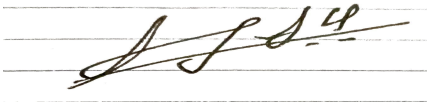
Name of Student: Sunny Satish Halkatti		
Roll Number: 17 (B)		**Lab Assignment Number: 5**
Title of Lab Assignment: Design a web page to demonstrate a connection-oriented architecture & disconnected architecture.		
DOP: 24/04/2023		**DOS: 02/05/2023**
CO: CO1	**PO:** PO3,PO5,PSO1, PSO2	**Faculty Signature:**

Student's Signature



AWTL - Practical-5

Aim: Design a web page to demonstrate a connection-oriented architecture & disconnected architecture.

Theory:

ADO.NET

ADO.NET is a data access technology used in Microsoft's .NET Framework for interacting with databases. It provides a set of classes and libraries that allow developers to create data-aware applications and supports various data sources such as SQL Server, Oracle, MySQL, and more. ADO.NET provides a unified programming model for working with data, including disconnected data access, data binding, and transaction management. It allows developers to access and manipulate data from a database through various objects such as DataReader, DataSet, and DataAdapter, which provide efficient and flexible ways to manage data in memory and interact with data sources. ADO.NET is a powerful tool for building data-driven applications in the .NET ecosystem.

Connected Architecture

Connected architecture is a traditional way of working with data in which a connection to the database is established, and data is accessed and manipulated directly from the database through the connection. In this model, a connection object is created to establish a connection to the database, and a command object is used to execute SQL statements against the database. The data is then read using a data reader, which provides a forward-only, read-only stream of data.

Disconnected Architecture

Disconnected architecture, on the other hand, is a modern approach in which data is first retrieved from the database and then disconnected from the database. This data is then manipulated in memory, and changes are later committed back to the database. In this model, a dataset is used to store data retrieved from the database, and a data adapter is used to fill the dataset with data and later update the database with changes made to the dataset.

Connection Object

In ADO.NET, **a connection object represents a connection to a data source such as a database**. It is used to establish and manage the connection to the data source and provides a communication channel between the application and the data source.

To create a connection object in ADO.NET, a connection string is required that contains information about the data source, such as the server name, database name, user name, and password. Once the connection string is created, a connection object can be instantiated using the SqlConnection class for SQL Server databases, the OracleConnection class for Oracle databases, and so on.

The connection object has various properties that can be set to configure the connection, such as the ConnectionString property to set the connection string, the Timeout property to set the time to wait for a connection to open, and the Database property to get or set the name of the current database.

The connection object also provides methods for opening and closing the connection, such as the Open() and Close() methods. When the connection is open, it can be used to execute SQL commands against the data source using command objects or to retrieve data using data readers or data adapters.

Connection String

In ADO.NET, a connection string is a string of parameters and values that provide information about the data source to connect to, such as a database. It contains key-value pairs that specify the server name, database name, user name, password, and other options required to establish a connection to the data source.
A typical connection string for SQL Server might look like this:
Server=myServerName;Database=myDataBase;UserId=myUsername;Password=myPassword;

Command Object

I**n ADO.NET, a command object represents a SQL statement or stored procedure to be executed against a data source such as a database.** It is used to execute commands against the data source and retrieve results, such as data from a table or the return value from a stored procedure.
To create a command object in ADO.NET, a connection object must be established first, which provides the connection to the data source. Once the connection is established, a command object can be instantiated using the appropriate command class, such as SqlCommand for SQL Server databases or OracleCommand for Oracle databases.
The command object has various properties that can be set to configure the command, such as the CommandText property to set the SQL statement or stored procedure to be executed, the CommandType property to specify the type of command (text or stored

procedure), and the CommandTimeout property to set the time to wait for a command to execute.
The command object also provides methods for executing the command and retrieving results, such as ExecuteNonQuery() to execute a command that does not return any results, ExecuteReader() to execute a command that returns a data reader or ExecuteScalar() to execute a command that returns a single value.

Data Reader

In ADO.NET, a data reader is an object that provides a forward-only, read-only stream of data from a data source such as a database. It is used to efficiently retrieve large amounts of data, such as from a query result, without loading the entire data set into memory.
To create a data reader in ADO.NET, a command object must first be created to execute the SQL query and return a result set. Once the command is executed, a data reader can be instantiated using the ExecuteReader() method of the command object.
The data reader provides methods for reading the data one row at a time, such as Read() to advance to the next row, and properties for accessing the values in each column, such as GetString() to get a string value, GetInt32() to get an integer value, and so on.

Data Adapter

In ADO.NET, a data adapter is an object that serves as a bridge between a data source and a dataset. **It is used to populate a dataset with data from a data source and to update the data source with changes made to the dataset.**
The data adapter works by executing commands against the data source, such as SQL statements or stored procedures, and using the resulting data to fill a dataset with tables and rows. It also provides methods for updating the data source with changes made to the dataset, such as updates, inserts, and deletes.
The data adapter has various properties that can be set to configure the adapter, such as the SelectCommand property to set the SQL statement or stored procedure to retrieve data, and the UpdateCommand property to set the command to update data in the data source.
The data adapter also provides methods for filling a dataset with data, such as Fill() to populate a dataset with the results of a query, and Update() to update the data source with changes made to the dataset.

Datasource

In ADO.NET, a data source refers to the physical location or container of data that can be accessed by an application. This can include a database, a file, a web service, or any other structured data storage system.

A data source is typically represented in ADO.NET by a connection object, which provides the connection to the data source. The connection object contains information about the location of the data source, such as the server name, database name, file path, or URL, as well as authentication information, such as a username and password.

Once a connection to a data source is established, various objects in ADO.NET can be used to access and manipulate the data, such as command objects, data readers, and data adapters.

Dataset

In ADO.NET, a dataset is an in-memory representation of a set of related tables and their relationships. It provides a disconnected, cache-like representation of data that can be manipulated and queried without a live connection to the data source.

A dataset can be thought of as a virtual database that can be queried and updated in much the same way as a physical database. It consists of a collection of DataTable objects that represent the individual tables in the dataset, and DataRelation objects that define the relationships between the tables.

The dataset can be populated with data from a data source using a data adapter, which fills the dataset with the results of a SQL query or stored procedure. The data adapter can also be used to update the data source with changes made to the dataset.

Once a dataset is populated, it can be queried and manipulated using LINQ to DataSet, which provides a powerful query syntax for querying and manipulating data in a dataset.

Design a web page to demonstrate a connection-oriented architecture.

A) Fetch Student details from a database such as Roll_no, Name, Program(eg. MCA), Course(eg. AWT), etc.

Code:

ConnectedDB.aspx

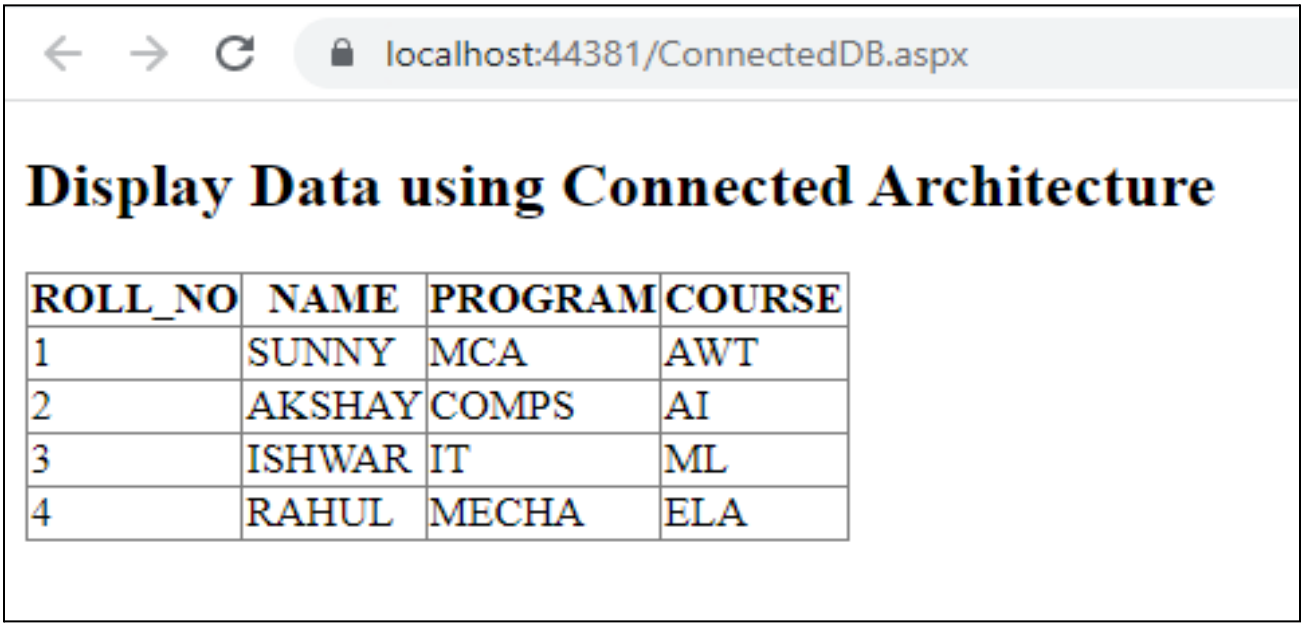
```
<%@ Page Language="C#" AutoEventWireup="true"
CodeBehind="ConnectedDB.aspx.cs" Inherits="Practical5.ConnectedDB" %>

<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
   <title></title>
</head>
<body>
   <form id="form1" runat="server">
      <div class="container">
         <h2>Display Data using Connected Architecture</h2>
         <asp:GridView ID="GridView1" runat="server"></asp:GridView>
      </div>
   </form>
</body>
</html>
```

ConnectedDB.aspx.cs

```
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
namespace Practical5
{
   public partial class ConnectedDB: System.Web.UI.Page
   {
```

```
protected void Page_Load(object sender, EventArgs e)
{
   try
   {
      SqlConnection conn = null;
      string connectionstring = "Data
Source=MCAB12-1\\SQLEXPRESS;Database=user17b;Integrated security=True";
      conn = new SqlConnection(connectionstring);
      {
         SqlCommand query = new SqlCommand("SELECT * FROM STUDENT",
conn);
         conn.Open();
         SqlDataReader rdr = query.ExecuteReader();
         GridView1.DataSource = rdr;
         GridView1.DataBind();
      }
      }
   catch (Exception ex)
   {
      Console.WriteLine(ex.ToString());
   }} }}
```

Output:



Design a web page to demonstrate a disconnected architecture.

B) Fetch Employee details from a database such as emp_id, Name, Designation(eg. Manager), Dept(eg. Sales), etc.

Code:

UnconnectedDB.aspx

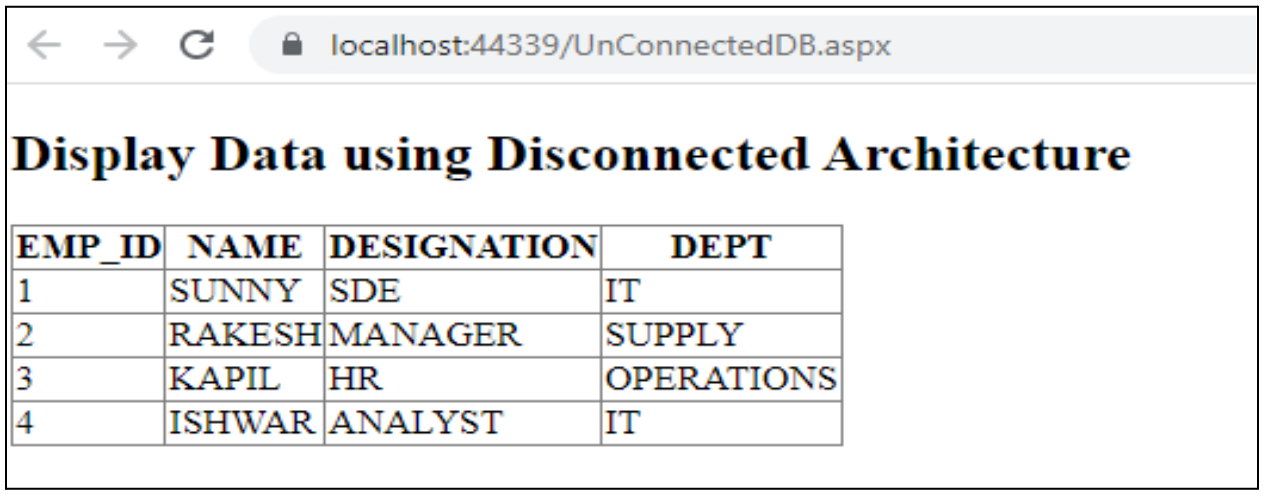
```
<%@ Page Language="C#" AutoEventWireup="true"
CodeBehind="UnConnectedDB.aspx.cs" Inherits="Practical5b.UnConnectedDB" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
   <title></title>
</head>
<body>
   <form id="form1" runat="server">
      <div class="container">
         <h2>Display Data using Disconnected Architecture</h2>
         <asp:GridView ID="GridView1" runat="server"></asp:GridView>
      </div>
   </form>
</body>
</html>
```

UnconnectedDB.aspx.cs

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.SqlClient;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
namespace Practical5b
{
   public partial class UnConnectedDB: System.Web.UI.Page
   {
```

```
    protected void Page_Load(object sender, EventArgs e)
    {
        try
        {
            SqlConnection conn = null;
            string connectionstring = "Data
Source=MCAB12-1\\SQLEXPRESS;Database=user17b;Integrated security=True";
            conn = new SqlConnection(connectionstring);
            {
                string query = "Select * from EMP";
                SqlDataAdapter da = new SqlDataAdapter(query, conn);
                DataSet ds = new DataSet();
                da.Fill(ds);
                GridView1.DataSource = ds;
                GridView1.DataBind();
            }

        }
        catch (Exception ex)
        {
            Console.WriteLine(ex.ToString());
        }
}}}
```

Output:



Conclusions:

We have successfully implemented connected & disconnected-oriented architecture in this practical.