# Assignment Case Study: Inventory Management System

**Problem Statement:** Inventory Management System

**Objective**: Design and implement an Inventory Management System using C# that demonstrates Object-Oriented Programming (OOP) concepts, specifically polymorphism, is/as operators, and interfaces. The system should manage a collection of products, support inventory operations like restocking and selling, and generate reports with product details. The solution should be suitable for beginner to intermediate learners, providing a clear and practical application of these OOP concepts.

### Requirements:

#### 1. System Overview:

- The system must manage different types of products (e.g., electronics and groceries) in an inventory.

- Each product type has unique attributes (e.g., warranty for electronics, expiry date for groceries) and common attributes (e.g., name, price, stock quantity).

- The system should support operations like adding products, restocking, selling, and displaying an inventory report.

#### 2. OOP Concepts to Demonstrate:

- Polymorphism: Allow different product types to provide specific implementations of a common method (e.g., displaying product details) using method overriding.

- is/as Operators: Use type checking and safe casting to handle product-specific attributes when generating reports or performing operations.

- Interfaces: Define a contract for inventory operations (e.g., restocking, selling) that all product types must follow, enabling flexible and extensible code.

#### 3. Functional Requirements:

- Product Management:

- Create a base `Product` class with common properties (e.g., Name, Price, Stock).

- Implement derived classes for specific product types (e.g., `Electronic` with warranty, `Grocery` with expiry date).

- Support polymorphic behavior for displaying product details.

- Inventory Operations:

- Define an interface (e.g., `IInventoryItem`) with methods for restocking, selling, and retrieving product details.

- Implement these methods in each product type to update stock and provide appropriate feedback.

- Inventory Manager:

- Create a class (e.g., `InventoryManager`) to manage a collection of products.

- Support adding products to the inventory.

- Allow performing operations (restock, sell) on products via the interface.

- Generate a report displaying details of all products, using type checking and casting to handle specific attributes.

- Type Checking and Casting:

- Use the `is` operator to check the type of a product.

- Use the `as` operator for safe casting to access type-specific properties.

#### 4. Constraints:

- Ensure stock quantities cannot go negative during sales.

- Validate inputs (e.g., restock/sell quantities must be positive).

- Keep the code simple and well-structured for educational purposes, avoiding overly complex features.

#### 5. Expected Output:

- Display confirmation messages for adding products, restocking, and selling.

- Generate an inventory report showing details of all products, including type-specific attributes (e.g., warranty for electronics, expiry date for groceries).

#### 6. Extensibility:

- The system should allow easy addition of new product types (e.g., clothing, books) without modifying existing code, leveraging interfaces and polymorphism.

**Example Scenario:**

- Add a laptop (electronic, $999.99, 10 units, 24-month warranty) and milk (grocery, $3.99, 50 units, expiry 2025-08-01) to the inventory.

- Restock 5 laptops and sell 3 laptops.

- Sell 10 units of milk.

- Display an inventory report showing updated stock and product-specific details.