

HOMework # 03

Contents:

ADO.NET, In-memory DataTable, DataSet, and DataAdapter

Objective:

Dear students, we will be engaging in hands-on practice with the technical contents mentioned above. Efficiently handling user data is an essential skill for becoming a proficient web developer, and we will be using **ado.net** and **in-memory** data handling for this purpose. **Let's practice** these concepts and enhance our proficiency as a .NET Developer!

Happy Learning!

Learning Resources:

- [Data Adapters – Microsoft Learn](#)
- [SqlDataAdapter Class – Microsoft Learn](#)



Scenario

Hello developers!

Do you still remember that you were working for *QueenShop* as a C# Web Developer, a clothing brand that sells the latest fashions designed especially to boost confidence. Previously, you were tasked with creating a local database to store order information. The purpose of that task was to enable *QueenShop* to efficiently track and manage orders. Now, you have some modifications from your stakeholder and a new requirement from your team lead. Your team lead wants to optimize querying database. He highlighted a case when the order's amount goes up beyond customer's sales preference. So he/she can reduce product quantity or can drop some product from the order. To incorporate this functionality, you need to handle user's run-time input using *in-memory* data handling techniques. The order details will be saved in database when user confirms the order. Further you have to update previous database schema according to UI/UX team's finalized prototype.

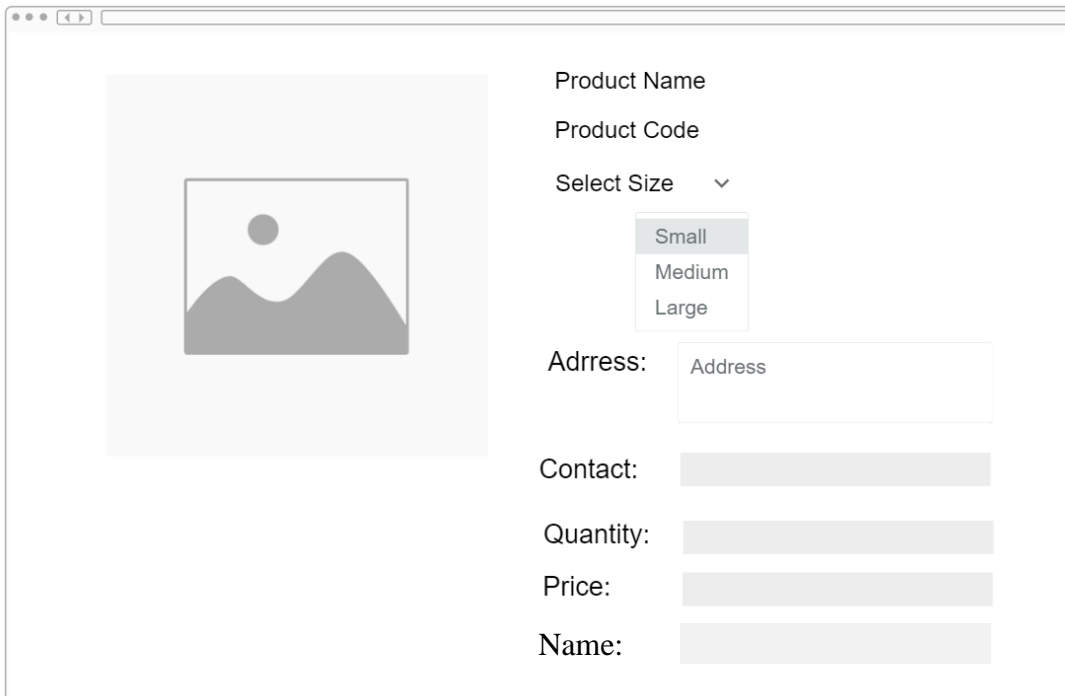
You're recommended to create a separate project for these updates. Don't disrupt existing code. We would first create a new system. Then slowly shift from old back-end to new back-end. This technique is extensively used for customer retention by UI/UX designers. (Just a tip for business point of view)

Task-1: (10 marks)

Following were the attributes for Orders table.

- a) Order ID
- b) Customer's CNIC
- c) Customer's Name
- d) Customer's Phone
- e) Customer's Address
- f) Product ID
- g) Price
- h) Size of Product (Large, Medium, Small etc.,)

Now, update your database schema based on following prototype:



Product Name

Product Code

Select Size ▼

- Small
- Medium
- Large

Address:

Contact:

Quantity:

Price:

Name:

The new attributes will be:

- | | |
|--------------------|--------------------|
| 1. ProductName | 5. CustomerContact |
| 2. ProductCode | 6. ProductQuantity |
| 3. ProductSize | 7. Price |
| 4. CustomerAddress | 8. OrderID |
| | 9. CustomerName |

Task-02 (5 Marks)

Create Customers table with following attributes:

- 1- CustomerID (a unique key for *QueenShop* with prefix “Cus-” and an arithmetic sequence. e.g., Cus-1, Cus-2, Cus-3 so on)
- 2- CustomerName (select most appropriate datatype and length)
- 3- CustomerAddress (select most appropriate datatype)
- 4- CustomerContact (select most appropriate datatype and length)

Task-03 (5 Marks)

Create Products Table with following attributes:

- 1- ProductCode: currently *QueenShop* sales only 5 unique designs
- 2- ProductName
- 3- ProductPrice
- 4- ProductPicture: A string to store image’s path

PQS-102 = Charm = 200\$ = “/images/charm.jpeg”

PQS-999 = Night = 400\$ = “/images/night. jpeg”

PQS-633 = Shine = 600\$ = “/images/shine. jpeg”

PQS-488 = Delight = 800\$ = “/images/delight. jpeg”

PQS-755 = Spark = 1000\$ = “/images/spark. jpeg”

Change Log:

- CustomerCNIC is dropped because order processing doesn’t need it. A unique key will be generated to store customer records in table instead of cnic.
- ProductId is replaced with ProductCode for matching company’s convention to call products by their ProductCode.

Task-4: (20 Marks)

Handling user input using disconnected database handling techniques:

- 1) Initialize a DataSet and a DataAdapter to retrieve order details from the database based on the customer's ID.
- 2) Implement functionality to modify order details locally in a DataTable within the DataSet, such as changing the quantity of products.
- 3) Allow users to add new orders by inserting rows into the DataTable.
- 4) Enable users to remove orders by deleting corresponding rows from the DataTable.
- 5) Upon confirmation of the order, synchronize the changes made in the DataTable with the database using the DataAdapter.
- 6) For above 5 sub-tasks, take input from the user!