PRACTICAL NO. 3: Database Programming in ASP.NET

Aim: To develop database applications using ADO.NET, LINQ to SQL, Entity Framework, and data controls in ASP.NET.

1. ADO.NET:

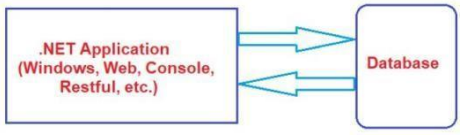
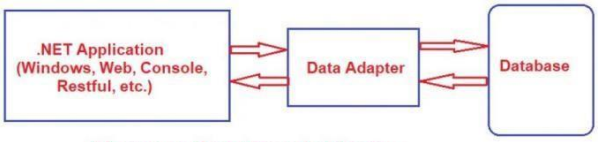
ADO (Active X Data Objects) is a rich set of classes, interfaces, structures and enumerated types that manage data access from various types of data stores.

In ASP.NET, ADO.NET is the primary way to interact with databases. If you are building a web application (ASP.NET Web Forms, MVC, or even ASP.NET Core), you will often use ADO.NET to:

1. Connect to a database
2. Execute SQL queries (like SELECT, INSERT, UPDATE, DELETE)
3. Retrieve data from the database
4. Populate data into controls (like GridView, DropDownList)
5. Perform transactions

ADO.NET provides a rich set of classes that facilitate database interaction. Below are some of the key classes:

Class Name	Description
Connection	Establishes a connection to the data source (e.g., SqlConnection, OleDbConnection).
Command	Executes SQL commands (e.g., SqlCommand, OleDbCommand).
DataReader	Reads data forward-only and read-only from a data source (e.g., SqlDataReader).
DataAdapter	Acts as a bridge between the DataSet and the database for filling data and updating changes (e.g., SqlDataAdapter).
DataSet	In-memory cache of data retrieved from the database. Can hold multiple tables.
DataTable	Represents a single table of in-memory data within a DataSet.
DataRow	Represents a single row within a DataTable.
DataColumn	Represents a single column within a DataTable.
CommandBuilder	Automatically generates SQL commands for single-table updates (e.g., SqlCommandBuilder).
Parameter	Represents a parameter to a SQL query or stored procedure (e.g., SqlParameter).

Connected Architecture	Disconnected Architecture
Connected architecture in **ASP.NET** refers to a data access model where the application maintains an active connection with the database while fetching or manipulating data. This approach is based on **ADO.NET** and is useful for scenarios where real-time data access is required.	**Disconnected architecture** in **ASP.NET** refers to a data access model where the application does not maintain a continuous connection to the database. Instead, it **fetches the data once**, stores it in memory (e.g., **DataSet**), and performs operations **offline**. This approach is based on **ADO.NET** and is suitable for scenarios where **scalability and reduced database load** are required.
 Connection Oriented Data Access Architecture	 Disconnected Data Access Architecture
Steps to access data from database table: 1. Establish a Connection using SqlConnection 2. Create a Command using SqlCommand 3. Execute the Command and retrieve data	**Steps to perform CRUD operations on table:** 1. Establish a Connection using SqlConnection. 2. Create a Data Adapter using SqlDataAdapter. 3. Fill the DataSet with the retrieved data.

using SqlDataReader 4. Process the Data 5. Close the Connection after use	4. Perform operations on the DataSet (Add, Update, Delete). 5. Update changes back to the database if required. 6. Close the Connection (but the DataSet remains available).
Advantages: 1. Real-time data access (always connected to DB) 2. Fast execution for read-only operations 3. Efficient for smaller datasets	**Advantages:** 1. Reduces database load (connection opens only when needed). 2. Scalable for large applications. 3. Faster operations on in-memory data. 4. Works well for applications with batch processing.
Disadvantages: 1. Requires an active database connection (resource-intensive) 2. Not suitable for large datasets (affects performance) 3. Less scalable compared to disconnected architecture	**Disadvantages:** 1. Data may become outdated (since it's stored in memory). 2. Uses more memory if dealing with large datasets. 3. Requires explicit update logic to sync changes back to the database.

2. Working with Stored Procedures (Simple and Parameterised):

A Stored Procedure is a precompiled SQL query stored in the database. It can accept parameters (input, output) and return values. It provides better performance, security, and maintainability compared to inline SQL queries.

- Simple Stored Procedure: No parameters — performs a direct query (like fetching all records).
- Parameterized Stored Procedure: Accepts parameters for filtering, inserting, updating, etc.

3. Data Bound Controls in ASP.net:

The **data-bound controls** provide different ways to display and manipulate data in **ASP.NET Web Forms**. If you need **grid-based** display, use **GridView**. If you need **custom layouts**, use **ListView** or **Repeater**.

Control	Description	Example Code
DataList	Displays data in a customizable, repeatable list format using templates.	<asp:DataList ID="DataList1" runat="server" DataSourceID="SqlDataSource1"> <ItemTemplate> Name: <%# Eval("Name") %> Age: <%# Eval("Age") %> </ItemTemplate> </asp:DataList>
DetailsView	Displays a single record in a table format with built-in editing, deleting, and inserting support.	<asp:DetailsView ID="DetailsView1" runat="server" DataSourceID="SqlDataSource1" AllowPaging="True"></asp:DetailsView>
FormView	Similar to DetailsView but allows full customization using templates to display a single record.	<asp:FormView ID="FormView1" runat="server" DataSourceID="SqlDataSource1"> <ItemTemplate> Name: <%# Eval("Name") %> Age: <%# Eval("Age") %> </ItemTemplate> </asp:FormView>
GridView	Displays data in a tabular format with built-in	<asp:GridView ID="GridView1" runat="server" DataSourceID="SqlDataSource1" AllowPaging="True"

Control	Description	Example Code
	sorting, paging, editing, and deleting functionalities.	AllowSorting="True"></asp:GridView>
ListView	A flexible, template-based control that supports customizable layouts, paging, and grouping.	<asp:ListView ID="ListView1" runat="server" DataSourceID="SqlDataSource1"> <ItemTemplate> Name: <%# Eval("Name") %> Age: <%# Eval("Age") %> </ItemTemplate> </asp:ListView>
Repeater	A lightweight, template-based control for displaying repeated data without built-in paging or sorting.	<asp:Repeater ID="Repeater1" runat="server" DataSourceID="SqlDataSource1"> <ItemTemplate> Name: <%# Eval("Name") %> Age: <%# Eval("Age") %> <hr /> </ItemTemplate> </asp:Repeater>

4. DataSource Controls in ASP.NET:

In ASP.NET, DataSource controls provide a declarative way to bind data to data-bound controls such as GridView, Repeater, and DropDownList. These controls abstract data retrieval logic, making it easier to work with databases, XML files, or other data sources without writing extensive code.

Types of DataSource Controls in ASP.NET: The following table summarizes the key DataSource controls available in ASP.NET, along with their descriptions and examples:

DataSource Control	Description	Example
SqlDataSource	Connects to a relational database (SQL Server, MySQL, etc.) using a connection string.	<asp:SqlDataSource ID="SqlDataSource1" runat="server" ConnectionString="your_connection_string" SelectCommand="SELECT * FROM Products"></asp:SqlDataSource>
ObjectDataSource	Binds to a business object or class instead of a database directly.	<asp:ObjectDataSource ID="ObjectDataSource1" runat="server" TypeName="ProductService" SelectMethod="GetAllProducts"></asp:ObjectDataSource>
XmlDataSource	Retrieves data from an XML file and transforms it using XSLT if needed.	<asp:XmlDataSource ID="XmlDataSource1" runat="server" DataFile="~/Products.xml"></asp:XmlDataSource>
LinqDataSource	Uses LINQ to query a data context (such as Entity Framework or LINQ to SQL).	<asp:LinqDataSource ID="LinqDataSource1" runat="server" ContextTypeName="ProductDataContext" TableName="Products"></asp:LinqDataSource>
AccessDataSource	Retrieves data from a Microsoft Access database.	<asp:AccessDataSource ID="AccessDataSource1" runat="server" DataFile="~/App_Data/Products.mdb" SelectCommand="SELECT * FROM Products"></asp:AccessDataSource>
EntityDataSource	Connects to an Entity Framework model and	<asp:EntityDataSource ID="EntityDataSource1" runat="server" ConnectionString="name=MyEntities"

DataSource Control	Description	Example
	provides data binding support.	DefaultContainerName="MyEntities" EntitySetName="Products"></asp:EntityDataSource>
SiteMapDataSource	Provides navigation data from a Web.sitemap file.	<asp:SiteMapDataSource ID="SiteMapDataSource1" runat="server"></asp:SiteMapDataSource>

5. LINQ in ASP.NET:

a. Introduction to LINQ: LINQ (Language-Integrated Query) enables querying collections, databases, XML, etc., using C#. Works with **LINQ to Objects, LINQ to SQL, LINQ to XML, LINQ to Entities**.

Advantages:

- Readability – Easier than raw SQL.
- Type Safety – Catches errors at compile-time.
- Productivity – Reduces boilerplate code.
- Interoperability – Works with multiple data sources.

Example Without LINQ (SQL Query in ADO.NET):

```
SqlCommand cmd = new SqlCommand("SELECT * FROM Employees WHERE Age > 30", conn);
SqlDataReader reader = cmd.ExecuteReader();
while (reader.Read()) { Console.WriteLine(reader["Name"]); }
```

Example With LINQ:

```
var result = from emp in employees where emp.Age > 30 select emp.Name;
foreach (var name in result) { Console.WriteLine(name); }
```

b. Mapping Data Model to Object Model: Data Model Mapping converts database tables into C# classes. Uses Entity Framework or LINQ to SQL.

Steps to Map Data Model:

1. **Create a Database Table:** CREATE TABLE Employees (Id INT PRIMARY KEY, Name NVARCHAR(50), Age INT);
2. **Define a C# Class:** public class Employee { public int Id { get; set; } public string Name { get; set; } public int Age { get; set; } }
3. **Query Using LINQ:** List<Employee> employees = new List<Employee> { new Employee { Id = 11, Name = "Sachin", Age = 27 } }; var olderEmployees = from emp in employees where emp.Age > 30 select emp;

c. LINQ Query Syntax:

LINQ provides two syntaxes:

1. Query Syntax (SQL-like): It is SQL-like, easy for beginners and used for Simple operations
   var result = from emp in employees where emp.Age > 30 select emp;
2. Method Syntax (Fluent API using Lambda): It is Concise, more flexible and Supports advanced queries
   var result = employees.Where(emp => emp.Age > 30);

d. Example: LINQ Query in ASP.NET: Retrieve Employees Older Than 30

Using Query Syntax: var result = from e in context.Employees where e.Age > 30 select e;

Using Method Syntax: var result = context.Employees.Where(e => e.Age > 30).ToList();

6. Entity Framework:

Entity Framework (EF) is an **Object-Relational Mapper (ORM)** for .NET applications, allowing developers to work with databases using .NET objects instead of writing SQL queries manually. It simplifies database operations by handling data access and manipulation using LINQ (Language-Integrated Query).

- **Key Features of Entity Framework:**
  1. **Abstraction of SQL** – Developers interact with objects instead of raw SQL queries.
  2. **Automatic Change Tracking** – Detects changes in objects and updates the database accordingly.
  3. **Migrations** – Helps manage database schema changes over time.
  4. **Lazy Loading & Eager Loading** – Optimizes how related data is retrieved.
  5. **Concurrency Handling** – Supports optimistic concurrency control.
  6. **Cross-Database Compatibility** – Works with SQL Server, PostgreSQL, MySQL, SQLite, etc.
- **EF Workflows:** There are three primary approaches to using Entity Framework:
  1. **Code-First** (Recommended for new projects): Define models as C# classes, and EF generates the database schema.
  2. **Database-First** (For existing databases): Reverse-engineers an existing database into C# models.
  3. **Model-First** (Less common): Uses a visual designer to create models and then generates the database.

References:

- https://learn.microsoft.com/en-us/dotnet/framework/data/adonet/
- https://dotnettutorials.net/lesson/connected-and-disconnected-architecture-in-ado-net/
- https://learn.microsoft.com/en-us/previous-versions/aspnet/ms247258(v=vs.100)
- https://learn.microsoft.com/en-us/dotnet/csharp/linq/get-started/introduction-to-linq-queries
- https://learn.microsoft.com/en-us/aspnet/mvc/overview/older-versions-1/models-data/creating-model-classes-with-linq-to-sql-cs
- https://www.csharp.com/article/understanding-linq-in-c-sharp-query-syntax-and-method-syntax/
- https://learn.microsoft.com/en-us/previous-versions/aspnet/bb907622(v=vs.100)
- https://en.wikipedia.org/wiki/Entity_Framework
- https://learn.microsoft.com/en-us/previous-versions/aspnet/ms247258(v=vs.100)

Exercise:

1. Design an ASP.NET web Applications using **Connected** Database **architecture** to add, update, delete and search customer information from database table. Consider following schema for customer_info table customer_info(cid primary key, cname, cadd)

Customer.aspx :

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Customer.aspx.cs"
Inherits="Practical_3.Question_1.Customer" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
   <title></title>
</head>
<body>
   <form id="form1" runat="server">
      <div>
         <h2>Customer Management</h2>
      Customer ID: <asp:TextBox ID="txtCid" runat="server"></asp:TextBox><br />
      Customer Name: <asp:TextBox ID="txtCname" runat="server"></asp:TextBox><br />
      Address: <asp:TextBox ID="txtCadd" runat="server"></asp:TextBox><br />

      <asp:Button ID="btnAdd" runat="server" Text="Add" OnClick="btnAdd_Click" />
      <asp:Button ID="btnUpdate" runat="server" Text="Update" OnClick="btnUpdate_Click" />
      <asp:Button ID="btnDelete" runat="server" Text="Delete" OnClick="btnDelete_Click" />
      <asp:Button ID="btnSearch" runat="server" Text="Search" OnClick="btnSearch_Click" /><br /><br />

      <asp:GridView ID="gvCustomers" runat="server" AutoGenerateColumns="True"></asp:GridView>
      </div>
   </form>
</body>
</html>
```

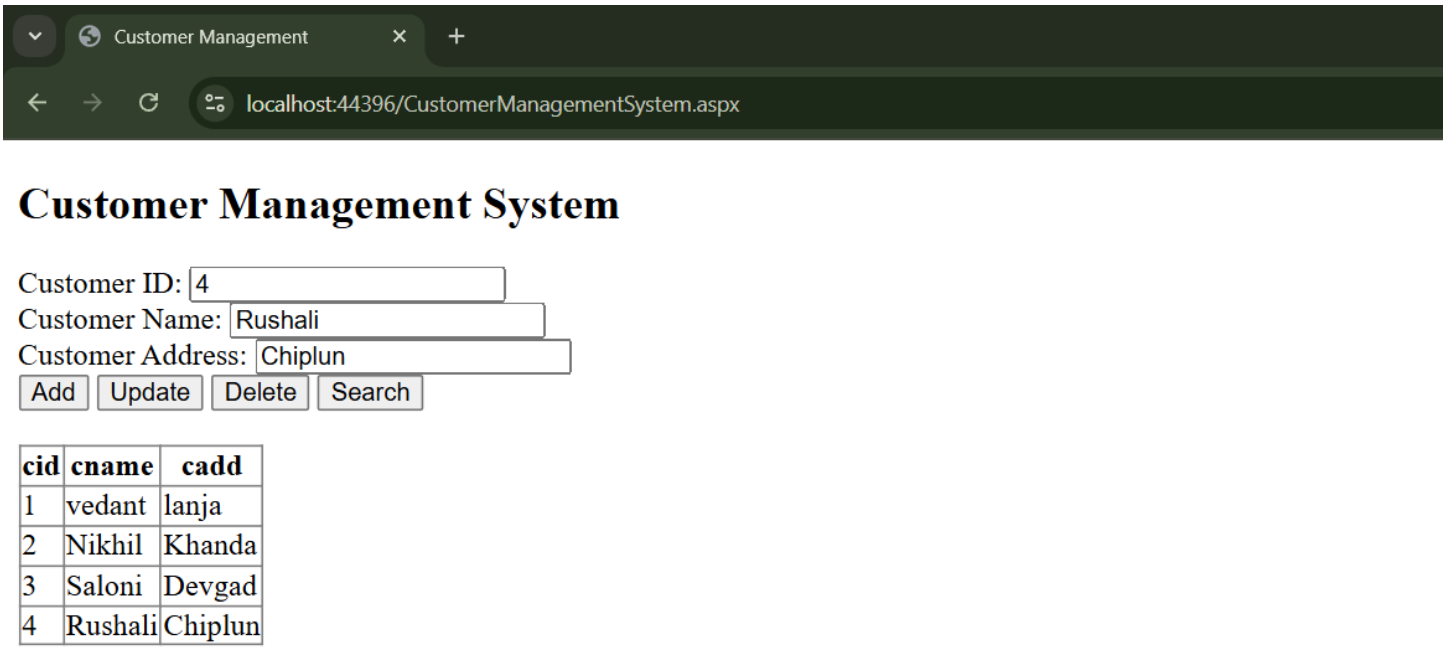
Customer.aspx.cs :

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Data;
using System.Data.SqlClient;
using System.Configuration;

namespace Practical_3.Question_1
{
   public partial class Customer : System.Web.UI.Page
   {
      string connectionString = ConfigurationManager.ConnectionStrings["DBCS"].ConnectionString;
```

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        LoadCustomers();
    }
}

protected void btnAdd_Click(object sender, EventArgs e)
{
    SqlConnection con = new SqlConnection(connectionString);
    string query = "INSERT INTO customer_info (cid, cname, cadd) VALUES (@cid, @cname, @cadd)";
    SqlCommand cmd = new SqlCommand(query, con);
    cmd.Parameters.AddWithValue("@cid", txtCid.Text);
    cmd.Parameters.AddWithValue("@cname", txtCname.Text);
    cmd.Parameters.AddWithValue("@cadd", txtCadd.Text);
    con.Open();
    cmd.ExecuteNonQuery();
    con.Close();
    LoadCustomers();
}

protected void btnUpdate_Click(object sender, EventArgs e)
{
    SqlConnection con = new SqlConnection(connectionString);
    string query = "UPDATE customer_info SET cname=@cname, cadd=@cadd WHERE cid=@cid";
    SqlCommand cmd = new SqlCommand(query, con);
    cmd.Parameters.AddWithValue("@cid", txtCid.Text);
    cmd.Parameters.AddWithValue("@cname", txtCname.Text);
    cmd.Parameters.AddWithValue("@cadd", txtCadd.Text);
    con.Open();
    cmd.ExecuteNonQuery();
    con.Close();
    LoadCustomers();
}

protected void btnDelete_Click(object sender, EventArgs e)
{
    SqlConnection con = new SqlConnection(connectionString);
    string query = "DELETE FROM customer_info WHERE cid=@cid";
    SqlCommand cmd = new SqlCommand(query, con);
    cmd.Parameters.AddWithValue("@cid", txtCid.Text);
    con.Open();
    cmd.ExecuteNonQuery();
    con.Close();
    LoadCustomers();
}

protected void btnSearch_Click(object sender, EventArgs e)
{
    SqlConnection con = new SqlConnection(connectionString);
    string query = "SELECT * FROM customer_info WHERE cid=@cid";
    SqlCommand cmd = new SqlCommand(query, con);
    cmd.Parameters.AddWithValue("@cid", txtCid.Text);
    con.Open();
    SqlDataReader reader = cmd.ExecuteReader();
    if (reader.Read())
    {
        txtCname.Text = reader["cname"].ToString();
        txtCadd.Text = reader["cadd"].ToString();
```

```
        }
        con.Close();
    }
    private void LoadCustomers()
    {
        SqlConnection con = new SqlConnection(connectionString);
        string query = "SELECT * FROM customer_info";
        SqlCommand cmd = new SqlCommand(query, con);
        SqlDataAdapter sda = new SqlDataAdapter(cmd);
        DataTable dt = new DataTable();
        sda.Fill(dt);
        gvCustomers.DataSource = dt;
        gvCustomers.DataBind();
    }
  }
}
```

Output :



2. Design an ASP.NET web Applications using **Disconnected** Database **architecture** to add, update, delete and search employee information from database table. Consider following schema for emp_info table emp_info(eid primary key, ename, designation, salary)

Employee.aspx :

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Employee.aspx.cs"
Inherits="Practical_3.Question_2.Employee" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
   <title></title>
</head>
<body>
   <form id="form1" runat="server">
      <div>
         <h2>Employee Management</h2>
      Employee ID: <asp:TextBox ID="txtEid" runat="server"></asp:TextBox><br />
      Employee Name: <asp:TextBox ID="txtEname" runat="server"></asp:TextBox><br />
      Designation: <asp:TextBox ID="txtDesignation" runat="server"></asp:TextBox><br />
      Salary: <asp:TextBox ID="txtSalary" runat="server"></asp:TextBox><br />

      <asp:Button ID="btnAdd" runat="server" Text="Add" OnClick="btnAdd_Click" />
      <asp:Button ID="btnUpdate" runat="server" Text="Update" OnClick="btnUpdate_Click" />
      <asp:Button ID="btnDelete" runat="server" Text="Delete" OnClick="btnDelete_Click" />
      <asp:Button ID="btnSearch" runat="server" Text="Search" OnClick="btnSearch_Click" /><br /><br />

      <asp:GridView ID="gvEmployees" runat="server" AutoGenerateColumns="True"></asp:GridView>
      </div>
   </form>
</body>
</html>
```

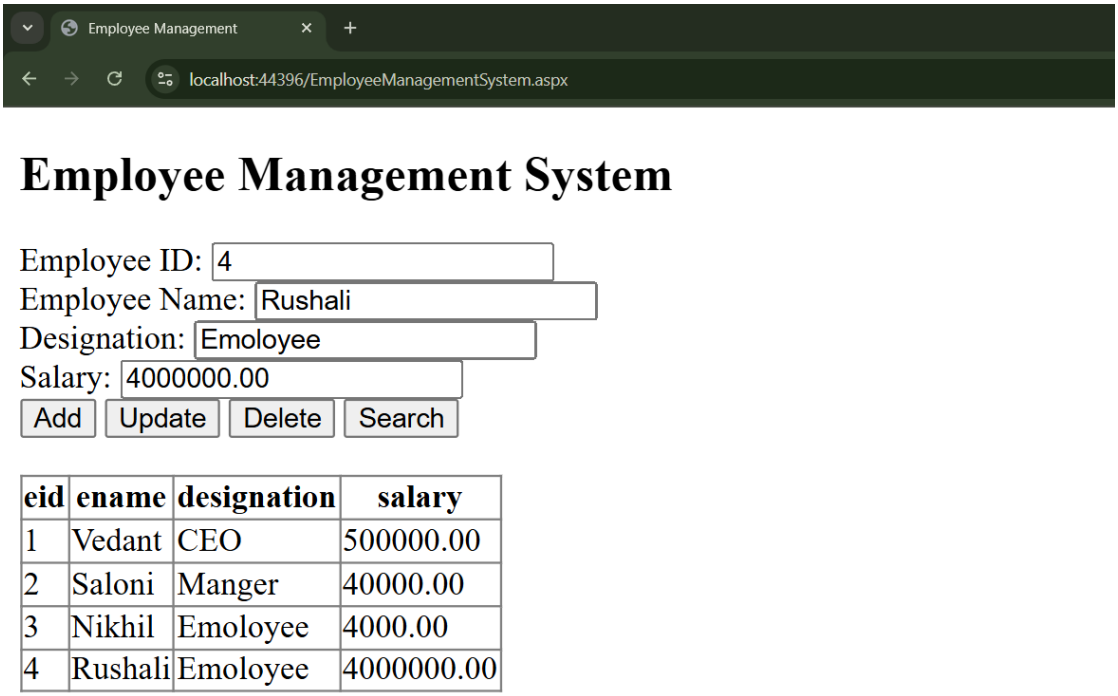
Employee.aspx.cs :

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Configuration;
using System.Data;
using System.Data.SqlClient;

namespace Practical_3.Question_2
{
   public partial class Employee : System.Web.UI.Page
   {
      string connectionString = ConfigurationManager.ConnectionStrings["DBCS2"].ConnectionString;
      DataTable dt = new DataTable();
```

```
SqlDataAdapter adapter;
SqlCommandBuilder builder;

private void LoadEmployees()
{
    dt = new DataTable();
    using (SqlConnection con = new SqlConnection(connectionString))
    {
        string query = "SELECT * FROM emp_info";
        adapter = new SqlDataAdapter(query, con);
        builder = new SqlCommandBuilder(adapter);
        adapter.Fill(dt);
    }
    gvEmployees.DataSource = dt;
    gvEmployees.DataBind();
}
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        LoadEmployees();
    }
}

protected void btnAdd_Click(object sender, EventArgs e)
{
    if (dt == null || dt.Columns.Count == 0)
    {
        LoadEmployees();
    }

    DataRow row = dt.NewRow();
    row["eid"] = txtEid.Text;
    row["ename"] = txtEname.Text;
    row["designation"] = txtDesignation.Text;
    row["salary"] = txtSalary.Text;
    dt.Rows.Add(row);

    adapter.InsertCommand = new SqlCommand("INSERT INTO emp_info (eid, ename, designation, salary) VALUES
(@eid, @ename, @designation, @salary)", new SqlConnection(connectionString));
    adapter.InsertCommand.Parameters.Add("@eid", SqlDbType.Int, 10, "eid");
    adapter.InsertCommand.Parameters.Add("@ename", SqlDbType.NVarChar, 50, "ename");
    adapter.InsertCommand.Parameters.Add("@designation", SqlDbType.NVarChar, 50, "designation");
    adapter.InsertCommand.Parameters.Add("@salary", SqlDbType.Decimal, 18, "salary");

    adapter.Update(dt);
    LoadEmployees();
}

protected void btnUpdate_Click(object sender, EventArgs e)
{
    if (dt == null || dt.Columns.Count == 0)
    {
        LoadEmployees();
    }

    foreach (DataRow row in dt.Rows)
    {
        if (row["eid"].ToString() == txtEid.Text)
        {
```

```
                row["ename"] = txtEname.Text;
                row["designation"] = txtDesignation.Text;
                row["salary"] = txtSalary.Text;
                break;
            }
        }

        adapter.UpdateCommand = new SqlCommand("UPDATE emp_info SET ename=@ename,
designation=@designation, salary=@salary WHERE eid=@eid", new SqlConnection(connectionString));
        adapter.UpdateCommand.Parameters.Add("@eid", SqlDbType.Int, 10, "eid");
        adapter.UpdateCommand.Parameters.Add("@ename", SqlDbType.NVarChar, 50, "ename");
        adapter.UpdateCommand.Parameters.Add("@designation", SqlDbType.NVarChar, 50, "designation");
        adapter.UpdateCommand.Parameters.Add("@salary", SqlDbType.Decimal, 18, "salary");

        adapter.Update(dt);
        LoadEmployees();
    }

    protected void btnDelete_Click(object sender, EventArgs e)
    {
        if (dt == null || dt.Columns.Count == 0)
        {
            LoadEmployees();
        }

        foreach (DataRow row in dt.Rows)
        {
            if (row["eid"].ToString() == txtEid.Text)
            {
                row.Delete();
                break;
            }
        }

        adapter.DeleteCommand = new SqlCommand("DELETE FROM emp_info WHERE eid=@eid", new
SqlConnection(connectionString));
        adapter.DeleteCommand.Parameters.Add("@eid", SqlDbType.Int, 10, "eid");

        adapter.Update(dt);
        LoadEmployees();
    }

    protected void btnSearch_Click(object sender, EventArgs e)
    {
        if (dt == null || dt.Columns.Count == 0)
        {
            LoadEmployees();
        }

        DataRow[] foundRows = dt.Select("eid = '" + txtEid.Text + "'");
        if (foundRows.Length > 0)
        {
            txtEname.Text = foundRows[0]["ename"].ToString();
            txtDesignation.Text = foundRows[0]["designation"].ToString();
            txtSalary.Text = foundRows[0]["salary"].ToString();
        }
    }
}
        }
```

Output :



3. Create **stored procedures** to add, update, delete and search employee information from emp_info table and design asp.net web form to call both procedures using connected architecture.

Employee.aspx :

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Employee.aspx.cs"
Inherits="Practical_3.Question_3.Employee" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
   <title></title>
</head>
<body>
   <form id="form1" runat="server">
      <div>
         <h2>Employee Management</h2>

      Employee ID: <asp:TextBox ID="txtEid" runat="server"></asp:TextBox><br />
      Name: <asp:TextBox ID="txtEname" runat="server"></asp:TextBox><br />
      Designation: <asp:TextBox ID="txtDesignation" runat="server"></asp:TextBox><br />
      Salary: <asp:TextBox ID="txtSalary" runat="server"></asp:TextBox><br />

      <asp:Button ID="btnAdd" runat="server" Text="Add_using_procedure" OnClick="btnAdd_Click" />
```

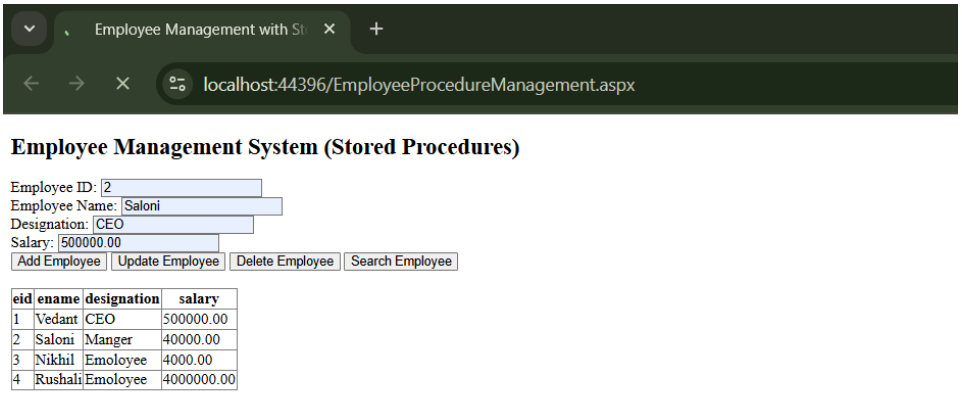
```
<asp:Button ID="btnUpdate" runat="server" Text="Update_using_procedure" OnClick="btnUpdate_Click" />
<asp:Button ID="btnDelete" runat="server" Text="Delete_using_procedure" OnClick="btnDelete_Click" />
<asp:Button ID="btnSearch" runat="server" Text="Search_using_procedure" OnClick="btnSearch_Click" /><br /><br
/>



asp:GridView ID="gvEmployees" runat="server" AutoGenerateColumns="True"></asp:GridView>
</div>
</form>
</body>
</html>
```

Employee.aspx.cs :

```
using System;
using System.Collections.Generic; using System.Linq;
using System.Web; using System.Web.UI;
using System.Web.UI.WebControls; using System.Configuration;
using System.Data;
using System.Data.SqlClient;

namespace Practical_3.Question_3
{
public partial class Employee : System.Web.UI.Page
{
string connectionString = ConfigurationManager.ConnectionStrings["DBCS3"].ConnectionString;

private void LoadEmployees()
{
using (SqlConnection con = new SqlConnection(connectionString))
{
SqlDataAdapter adapter = new SqlDataAdapter("SELECT * FROM emp_info", con); DataTable dt = new DataTable();
adapter.Fill(dt); gvEmployees.DataSource = dt; gvEmployees.DataBind();
}
}
protected void Page_Load(object sender, EventArgs e)
{
if (!IsPostBack)
{
LoadEmployees();
}
}

protected void btnAdd_Click(object sender, EventArgs e)
{
SqlConnection con = new SqlConnection(connectionString); SqlCommand cmd = new SqlCommand("sp_AddEmployee", con);
cmd.CommandType = CommandType.StoredProcedure; cmd.Parameters.AddWithValue("@eid", txtEid.Text);
cmd.Parameters.AddWithValue("@ename", txtEname.Text); cmd.Parameters.AddWithValue("@designation", txtDesignation.Text);
cmd.Parameters.AddWithValue("@salary", txtSalary.Text); con.Open();
cmd.ExecuteNonQuery(); con.Close(); LoadEmployees();
}
```

```
protected void btnUpdate_Click(object sender, EventArgs e)
{
SqlConnection con = new SqlConnection(connectionString); SqlCommand cmd = new SqlCommand("sp_UpdateEmployee", con);
cmd.CommandType = CommandType.StoredProcedure; cmd.Parameters.AddWithValue("@eid", txtEid.Text);
cmd.Parameters.AddWithValue("@ename", txtEname.Text); cmd.Parameters.AddWithValue("@designation", txtDesignation.Text);
cmd.Parameters.AddWithValue("@salary", txtSalary.Text); con.Open();
cmd.ExecuteNonQuery(); con.Close(); LoadEmployees();
}

protected void btnDelete_Click(object sender, EventArgs e)
{
SqlConnection con = new SqlConnection(connectionString); SqlCommand cmd = new SqlCommand("sp_DeleteEmployee", con);
cmd.CommandType = CommandType.StoredProcedure; cmd.Parameters.AddWithValue("@eid", txtEid.Text);
con.Open(); cmd.ExecuteNonQuery(); con.Close(); LoadEmployees();
}

protected void btnSearch_Click(object sender, EventArgs e)
{
SqlConnection con = new SqlConnection(connectionString); SqlCommand cmd = new SqlCommand("sp_SearchEmployee", con);
cmd.CommandType = CommandType.StoredProcedure; cmd.Parameters.AddWithValue("@eid", txtEid.Text);
con.Open();
SqlDataReader reader = cmd.ExecuteReader(); if (reader.Read())
{
txtEname.Text = reader["ename"].ToString(); txtDesignation.Text = reader["designation"].ToString(); txtSalary.Text =
reader["salary"].ToString();
}
con.Close();
}
}
}
```

Output :



4. Design asp.net web application to demonstrate use of DataList, DetailsView, FormView, GridView, ListView and Repeater Data Bound Controls. (Use ProductDetails table).

Product.aspx :

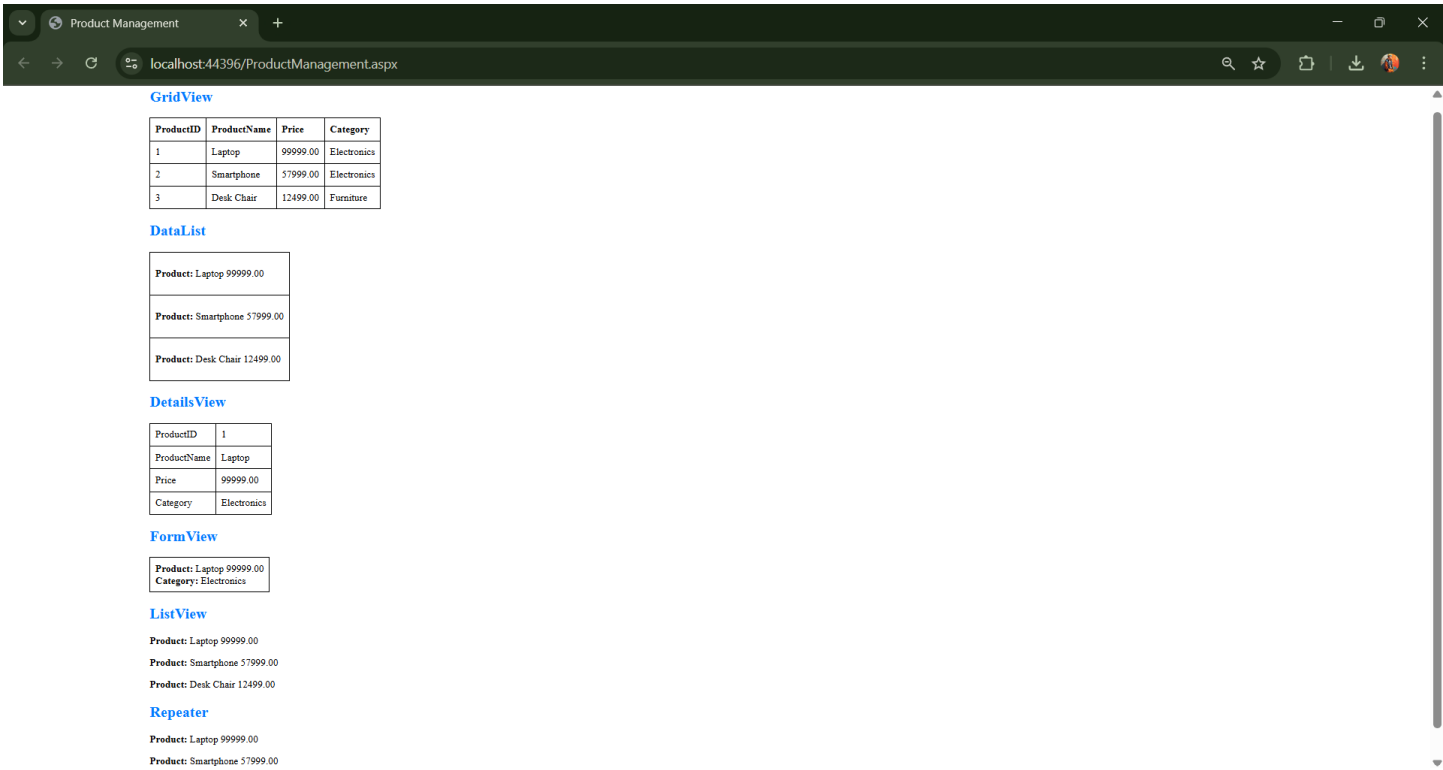
```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Product.aspx.cs"
Inherits="Practical_3.Question_4.Product" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <asp:DataList ID="DataList1" runat="server" DataKeyField="ProductID"
DataSourceID="SqlDataSource1">
                <ItemTemplate>
                    ProductID:
                    <asp:Label ID="ProductIDLabel" runat="server" Text='<%#
Eval("ProductID") %>' />
                    <br />
                    ProductName:
                    <asp:Label ID="ProductNameLabel" runat="server" Text='<%#
```

```
Eval("ProductName") %>' />
                    <br />
                    Price:
                    <asp:Label ID="PriceLabel" runat="server" Text='<%# Eval("Price") %>'
/>
                    <br />
                    Category:
                    <asp:Label ID="CategoryLabel" runat="server" Text='<%#
Eval("Category") %>' />
                    <br />
<br />
                </ItemTemplate>
            </asp:DataList>
            <asp:SqlDataSource ID="SqlDataSource1" runat="server"
ConnectionString="<%$ ConnectionStrings:ConnectionString %>" SelectCommand="SELECT * FROM
[ProductDetails]"></asp:SqlDataSource>
            <br />
            <br />
            <asp:DetailsView ID="DetailsView1" runat="server" AutoGenerateRows="False"
DataKeyNames="ProductID" DataSourceID="SqlDataSource1" Height="50px" Width="125px">
                <Fields>
                    <asp:BoundField DataField="ProductID" HeaderText="ProductID"
ReadOnly="True" SortExpression="ProductID" />
                    <asp:BoundField DataField="ProductName" HeaderText="ProductName"
SortExpression="ProductName" />
                    <asp:BoundField DataField="Price" HeaderText="Price"
SortExpression="Price" />
                    <asp:BoundField DataField="Category" HeaderText="Category"
SortExpression="Category" />
                </Fields>
            </asp:DetailsView>
            <br />
            <br />
            <asp:FormView ID="FormView1" runat="server" DataKeyNames="ProductID"
DataSourceID="SqlDataSource1">
                <EditItemTemplate>
                    ProductID:
                    <asp:Label ID="ProductIDLabel1" runat="server" Text='<%#
Eval("ProductID") %>' />
                    <br />
                    ProductName:
                    <asp:TextBox ID="ProductNameTextBox" runat="server" Text='<%#
Bind("ProductName") %>' />
                    <br />
                    Price:
                    <asp:TextBox ID="PriceTextBox" runat="server" Text='<%#
Bind("Price") %>' />
                    <br />
                    Category:
                    <asp:TextBox ID="CategoryTextBox" runat="server" Text='<%#
Bind("Category") %>' />
                    <br />
                    <asp:LinkButton ID="UpdateButton" runat="server"
CausesValidation="True" CommandName="Update" Text="Update" />
                     <asp:LinkButton ID="UpdateCancelButton" runat="server"
CausesValidation="False" CommandName="Cancel" Text="Cancel" />
                </EditItemTemplate>
                <InsertItemTemplate>
                    ProductID:
```

```
                    <asp:TextBox ID="ProductIDTextBox" runat="server" Text='<%#
Bind("ProductID") %>' />
                    <br />
                    ProductName:
                    <asp:TextBox ID="ProductNameTextBox" runat="server" Text='<%#
Bind("ProductName") %>' />
                    <br />
                    Price:
                    <asp:TextBox ID="PriceTextBox" runat="server" Text='<%#
Bind("Price") %>' />
                    <br />
                    Category:
                    <asp:TextBox ID="CategoryTextBox" runat="server" Text='<%#
Bind("Category") %>' />
                    <br />
                    <asp:LinkButton ID="InsertButton" runat="server"
CausesValidation="True" CommandName="Insert" Text="Insert" />
                     <asp:LinkButton ID="InsertCancelButton" runat="server"
CausesValidation="False" CommandName="Cancel" Text="Cancel" />
                </InsertItemTemplate>
                <ItemTemplate>
                    ProductID:
                    <asp:Label ID="ProductIDLabel" runat="server" Text='<%#
Eval("ProductID") %>' />
                    <br />
                    ProductName:
                    <asp:Label ID="ProductNameLabel" runat="server" Text='<%#
Bind("ProductName") %>' />
                    <br />
                    Price:
                    <asp:Label ID="PriceLabel" runat="server" Text='<%# Bind("Price") %>'
/>
                    <br />
                    Category:
                    <asp:Label ID="CategoryLabel" runat="server" Text='<%#
Bind("Category") %>' />
                    <br />

                </ItemTemplate>
            </asp:FormView>
            <br />
            <br />
            <asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="False"
DataKeyNames="ProductID" DataSourceID="SqlDataSource1">
                <Columns>
                    <asp:BoundField DataField="ProductID" HeaderText="ProductID"
ReadOnly="True" SortExpression="ProductID" />
                    <asp:BoundField DataField="ProductName" HeaderText="ProductName"
SortExpression="ProductName" />
                    <asp:BoundField DataField="Price" HeaderText="Price"
SortExpression="Price" />
                    <asp:BoundField DataField="Category" HeaderText="Category"
SortExpression="Category" />
                </Columns>
            </asp:GridView>
            <br />
            <br />
            <asp:ListView ID="ListView1" runat="server" DataSourceID="SqlDataSource1">
    <ItemTemplate>
```

```
        ProductID:
        <asp:Label ID="ProductIDLabel" runat="server" Text='<%# Eval("ProductID") %>' />
        <br />
        ProductName:
        <asp:Label ID="ProductNameLabel" runat="server" Text='<%# Eval("ProductName") %>'
/>
        <br />
        Price:
        <asp:Label ID="PriceLabel" runat="server" Text='<%# Eval("Price") %>' />
        <br />
        Category:
        <asp:Label ID="CategoryLabel" runat="server" Text='<%# Eval("Category") %>' />
        <br />
        <hr />
    </ItemTemplate>
</asp:ListView>
            <br />
            <br />
            <asp:Repeater ID="Repeater1" runat="server" DataSourceID="SqlDataSource1">
            </asp:Repeater>
        </div>
    </form>
</body>
</html>
```

Output :


Product Management
localhost:44396/ProductManagement.aspx
GridView
ProductID ProductName Price Category
1 Laptop 99999.00 Electronics
2 Smartphone 57999.00 Electronics
3 Desk Chair 12499.00 Furniture
DataList
Product: Laptop 99999.00
Product: Smartphone 57999.00
Product: Desk Chair 12499.00
DetailsView
ProductID 1
ProductName Laptop
Price 99999.00
Category Electronics
FormView
Product: Laptop 99999.00
Category: Electronics
ListView
Product: Laptop 99999.00
Product: Smartphone 57999.00
Product: Desk Chair 12499.00
Repeater
Product: Laptop 99999.00
Product: Smartphone 57999.00

5. Design asp.net web form to find odd numbers from integer Array using **LINQ**.

FindOddNumbers.aspx :

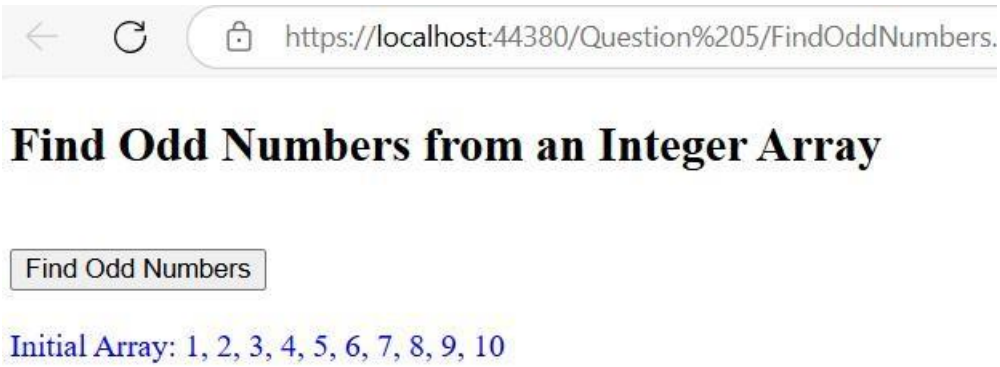
```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="FindOddNumbers.aspx.cs"
Inherits="Practical_3.Question_5.FindOddNumbers" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <h2>Find Odd Numbers from an Integer Array</h2>
            <br />
            <asp:Button ID="btnFindOdd" runat="server" Text="Find Odd Numbers"
OnClick="btnFindOdd_Click" />
            <br /><br />
            <asp:Label ID="lblResult" runat="server" ForeColor="Blue"></asp:Label>
        </div>
    </form>
</body>
</html>
```

FindOddNumbers.aspx.cs :

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace Practical_3.Question_5
{
    public partial class FindOddNumbers : System.Web.UI.Page
    {
        // Define an integer array
        int[] numbers = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
        protected void Page_Load(object sender, EventArgs e)
        {
            if (!IsPostBack)
            {
                lblResult.Text = "Initial Array: " + string.Join(", ", numbers);
            }
        }

        protected void btnFindOdd_Click(object sender, EventArgs e)
        {
            // Use LINQ to filter odd numbers
            var oddNumbers = numbers.Where(n => n % 2 != 0);
```

```
        // Display result
        lblResult.Text = "Odd Numbers: " + string.Join(", ", oddNumbers);
    }
}
    }
```

Output :



6. Design asp.net web application to perform add, update, delete and search operations on emp_info Table using **LINQ to SQL**.

Question 6.aspx :

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Question06.aspx.cs"
Inherits="Practical_03.Question06" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<title></title>
</head>
<body>
<form id="form1" runat="server">
<div>
<h2>Employee Information Management LINQ to SQL</h2>
<asp:Label ID="Label1" runat="server" Text="Enter Employee ID: "></asp:Label>
<asp:TextBox ID="txtEid" runat="server"></asp:TextBox>
<br />
<asp:Label ID="Label2" runat="server" Text="Enter Employee Name: "></asp:Label>
<asp:TextBox ID="txtEname" runat="server"></asp:TextBox>
<br />
<asp:Label ID="Label3" runat="server" Text="Enter Designation: "></asp:Label>
<asp:TextBox ID="txtDesignation" runat="server"></asp:TextBox>
<br />
<asp:Label ID="Label4" runat="server" Text="Enter Salary: "></asp:Label>
<asp:TextBox ID="txtSalary" runat="server"></asp:TextBox>
<br />
<asp:Label ID="lblMessage" runat="server" ForeColor="Red"></asp:Label>
<br />
<asp:GridView ID="GridView1" runat="server"></asp:GridView>
```

```
<br />
<asp:Button ID="addbtn" runat="server" Text="Add" OnClick="addbtn_Click" />
<asp:Button ID="dltbtn" runat="server" Text="Delete" OnClick="dltbtn_Click" />
<asp:Button ID="updatebtn" runat="server" Text="Update" OnClick="updatebtn_Click" />
<asp:Button ID="srchbtn" runat="server" Text="Search" OnClick="srchbtn_Click" />
</div>
</form>
</body>
</html>
```

Question6.aspx.cs :

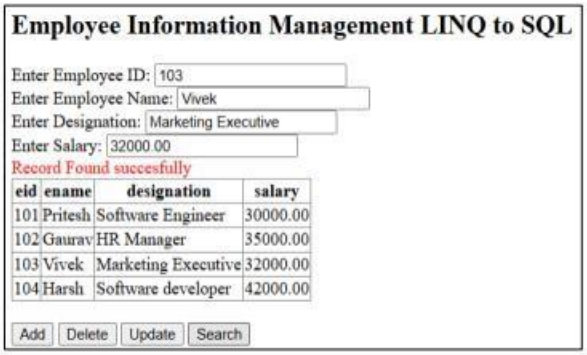
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
namespace Practical_03
{
public partial class Question06 : System.Web.UI.Page
{
DataClasses1DataContext dc = new DataClasses1DataContext();
public void clearTextBox()
{
txtEid.Text = "";
txtEname.Text = "";
txtDesignation.Text = "";
txtSalary.Text = "";
}
public void showData()
{
try
{
var q = from a in dc.GetTable<emp_info>() select a;
GridView1.DataSource = q;
GridView1.DataBind();
}
catch (Exception ex)
{
lblMessage.Text = "Exception Caught !" + ex.Message;
}
}
protected void Page_Load(object sender, EventArgs e)
{
showData();
}
protected void addbtn_Click(object sender, EventArgs e)
{
try
{
emp_info objAdd = new emp_info();
objAdd.eid = Convert.ToByte(txtEid.Text);
objAdd.ename = Convert.ToString(txtEname.Text);
objAdd.designation = Convert.ToString(txtDesignation.Text);
objAdd.salary = Convert.ToDecimal(txtSalary.Text);
dc.emp_infos.InsertOnSubmit(objAdd);
```

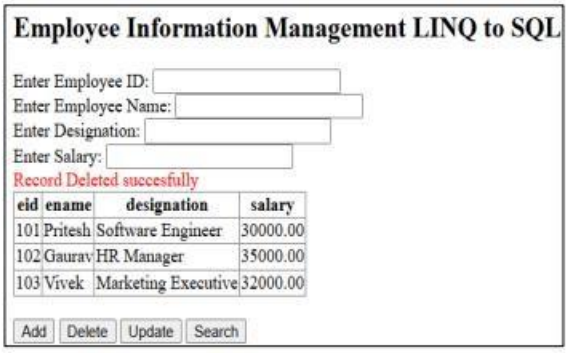
```
dc.SubmitChanges();
lblMessage.Text = "Record Inserted successfully";
clearTextBox();
}
catch (Exception ex)
{
lblMessage.Text = "Exception Caught !" + ex.Message;
}
finally
{
showData();
clearTextBox();
}
}
protected void dltbtn_Click(object sender, EventArgs e)
{
try
{
emp_info objDel = dc.emp_infos.Single(emp_info => emp_info.eid == Convert.ToInt16(txtEid.Text));
if (objDel != null)
{
dc.emp_infos.DeleteOnSubmit(objDel);
dc.SubmitChanges();
lblMessage.Text = "Record Deleted succesfully";
}
else
{
lblMessage.Text = "Record not found";
}
clearTextBox();
}
catch (Exception ex)
{
lblMessage.Text = "Exception Caught !" + ex.Message;
}
finally
{
showData();
clearTextBox();
}
}
protected void updatebtn_Click(object sender, EventArgs e)
{
try
{
emp_info objupdate = dc.emp_infos.Single(emp_info => emp_info.eid ==
Convert.ToInt16(txtEid.Text));
if (objupdate != null)
{
objupdate.ename = Convert.ToString(txtEname.Text);
objupdate.designation = Convert.ToString(txtDesignation.Text);
objupdate.salary = Convert.ToDecimal(txtSalary.Text);
dc.SubmitChanges();
lblMessage.Text = "Record Updated succesfully";
}
else
{
lblMessage.Text = "Record not found..!!";
}
}
```

```
catch (Exception ex)
{
lblMessage.Text = "Exception Caught !" + ex.Message;
}
finally
{
showData();
clearTextBox();

}
}
protected void srchbtn_Click(object sender, EventArgs e)
{
try
{
emp_info objSearch = dc.emp_infos.Single(emp_info => emp_info.eid ==
Convert.ToInt16(txtEid.Text));
if (objSearch != null)
{
txtEid.Text = Convert.ToString(objSearch.eid);
txtEname.Text = objSearch.ename;
txtDesignation.Text = objSearch.designation;
txtSalary.Text = Convert.ToString(objSearch.salary);
lblMessage.Text = "Record Found succesfully";
}
else
{
lblMessage.Text = "Record not found..!!";
}
}
catch (Exception ex)
{
lblMessage.Text = "Exception Caught !" + ex.Message;
}
finally
{
showData();
}
}
}
}
```

Output :

Output





7. Design asp.net web application to perform add, update, delete and search operations on ProductDetails Table using **Entity Framework**.

Question7.aspx :

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="EntityFramework.aspx.cs"
Inherits="Practical03.EntityFramework" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<title></title>
</head>
<body>
<form id="form2" runat="server">
<div>
<h1>Databse Access Using Entity Framework</h1>
<asp:Label ID="lblid" runat="server" Text="Enter Id : "></asp:Label>
<asp:TextBox ID="txtid" runat="server"></asp:TextBox><br />
<asp:Label ID="lblname" runat="server" Text="Enter Product Name : "></asp:Label>
<asp:TextBox ID="txtname" runat="server"></asp:TextBox><br />
<asp:Label ID="lblprice" runat="server" Text="Enter Product Price : "></asp:Label>
<asp:TextBox ID="txtprice" runat="server"></asp:TextBox><br />
<asp:Label ID="lbldes" runat="server" Text="Enter Product Description : "></asp:Label>
<asp:TextBox ID="txtdes" runat="server"></asp:TextBox><br />
<asp:GridView ID="GridView1" runat="server"></asp:GridView><br />
<asp:Label ID="lblmessage" runat="server" Text="Label"></asp:Label><br />
<asp:Button ID="btnadd" runat="server" Text="Add" OnClick="btnadd_Click" />
<asp:Button ID="brndel" runat="server" Text="Delete" OnClick="brndel_Click" />
<asp:Button ID="btnup" runat="server" Text="Update" OnClick="btnup_Click" />
<asp:Button ID="btnsearch" runat="server" Text="Search" OnClick="btnsearch_Click" />
</div>
</form>
</body>
</html>
```

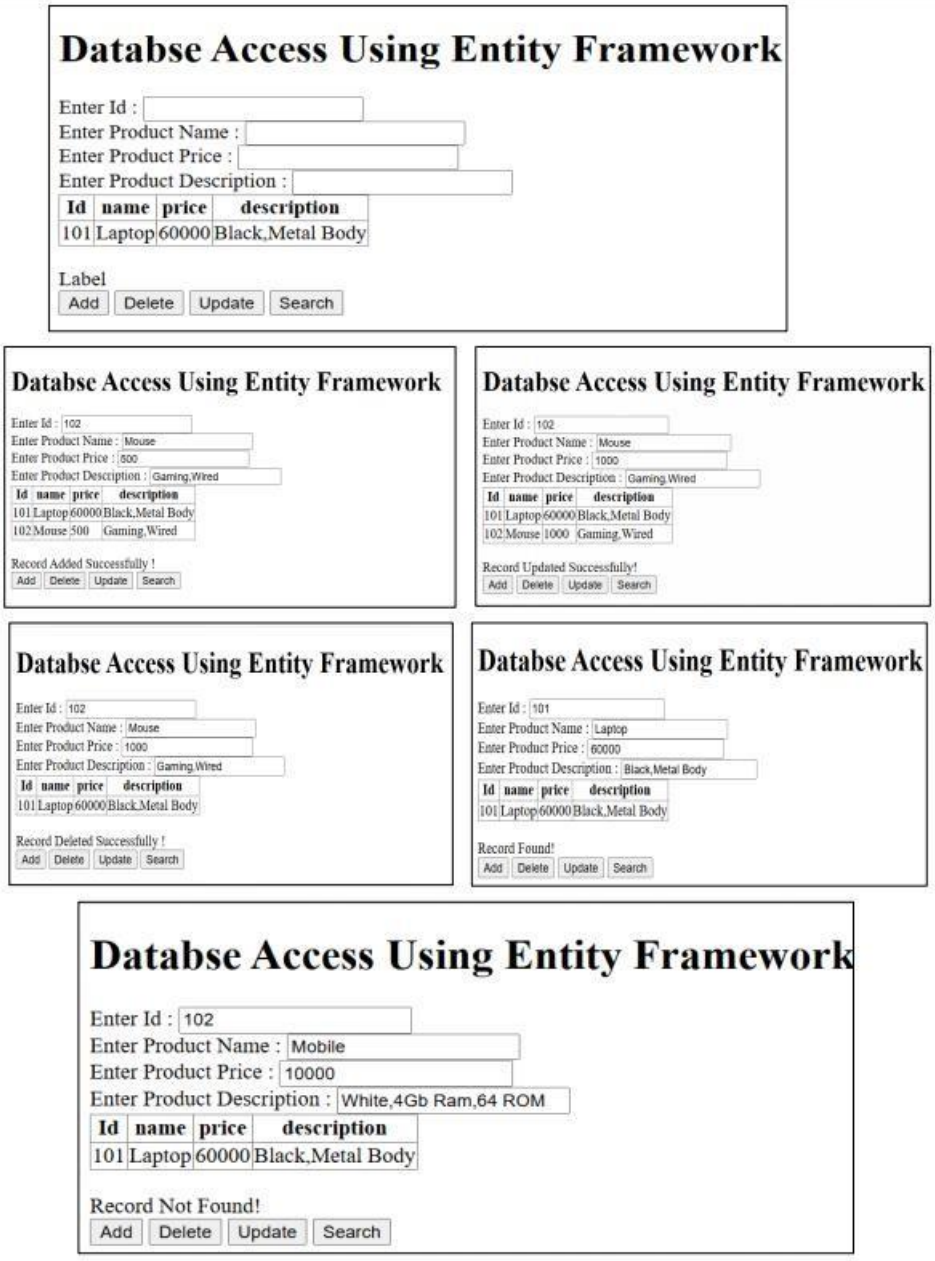
Question7.aspx.cs :

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
namespace Practical03
{
public partial class EntityFramework : System.Web.UI.Page
{
ProductDBEntities db = new ProductDBEntities();
public void showdata()
{
try
{
```

```
GridView1.DataSource = db.ProductDetails.ToList();
GridView1.DataBind();
}
catch (Exception ex)
{
lblmessage.Text = "Exception Caught !" + ex.Message;
}
}
protected void Page_Load(object sender, EventArgs e)
{
showdata();
}
protected void btnadd_Click(object sender, EventArgs e)
{
try
{
ProductDetail productObjAdd = new ProductDetail();
productObjAdd.Id = Convert.ToInt32(txtid.Text);
productObjAdd.name = txtname.Text;
productObjAdd.price = Convert.ToInt32(txtprice.Text);
productObjAdd.description = txtdes.Text;
db.ProductDetails.Add(productObjAdd);
db.SaveChanges();
lblmessage.Text = "Record Added Successfully !";
showdata();
}
catch (Exception ex)
{
lblmessage.Text = "Exception caught !" + ex.Message;
}
}
protected void brndel_Click(object sender, EventArgs e)
{
try
{
ProductDetail productObjDelete = db.ProductDetails.Find(Convert.ToInt32(txtid.Text));
if (productObjDelete != null)
{
db.ProductDetails.Remove(productObjDelete);
db.SaveChanges();
lblmessage.Text = "Record Deleted Successfully !";
showdata();
}
else
{
lblmessage.Text = "Record Not Found !";
}
}
catch (Exception ex)
{
lblmessage.Text = "Exception Caught!" + ex.Message;
}
}
protected void btnup_Click(object sender, EventArgs e)
{
try
{
int productId = Convert.ToInt32(txtid.Text);
ProductDetail productObjUpdate = db.ProductDetails.Find(productId);
if (productObjUpdate != null)
```

```
{
productObjUpdate.name = txtname.Text;
productObjUpdate.price = Convert.ToInt32(txtprice.Text);
productObjUpdate.description = txtdes.Text;
db.SaveChanges();
lblmessage.Text = "Record Updated Successfully!";
}
Else
{
lblmessage.Text = "Record Not Found!";
}
showdata();
}
catch (Exception ex)
{
lblmessage.Text = "Exception Caught! " + ex.Message;
}
}
protected void btnsearch_Click(object sender, EventArgs e)
{
try
{
int productId = Convert.ToInt32(txtid.Text);
ProductDetail productObjSearch = db.ProductDetails.Find(productId);
if (productObjSearch != null)
{
txtname.Text = productObjSearch.name;
txtprice.Text = productObjSearch.price.ToString();
txtdes.Text = productObjSearch.description;
lblmessage.Text = "Record Found!";
}
else
{
lblmessage.Text = "Record Not Found!";
}
}
catch (Exception ex)
{
lblmessage.Text = "Exception Caught! " + ex.Message;
}
}
}
}
```

Output :


Databse Access Using Entity Framework
Enter Id :
Enter Product Name :
Enter Product Price :
Enter Product Description :
Id name price description
101 Laptop 60000 Black,Metal Body
Label
Add Delete Update Search
Databse Access Using Entity Framework
Enter Id : 102
Enter Product Name : Mouse
Enter Product Price : 500
Enter Product Description : Gaming,Wired
Id name price description
101 Laptop 60000 Black,Metal Body
102 Mouse 500 Gaming,Wired
Record Added Successfully !
Add Delete Update Search
Databse Access Using Entity Framework
Enter Id : 102
Enter Product Name : Mouse
Enter Product Price : 1000
Enter Product Description : Gaming,Wired
Id name price description
101 Laptop 60000 Black,Metal Body
102 Mouse 1000 Gaming,Wired
Record Updated Successfully!
Add Delete Update Search
Databse Access Using Entity Framework
Enter Id : 102
Enter Product Name : Mouse
Enter Product Price : 1000
Enter Product Description : Gaming,Wired
Id name price description
101 Laptop 60000 Black,Metal Body
Record Deleted Successfully !
Add Delete Update Search
Databse Access Using Entity Framework
Enter Id : 101
Enter Product Name : Laptop
Enter Product Price : 60000
Enter Product Description : Black,Metal Body
Id name price description
101 Laptop 60000 Black,Metal Body
Record Found!
Add Delete Update Search
Databse Access Using Entity Framework
Enter Id : 102
Enter Product Name : Mobile
Enter Product Price : 10000
Enter Product Description : White,4Gb Ram,64 ROM
Id name price description
101 Laptop 60000 Black,Metal Body
Record Not Found!
Add Delete Update Search