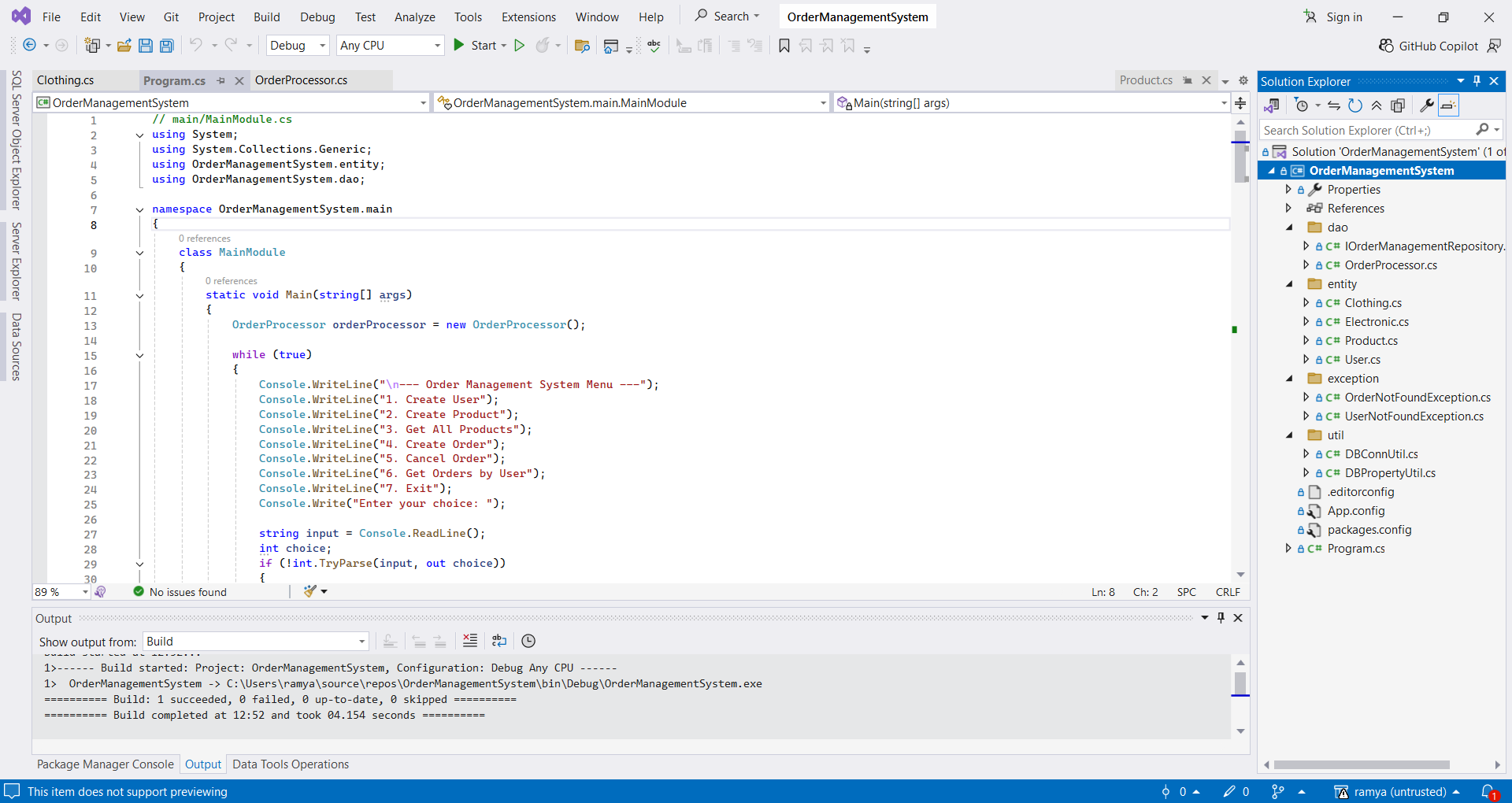
## **ORDER MANAGEMENT SYSTEM – PROJECT CODE**

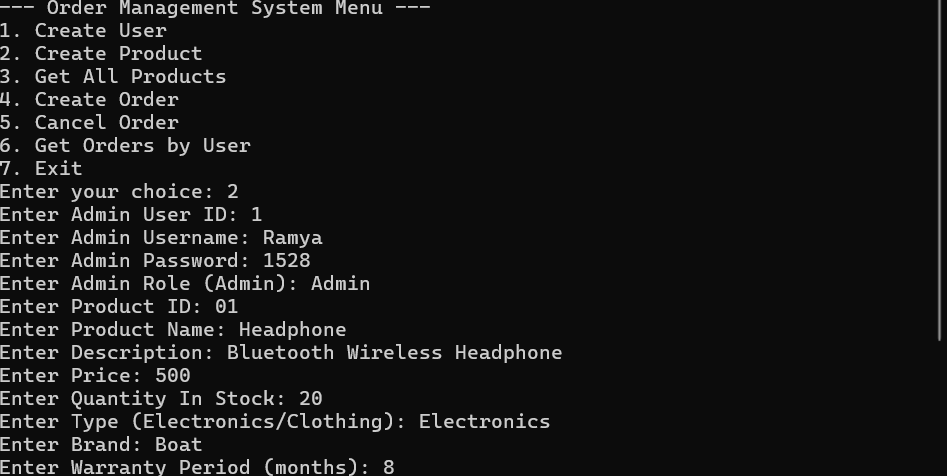
### **Student Name: *Ramya Arumugam***

### **Date: 27-06-2025**

### **Company: Hexaware Technologies Coding Challenge**

Output





CREATE DATABASE OrderDB;

USE OrderDB;

CREATE TABLE Users (

UserId INT PRIMARY KEY,

Username VARCHAR(100),

Password VARCHAR(100),

Role VARCHAR(50) -- 'Admin' or 'User'

);

CREATE TABLE Products (

ProductId INT PRIMARY KEY,

ProductName VARCHAR(100),

Description VARCHAR(200),

Price FLOAT,

QuantityInStock INT,

Type VARCHAR(50)

);

CREATE TABLE Electronics (

ProductId INT PRIMARY KEY,

Brand VARCHAR(100),

WarrantyPeriod INT,

FOREIGN KEY (ProductId) REFERENCES Products(ProductId)

);

CREATE TABLE Clothing (

ProductId INT PRIMARY KEY,

Size VARCHAR(20),

Color VARCHAR(50),

FOREIGN KEY (ProductId) REFERENCES Products(ProductId)

);

CREATE TABLE Orders (

OrderId INT IDENTITY(1,1) PRIMARY KEY,

UserId INT,

FOREIGN KEY (UserId) REFERENCES Users(UserId)

);

CREATE TABLE OrderItems (

OrderItemId INT IDENTITY(1,1) PRIMARY KEY,

OrderId INT,

ProductId INT,

FOREIGN KEY (OrderId) REFERENCES Orders(OrderId),

FOREIGN KEY (ProductId) REFERENCES Products(ProductId)

);

// ---------- App.config ----------

<?xml version="1.0" encoding="utf-8" ?>

<configuration>

<connectionStrings>

<add name="DbConnection"

connectionString="Data Source=localhost;Initial Catalog=OrderDB;Integrated Security=True"

providerName="System.Data.SqlClient" />

</connectionStrings>

</configuration>

// ---------- util\DBPropertyUtil.cs ----------

using System.Configuration;

namespace OrderManagementSystem.util

{

public class DBPropertyUtil

{

public static string GetConnectionString(string name)

{

return ConfigurationManager.ConnectionStrings[name].ConnectionString;

}

}

}

//---------- util\DBConnUtil.cs ----------

using System.Data.SqlClient;

namespace OrderManagementSystem.util

{

public class DBConnUtil

{

public static SqlConnection GetConnection()

{

string connStr = DBPropertyUtil.GetConnectionString("DbConnection");

return new SqlConnection(connStr);

}

}

}

// ---------- entity\Product.cs ----------

namespace OrderManagementSystem.entity

{

public class Product

{

public int ProductId { get; set; }

public string ProductName { get; set; }

public string Description { get; set; }

public double Price { get; set; }

public int QuantityInStock { get; set; }

public string Type { get; set; }

public Product() { }

public Product(int id, string name, string desc, double price, int qty, string type)

{

ProductId = id;

ProductName = name;

Description = desc;

Price = price;

QuantityInStock = qty;

Type = type;

}

}

}

// ---------- entity\Electronics.cs ----------

using OrderManagementSystem.entity;

namespace OrderManagementSystem.entity

{

public class Electronics : Product

{

public string Brand { get; set; }

public int WarrantyPeriod { get; set; }

public Electronics(int id, string name, string desc, double price, int qty, string type, string brand, int warranty)

: base(id, name, desc, price, qty, type)

{

Brand = brand;

WarrantyPeriod = warranty;

}

}

}

// ---------- entity\Clothing.cs ----------

using OrderManagementSystem.entity;

namespace OrderManagementSystem.entity

{

public class Clothing : Product

{

public string Size { get; set; }

public string Color { get; set; }

public Clothing(int id, string name, string desc, double price, int qty, string type, string size, string color)

: base(id, name, desc, price, qty, type)

{

Size = size;

Color = color;

}

}

}

// ---------- entity\User.cs ----------

namespace OrderManagementSystem.entity

{

public class User

{

public int UserId { get; set; }

public string Username { get; set; }

public string Password { get; set; }

public string Role { get; set; }

public User(int id, string uname, string pwd, string role)

{

UserId = id;

Username = uname;

Password = pwd;

Role = role;

}

}

}

// ---------- exception\UserNotFoundException.cs ----------

using System;

namespace OrderManagementSystem.exception

{

public class UserNotFoundException : Exception

{

public UserNotFoundException(string message) : base(message) { }

}

}

// ---------- exception\OrderNotFoundException.cs ----------

using System;

namespace OrderManagementSystem.exception

{

public class OrderNotFoundException : Exception

{

public OrderNotFoundException(string message) : base(message) { }

}

}

// ---------- dao\IOrderManagementRepository.cs ----------

using System.Collections.Generic;

using OrderManagementSystem.entity;

namespace OrderManagementSystem.dao

{

public interface IOrderManagementRepository

{

void CreateUser(User user);

void CreateProduct(User admin, Product product);

void CreateOrder(User user, List<Product> products);

void CancelOrder(int userId, int orderId);

List<Product> GetAllProducts();

List<Product> GetOrderByUser(User user);

}

}

//----------dao\OrderProcessor.cs-----------------

// dao/OrderProcessor.cs

using System;

using System.Collections.Generic;

using System.Data.SqlClient;

using OrderManagementSystem.entity;

using OrderManagementSystem.exception;

using OrderManagementSystem.util;

namespace OrderManagementSystem.dao

{

public class OrderProcessor : IOrderManagementRepository

{

SqlConnection conn = DBConnUtil.GetConnection();

public void CreateUser(User user)

{

try

{

conn.Open();

string query = "INSERT INTO Users (UserId, Username, Password, Role) VALUES (@UserId, @Username, @Password, @Role)";

SqlCommand cmd = new SqlCommand(query, conn);

cmd.Parameters.AddWithValue("@UserId", user.UserId);

cmd.Parameters.AddWithValue("@Username", user.Username);

cmd.Parameters.AddWithValue("@Password", user.Password);

cmd.Parameters.AddWithValue("@Role", user.Role);

cmd.ExecuteNonQuery();

Console.WriteLine("User created successfully.");

}

catch (Exception ex)

{

Console.WriteLine("Error: " + ex.Message);

}

finally

{

conn.Close();

}

}

public void CreateProduct(User user, Product product)

{

if (user.Role != "Admin")

{

Console.WriteLine("Only Admin can create products.");

return;

}

try

{

conn.Open();

string query = "INSERT INTO Products (ProductId, ProductName, Description, Price, QuantityInStock, Type) VALUES (@ProductId, @ProductName, @Description, @Price, @QuantityInStock, @Type)";

SqlCommand cmd = new SqlCommand(query, conn);

cmd.Parameters.AddWithValue("@ProductId", product.ProductId);

cmd.Parameters.AddWithValue("@ProductName", product.ProductName);

cmd.Parameters.AddWithValue("@Description", product.Description);

cmd.Parameters.AddWithValue("@Price", product.Price);

cmd.Parameters.AddWithValue("@QuantityInStock", product.QuantityInStock);

cmd.Parameters.AddWithValue("@Type", product.Type);

cmd.ExecuteNonQuery();

if (product.Type == "Electronics" && product is Electronics electronics)

{

string query2 = "INSERT INTO Electronics (ProductId, Brand, WarrantyPeriod) VALUES (@ProductId, @Brand, @WarrantyPeriod)";

SqlCommand cmd2 = new SqlCommand(query2, conn);

cmd2.Parameters.AddWithValue("@ProductId", electronics.ProductId);

cmd2.Parameters.AddWithValue("@Brand", electronics.Brand);

cmd2.Parameters.AddWithValue("@WarrantyPeriod", electronics.WarrantyPeriod);

cmd2.ExecuteNonQuery();

}

else if (product.Type == "Clothing" && product is Clothing clothing)

{

string query3 = "INSERT INTO Clothing (ProductId, Size, Color) VALUES (@ProductId, @Size, @Color)";

SqlCommand cmd3 = new SqlCommand(query3, conn);

cmd3.Parameters.AddWithValue("@ProductId", clothing.ProductId);

cmd3.Parameters.AddWithValue("@Size", clothing.Size);

cmd3.Parameters.AddWithValue("@Color", clothing.Color);

cmd3.ExecuteNonQuery();

}

Console.WriteLine("Product created successfully.");

}

catch (Exception ex)

{

Console.WriteLine("Error: " + ex.Message);

}

finally

{

conn.Close();

}

}

public List<Product> GetAllProducts()

{

List<Product> productList = new List<Product>();

try

{

conn.Open();

string query = "SELECT \* FROM Products";

SqlCommand cmd = new SqlCommand(query, conn);

SqlDataReader reader = cmd.ExecuteReader();

while (reader.Read())

{

Product p = new Product

{

ProductId = reader.GetInt32(0),

ProductName = reader.GetString(1),

Description = reader.GetString(2),

Price = reader.GetDouble(3),

QuantityInStock = reader.GetInt32(4),

Type = reader.GetString(5)

};

productList.Add(p);

}

reader.Close();

}

catch (Exception ex)

{

Console.WriteLine("Error: " + ex.Message);

}

finally

{

conn.Close();

}

return productList;

}

public void CreateOrder(User user, List<Product> products)

{

try

{

conn.Open();

// Check if user exists

SqlCommand checkUser = new SqlCommand("SELECT COUNT(\*) FROM Users WHERE UserId=@UserId", conn);

checkUser.Parameters.AddWithValue("@UserId", user.UserId);

int count = (int)checkUser.ExecuteScalar();

if (count == 0)

{

CreateUser(user); // Create user if not exists

}

// Create Order

SqlCommand insertOrder = new SqlCommand("INSERT INTO Orders (UserId) OUTPUT INSERTED.OrderId VALUES (@UserId)", conn);

insertOrder.Parameters.AddWithValue("@UserId", user.UserId);

int orderId = (int)insertOrder.ExecuteScalar();

foreach (var product in products)

{

SqlCommand insertItem = new SqlCommand("INSERT INTO OrderItems (OrderId, ProductId) VALUES (@OrderId, @ProductId)", conn);

insertItem.Parameters.AddWithValue("@OrderId", orderId);

insertItem.Parameters.AddWithValue("@ProductId", product.ProductId);

insertItem.ExecuteNonQuery();

}

Console.WriteLine("Order created successfully.");

}

catch (Exception ex)

{

Console.WriteLine("Error: " + ex.Message);

}

finally

{

conn.Close();

}

}

public void CancelOrder(int userId, int orderId)

{

try

{

conn.Open();

SqlCommand check = new SqlCommand("SELECT COUNT(\*) FROM Orders WHERE OrderId=@OrderId AND UserId=@UserId", conn);

check.Parameters.AddWithValue("@OrderId", orderId);

check.Parameters.AddWithValue("@UserId", userId);

int found = (int)check.ExecuteScalar();

if (found == 0)

throw new OrderNotFoundException("Order not found for this user.");

SqlCommand deleteItems = new SqlCommand("DELETE FROM OrderItems WHERE OrderId=@OrderId", conn);

deleteItems.Parameters.AddWithValue("@OrderId", orderId);

deleteItems.ExecuteNonQuery();

SqlCommand deleteOrder = new SqlCommand("DELETE FROM Orders WHERE OrderId=@OrderId", conn);

deleteOrder.Parameters.AddWithValue("@OrderId", orderId);

deleteOrder.ExecuteNonQuery();

Console.WriteLine("Order cancelled successfully.");

}

catch (OrderNotFoundException ex)

{

Console.WriteLine("Custom Exception: " + ex.Message);

}

catch (Exception ex)

{

Console.WriteLine("Error: " + ex.Message);

}

finally

{

conn.Close();

}

}

public List<Product> GetOrderByUser(User user)

{

List<Product> list = new List<Product>();

try

{

conn.Open();

string query = @"

SELECT p.ProductId, p.ProductName, p.Description, p.Price, p.QuantityInStock, p.Type

FROM Products p

INNER JOIN OrderItems oi ON p.ProductId = oi.ProductId

INNER JOIN Orders o ON oi.OrderId = o.OrderId

WHERE o.UserId = @UserId";

SqlCommand cmd = new SqlCommand(query, conn);

cmd.Parameters.AddWithValue("@UserId", user.UserId);

SqlDataReader reader = cmd.ExecuteReader();

while (reader.Read())

{

Product p = new Product

{

ProductId = reader.GetInt32(0),

ProductName = reader.GetString(1),

Description = reader.GetString(2),

Price = reader.GetDouble(3),

QuantityInStock = reader.GetInt32(4),

Type = reader.GetString(5)

};

list.Add(p);

}

reader.Close();

}

catch (Exception ex)

{

Console.WriteLine("Error: " + ex.Message);

}

finally

{

conn.Close();

}

return list;

}

}

}

// ---------- Program.cs ----------

// main/MainModule.cs

using System;

using System.Collections.Generic;

using OrderManagementSystem.entity;

using OrderManagementSystem.dao;

namespace OrderManagementSystem.main

{

class MainModule

{

static void Main(string[] args)

{

OrderProcessor orderProcessor = new OrderProcessor();

while (true)

{

Console.WriteLine("\n--- Order Management System Menu ---");

Console.WriteLine("1. Create User");

Console.WriteLine("2. Create Product");

Console.WriteLine("3. Get All Products");

Console.WriteLine("4. Create Order");

Console.WriteLine("5. Cancel Order");

Console.WriteLine("6. Get Orders by User");

Console.WriteLine("7. Exit");

Console.Write("Enter your choice: ");

string input = Console.ReadLine();

int choice;

if (!int.TryParse(input, out choice))

{

Console.WriteLine("Invalid input. Please enter a number.");

continue;

}

switch (choice)

{

case 1:

CreateUser(orderProcessor);

break;

case 2:

CreateProduct(orderProcessor);

break;

case 3:

GetAllProducts(orderProcessor);

break;

case 4:

CreateOrder(orderProcessor);

break;

case 5:

CancelOrder(orderProcessor);

break;

case 6:

GetOrdersByUser(orderProcessor);

break;

case 7:

Console.WriteLine("Exiting application.");

return;

default:

Console.WriteLine("Invalid choice. Try again.");

break;

}

}

}

static void CreateUser(OrderProcessor op)

{

Console.Write("Enter User ID: ");

int userId = int.Parse(Console.ReadLine());

Console.Write("Enter Username: ");

string username = Console.ReadLine();

Console.Write("Enter Password: ");

string password = Console.ReadLine();

Console.Write("Enter Role (Admin/User): ");

string role = Console.ReadLine();

User user = new User(userId, username, password, role);

op.CreateUser(user);

}

static void CreateProduct(OrderProcessor op)

{

Console.Write("Enter Admin User ID: ");

int adminUserId = int.Parse(Console.ReadLine());

Console.Write("Enter Admin Username: ");

string adminUsername = Console.ReadLine();

Console.Write("Enter Admin Password: ");

string adminPassword = Console.ReadLine();

Console.Write("Enter Admin Role (Admin): ");

string adminRole = Console.ReadLine();

User adminUser = new User(adminUserId, adminUsername, adminPassword, adminRole);

Console.Write("Enter Product ID: ");

int productId = int.Parse(Console.ReadLine());

Console.Write("Enter Product Name: ");

string productName = Console.ReadLine();

Console.Write("Enter Description: ");

string description = Console.ReadLine();

Console.Write("Enter Price: ");

double price = double.Parse(Console.ReadLine());

Console.Write("Enter Quantity In Stock: ");

int quantity = int.Parse(Console.ReadLine());

Console.Write("Enter Type (Electronics/Clothing): ");

string type = Console.ReadLine();

if (type.Equals("Electronics", StringComparison.OrdinalIgnoreCase))

{

Console.Write("Enter Brand: ");

string brand = Console.ReadLine();

Console.Write("Enter Warranty Period (months): ");

int warranty = int.Parse(Console.ReadLine());

Electronics product = new Electronics(productId, productName, description, price, quantity, type, brand, warranty);

op.CreateProduct(adminUser, product);

}

else if (type.Equals("Clothing", StringComparison.OrdinalIgnoreCase))

{

Console.Write("Enter Size: ");

string size = Console.ReadLine();

Console.Write("Enter Color: ");

string color = Console.ReadLine();

Clothing product = new Clothing(productId, productName, description, price, quantity, type, size, color);

op.CreateProduct(adminUser, product);

}

else

{

Console.WriteLine("Invalid product type.");

}

}

static void GetAllProducts(OrderProcessor op)

{

List<Product> products = op.GetAllProducts();

Console.WriteLine("\n--- Product List ---");

foreach (var p in products)

{

Console.WriteLine($"ID: {p.ProductId}, Name: {p.ProductName}, Type: {p.Type}, Price: {p.Price}");

}

}

static void CreateOrder(OrderProcessor op)

{

Console.Write("Enter User ID: ");

int userId = int.Parse(Console.ReadLine());

Console.Write("Enter Username: ");

string username = Console.ReadLine();

Console.Write("Enter Password: ");

string password = Console.ReadLine();

Console.Write("Enter Role (User/Admin): ");

string role = Console.ReadLine();

User user = new User(userId, username, password, role);

List<Product> productsToOrder = new List<Product>();

while (true)

{

Console.Write("Enter Product ID to add to order (or 'done' to finish): ");

string input = Console.ReadLine();

if (input.ToLower() == "done")

break;

if (int.TryParse(input, out int prodId))

{

// Here, for simplicity, we create dummy product with just ID.

// You may want to get full product details from DB.

Product p = new Product { ProductId = prodId };

productsToOrder.Add(p);

}

else

{

Console.WriteLine("Invalid input.");

}

}

op.CreateOrder(user, productsToOrder);

}

static void CancelOrder(OrderProcessor op)

{

Console.Write("Enter User ID: ");

int userId = int.Parse(Console.ReadLine());

Console.Write("Enter Order ID: ");

int orderId = int.Parse(Console.ReadLine());

op.CancelOrder(userId, orderId);

}

static void GetOrdersByUser(OrderProcessor op)

{

Console.Write("Enter User ID: ");

int userId = int.Parse(Console.ReadLine());

Console.Write("Enter Username: ");

string username = Console.ReadLine();

Console.Write("Enter Password: ");

string password = Console.ReadLine();

Console.Write("Enter Role (User/Admin): ");

string role = Console.ReadLine();

User user = new User(userId, username, password, role);

var products = op.GetOrderByUser(user);

Console.WriteLine("\n--- Products Ordered by User ---");

foreach (var p in products)

{

Console.WriteLine($"ID: {p.ProductId}, Name: {p.ProductName}, Type: {p.Type}, Price: {p.Price}");

}

}

}

}