Name : Saad Humayun
RegNo: Fa22-bse-012

Case Study: Bookstore System

Project Overview:

The **Bookstore** project is a software application developed to manage various aspects of an online bookstore. This system allows customers to browse, search, and purchase books, while also enabling administrative functions like managing inventory and updating order statuses. The application is designed with a three-layer architecture (Presentation, Business Logic, and Data Access), ensuring separation of concerns and making the system modular and easy to maintain. Additionally, the project applies the GRASP (General Responsibility Assignment Software Patterns) principles to ensure clear responsibility assignment across different components.

Core Features/Use Cases:

The system is designed to cater to two types of users: **customers** and **admins**. The major use cases in the project are as follows:

Customer Use Cases:

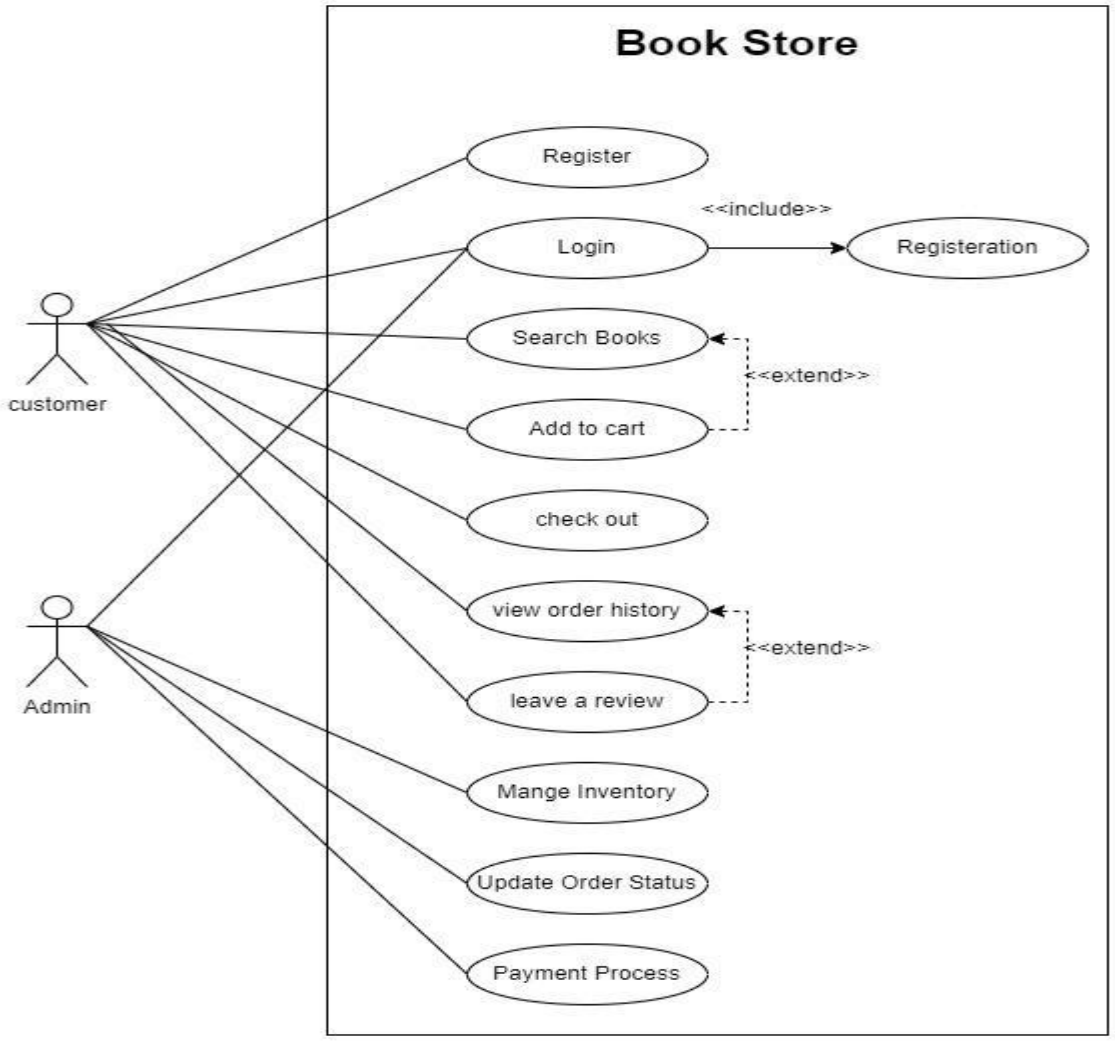
1. **Register**: Allows new customers to create an account.
2. **Login**: Provides access to the system for existing customers, including secure login validation.
3. **Search Books**: Enables customers to search for books by title, author, or ISBN.
4. **Add to Cart**: Adds selected books to the shopping cart for future purchase.
5. **Checkout**: Completes the purchase process and generates an order.
6. **View Order History**: Allows customers to view their past orders.
7. **Leave a Review**: Customers can leave feedback or reviews on purchased books.

Admin Use Cases:

1. **Manage Inventory**: Enables admins to add, update, or remove books from the inventory.
2. **Update Order Status**: Allows admins to change the status of orders, such as marking them as "shipped" or "delivered."
3. **Payment Process**: Manages payment confirmations and transactions within the bookstore.

Each of these use cases was outlined in the **Use Case Diagram**, which helped identify key interactions between customers, admins, and the system's primary functions.

2. UseCase Diagram:



---

3. Selected Use Case: Search Books

Description:
The **Search Books** use case allows a **Customer** to find books in the bookstore's inventory by entering keywords such as title, author, or genre. The system then retrieves and displays relevant results for the customer to view.

Primary Actor:
Customer

Pre-Conditions:

- Customer must be on web or app.

Main Success Scenario:

1. The customer enters a title or name in the search field.
2. The system retrieves matching book from the inventory.
3. The customer views the results list.

Post-Conditions:

- The system display matching book, or show "No results found" if no match.

---

4. Three-Layer Architecture

I. Data Access Layer (DAL):

- **Files**: `Book.java` and `BookDataAccess.java`
- **Purpose**: This layer is responsible for managing data. The `Book` class represents the data structure, holding book details like title, author, and ISBN. `BookDataAccess` is where the data is stored and accessed, including the method `searchBook()` to search books by title.
- **Role**: Encapsulates and manages data, separating data operations from the business and UI layers.

II. Business Logic Layer (BLL):

- **File**: `BookService.java`
- **Purpose**: Contains the core business logic. In `BookService`, you have `searchBookByTitle()`, which interacts with `BookDataAccess` to retrieve book data and prepares it for the UI.
- **Role**: Acts as the intermediary, ensuring data integrity and applying business rules.

III. Presentation Layer (PL):

- **File**: `SearchBookUI.java`
- **Purpose**: This layer manages user interactions. It allows the user to search for a book, triggering a call to `BookService` and displaying the result in the user interface.
- **Role**: Provides a user-friendly interface for book searches, isolating user interaction logic from business and data logic.

---

5. Application of GRASP Principles

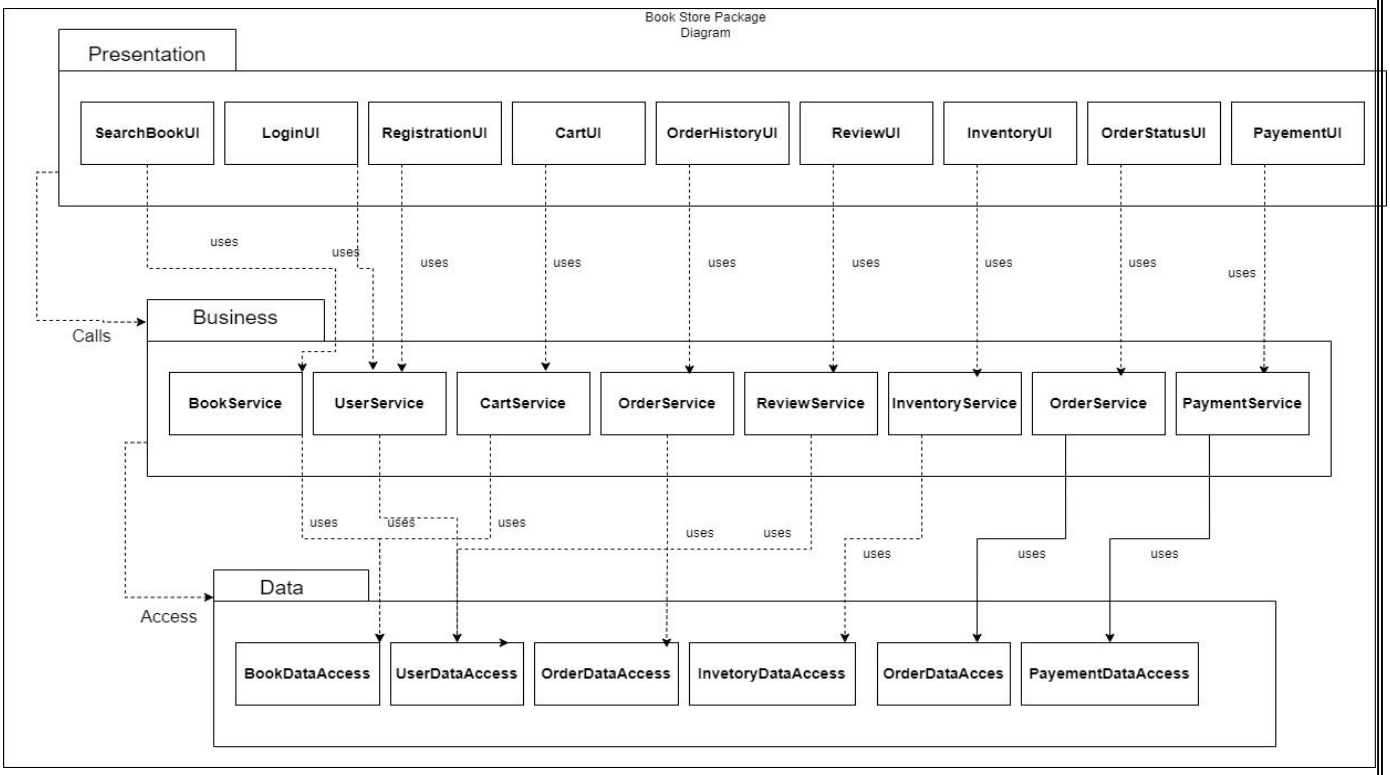
Information Expert:

- **Where**: In the `Book` and `BookDataAccess` classes.
- **Explanation**: The `Book` class is responsible for storing its own information, making it the "expert" on book data. `BookDataAccess` is the expert on managing the collection of books, as it knows how to store and search through book data.

Controller:

- **Where**: `BookService` acts as the controller in this project.
- **Explanation**: `BookService` handles the flow between the UI and data layers. It accepts the user input, processes it (e.g., searches for a book), and then returns the result to the UI.

6. Package Diagram:



Package Diagram Explanation:

The **Package Diagram** for the Bookstore project visually represents the architecture's structure and the relationships between different layers. Here's a summary:

Presentation Layer:

1. Contains UI packages such as `SearchBookUI`, `LoginUI`, `RegistrationUI`, and more.
2. Each UI package is responsible for handling specific user interactions and uses the respective services from the **Business Logic Layer**.

Business Logic Layer:

1. Includes services like `BookService`, `UserService`, `CartService`, `OrderService`, etc.
2. Each service class connects to the **Data Access Layer** to retrieve or update data based on the user's request.
3. The services encapsulate the main functionality and business rules, following GRASP principles to assign responsibilities appropriately.

Data Access Layer:

1. Consists of data access classes such as `BookDataAccess`, `UserDataAccess`, and `OrderDataAccess`.
2. These classes are responsible for accessing data stored in CSV files and are only accessed by the **Business Logic Layer**.
3. This separation allows flexibility in replacing the CSV data store with a different database if needed, without affecting other layers.

Implementation Highlights:

In the implementation, a **three-layer approach** ensures the project is well-structured and each part can be independently modified or expanded. Using Java, classes were structured in packages that align with each layer:

- `data` package for the **Data Access Layer**,
- `business` package for the **Business Logic Layer**, and
- `presentation` package for the **Presentation Layer**.