Oren Pildus Designs

Online Store Demo

(C# & WPF Project)

1. About This Project

This project intends to demo an online store for displaying and selling various products. It was built to demonstrate knowledge and implementation of C# and Entity Framework Programming fundamentals and principles,

2. Built With :

The project was Built with Entity Framework Core as the data access technology, and SQL Server as the database structure. Though not mandatory, the GUI for the demo was built with WPF graphical environment.

Programming	Data Access	Database	GUI
C#	Entity Framework Core.	SQL Server	WPF

The project loosely implements the **Factory** design pattern.

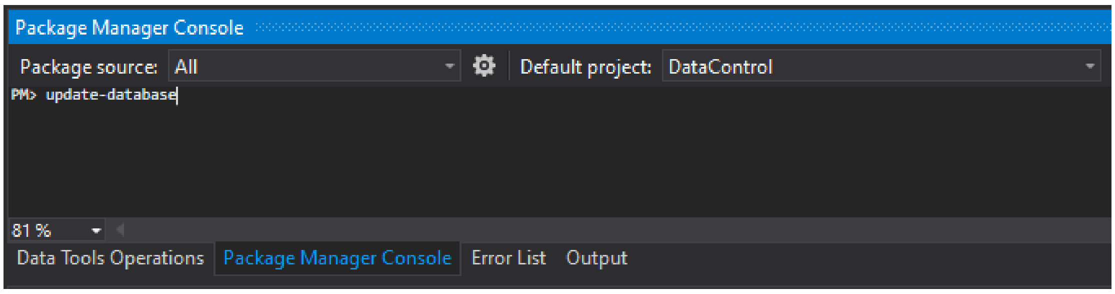
3. Getting Started:

- ***Download the code source:***

Download or clone the Github code from: www.github.com....

- ***Building the database structure:***

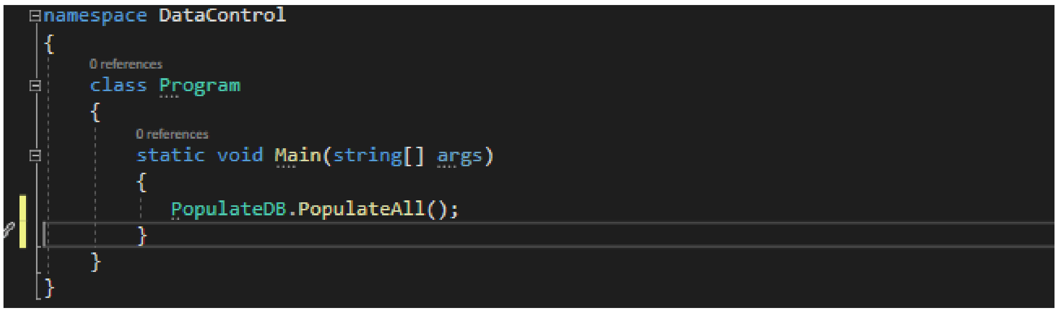
The project contains one final migration file. Using the Package manager console, run command "Update-Database" to apply the migration file to the SQL Server.



```
PM> update-database
```

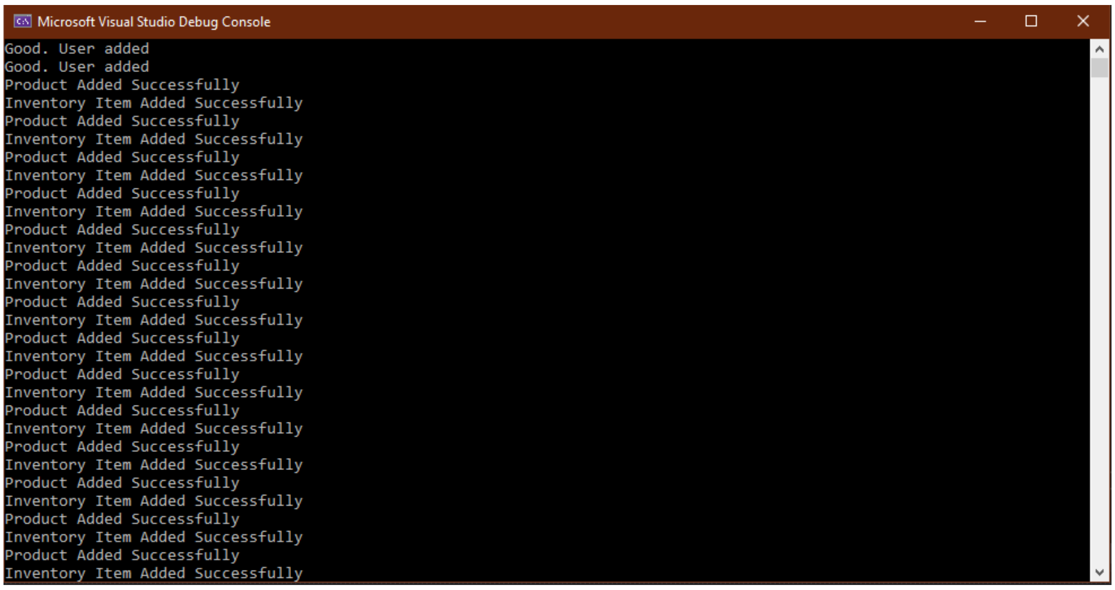
- ***Populating the database:***

Set Project "Data Control" as startup project; make sure "Program.CS" contains the following code:



```
namespace DataControl
{
    class Program
    {
        static void Main(string[] args)
        {
            PopulateDB.PopulateAll();
        }
    }
}
```

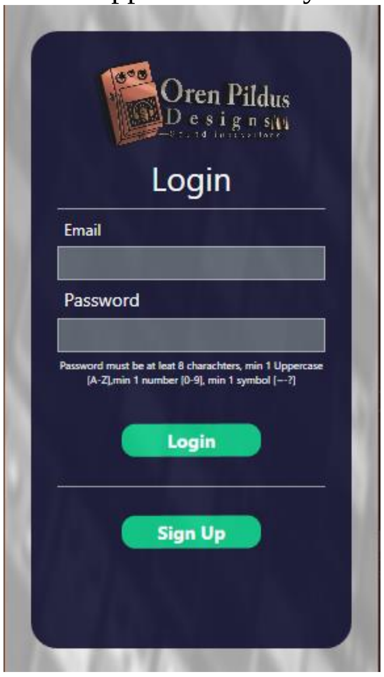
Build And Run solution. The output Window should output the following:



```
Good. User added
Good. User added
Product Added Successfully
Inventory Item Added Successfully
Product Added Successfully
Inventory Item Added Successfully
Product Added Successfully
Inventory Item Added Successfully
Product Added Successfully
Inventory Item Added Successfully
Product Added Successfully
Inventory Item Added Successfully
Product Added Successfully
Inventory Item Added Successfully
Product Added Successfully
Inventory Item Added Successfully
Product Added Successfully
Inventory Item Added Successfully
Product Added Successfully
Inventory Item Added Successfully
Product Added Successfully
Inventory Item Added Successfully
Product Added Successfully
Inventory Item Added Successfully
Product Added Successfully
Inventory Item Added Successfully
Product Added Successfully
Inventory Item Added Successfully
Product Added Successfully
Inventory Item Added Successfully
```

- ***Running the Demo:***

Set project OPD_GUI as startup project, Build And Run solution. The Login screen should appear to allow you to log in:



Password must be at leat 8 charachters, min 1 Uppercase [A-Z],min 1 number [0-9], min 1 symbol [~-?]

Admin user:
    Email address: admin@admin.com
    Password: Admin123!
Regular User:
    Email address: user@user.com
    Password: User123!

4. Entities, Relations design and logic:

The project implements EF TPT (Table-per-Type) inheritance model. Table-per-type inheritance uses a separate table in the database to maintain data for non-inherited properties and key properties for each type in the inheritance hierarchy.
The main model for the products section is an abstract class ***Product*** from which each specific product type model is inherent:

```
public abstract class Product
{
    [Key]
    public int ProductId { get; set; }
    [Required]
    public string ProductName { get; set; }
    [Required]
    public double ProductPrice { get; set; }
    [Required]
    public ProductTypes ProductType { get; set; }

    public EffectTypes EffectType { get; set; }
}
```

There are currently 3 product types models (***Pedal, Board, Component***), each has unique Properties:

```
[Table("Pedals")]
public class Pedal : Product
{
    public string PedalDescription { get; set; }
}

[Table("Boards")]
public class Board : Product
{
    [Required]
    public double BoardWidth { get; set; }
    [Required]
    public double BoardHeight { get; set; }
}

[Table("Components")]
public class Component : Product
{
    [Required]
    public int QuantityPerLot { get; set; }
    public ComponentTypes ComponentType { get; set; }
}
```

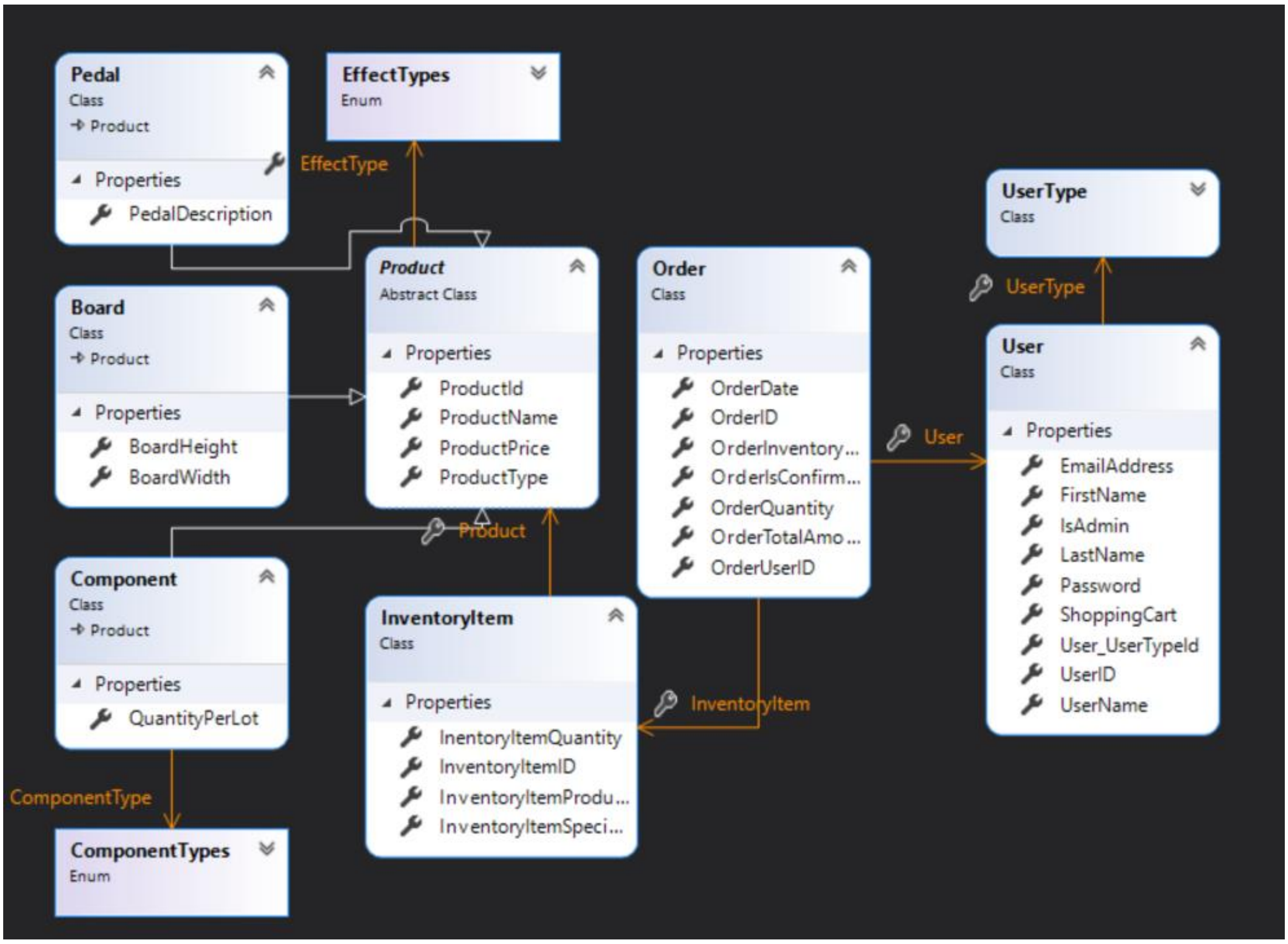
Furthermore, there are several more entities in the project:

InventoryItem *(Table Inventory)* – Handles the actual amount of items from a certain product. Updated by explicit inventory creation/modifications, or as a result of User's oders/returns.

Order *(Table Orders)* – Handles inventory traffic to/from users.

User *(Table Users)* – Holds the details of each user, and the user's type. Upon signing up and registering a User, **the selected password is being encrypted using SHA256 hash function.**

UserType *(Table UsersTypes)* – holds the various user's types.



5. Usage:

The DataControl assembly revels a Utils folder to the customer, which offer a various data retrieving / manipulation methods to be called via static classes. Following is a key list of these methods:

Admin Only Methods:

- Adding Products (AddProduct (Overloaded for various product types))
- Editing Products (EditProduct (Overloaded for various product types))
- Delete Product (DeleteProduct by ID)

- Adding Inventory record
- Explicit Inventory editing (Oppose to editing via order) – allow admins to return/refund items to inventory.
- Retrieve all orders in the system.

*not implemented in the GUI due to lack of time.

General usage Methods:

- Log in / Sign up methods.
- Retrieve Products List (Overloaded for various filters requests)
- Add Order / Edit order (Confirming order to move from shopping cart to be removed from inventory).
- Retrieve list of user's confirmed orders (Overloaded for various filters requests)
- Edit User details and password