Second Assignment - Advanced Library System

Objective

Enhance the library project to support new features and advanced technologies.

User Journey

This assignment explains the operation of a library management system through the experiences of three users: John, a library member; Tarek, an author; Sarah, a CMS administrator and her intern, Ralph. Their journeys illustrate how different entities and processes interact within the system.

John's Journey (Library Member)

John, a university student, loves reading and frequently visits his local library. One day, he decides to visit the library's website to sign up. He provides his email, verifies it using a one-time password (OTP), and creates a password. Once registered, John logs in and becomes a member of the library and can **login** at any time.

As a registered member, John has a profile where his borrowing books history, feedback, and reviews are stored. His profile also tracks his **Return Rate**, a percentage that reflects how often he returns books on time. John knows that if his return rate drops below 30%, he won't be allowed to borrow books anymore.

John decides to visit a library branch near his home. The branch's **Inventory** shows that his favorite book, *The Great Gatsby*, has 10 total copies, but only 3 are available for borrowing. The inventory system ensures **real-time tracking** of each book's availability in the branch. Because John meets the minimum age requirement for borrowing the book, he proceeds to check it out.

When John borrows the book, the system creates a new **Borrow History** record. This record stores details like John's member ID, the book ID, the branch ID, and the borrowing date, while also setting a return deadline based on library policies. The **Branch Inventory** updates immediately, reducing the number of available copies by one. A **real-time event** is triggered to reflect the new availability on the branch's details page for that specific book.

> This number would be reflected in the already finished *Get Book By Id* of the first assignment, which should now be filtered by branch. In the frontend, there would be a **live tracking** section in the details page.

If the book is authored by someone in the system, the author receives a notification, listing who borrowed his book and when.

John enjoys his book but knows he must return it on time to maintain his return rate. If he returns the book late, the borrow history will flag it as overdue, and his return rate will decrease. When he returns the book on time, the system updates the branch inventory to increase the number of available copies using **another real time event.**

After reading, John leaves a review for *The Great Gatsby*. Reviews include a comment and a star rating (1–5). The system displays these reviews based on their popularity, which is determined by how many members **like** a specific review. A member may undo his **like.**

> Only books that are **open to review** may have reviews added to them.

At midnight every day, a scheduled job runs to log any overdue books and adjust return rates as needed. This ensures the system stays up to date and fair for all members.

Tarek's Journey (Author)

Tarek is an author who wants to add his books to the library system. Unlike John, Tarek cannot sign up on the website. Instead, he must contact a **CMS User** like Sarah to be added to the system. Once added, Tarek receives a pin code via email, which he uses to log in. If
needed, Sarah can generate a new pin code at any time and send it to Tarek's email.

After logging in, Tarek can request the addition of a book to the library. To do this, he submits the book's details to **a specific library**, including:

- Title

Genre

ISBN

Description

Minimum age requirement

Number of copies requested

A PDF of the book

Cover Image of the Book

If the book is open to reviews

This request is sent to the CMS for approval.

To better understand how authors like Tarek interact with the system, the library provides the following analytics:

1. **Book Requests by Status**:
   Breakdown of book requests by status (Approved, Rejected, Pending).
2. **Average Approval Time**:
   Calculates the average time it takes for a book to be approved, measured from the submission date to the approval
date. 3. **Branch Distribution for Author's Books**:
   Total copies of books distributed per branch for a specific author

Sarah's Journey (CMS User)

Sarah is a CMS administrator responsible for managing the library system. She is the administrator, has full access to the CMS and performs several key functions (in addition of the other functions of the previous assignment).

Sarah can add additional CMS users to help her manage the library system through the "Create CMS user" page. In that page, she specifies the email, full name and **role** of that CMS user, an email is then sent to that user with a password so that he can login to the CMS. There are currently Dynamic creation of roles within the CMS

This role-based access ensures a clear division of responsibilities. An Administrator has permission to everything.

The first CMS user, Sarah, needs to be added by you, as there is no sign up for the CMS.

1. **Manage Authors**:
   Add new authors to the system by sending them a pin code via email.
   Generate and send new pin codes when needed.
2. **Branches Management**
   CRUD for branches
3. **View Book Requests and approve them**:
   Table to view all book requests
   Review book addition requests from authors like Tarek which are **pending** until she eithers **approves** or **reject them**
4. **Distribute Books**:
   Allocate a specific number of copies to each branch when a book is approved and adding the **number of borrowable days** to
it. 5. **Monitor the Dashboard**:
   View insights such as:
      Total number of books.
      Total overdue books.
      Most popular authors (top 5, with a "View More" option for paginated results).
      Most popular books (top 5, with a "View More" option for paginated results).
   Analyze branch-specific data, including:
      The most popular authors within a specific branch.
      A graph showing the number of books borrowed over time (filterable by day or month).
         Response Example:

```
1 [ { numberOfBooks: 10, date: 2025-1-1 }, { numberOfBooks: 12, date: 2025-1-2 }]
```

Sarah works with an Dynamic creation of roles

within the CMS , who also has access to the CMS but with limited permissions already mentioned above.

Make sure to convert the previous assignment to Nest JS and add the features explained in the user journey above.

Code Structure and General Guidelines

With the addition of branches, make sure the previous APIs are adapted to accommodate the new requirements. Make sure to include guards, interceptors (when possible) pipes (when possible).

Use Nest JS base structure and add to it while preserving it.

Have generic and reusable functions be in a util folder.

Avoid using **any**, have a types folder to include all required types for the application

Add **indexes** to your schemas in the correct places

Bonus Points

Add Jest tests

Add Obsidian documentation (add to the documentation folder)

Documentation

Add in a **Documentation Folder** that you will include on the same level of your [redacted] file the following: A well structured postman with folders and examples (Export it as JSON). **Make sure to SAVE with EXAMPLES.** The exported collections and schemas with **sufficient** data for testing

Assignments that do not have the documentation folder with the required points mentioned above will **fail.**

Final Notes

There are multiple ways to approach this assignment.

Test everything before submitting the assignment.
Try your best to finish as many functionalities as possible. Having part of the application finished, working and well tested will get you a better grade than handing all of the functionalities with most of them not working.

Use **Prettier.**

May luck be on your side.