Dean H - Library Management System Design Document

Introduction

The Library Management System is a web-based application designed to streamline the operations of a library, including cataloging books, managing reservations, and providing a user-friendly interface for users to discover and reserve books. Targeted users include library staff and library users each with distinct roles and interactions within the system.

Key Features:

- **Book Catalog:** Allows librarians to add, update, and delete books from the library catalog, including details such as title, author, genre, publication year, and description.
- **Book Reservation**: Enables users to search for books based on various criteria, such as title, author, genre, and publication year, and reserve books for checkout.
- **User Authentication:** Implements user authentication and authorization, ensuring that only authenticated users can perform certain actions, such as making reservations or editing book details.
- **Admin Panel:** Provides administrators with access to administrative features, such as managing user accounts, viewing reservation statistics, and generating reports.
- **User Profiles:** Allows users to create and manage their profiles, including personal information, reservation history, and account settings.

Benefits:

- **Efficient Book Management:** Simplifies the process of cataloging and managing books within the library, reducing manual effort and minimizing errors.
- **Improved User Experience:** Offers users a user-friendly interface to discover, search for, and reserve books, enhancing their overall experience and satisfaction.
- **Enhanced Accessibility:** Facilitates access to library resources by providing online reservation capabilities, allowing users to reserve books remotely at their convenience.
- **Data Insights:** Enables administrators to gain insights into library usage patterns, reservation trends, and popular book genres through built-in reporting and analytics features.

In summary, the Library Management System is a valuable tool for libraries to digitize their operations, improve user engagement, and optimize resource utilization.

System Architecture

The Library Management System follows a three-tier architecture, consisting of the presentation layer, the application layer, and the data layer. Each layer has specific responsibilities and interacts with the others to fulfill system requirements.

Presentation Layer

The presentation layer is responsible for interacting with users and displaying the user interface. It includes web pages, forms, and graphical elements that users interact with to perform actions such as searching for books, making reservations, and managing profiles. The presentation layer is implemented using Razor Pages, a feature of ASP.NET Core for creating dynamic web pages.

Application Layer

The application layer contains the business logic and processes user requests received from the presentation layer. It handles tasks such as validating user input, processing reservation requests, querying the database for book information, and applying business rules. This layer communicates with the data layer to retrieve and store information in the database. It includes components such as controllers, models, and services responsible for executing specific functions within the system.

Data Layer

The data layer manages the storage and retrieval of data from the database. It stores information related to books, users, reservations, and other system entities. The data layer uses a relational database management system (RDBMS) to store structured data efficiently. Entity Framework Core, an object-relational mapping (ORM) framework, is utilized to interact with the database, execute queries, and perform CRUD (Create, Read, Update, Delete) operations.

Integration with External Services

The system integrates with external services such as Microsoft Identity for user authentication and authorization. Microsoft Identity provides features for user management, role-based access control, and secure authentication mechanisms, enhancing the security and reliability of the system.

Scalability and Performance Considerations

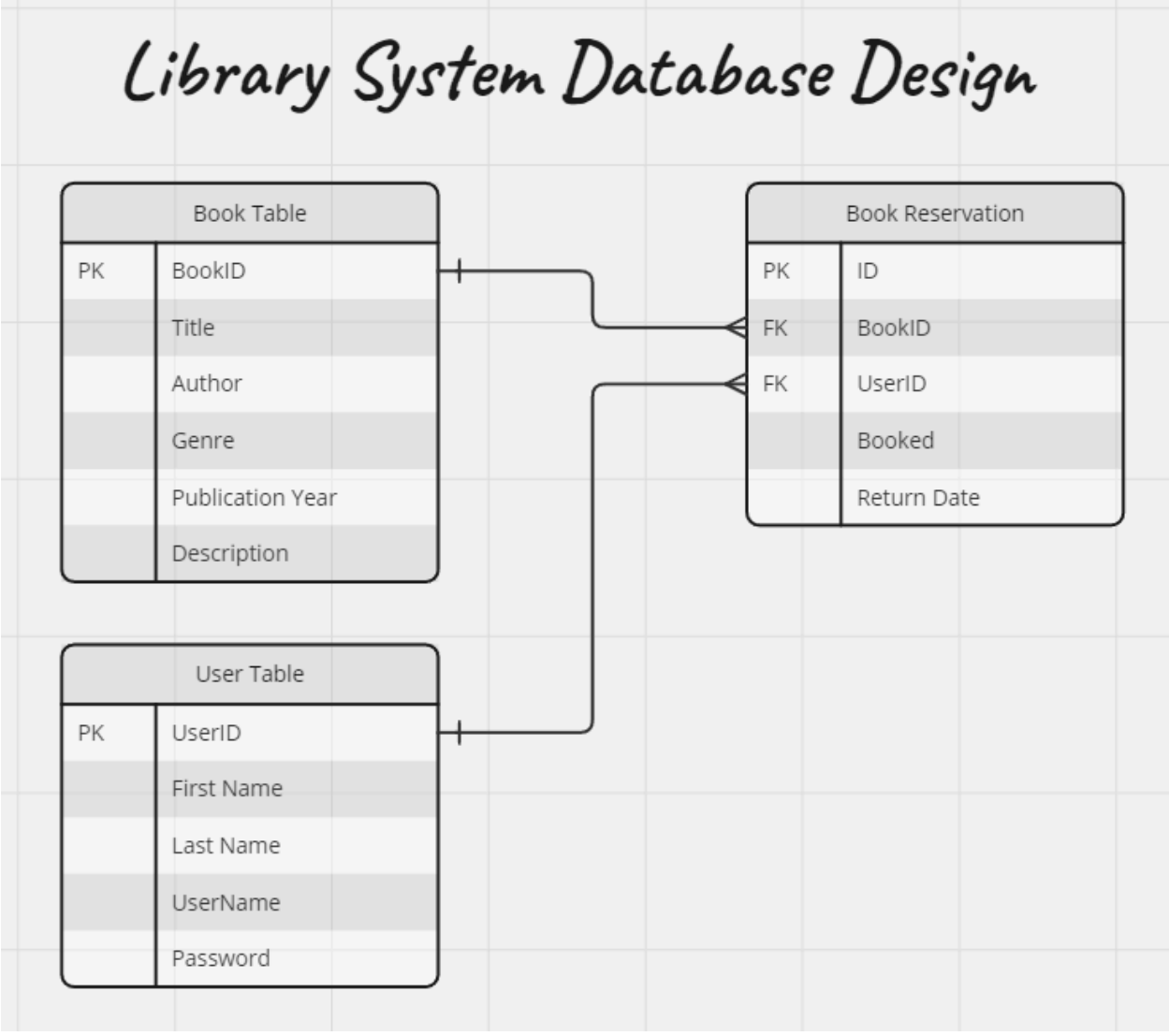
The system is designed to be scalable, allowing for future expansion to accommodate increased user traffic, data volume, and system complexity. Measures such as database indexing, caching, and asynchronous processing are implemented to optimize performance and ensure responsiveness, even under high load conditions.

Deployment Environment

The system can be deployed on cloud platforms such as Microsoft Azure. Continuous integration and continuous deployment (CI/CD) pipelines are utilized to automate the deployment process and ensure smooth releases and updates.

In summary, the three-tier architecture of the Library Management System provides a structured approach to system design, ensuring modularity, scalability, and maintainability while meeting the functional and non-functional requirements of the application.

Database Design



Components

The project is broken down into three parts: Model, View and Controllers.

Model

An example of an Model used in this project is Book Table

It has the properties of
ID - Unique Primary Key
Title - Book title
Author
Genre
PublicationYear
Description

Another Model is Book Reservations

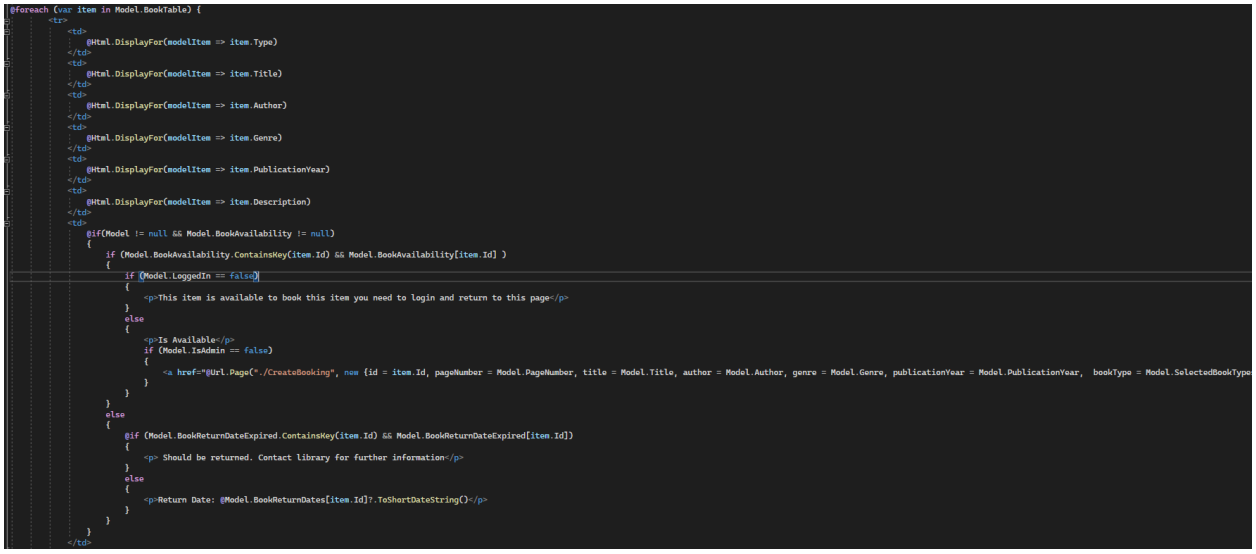
Which has the properties of
ID - Unique Primary key for each book reservation entry
BookID - The Secondary key ID from the Book Table to identify the book
UserId - The Secondary key ID from the User Table to identify the user
Booked - If the book is still booked out
ReturnDate - When the book should be returned

In the database, the relationship between Book Table and Book Reservations is one-to-many, where one book can have multiple bookings.

View

The View component utilizes Razor Pages along with HTML, CSS, and C# to create a dynamic and interactive user interface. Razor Pages handle both the structural markup of the page and the logic execution and rendering.

The View component is responsible for displaying data from multiple tables, handling data manipulation, and presenting the requested information in a user-friendly format.



```
@foreach (var item in Model.BookTable) {
    <tr>
        <td>
            @Html.DisplayFor(modelItem => item.Type)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Title)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Author)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Genre)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.PublicationYear)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Description)
        </td>
        <td>
            @if(Model != null && Model.BookAvailability != null)
            {
                if (Model.BookAvailability.ContainsKey(item.Id) && Model.BookAvailability[item.Id] )
                {
                    if (Model.LoggedIn == false)
                    {
                        <p>This item is available to book this item you need to login and return to this page</p>
                    }
                    else
                    {
                        <p>Is Available</p>
                        if (Model.IsAdmin == false)
                        {
                            <a href="@Url.Page("./CreateBooking", new {id = item.Id, pageNumber = Model.PageNumber, title = Model.Title, author = Model.Author, genre = Model.Genre, publicationYear = Model.PublicationYear,  bookType = Model.SelectedBookType
                        }
                    }
                }
                else
                {
                    @if (Model.BookReturnDateExpired.ContainsKey(item.Id) && Model.BookReturnDateExpired[item.Id])
                    {
                        <p> Should be returned. Contact library for further information</p>
                    }
                    else
                    {
                        <p>Return Date: @Model.BookReturnDates[item.Id]?.ToShortDateString()</p>
                    }
                }
            }
        </td>
```

Controllers

Controllers manage request handling, routing, and data manipulation in the application. They enforce validations on CRUD operations and handle authentication and error handling.

Key responsibilities of Controllers include routing requests, processing data from the database, and coordinating interactions between the View and Model components.

Overall, this architecture promotes separation of concerns, modularity, and maintainability in web application development.