**Stage 1**

**Title: DrinkHub**

**Description:**

DrinkHub is a simplified online store specializing in selling a variety of beverages, including whiskeys, craft beers, wines, and non-alcoholic drinks. The platform will offer basic product listings, customer reviews, and a simple shopping cart for easy checkout.

**Features:**

1. **User Accounts**: Customers can create accounts to manage their orders and reviews.
2. **Product Listings**: Pages for each product, including descriptions and customer reviews.
3. **Search and Filtering**: Basic search functionality to find drinks by name.
4. **Shopping Cart and Checkout**: A simple shopping cart experience with basic payment options.

**Technology Stack:**

* **Backend**: Java, Spring Boot, Spring Data JPA
* **Database**: Postgres
* **Frontend**: Thymeleaf, HTML, CSS
* **Security**: Spring Security
* **Build Tool**: Maven

**Simplified Project Idea 2: BookBazaar**

**Title: BookBazaar**

**Description:**

BookBazaar is a simplified online bookstore offering a selection of books across various genres. The platform will feature basic book listings, customer reviews, and a simple shopping cart for purchasing books.

**Features:**

1. **User Accounts**: Users can create accounts to manage their orders and reviews.
2. **Book Listings**: Pages for each book, including descriptions and customer reviews.
3. **Search and Filtering**: Basic search functionality to find books by title or author.
4. **Shopping Cart and Checkout**: A simple shopping cart experience with basic payment options.
5. **Review System**: Users can leave reviews and ratings for books.

**Technology Stack:**

* **Backend**: Java, Spring Boot, Spring Data JPA
* **Database**: PostgreSQL (production)
* **Frontend**: Thymeleaf, HTML, CSS
* **Security**: Spring Security
* **Build Tool**: Maven

Github: https://github.com/Darbi23/drinkhub/tree/main

Trello: <https://trello.com/b/wMwehMSh/drinkhub>

**Stage 2**

**Functional Requirements**

1. **User Registration and Authentication**: Users should be able to create accounts, log in, and log out securely.
2. **Product Management**:
   * Admin users should be able to add, edit, and delete products.
   * Products should include attributes such as name, description, price, category, and reviews.
3. **Product Browsing and Searching**:
   * Users should be able to browse products by categories.
   * Users should be able to search for products by name.
   * Users should be able to sort product by price
   * Users should be able to sort product by name(alphabetically)
   * Users should be able to use pagination.
4. **Shopping Cart**:
   * Users should be able to add products to a shopping cart.
   * Users should be able to view and modify their cart.
   * Users should be able to remove products from the cart.
5. **Checkout and Payment**:
   * Users should be able to proceed to checkout and enter shipping information.
   * Users should be able to make payments.
6. **Order Management**:
   * Users should be able to view their order history.
   * Admin users should be able to view and manage all orders.

**Non-Functional Requirements**

1. **Performance**:
   * The system should handle up to 1000 concurrent users.
   * Page load times should not exceed 3 seconds under normal load.
2. **Security**:
   * The system should protect against common security threats such as SQL injection and cross-site scripting (XSS).
3. **Usability**:
   * The interface should be intuitive and easy to navigate.
   * Users should be able to complete key tasks (e.g., registration, purchase) with minimal steps.

**Step 2: Design Use Cases for the System**

1. **User Registration and Authentication**:
   * **Actors**: User
   * **Description**: Users register for an account, log in, and log out.
   * **Preconditions**: None
   * **Postconditions**: User account is created, user is logged in, or user is logged out.
2. **Product Management**:
   * **Actors**: Admin
   * **Description**: Admin users add, edit, and delete products.
   * **Preconditions**: Admin user is logged in.
   * **Postconditions**: Product information is updated in the system.
3. **Product Browsing and Searching**:
   * **Actors**: User
   * **Description**: Users browse products by category or search by name.
   * **Preconditions**: None
   * **Postconditions**: Products matching the search criteria are displayed.
4. **Shopping Cart**:
   * **Actors**: User
   * **Description**: Users add products to their cart, view, and modify their cart.
   * **Preconditions**: User is logged in.
   * **Postconditions**: Shopping cart is updated.
5. **Checkout and Payment**:
   * **Actors**: User
   * **Description**: Users proceed to checkout, enter shipping information, and make payments.
   * **Preconditions**: User is logged in and has items in the cart.
   * **Postconditions**: Order is placed and payment is processed.
6. **Order Management**:
   * **Actors**: User, Admin
   * **Description**: Users view their order history; admins manage all orders.
   * **Preconditions**: User or admin is logged in.
   * **Postconditions**: Order details are displayed or updated.

**Step 3: Identify Objects, Classes, and Relationships in the System**

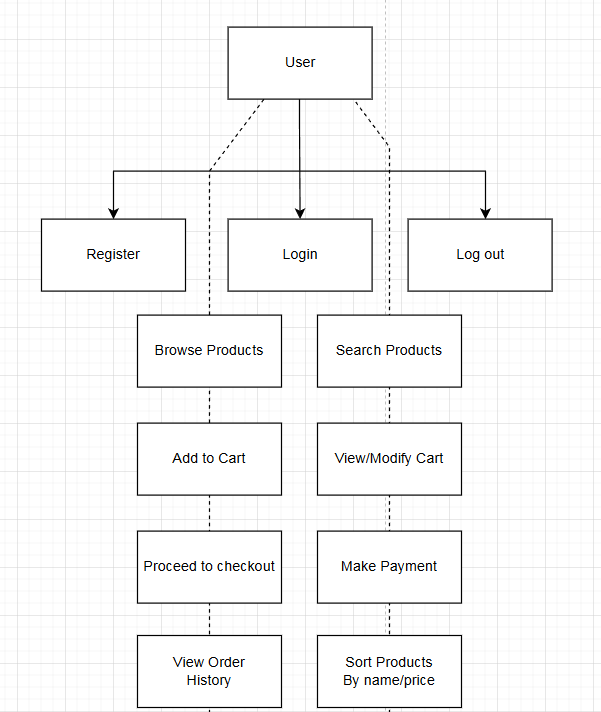
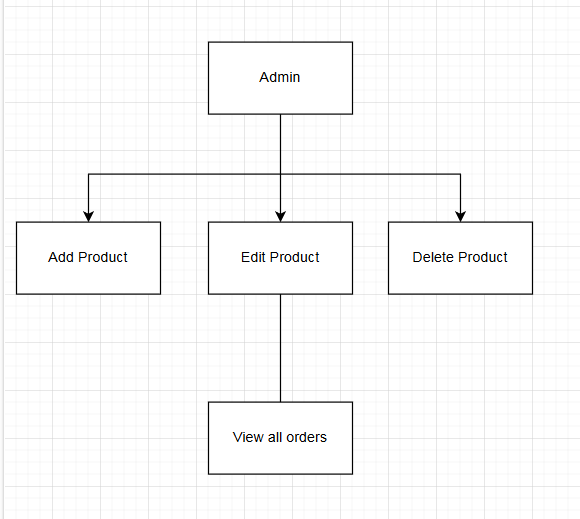
**Objects and Classes**

1. **User**:
   * Attributes: id, username, password, email, roles
2. **Product**:
   * Attributes: id, name, description, price, category, reviews
3. **Review**:
   * Attributes: id, userId, productId, rating, comment
4. **Order**:
   * Attributes: id, userId, productList, totalAmount, orderDate, status
5. **Cart**:
   * Attributes: userId, productList
6. **Category**:
   * Attributes: id, name

**Relationships**

* **User** has a one-to-many relationship with **Order**.
* **Product** has a many-to-one relationship with **Category**.
* **Order** has a many-to-many relationship with **Product**.
* **Cart** has a many-to-many relationship with **Product**.
* **User** has a one-to-many relationship with **Cart.**

**Step 4: Design Class Diagrams**



**Stage 3**

**Entities**:

1. **User**: Represents users of the system.
2. **Role**: Enum to represent user roles.
3. **Category**: Represents categories of products.
4. **Product**: Represents products available in the store.
5. **CustomerOrder**: Represents orders made by users.
6. **OrderProduct**: Represents the associative entity between CustomerOrder and Product (to handle many-to-many relationships).
7. **Address**: Represents addresses for users.
8. **Payment**: Represents payment information for orders.

**Updated Relationships Between Entities**

**Relationships**:

1. **User** and **Order**:
   * One User can place many Orders.
   * One Order is placed by one User.
2. **Order** and **Product** (via OrderProduct):
   * One Order can contain many Products.
   * One Product can be part of many Orders.
3. **User** and **Address**:
   * One User can have many Addresses.
   * One Address belongs to one User.
4. **Order** and **Payment**:
   * One Order can have one Payment.
   * One Payment is associated with one Order.
5. **Product** and **Category**:
   * One Category can have many Products.
   * One Product belongs to one Category.

**Updated Cardinality and Ordinality of the Relationships**

**Cardinality**:

1. **User** to **Order**: One-to-Many (1)
2. **Order** to **Product**: Many-to-Many (M)
3. **User** to **Address**: One-to-Many (1)
4. **Order** to **Payment**: One-to-One (1:1)
5. **Category** to **Product**: One-to-Many (1)

**Attributes for the Entities**

**Attributes**:

**User**:

* id: Primary Key
* username: Unique, Not Null
* password: Not Null
* email: Unique, Not Null
* roles: Enum, Not Null

**Role** (Enum):

* ADMIN
* USER

**Category**:

* id: Primary Key
* name: Not Null

**Product**:

* id: Primary Key
* name: Not Null
* description
* price: Not Null
* categoryId: Foreign Key, Not Null

**Order**:

* id: Primary Key
* userId: Foreign Key, Not Null
* totalAmount: Not Null
* orderDate: Not Null
* status

**OrderProduct** (Associative Entity):

* orderId: Foreign Key, Not Null
* productId: Foreign Key, Not Null

**Address**:

* id: Primary Key
* userId: Foreign Key, Not Null
* street: Not Null
* city: Not Null
* state
* postalCode: Not Null
* country: Not Null

**Payment**:

* id: Primary Key
* orderId: Foreign Key, Not Null
* amount: Not Null
* paymentDate: Not Null
* paymentMethod: Not Null

