Online Library Management System

Objective

Objective of this Task is build Rest APIs for Library Management System .

Description

Being an online library system, Platform will give the capability for anyone to create their own library. For each Library, we would have 2 type of user profiles (LibraryAdmin, Readers).
Where Owner or Creator of the Library can directly onboard LibraryAdmin and he/she can share a sign-up portal to Readers (Fancy enough). For now, System would target only one LibraryAdmin.

There is a huge number of different books published by a publication which is written by several authors, and some of which are newer versions of same book. Which is why standard ISBN number is given to each book which globally identifies a unique book. And so would do our Online Library system.

Profiles and Roles

- Admin:
    - Add or Remove Books
    - Update the details of Books, like number of copies
    - List Issue Requests
    - Approve or disapprove issue request from a user
    - Issue Info for a Reader
- Reader:
    - Raise an issue request for a book.
    - Search book by Title, Author, Publisher

Database Info

Database Tables would be as given below.

Library: (ID, Name)

Users: (ID, Name, Email, ContactNumber, Role, LibID)

BookInventory : (ISBN, LibID, Title, Authors, Publisher, Version, TotalCopies, AvailableCopies)

RequestEvents: (ReqID, BookID, ReaderID, RequestDate, ApprovalDate, ApproverID, RequestType)

IssueRegistery: (IssueID, ISBN, ReaderID, IssueApproverID, IssueStatus, IssueDate, ExpectedReturnDate, ReturnDate, ReturnApproverID)

Flows:

In the flows, user authentication and authorisation are most important, so for all the flows a base need would be for every user to pass their email address. System would use basic authentication (if user exists in table, then it's a valid user) and authorisation (user will have the access based on the role)

Library Owner Flows:

Library Creation: This would include registering a new Library and adding a user with Role "Owner". Library cannot be registered, if the same name library already exists. If the same name library already exists, we need to ask user to define a new name.

Reader:

Search Book: Reader would need capability to search for book by title, author, publisher and also need to know the status if book is available or not. If a book is not available, user would need to know the latest when the book will be available.

Raise an issue request: With book ID and email address, the reader can raise issue request.
Book's availability is defined as the copies of same book which are not issued yet, are the available books. If a request is being raised for the book which is not available, the request should get rejected immediately.

Library Admin's Flows:

Adding Books: While adding the books in inventory, book details will be passed, along with LibAdmin's email address. If the book is already registered, expectation is that system would automatically increment the number of copies.

Removing Book: To remove a book, we'll keep decrementing the number of copies till 0. The issued copies of the books cannot be removed.

Update Book: To update the details of book, user is expected to pass the ISBN and the information to be updated.

List Issue Request: For Approving or Rejecting any issue request, Admin would first list the issue requests in his Library

Approve or Reject Issue Request: From the listing, Admin would pick the request ID and approve the issue request, setting the request with approver ID and approval date, and setting the issue registry accordingly.

Tasks:

Task 1: Design and Create Database as per above information.

Here, we'll need to design and create Database as per above information, with proper field types and relations. Here we'll need to make sure that all the SQL Queries used should be available in SQL Files and the ER Diagram should be available.

Task 2: Design and Implement Backend APIs

The APIs should cover the validation of role and the anonymity as mentioned in the flows.

As there could be multiple Readers and multiple Admins requesting in parallel, the ***ACID properties*** should be made sure.

Best Practice and Modularisation should be there in APIs.

Unit Testcases of the APIs should be there with maximum code-coverage checked and exported as report.

Reference Links

1. https://gorm.io/
2. https://github.com/gin-gonic/gin
3. https://medium.com/@rosaniline/unit-testing-gorm-with-go-sqlmock-in-go-93cbce1f6b5b
4. https://blog.logrocket.com/angular-unit-testing-tutorial/
5. https://www.freecodecamp.org/news/how-to-write-unit-tests-in-react/
6. https://medium.com/mindful-engineering/getting-started-with-writing-test-cases-in-flutter-d5f432c4e680
7. https://blogs.halodoc.io/golang-unit-testing/
8. https://roadmap.sh/flutter
9. https://roadmap.sh/angular
10. https://roadmap.sh/react