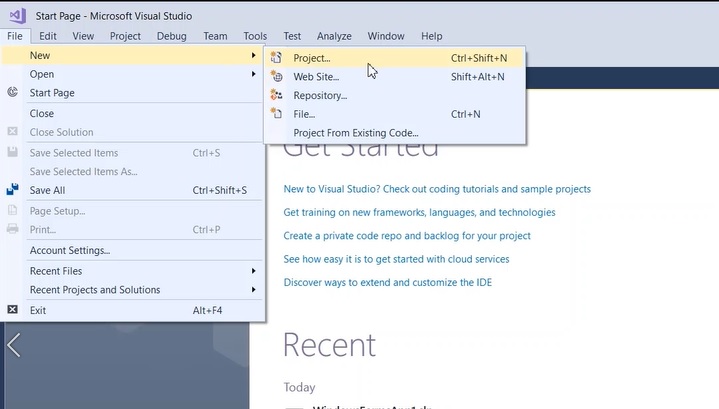
**Lab 11: DB CONNECTIVITY WITH ASP.NET CRUD APP**

**This is reference of windows form application however your task is to create WEB FORM asp.net application as practiced in class**

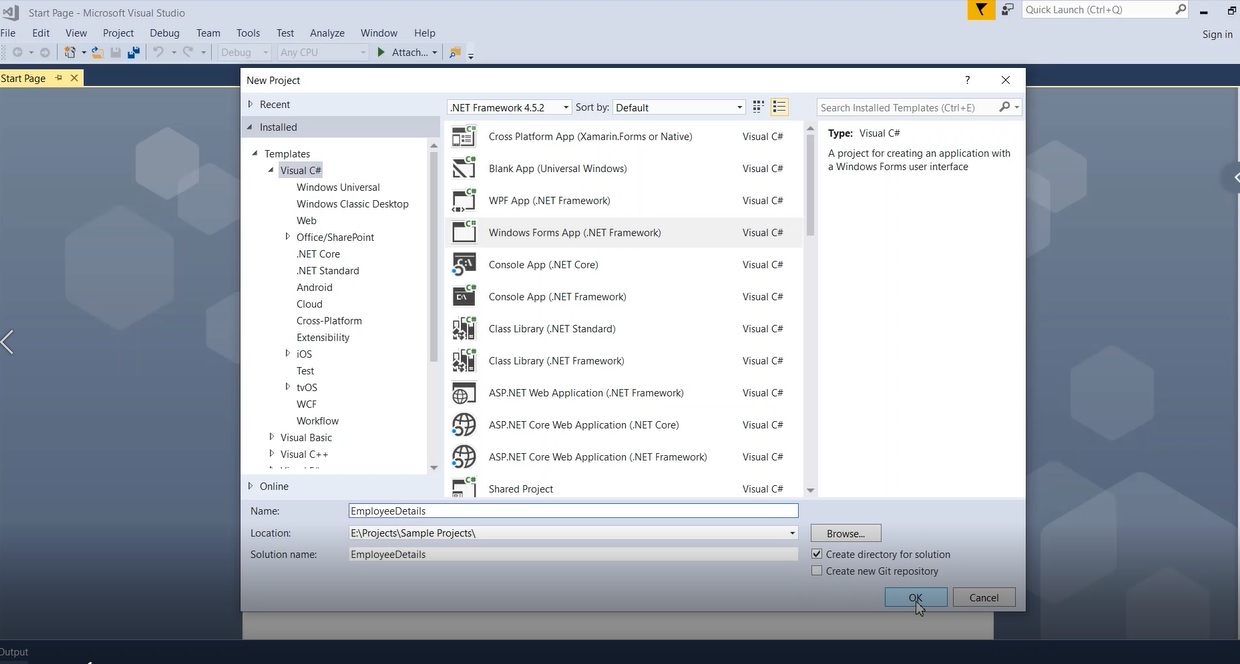
**Step 1:**

Open Visual Studio > File > New > Click **Project.**



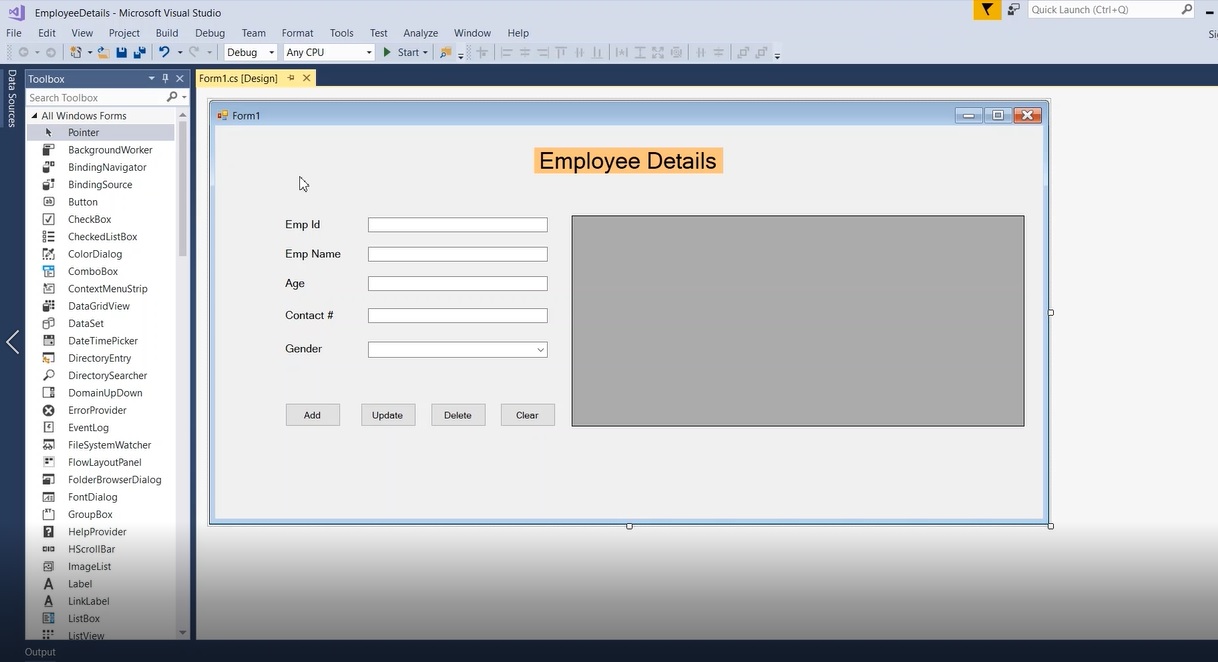
**Step 2:**

In New Project dialog, Select Visual C# templates > Choose .Net Framwork > Select "Windown Forms App (.Net Framework)"



**Step 3:**

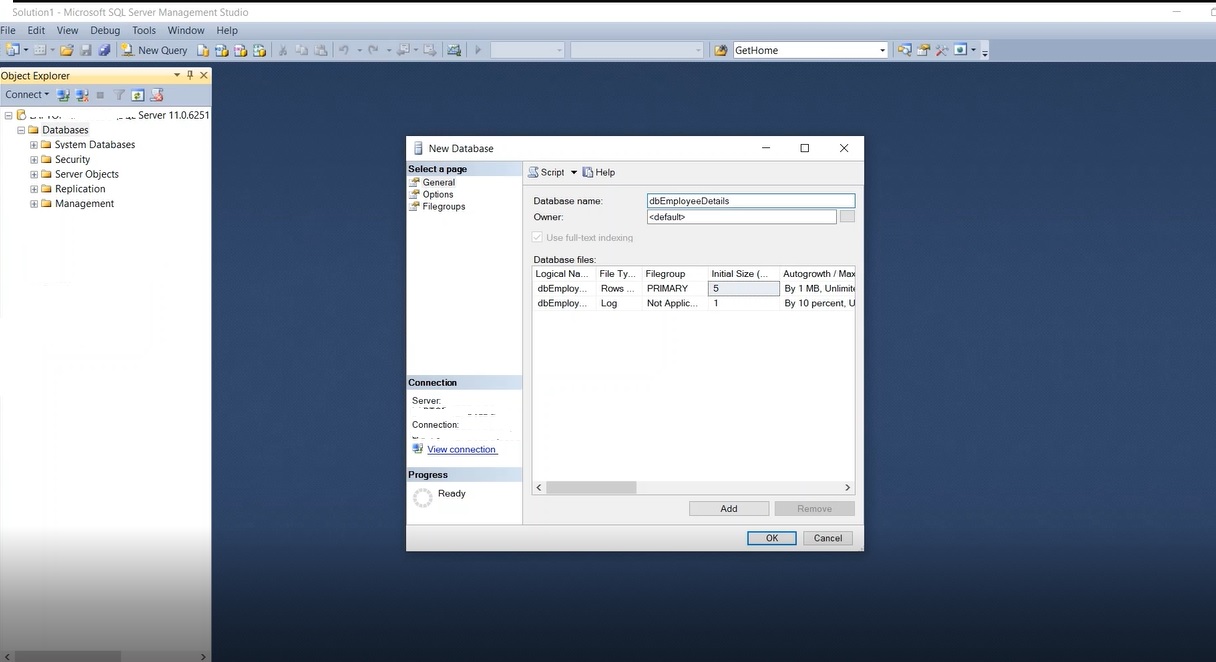
Once the application created successfully, We will start adding controls from the **Toolbox** section as shown below and provide proper naming for controls.



**Step 4:**

Next, We will create a database and table in SQL Server. Open "Microsoft SQL Server Management Studio" > Right Click "**Database**" folder > Click "New Database".

The below "New Database" dialog, provide a new database name as "**dbEmployeeDetails**" and click **OK.**



**Step 5:**

Run below new table creation query in SQL window to create a new Emp\_details table.

CREATE TABLE [dbo].[Emp\_details](

[EmpId] [int] NOT NULL,

[EmpName] [nvarchar](50) NULL,

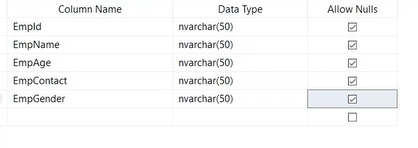
[EmpAge] [int] NULL,

[EmpContact] [int] NULL,

[EmpGender] [nchar](10) NULL

)

You can also verify table columns by Right Click newly created table and click "Design". The table looks like below in SQL studio,

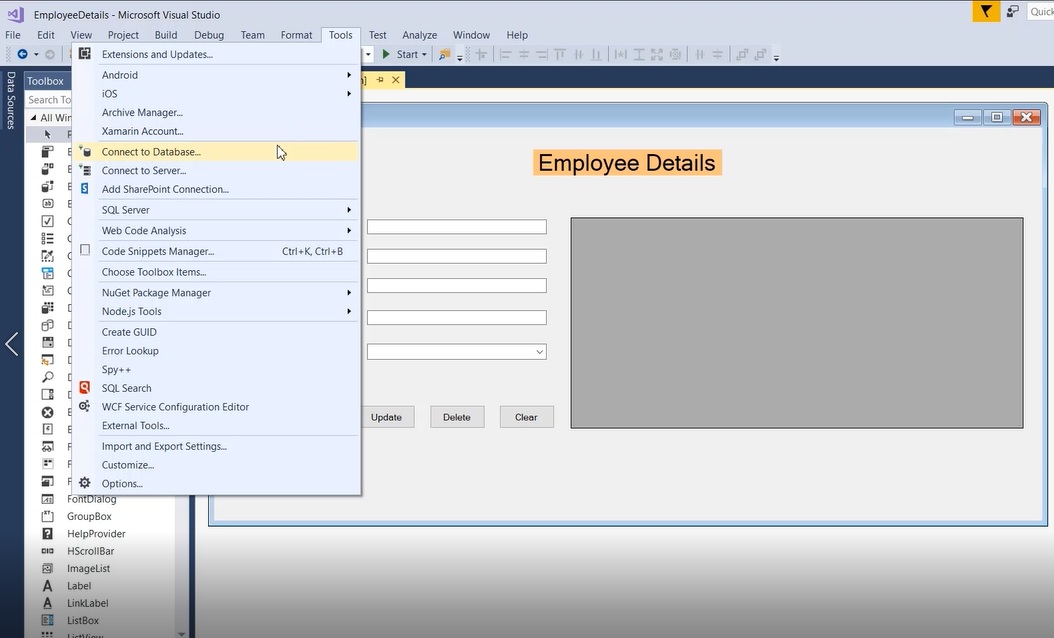


**Step 1:**

How to connect database from Windows application?

We will add database connection easily in windows application in simple steps as shown below,

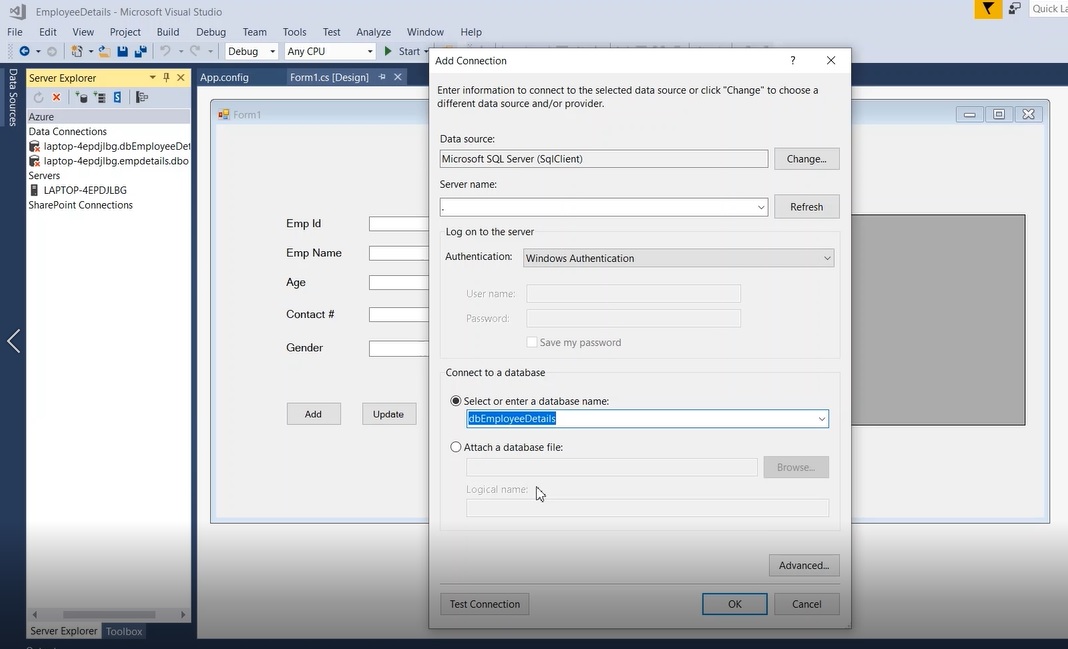
Go to Tools menu > Click "Connect to Database".



**Step 2:**

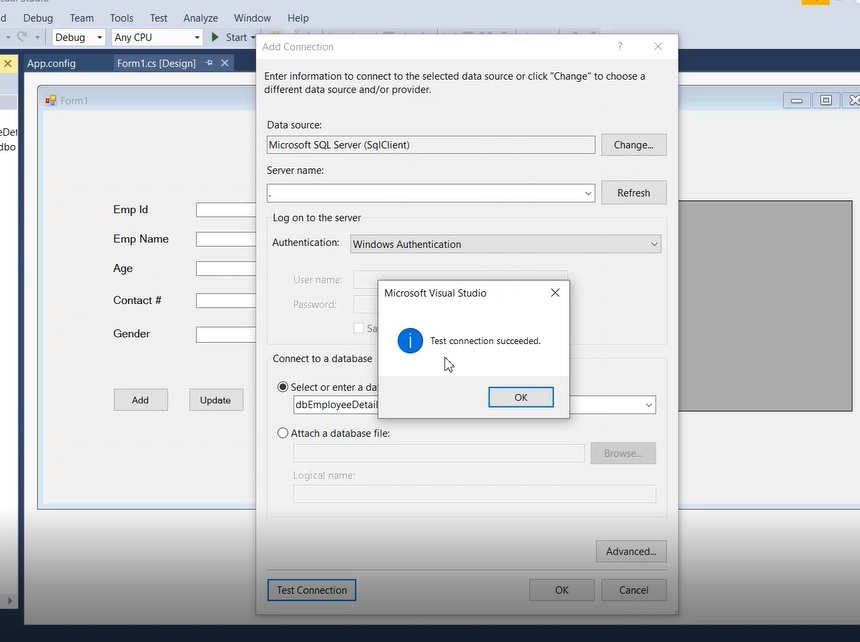
By default, the data source is selected as Microsoft SQL Server (SQL Client).

Provide "Server name" to . or localhost. and Select our database in the dropdown as "dbEmployeeDetails"



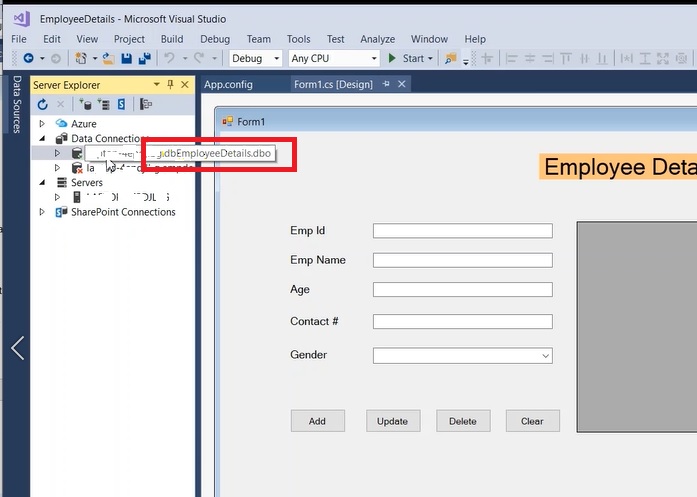
**Step 3:**

We need to verify the database connection by clicking the "Test Connection" button as shown below, Once the connection is verified you will see the "**Test connection succeeded**" message box.



**Step 4:**

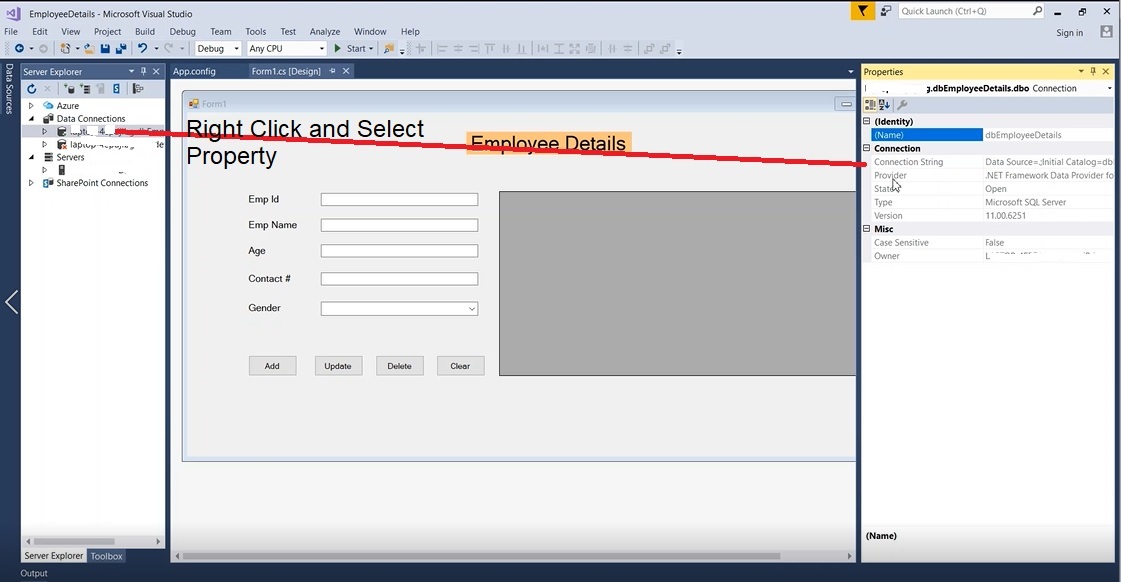
Once the database connection is created, then you will see the connection details as shown below,



**Step 5:**

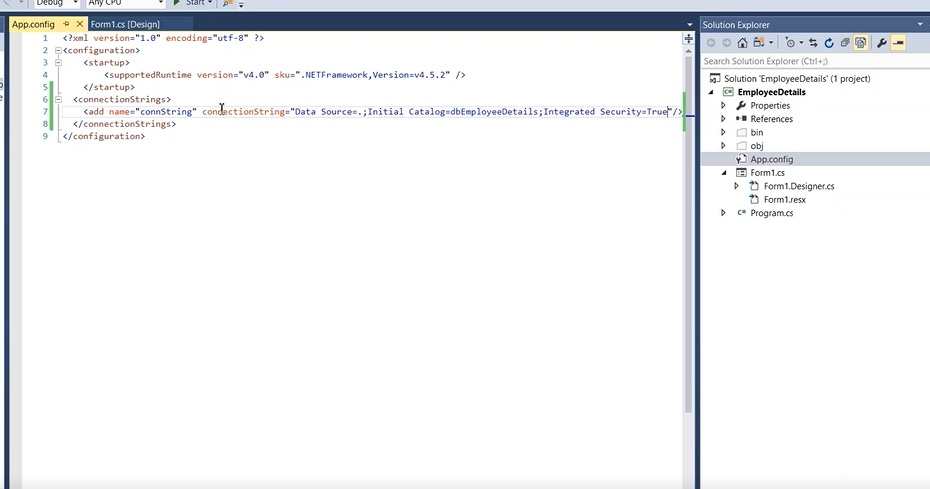
How do I know the connection string?

Right Click database connection > Properties > Copy "Connection string" for our reference as shown below,



**Step 6:**

As a first step, We need to add a connection string in "App.config" file to connect database as shown below,



**Step 7:  Insert Employee Details**

First, we will have to create an Employee class and its properties,

public class Employee

{

public string EmpId { get; set; }

public string EmpName { get; set; }

public string Age { get; set; }

public string ContactNo { get; set; }

}

Add read connection string from App.config using Configuration manager,

private static string myConn = ConfigurationManager.ConnectionStrings["connString"].ConnectionString;

Below method accepts employee details and opens SQL connection and command to insert the employee details in the database using INSERT query as shown below,  if success, then returns true or else false.

private const string InsertQuery = "Insert Into Emp\_details(EmpId, EmpName, EmpAge, EmpContact, EmpGender) Values (@EmpId, @EmpName, @EmpAge, @EmpContact, @EmpGender)";

public bool InsertEmployee(Employee employee)

{

int rows;

using (SqlConnection con = new SqlConnection(myConn))

{

con.Open();

using (SqlCommand com = new SqlCommand(InsertQuery, con))

{

com.Parameters.AddWithValue("@EmpId", employee.EmpId);

com.Parameters.AddWithValue("@EmpName", employee.EmpName);

com.Parameters.AddWithValue("@EmpAge", employee.Age);

com.Parameters.AddWithValue("@EmpContact", employee.ContactNo);

com.Parameters.AddWithValue("@EmpGender", employee.Gender);

rows = com.ExecuteNonQuery();

}

}

return (rows > 0) ? true: false;

}

**Step 8: Update Employee**

The UpdateEmployee method accepts updated employee details and opens SQL connection and updates the employee details into the database using UPDATE query as shown below,  if success, then returns true or else false.

private const string UpdateQuery = "Update Emp\_details set EmpName=@EmpName, EmpAge=@EmpAge, EmpContact=@EmpContact, EmpGender=@EmpGender where EmpId=@EmpId";

public bool UpdateEmployee(Employee employee)

{

int rows;

using (SqlConnection con = new SqlConnection(myConn))

{

con.Open();

using (SqlCommand com = new SqlCommand(UpdateQuery, con))

{

com.Parameters.AddWithValue("@EmpName", employee.EmpName);

com.Parameters.AddWithValue("@EmpAge", employee.Age);

com.Parameters.AddWithValue("@EmpContact", employee.ContactNo);

com.Parameters.AddWithValue("@EmpGender", employee.Gender);

com.Parameters.AddWithValue("@EmpId", employee.EmpId);

rows = com.ExecuteNonQuery();

}

}

return (rows > 0) ? true : false;

}

**Step 9: Delete Employee**

The DeleteEmplyee method passes the employee id to delete the employee details in the database using DELETE query. if success, then returns true or else false.

private const string DeleteQuery = "Delete from Emp\_details where EmpId=@EmpId";

public bool DeleteEmployee(Employee employee)

{

int rows;

using (SqlConnection con = new SqlConnection(myConn))

{

con.Open();

using (SqlCommand com = new SqlCommand(DeleteQuery, con))

{

com.Parameters.AddWithValue("@EmpId", employee.EmpId);

rows = com.ExecuteNonQuery();

}

}

return (rows > 0) ? true : false;

}

**Step 10: Retrieve Employee Details**

The method GetEmplyees returns all the employees which are stored in the database using SELECT query as shows below,

private const string SelectQuery = "Select \* from Emp\_details";

public DataTable GetEmployees()

{

var datatable = new DataTable();

using (SqlConnection con = new SqlConnection(myConn))

{

con.Open();

using(SqlCommand com = new SqlCommand(SelectQuery, con))

{

using(SqlDataAdapter adapter = new SqlDataAdapter(com))

{

adapter.Fill(datatable);

}

}

}

return datatable;

}

The Employee class now look likes below,

using System;

using System.Collections.Generic;

using System.Configuration;

using System.Data;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace EmployeeDetails

{

class Employee

{

private static string myConn = ConfigurationManager.ConnectionStrings["connString"].ConnectionString;

public string EmpId { get; set; }

public string EmpName { get; set; }

public string Age { get; set; }

public string ContactNo { get; set; }

public string Gender { get; set; }

private const string SelectQuery = "Select \* from Emp\_details";

private const string InsertQuery = "Insert Into Emp\_details(EmpId, EmpName, EmpAge, EmpContact, EmpGender) Values (@EmpId, @EmpName, @EmpAge, @EmpContact, @EmpGender)";

private const string UpdateQuery = "Update Emp\_details set EmpName=@EmpName, EmpAge=@EmpAge, EmpContact=@EmpContact, EmpGender=@EmpGender where EmpId=@EmpId";

private const string DeleteQuery = "Delete from Emp\_details where EmpId=@EmpId";

public DataTable GetEmployees()

{

var datatable = new DataTable();

using (SqlConnection con = new SqlConnection(myConn))

{

con.Open();

using(SqlCommand com = new SqlCommand(SelectQuery, con))

{

using(SqlDataAdapter adapter = new SqlDataAdapter(com))

{

adapter.Fill(datatable);

}

}

}

return datatable;

}

public bool InsertEmployee(Employee employee)

{

int rows;

using (SqlConnection con = new SqlConnection(myConn))

{

con.Open();

using (SqlCommand com = new SqlCommand(InsertQuery, con))

{

com.Parameters.AddWithValue("@EmpId", employee.EmpId);

com.Parameters.AddWithValue("@EmpName", employee.EmpName);

com.Parameters.AddWithValue("@EmpAge", employee.Age);

com.Parameters.AddWithValue("@EmpContact", employee.ContactNo);

com.Parameters.AddWithValue("@EmpGender", employee.Gender);

rows = com.ExecuteNonQuery();

}

}

return (rows > 0) ? true: false;

}

public bool UpdateEmployee(Employee employee)

{

int rows;

using (SqlConnection con = new SqlConnection(myConn))

{

con.Open();

using (SqlCommand com = new SqlCommand(UpdateQuery, con))

{

com.Parameters.AddWithValue("@EmpName", employee.EmpName);

com.Parameters.AddWithValue("@EmpAge", employee.Age);

com.Parameters.AddWithValue("@EmpContact", employee.ContactNo);

com.Parameters.AddWithValue("@EmpGender", employee.Gender);

com.Parameters.AddWithValue("@EmpId", employee.EmpId);

rows = com.ExecuteNonQuery();

}

}

return (rows > 0) ? true : false;

}

public bool DeleteEmployee(Employee employee)

{

int rows;

using (SqlConnection con = new SqlConnection(myConn))

{

con.Open();

using (SqlCommand com = new SqlCommand(DeleteQuery, con))

{

com.Parameters.AddWithValue("@EmpId", employee.EmpId);

rows = com.ExecuteNonQuery();

}

}

return (rows > 0) ? true : false;

}

}

}

Now we have completed all the operations and able to Insert, Update, Delete and Retrieve the employee details using the above methods.

**Step 11: How do I do the CURD operations using windows forms?**

Double click "Add" button below,



It will automatically open the Form1.cs and creates the below click event method and we need to write a logic over here.

private void btnAdd\_Click(object sender, EventArgs e)

{

}

The Form1.cs contains Add, Update, Delete, and Clear button click events when the user clicks the button then the event triggers and performs selected operations, as shown below.

using System;

using System.Windows.Forms;

namespace EmployeeDetails

{

public partial class Form1 : Form

{

Employee employee = new Employee();

public Form1()

{

InitializeComponent();

dgvEmployeeDetails.DataSource = employee.GetEmployees();

}

// Add employee details when clicks the Add button

private void btnAdd\_Click(object sender, EventArgs e)

{

employee.EmpId = txtEmpId.Text;

employee.EmpName = txtEmpName.Text;

employee.Age = txtAge.Text;

employee.ContactNo = txtContactNo.Text;

employee.Gender = cboGender.SelectedItem.ToString();

// Call Insert Employee method to insert the employee details to database

var success = employee.InsertEmployee(employee);

// Refresh the grid to show the updated employee details

dgvEmployeeDetails.DataSource = employee.GetEmployees();

if (success)

{

// Clear controls once the employee is inserted successfully

ClearControls();

MessageBox.Show("Employee has been added successfully");

}

else

MessageBox.Show("Error occured. Please try again...");

}

// Update selected employee details when clicks the update button

private void btnUpdate\_Click(object sender, EventArgs e)

{

employee.EmpId = txtEmpId.Text;

employee.EmpName = txtEmpName.Text;

employee.Age = txtAge.Text;

employee.ContactNo = txtContactNo.Text;

employee.Gender = cboGender.SelectedItem.ToString();

// Call Update Employee method to update the selected employee details to database

var success = employee.UpdateEmployee(employee);

// Refresh the grid to show the updated employee details

dgvEmployeeDetails.DataSource = employee.GetEmployees();

if (success)

{

// Clear controls once the employee is inserted successfully

ClearControls();

MessageBox.Show("Employee has been added successfully");

}

else

MessageBox.Show("Error occured. Please try again...");

}

// Delete selected employee when clicks the delete button

private void btnDelete\_Click(object sender, EventArgs e)

{

employee.EmpId = txtEmpId.Text;

// Call DeleteEmployee method to delete the selected employee from database

var success = employee.DeleteEmployee(employee);

// Refresh the grid to show the updated employee details

dgvEmployeeDetails.DataSource = employee.GetEmployees();

if (success)

{

// Clear controls once the employee is inserted successfully

ClearControls();

MessageBox.Show("Employee has been added successfully");

}

else

MessageBox.Show("Error occured. Please try again...");

}

// Clear all controls when clicks clear button

private void btnClear\_Click(object sender, EventArgs e)

{

ClearControls();

}

private void ClearControls()

{

txtEmpId.Text = "";

txtEmpName.Text = "";

txtAge.Text = "";

txtContactNo.Text = "";

cboGender.Text = "";

}

// This data grid event triggers when select the grid rows and populate the controls with selected employee details

private void dgvEmployeeDetails\_RowHeaderMouseClick(object sender, DataGridViewCellMouseEventArgs e)

{

var index = e.RowIndex;

txtEmpId.Text = dgvEmployeeDetails.Rows[index].Cells[0].Value.ToString();

txtEmpName.Text = dgvEmployeeDetails.Rows[index].Cells[1].Value.ToString();

txtAge.Text = dgvEmployeeDetails.Rows[index].Cells[2].Value.ToString();

txtContactNo.Text = dgvEmployeeDetails.Rows[index].Cells[3].Value.ToString();

cboGender.Text = dgvEmployeeDetails.Rows[index].Cells[4].Value.ToString();

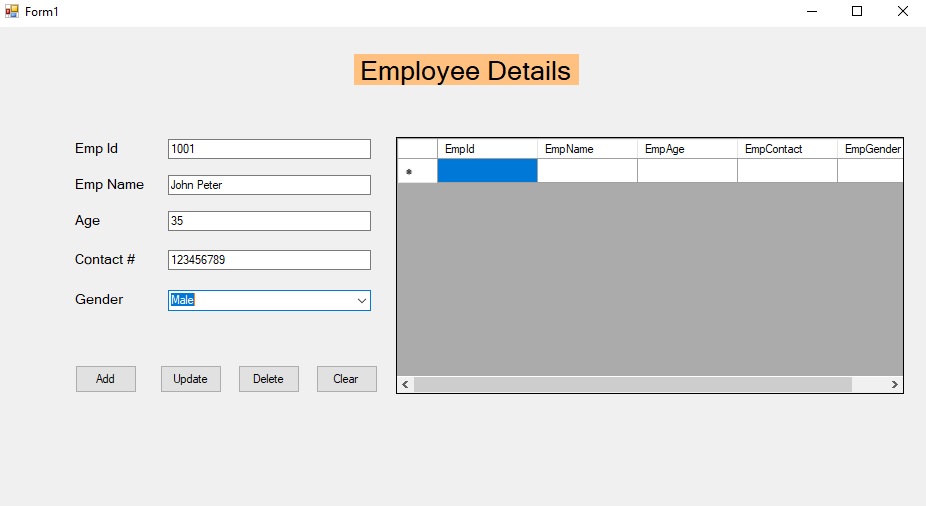
}

}

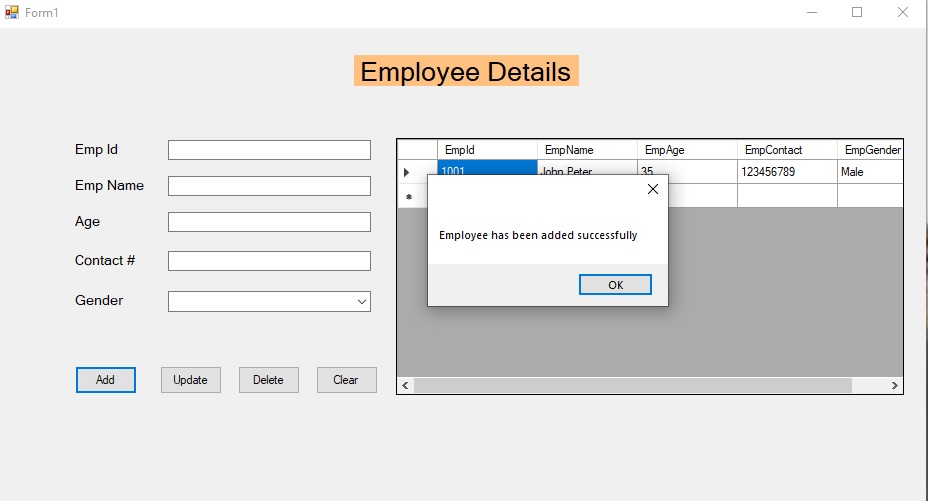
}

**Output:**

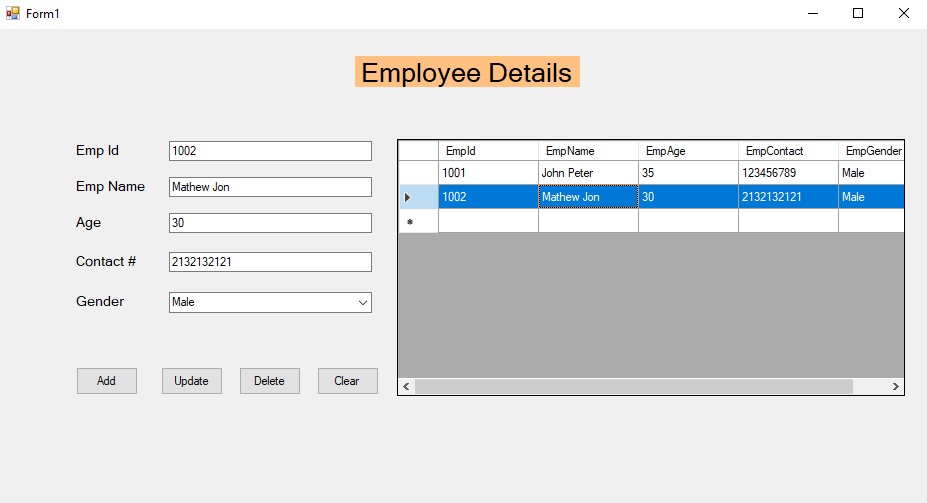
Add employee details and Click "Add" button,



The employee details are successfully inserted.



When you select the grid row (employee)  selected employee details will be populated in the controls for Update or Delete purpose



TASK:

Your task is to create CRUD application using Webforms in Asp.net for Employees table as discussed in class.

Note: Project is uploaded in miscellaneous section

CREATE PROCEDURE GetData

AS

BEGIN

SET NOCOUNT ON;

SELECT \*

FROM emp;

END

CREATE PROCEDURE InsertData

@name varchar(50),

@address varchar(50),

@dept varchar(50)

-- Add more parameters as needed

AS

BEGIN

SET NOCOUNT ON;

-- Insert statement

INSERT INTO emp (name,address, department)

VALUES (@name, @address,@dept);

-- You can add more columns and values as needed

-- Optional: Check the number of affected rows

DECLARE @RowCount INT;

SET @RowCount = @@ROWCOUNT;

-- Optional: Return the number of affected rows

SELECT @RowCount AS 'RowsInserted';

END

CREATE PROCEDURE GetData

AS

BEGIN

SET NOCOUNT ON;

SELECT \*

FROM emp;

END

CREATE PROCEDURE InsertData

@name varchar(50),

@address varchar(50),

@dept varchar(50)

-- Add more parameters as needed

AS

BEGIN

SET NOCOUNT ON;

-- Insert statement

INSERT INTO emp (name,address, department)

VALUES (@name, @address,@dept);

-- You can add more columns and values as needed

-- Optional: Check the number of affected rows

DECLARE @RowCount INT;

SET @RowCount = @@ROWCOUNT;

-- Optional: Return the number of affected rows

SELECT @RowCount AS 'RowsInserted';

END

CREATE PROCEDURE UpdateData

@id int,

@name varchar(50),

@address varchar(50),

@dept varchar(50)

AS

BEGIN

SET NOCOUNT ON;

-- Update statement

UPDATE emp

SET name = @Parameter1,

Column2 = @Parameter2

WHERE id=@id;

-- You can add more columns and values to update, and customize the WHERE clause based on your requirements

END