BOOKSTORE WITH RECOMMENDATION

This project is a web-based bookstore application that allows users to browse, search for, and purchase books. It features a recommendation system that suggests books based on the user's browsing and purchase history, as well as ratings and reviews, books can also be recommended based on a form that the user fills out.

Features

- User Registration and Authentication: Users can sign up, log in, and manage their profiles.
- Book Catalog: Users can browse and search the catalog by various criteria (title, author, genre, etc.).
- Book Details: Detailed information about books, including descriptions, ratings, and reviews.
- Shopping Cart: Users can add books to their shopping cart and proceed to checkout.
- Recommendations: The system recommends books based on the user's preferences and previous interactions with the application, also based on a form.
- Admin Panel: Admin users can add, edit, or remove books, manage orders, and view user activities.

Data base

1. User Entity:

-A User can have one ShoppingCart. -A User can have multiple instances of UserShipping. -A User can have multiple instances of UserPayment. -A User can have multiple Orders. -A User can have multiple UserRoles.

2. ShoppingCart Entity:

-A ShoppingCart belongs to one User. -A ShoppingCart can contain multiple CartItems.

3. CartItem Entity:

-A CartItem is linked to one Book. -A CartItem is linked to one Order. -A CartItem can be a part of multiple BookToCartItems. -A CartItem is associated with one ShoppingCart.

4. Order Entity:

-An Order can contain multiple CartItems. -An Order is associated with one User. -An Order is linked to one ShippingAddress. -An Order is linked to one BillingAddress. -An Order is linked to one Payment.

5. Book Entity:

-A Book can be a part of multiple CartItems through BookToCartItem.

6. BookToCartItem Entity:

-A BookToCartItem associates a Book with a CartItem.

7. Payment Entity:

-A Payment is linked to one Order. -A Payment is linked to one UserBilling instance.

8. UserPayment Entity:

-A UserPayment belongs to one User. -A UserPayment is linked to one UserBilling.

9. UserBilling Entity:

-A UserBilling is associated with one UserPayment.

10. ShippingAddress Entity:

-A ShippingAddress is linked to one Order.

11. UserShipping Entity:

-A UserShipping belongs to one User.

These relationships are typically managed by JPA (Java Persistence API) using annotations like @OneToOne, @OneToMany, @ManyToOne, and @ManyToMany. The @JoinColumn annotation is used to define the actual foreign key in the database.

Technology Stack

- Spring Boot: For creating the web application and handling backend logic.
- Spring Data JPA: For database operations, making it easier to work with relational data.
- Thymeleaf: As the template engine for rendering views.
- MySQL: For the database to store book and user information.
- Spring Security: For authentication and authorization.
- Maven: For project management and build tool.

Backend

- Endpoints

POST /auth/register Description: Registers a new user with the provided registration information. Request Body: RegistrationBody Contains the registration information necessary for creating a new user account. Response: This endpoint does not explicitly return a response body. Upon successful registration, the user is registered in the system without any confirmation message. Errors during registration (e.g., user already exists) are handled through exceptions. Exceptions: UserAlreadyExistsException: Thrown if an attempt is made to register a user that already exists in the system.

-GET /auth/find Description: Finds a user based on the provided registration information. Request Body: RegistrationBody Contains the user information used for finding a user account. Response: This endpoint does not explicitly return a response body. It's assumed to perform some form of user lookup and handle the result internally or through exceptions. Note: Using @RequestBody in a GET request is unconventional and may not be supported by all clients or proxies.

-DELETE /auth/delete Description: Deletes a user based on the provided registration information. Request Body: RegistrationBody Contains the user information necessary for deleting a user account. Response: This endpoint does not explicitly return a response body. Upon successful deletion, the user is removed from the system without any confirmation message.

-PUT /auth/update Description: Updates a user's information based on the provided registration information. Request Body: RegistrationBody Contains the user information to be updated. Response: This endpoint does not explicitly return a response body.

Upon successful update, the user's information is updated in the system without any confirmation message. Models RegistrationBody The RegistrationBody model contains information about the user, which is used across various endpoints for registering, finding, deleting, and updating users. The specific fields and validation constraints of this model are not detailed in the provided code snippet.

Frontend

Bibliography