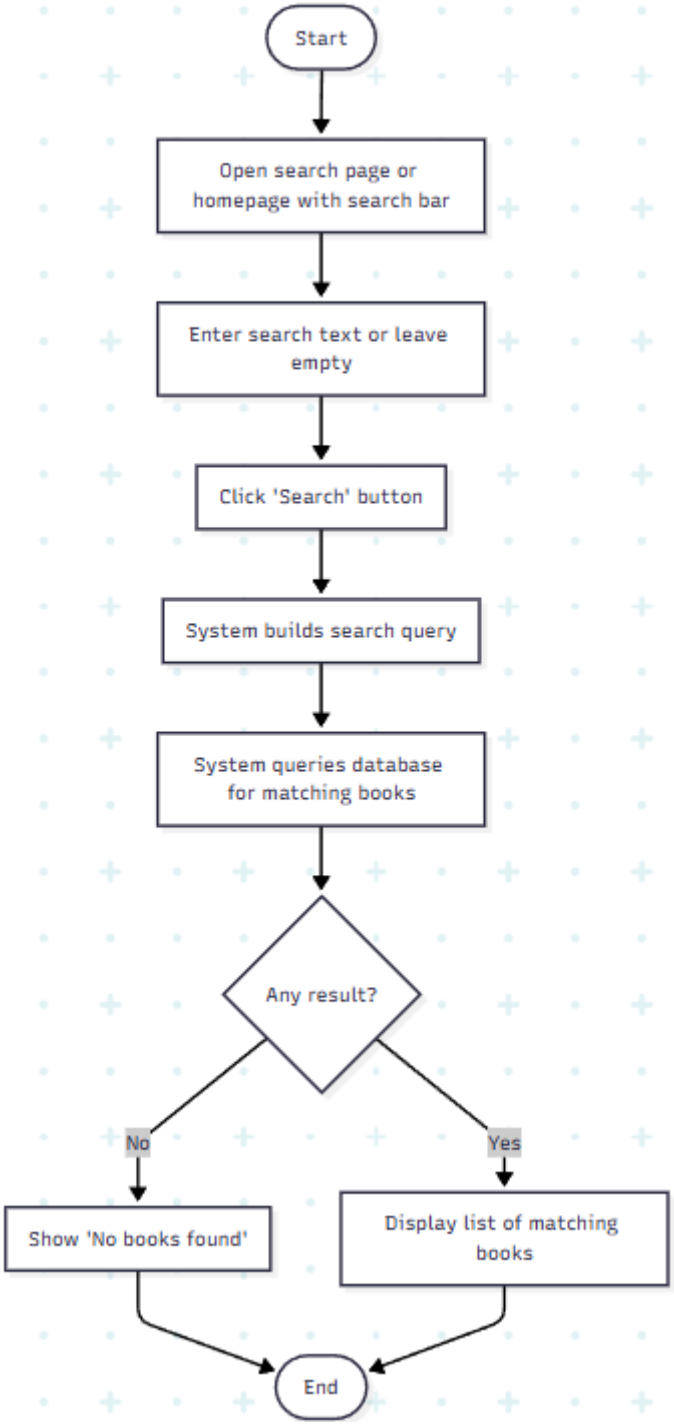


Activity diagram

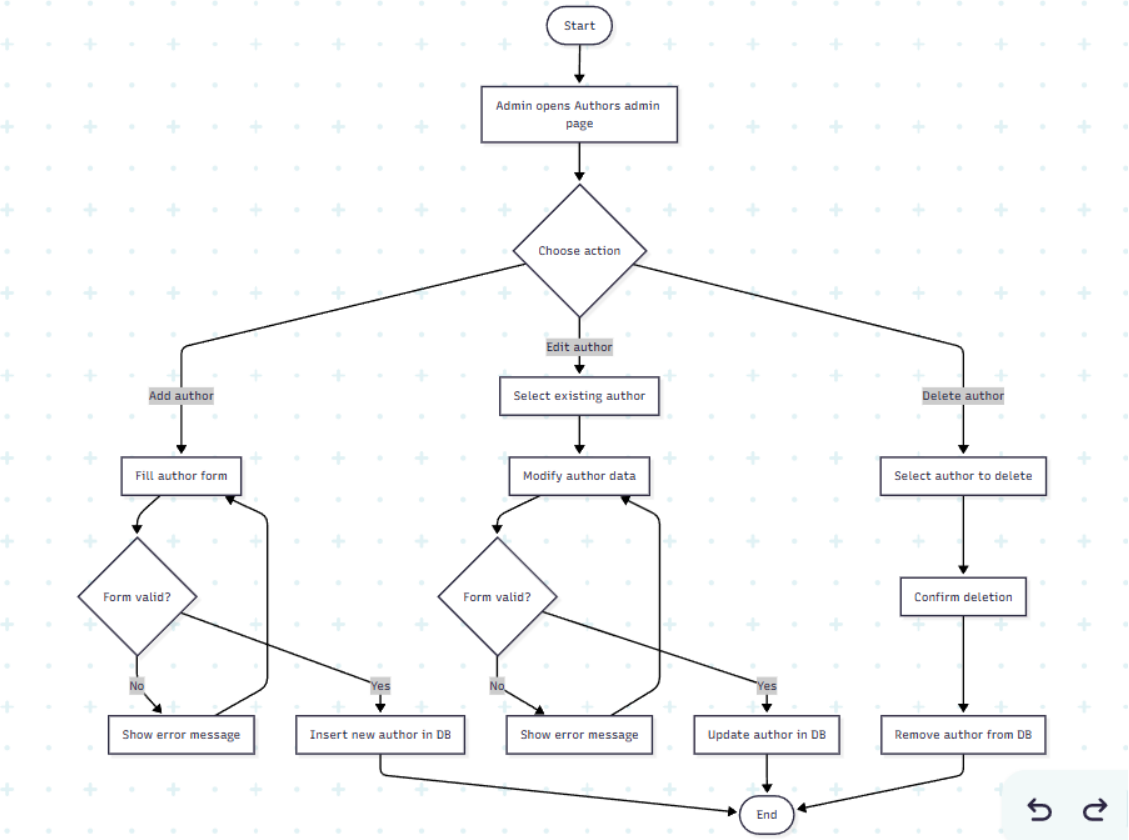
The three of Renaud (sign in, log in and search book global)

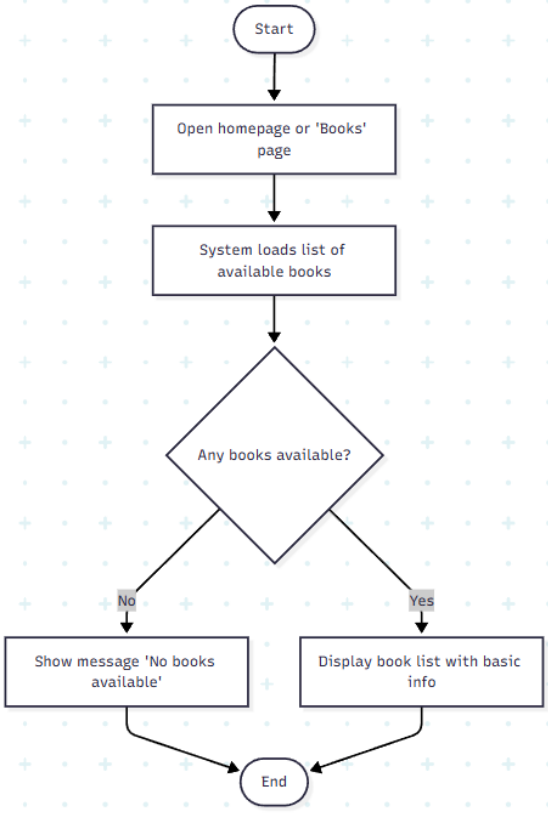


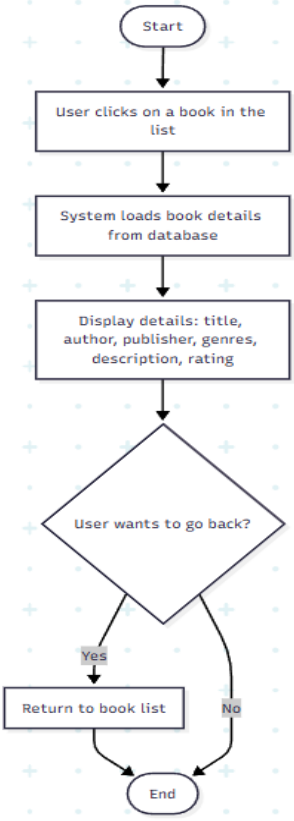



Start
Open search page or homepage with search bar
Enter search text or leave empty
Click 'Search' button
System builds search query
System queries database for matching books
Any result?
No
Yes
Show 'No books found'
Display list of matching books
End

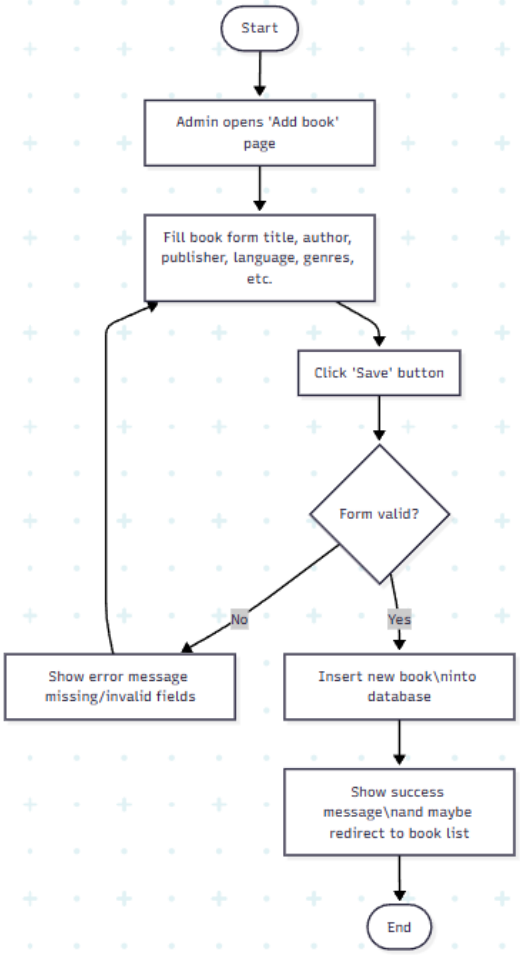
The three of Max (manage authors, check book list and view book details)

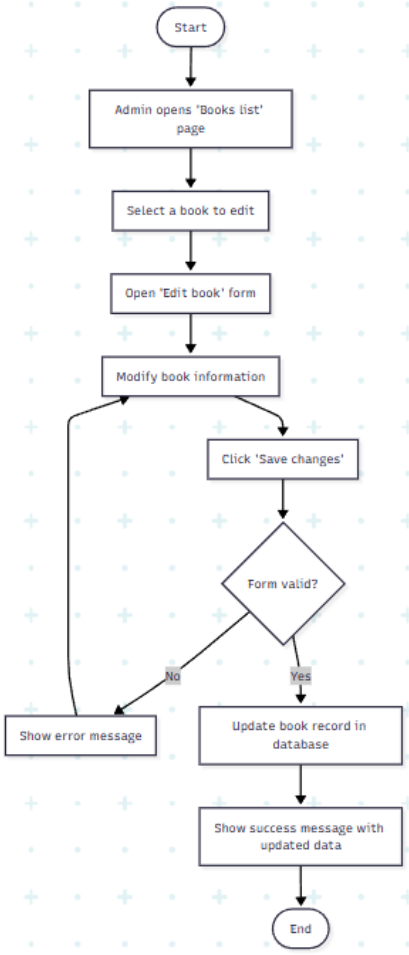


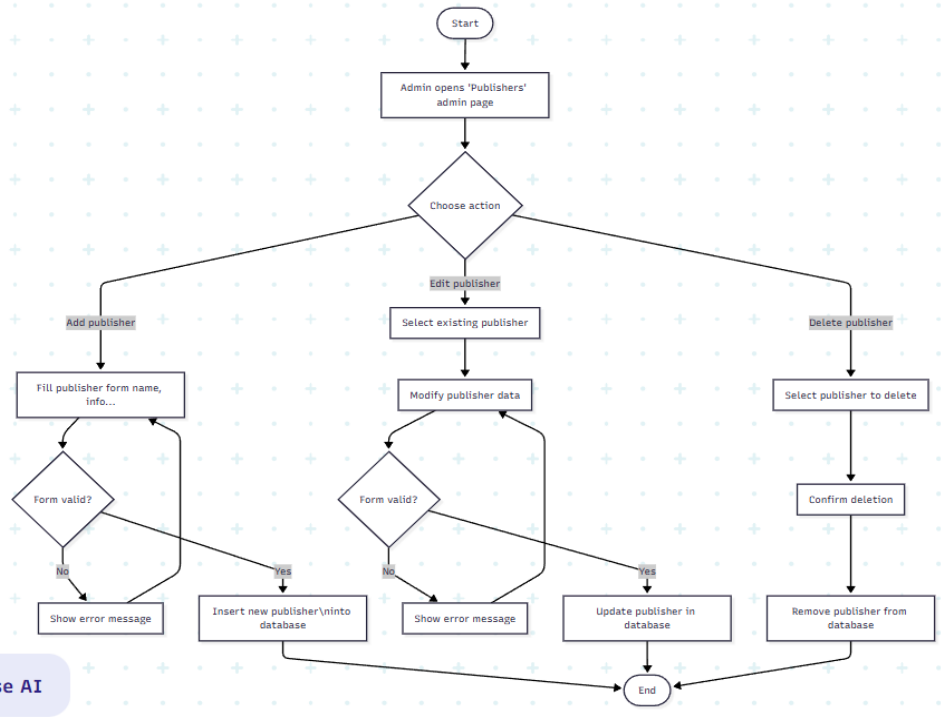




The three of Souhaib (add books, edit books, manage publishers)





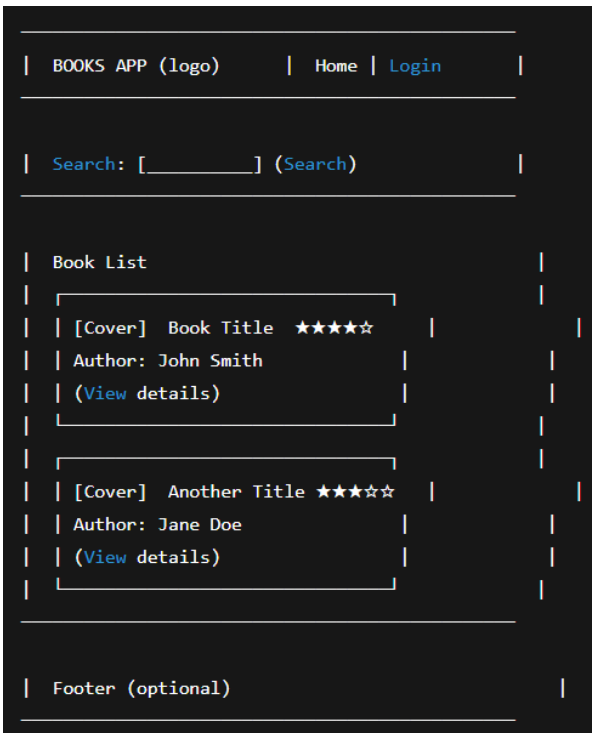


Wireframes

There is one wireframe per person.

Renaud (manage authors)

Max (check book list)



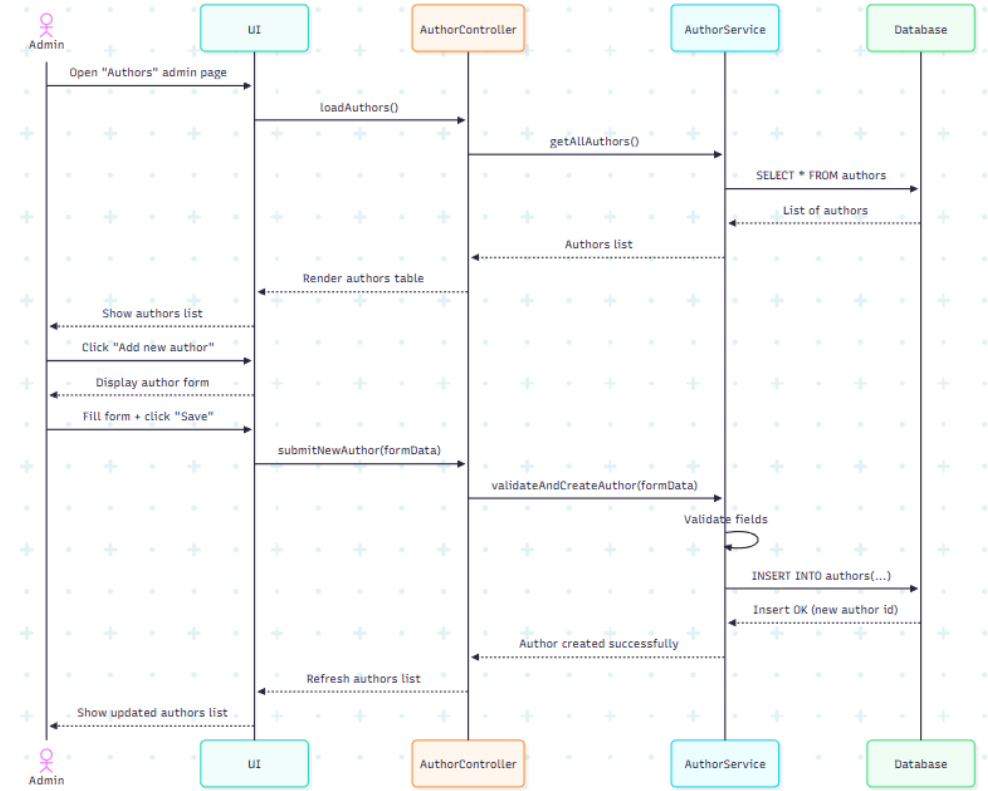
Souhaib (add book)

```
_____________________________________________
|  BOOKS APP (logo)     | Admin Panel | Logout |
_____________________________________________

|          ADD A NEW BOOK                     |
_____________________________________________

|  Title:           [__________________]      |
|  Author:          [Dropdown ▼]              |
|  Publisher:       [Dropdown ▼]              |
|  Language:        [Dropdown ▼]              |
|  Genres:          [ ] Fantasy   [ ] Drama   |
|                   [ ] Action    [ ] Sci-Fi  |
|                                             |
|  Description:     [____________________]    |
|                   [____________________]    |
_____________________________________________

|  ( Save )                ( Cancel )         |
_____________________________________________

| Footer                                      |
_____________________________________________
```

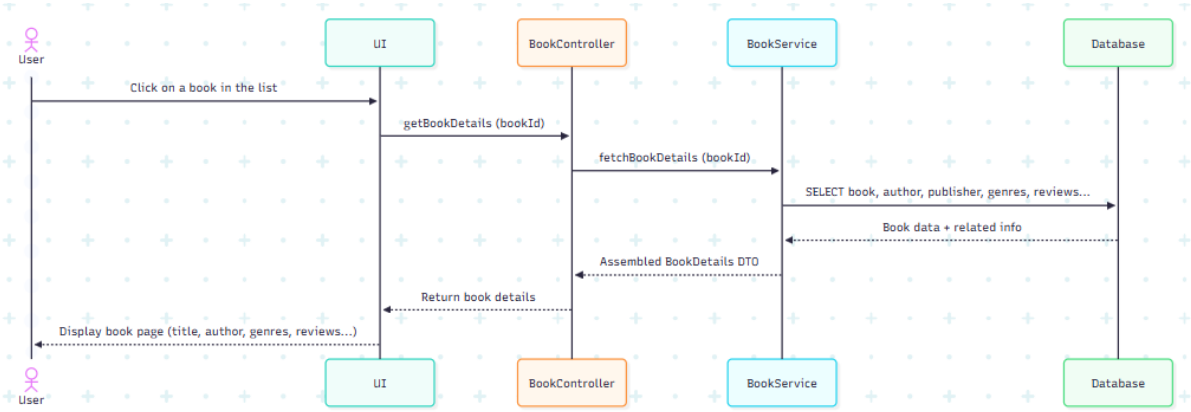
Sequence diagram

There is one sequence diagram per person.

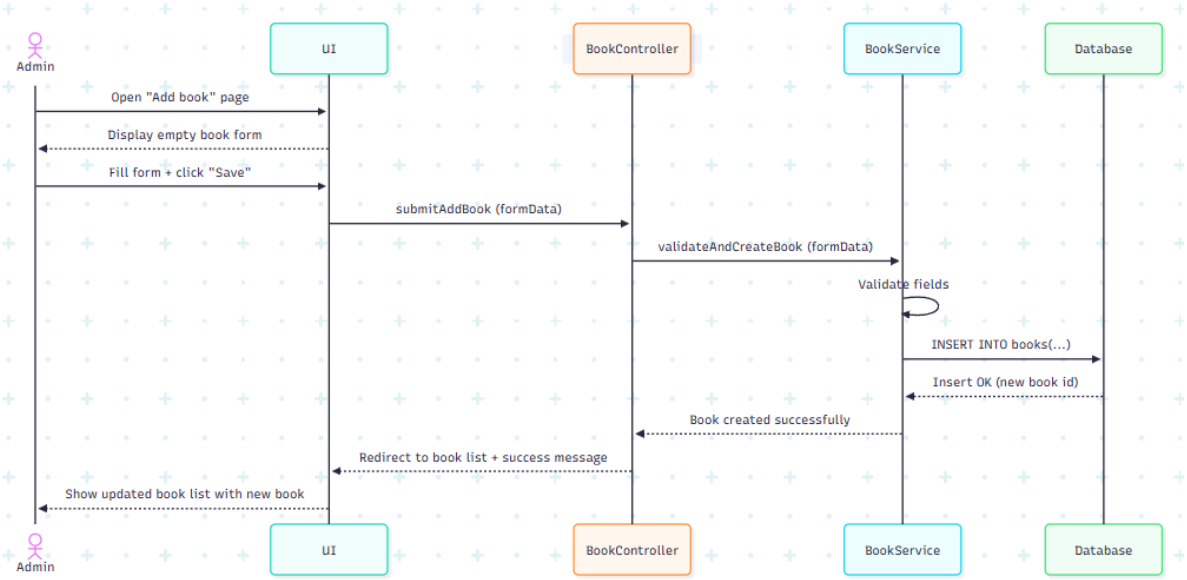
Renaud (manage authors)



Max (view books details)



Souhaib (add book)



Class diagram

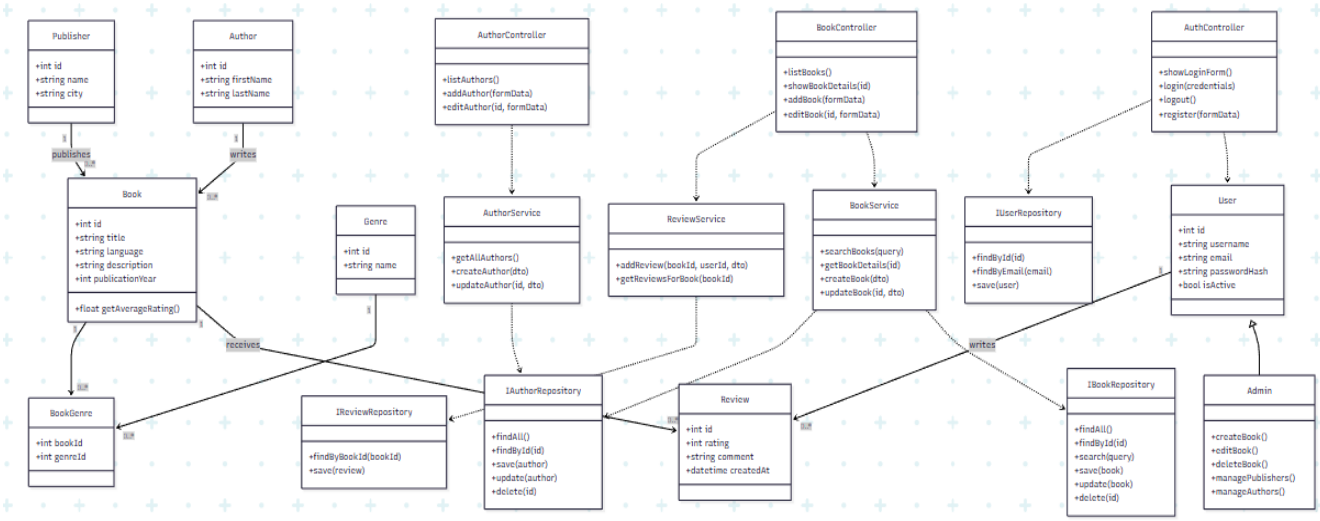
The class diagram shows the internal structure of the **Books App** using object-oriented design.

It separates the system into three layers: the domain layer (Book, Author, Publisher, User, etc.), the application layer (controllers and services), and the Persistence layer (repository interfaces).

The relationships between classes describe how books, authors, publishers, and reviews interact, while inheritance between User and Admin illustrates role specialization.

The dependencies between controllers, services, and repositories highlight the system's data and logic flow.

Overall, the diagram provides a clear and structured view of how the application is organized.



Component diagram

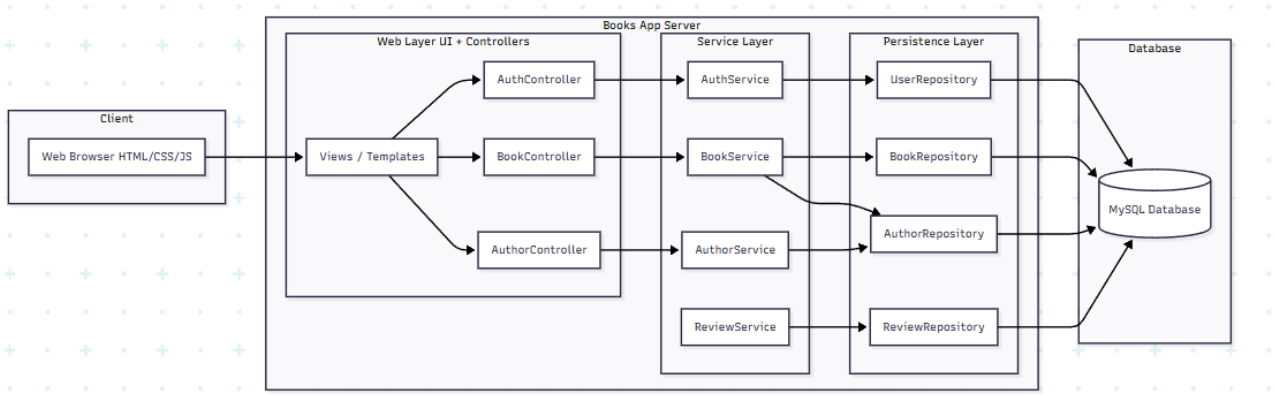
The component diagram illustrates the high-level architecture of the Books App.

The system is divided into three main layers: the Web Layer, which contains the UI and controllers; the Service Layer, which implements the business logic; and the Persistence Layer, which handles data access through repository components.

The client interacts with the application through the browser, which communicates with controllers.

Controllers delegate operations to services, and services retrieve or store data through repositories connected to the database.

This diagram provides a clear overview of how the major components of the application interact and how responsibilities are distributed across layers.



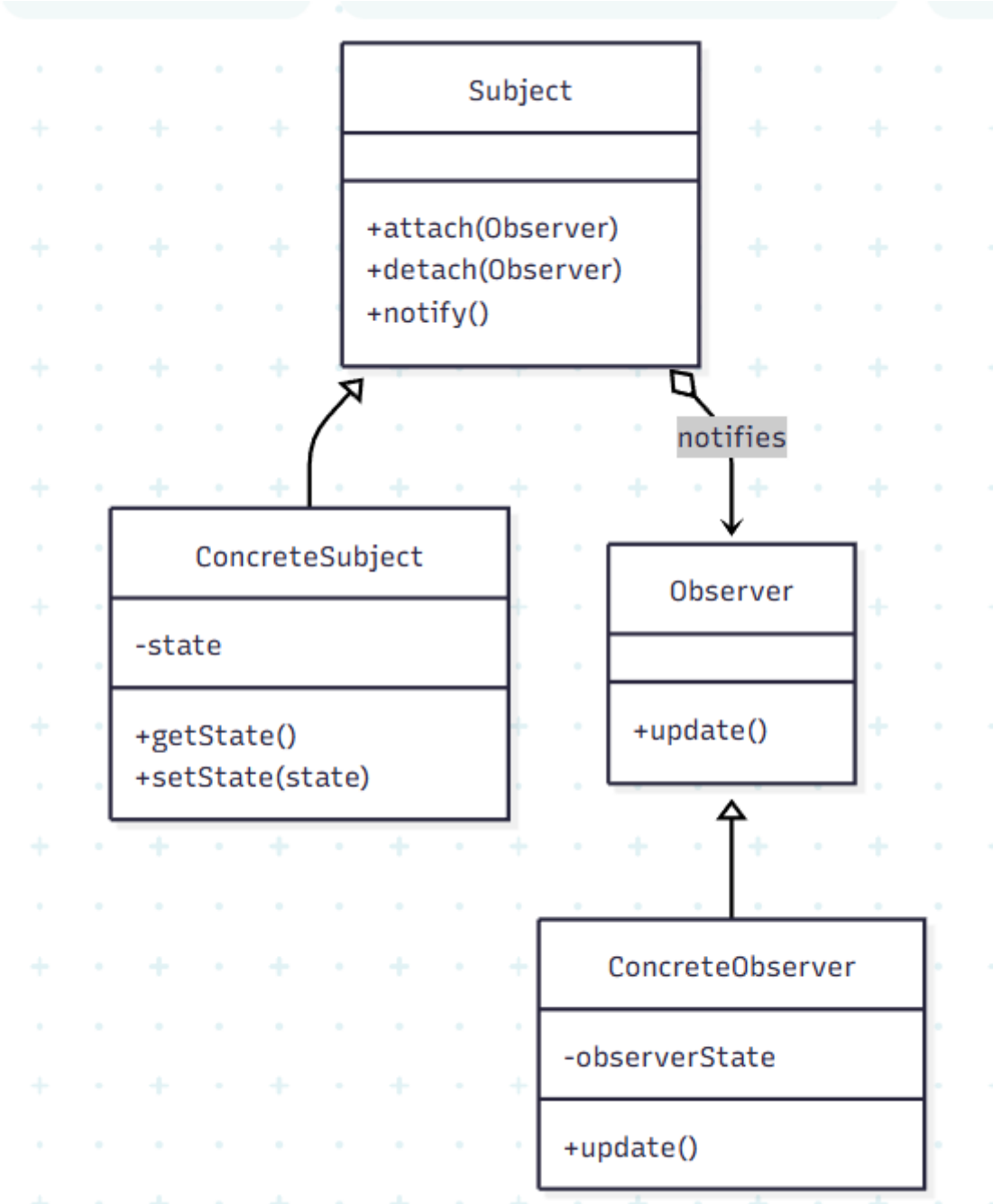
Behavioral diagram

When an admin adds a new book, the **BookService** can act as a Subject.

Any UI component displaying the list of books becomes an Observer.

When the service inserts the new book, it calls notify(), and all observers automatically refresh their displayed data.

This makes the system easier to maintain and prevents inconsistencies in the interface.



Gantt diagram