**Case Study: Enhancing a Flutter Shop App**

**Background**

Company Y is developing an **e-commerce app** using Flutter. However, the current implementation has issues with product management, order processing, and user interactions. The goal is to identify these issues, propose solutions, and establish a **domain model, sequence diagram, and ERD diagram** for an improved version.

**Domain Model**

**Entities & Attributes**

* **Product**
  + id (int): Unique identifier
  + name (String): Product title
* **Category**
  + id (int): Unique identifier
  + name (String): Category name
* **User**
  + id (int): Unique identifier
  + username (String): Login name
* **Order**
  + id (int): Unique identifier
  + totalAmount (double): Order total
* **OrderItem**
  + id (int): Unique identifier
  + quantity (int): Ordered quantity

**Cardinality**

* A **Category** can have multiple **Products**, but each **Product** belongs to only one **Category**.
* A **User** can place multiple **Orders**, but each **Order** is linked to only one **User**.
* An **Order** contains multiple **OrderItems**, but each **OrderItem** belongs to a single **Order**.
* Each **OrderItem** is associated with exactly one **Product**, and a **Product** can appear in multiple **OrderItems**.
* A **User** can add multiple **Products** to their cart, but each **Product** can belong to multiple Users' carts.

**Relationships**

 **A Category contains multiple Products, where each Product can belong to a Category but can also exist independently of it.**

 **Each OrderItem references exactly one Product, which provides the details of the item. If an OrderItem is deleted, its Product reference is also removed, as the Product is tightly bound to the OrderItem.**

 **A User places multiple Orders, meaning a User can create many Orders, and each Order must have a User who created it.**

 **A User can add multiple Products to their cart, and each Product can appear in multiple Users' carts, forming a many-to-many association.**

**Sequence Diagram**

**Actors & Objects:**

* **User**
* **App Controller**
* **Authentication System**
* **Cart System**
* **Payment Gateway**
* **Order Management**

**Flow:**

**1. User Authentication (Login/Register)**

1. **User → App Controller: Requests to log in or register.**
2. **App Controller → Authentication System: Sends authentication request (credentials for login or new account details for registration).**
3. **Authentication System → App Controller: Returns authentication success or failure response.**
4. **App Controller → User: Notifies user of success or failure.**

***(If login is successful, proceed to shopping; otherwise, user retries.)***

**2. Shopping & Checkout Process**

1. **User → App Controller: Adds product to the cart.**
2. **App Controller → Cart System: Sends request to update cart.**
3. **Cart System → App Controller: Confirms cart update.**
4. **User → App Controller: Proceeds to checkout.**

**3. Payment Process**

1. **App Controller → Payment Gateway: Requests available payment methods.**
2. **Payment Gateway → App Controller: Returns payment options.**
3. **User → App Controller: Selects a payment method.**
4. **App Controller → Payment Gateway: Sends payment request.**
5. **Payment Gateway → App Controller: Returns payment success or failure response.**
6. **App Controller → Order Management: Sends order confirmation request if payment is successful.**
7. **Order Management → App Controller: Confirms order storage.**
8. **App Controller → User: Notifies order confirmation or payment failure.**

**Entity-Relationship Model (ERD)**

**Entities & Relationships:**

* **Product (1..\*) → (1) Category**
  + A category can have multiple products, but a product belongs to only one category.
* **User (1) → (0..\*) Order**
  + A user can place multiple orders, but each order is linked to only one user.
* **Order (1) → (1..\*) OrderItem**
  + An order consists of multiple order items, but an order item is linked to a single order.
* **OrderItem (1) → (1) Product**
  + Each order item references exactly one product, and a product can appear in multiple order items.
* **User (M) → (M) Product (through Cart)**
  + A user can add multiple products to their cart, and each product can appear in multiple users' carts.