***Team members:***

| Name | Id |
|---|---|
| **1-Shehab Ahmed Soliman Magdy** | **21-01626** |
| **2-Manar Ebrahim Ahmed** | **21-00961** |
| **3-Menna Salah Abdelhamed** | **21-01075** |
| **4-Menna Mahmoud Mohamed** | **21-01069** |
| **5-Shahd Nady Sayed** | **21-01615** |
| **6- Tassbeeh Hassan Khalaf** | **21-01666** |

# Project:

**An Inventory Management System (IMS):** is a software application designed to efficiently track, organize, and manage a company's inventory levels, orders, sales, and deliveries. It serves as a critical tool for businesses to maintain optimal stock levels, reduce waste, and improve supply chain efficiency.

When using **design patterns** in an **Inventory Management System (IMS)**, each pattern serves a specific purpose to make the system more modular, maintainable, and scalable.

## Here's a concise summary of what each of the design patterns :

### 1. Singleton Pattern:

- **Purpose:** Ensures that a class has only one instance and provides a global point of access to it.
- **Key Use Case**: Managing shared resources, such as a configuration manager or a database connection

### 2. Factory Pattern:

- **Purpose:** Provides a way to create objects without specifying their exact class.
- **Key Use Case:** Centralizing and simplifying object creation logic, often based on input parameters or conditions.

### 3. Proxy Pattern:

- **Purpose:** Acts as an intermediary to control access to an object, adding additional functionality like security or lazy initialization.
- **Key Use Case:** Controlling access to sensitive resources, optimizing performance, or managing remote objects.

### 4. Decorator Pattern:

- **Purpose:** Dynamically adds or modifies the behavior of an object without changing its structure.
- **Key Use Case:** Extending object functionality (e.g., adding features like discounts or logging) while keeping the core class unchanged.

### 5. Observer Pattern:

- **Purpose:** Defines a dependency between objects so that when one object changes state, its dependents (observers) are notified.
- **Key Use Case:** Implementing event-driven systems (e.g., notifying users of stock changes or order updates).

## The five patterns we used in our project:

**1-Singleton:** for Database Connection Manager.

**2-Factory:** used in product factory to create object kind of product.

**3-proxy:** used in discount to make discount or not.

**4- Observer:** Represents a product in the inventory system. Implements the Observer designpattern to notify observers when the stock quantity changes.

**5-Decrator:** used in product decorator for adding discount to product

## The Classes we used:

- **Class ConnectionProvider:** Establish the connection to the database.
- **Class InventoryManager:** Add product to the database.
- **interface Discount**: has method to applyDiscount.
- **Class InventoryGUI:** Features of InventoryGUI is adding product based on product details
- **Class Product:** has product details.
- **Class ProductDecorator & Discount product :** they extends from product and implement Decorator design pattern for Adding discount to product.
- **Class ProductFactory:** to create a new product instance
- **Class RealDiscount:** performs actual discount calculation.
- **Class DiscountProxy:** Proxy class that controls access to the real discount.