Online Library Portal

Objective:

Create an online library portal that allows users to manage and borrow books. The system will include various roles with specific permissions, and users will interact through a web portal.

Learning Outcomes:

- Learn basics of ROR and Golang for Backend
- Learn basics of React for Frontend
- Learn about design patterns used in industry
- Learn about coding patterns used in industry
- Learn about frameworks/components used in React (e.g., Material UI)
- Learn about unit testing and its importance. How to modularize unit tests

Task Description and Functional Requirements:

User Roles and Permissions:

Admin:
- Can create/update/delete books.
- Can manage library staff (librarians)
- Can view all books and borrowing records.

Librarian:
- Can add/update/delete books.
- Can manage borrowing records.

Member:
- Can search and view books.
- Can borrow and return books.
- Can view their borrowing history.

Core Features:

Book Management:
- Admins and Librarians can create, update, and delete books.
- Books have details like title, author, publication date, genre, and availability status.

User Management:
- Admins can manage library staff.
- Users can register and log in to the portal.

Borrowing System:
- Members can borrow available books.
- Members can return borrowed books.
- Borrowing history is maintained for each member.
- Number of books available should be included in the UI.

Technical Requirements

Backend:

- User Service: Written in ROR, handles user authentication and authorization.
- Library Service: Written in GO, handles book management and borrowing system.
- Database: MySQL for relational data (users, books, borrowing records), MongoDB(logging)

Frontend:

- Implemented in React.js with Material UI.
- Responsive design for different device sizes.
- Interactive UI for book searching, borrowing, and returning.

Evaluation Criteria

Functionality:

- The system meets all specified requirements.
- Roles and permissions are correctly enforced.

Code Quality:

- Code follows industry standards and best practices.
- Proper use of design patterns and architecture.

Testing:

- Comprehensive unit tests and end-to-end tests.
- High test coverage and passing test cases.

Usability:

- User-friendly and responsive frontend.
- Clear and concise documentation.