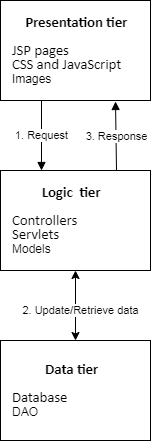
**2. Architecture description**

**Overall Architecture**

The application is use 3-tier architecture pattern, which is consist of Presentation tier, Logic tier and Data tier. The Presentation tier contains the front-end functions which responsible for user interaction and presentation, Logic tier handle user input data in Presentation tier and run Business Logic then save or retrieve data in Data tier, Data tier save business data, 3-tier architecture pattern can ensure clear separation between front and back-end.

**Architecture Diagram**

****

Use Case: Register, Login and browse Catalog, Add to Cart

1. Customer open index page to register.

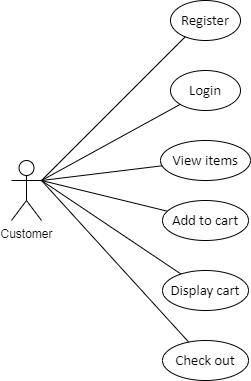
2. Login the system.

3. View catalog items page.

4. Add items to cart.

5. Display items in the cart and bill.

6. Check out to purchase the items in the shopping cart.



**3. Design description**

**Design decisions and trade-offs**

Design pattern: MVC and DAO.

MVC design pattern is divide system in 3 tiers,

**Benefits of the MVC Design pattern:**

1. Modularity and Separation of Concerns: The primary benefits of MVC is its clear separation of concerns, it allows developers to work on different components without affecting others.

2. Reusability: Because of the separation of concerns, components can be reused in other tier, like model can be used in different views.

3. Ease of Maintenance: With clear divisions between components, maintenance becomes easier. Bug fixes, updates, or enhancements can be made to one component without affecting the others, minimizing the risk of unintended consequences.

4. Parallel Development: Multiple developers can work simultaneously on different tiers without conflicts, speeding up the development process.

5. Scalability: Different tier can be deployed in different servers, can scale up certain tier without affecting other tiers.

6. Flexibility in User Interface: The separation of the view and data make it easy to display the same data in different format.

**Trade-offs of the MVC Design pattern:**

1. Complexity: Developers need to understand how components interact with each other and how they communicate with each other.

2. Increased Development Time: It spend more time in the early stages because developers need to create individual components and design interface between them.

3. Overhead: Separation of tiers lead to additional overhead in communication between components.

4. Maintenance Challenges: It hard to maintenance code when the interactions between components become complex.

The DAO design pattern is used to separate the data access code from the rest of the business logic of the application.

**Benefits of the DAO Design pattern:**

1. Separation of Concerns: It separates database interaction and query logic from business logic, ensuring that changes to the database schema do not directly affect business logic.

2. Code Reusability: DAO encapsulate data access logic and can be reused in the rest of the application code.

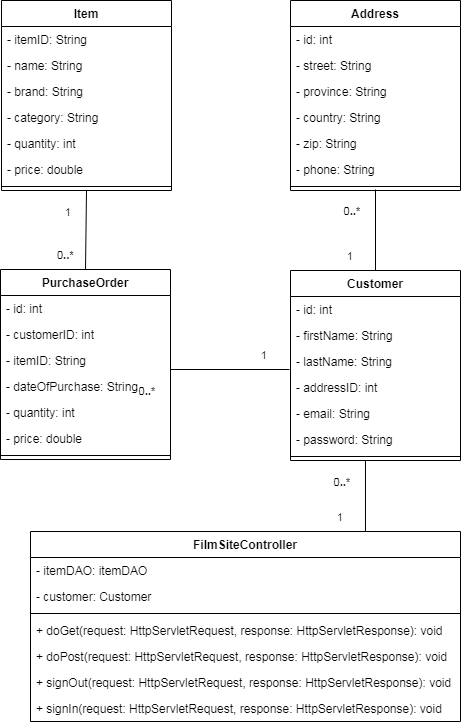
3. Flexibility: DAO provide access interface to database, it can easy change database instance without affect the rest of the application code.

**Trade-offs of the DAO Design pattern:**

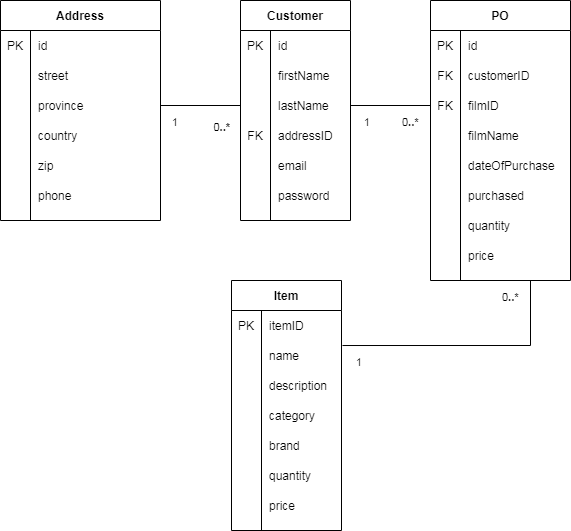
1. Complexity: Implementing the DAO pattern can introduce additional complexity, especially in smaller applications, where this separation may not provide significant benefits.

2. Development Overhead: Introducing DAO requires additional development effort to make access interface with the Logic tier.

**UML package/class diagram**



**Database schema diagrams**

****

**4. Advanced Features**

Prevent SQL injection by use PreparedStatement instead of Statement to execute the sql query,

PreparedStatement not use concatenating user input data as sql command, it use setter methods to set parameterized value.