C# and .Net framework

1. Design and implement a Student Registration form using C# and Windows Forms. The form should allow users to input and save student details into a database.

AIM:

To design and implement a Student Registration Form using C# and Windows Forms, allowing users to input student details and save them to a database.

PROCEDURE:

- **Design the Form**: Create a Windows Forms application in Visual Studio and design a form with input fields for Name, Age, Gender, Email, and Contact, along with a Save button.
- **Setup Database**: Create a SQL Server database called `StudentDB` with a table `Students` to store the student details.
- **Implement Database Connection**: Use ADO.NET to connect to the `StudentDB` database.
- **Save Button Functionality**: Add code to the `Save` button's click event to insert student details into the `Students` table in the database.
- **Run the Application**: Test the form by entering details and checking the database to ensure the data is saved correctly.

PROGRAM:

Database Setup:

```
CREATE DATABASE StudentDB;

USE StudentDB;

CREATE TABLE Students (
    StudentID INT PRIMARY KEY IDENTITY,
    Name NVARCHAR(50),
    Age INT,
    Gender NVARCHAR(10),
    Email NVARCHAR(50),
    Contact NVARCHAR(15)
);
```

C# Code:

1. **Form Design** (Add fields for Name, Age, Gender, Email, and Contact in Windows Forms Designer).
2. **Code for Student Registration Form**:

```
using System;
using System.Data.SqlClient;
using System.Windows.Forms;

namespace StudentRegistrationApp
{
   public partial class StudentRegistrationForm : Form
   {
      public StudentRegistrationForm()
      {
         InitializeComponent();
      }

      string connectionString = "Data Source=YourServerName;Initial Catalog=StudentDB;Integrated
Security=True";

      private void btnSave_Click(object sender, EventArgs e)
      {
         string name = txtName.Text;
         int age = int.Parse(txtAge.Text);
         string gender = comboGender.SelectedItem.ToString();
         string email = txtEmail.Text;
         string contact = txtContact.Text;
         using (SqlConnection conn = new SqlConnection(connectionString))
         {
            string query = "INSERT INTO Students (Name, Age, Gender, Email, Contact) VALUES (@Name,
@Age, @Gender, @Email, @Contact)";
            SqlCommand cmd = new SqlCommand(query, conn);
            cmd.Parameters.AddWithValue("@Name", name);
            cmd.Parameters.AddWithValue("@Age", age);
            cmd.Parameters.AddWithValue("@Gender", gender);
            cmd.Parameters.AddWithValue("@Email", email);
            cmd.Parameters.AddWithValue("@Contact", contact);
            conn.Open();
            int result = cmd.ExecuteNonQuery();

            if (result > 0)
            {
               MessageBox.Show("Student registered successfully.");
            }
            else
            {
               MessageBox.Show("Failed to register student.");
            }
         }
      }
   }
}
```

INPUT:

- **Name**: John Doe
- **Age**: 21
- **Gender**: Male
- **Email**: johndoe@example.com
- **Contact**: 1234567890

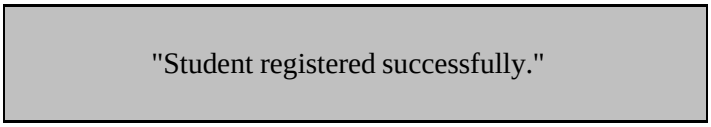
OUTPUT:

Form Output (on Successful Submission)

1. **User Input Form**
   The form would look like this in Windows Forms:

2. **Confirmation Message** After clicking the **Save** button, the form displays this message:

Message Box:



Database Output:

Query: SELECT * FROM Students;

SQL Table Output:

Student Id	Name	Age	Gender	Email	Contact
1	John Doe	21	Gender	johndoe@example.com	1234567890

2. DesignandimplementaStudentFeePaymentSystemusingC#andWindowsForms.The application should allow students to enter their details, pay their fees, and generate a bill with a unique bill number.

AIM:

To design and implement a Student Fee Payment System using C# and Windows Forms, allowing students to enter their details, pay their fees, and generate a bill with a unique bill number.

PROCEDURE:

- **Create a Windows Forms Application**: Open Visual Studio, create a new Windows Forms Application project for the Student Fee Payment System.
- **Design the Form**: Add controls for student details (Student ID, Name, Course), fee amount, and payment method. Include buttons for **Pay** and **Generate Bill**.
- **Setup Database**: Create a SQL Server database called `StudentFeesDB` with tables for `Students` and `Payments` to store student details and payment records.
- **Database Connection**: Use ADO.NET to establish a connection to the `StudentFeesDB` database.
- **Test the Application**: Run the application to ensure data is correctly entered, saved, and a bill is generated with a unique bill number.

PROGRAM:

```
-- Create the database
CREATE DATABASE StudentFeesDB;

USE StudentFeesDB;

-- Create the Students table
CREATE TABLE Students (
    StudentID INT PRIMARY KEY,
    Name NVARCHAR(50),
    Course NVARCHAR(50)
);

-- Create the Payments table
CREATE TABLE Payments (
    PaymentID INT PRIMARY KEY IDENTITY,
    StudentID INT FOREIGN KEY REFERENCES Students(StudentID),
    Amount DECIMAL(10, 2),
    PaymentDate DATETIME,
    BillNumber UNIQUEIDENTIFIER DEFAULT NEWID()
);
```

C# Code:

1. **Form Design**: Add TextBoxes for `Student ID`, `Name`, and `Course`, a `Fee Amount` field, and buttons for **Pay** and **Generate Bill**.
2. **Code for Fee Payment System Form**:

```
using System;

using System.Data.SqlClient;

using System.Windows.Forms;

namespace StudentFeePaymentApp

{

   public partial class FeePaymentForm : Form

   {

      public FeePaymentForm(){

         InitializeComponent();

      }

      string connectionString = "Data Source=YourServerName;Initial
Catalog=StudentFeesDB;Integrated Security=True";

      private void btnPay_Click(object sender, EventArgs e){

         int studentID = int.Parse(txtStudentID.Text);

         string name = txtName.Text;

         string course = txtCourse.Text;

         decimal amount = decimal.Parse(txtAmount.Text);

         DateTime paymentDate = DateTime.Now;

         using (SqlConnection conn = new SqlConnection(connectionString)){

            conn.Open();

            string insertStudent = "IF NOT EXISTS (SELECT * FROM Students WHERE
StudentID = @StudentID) " +

                                   "INSERT INTO Students (StudentID, Name, Course) VALUES
(@StudentID, @Name, @Course)";
```

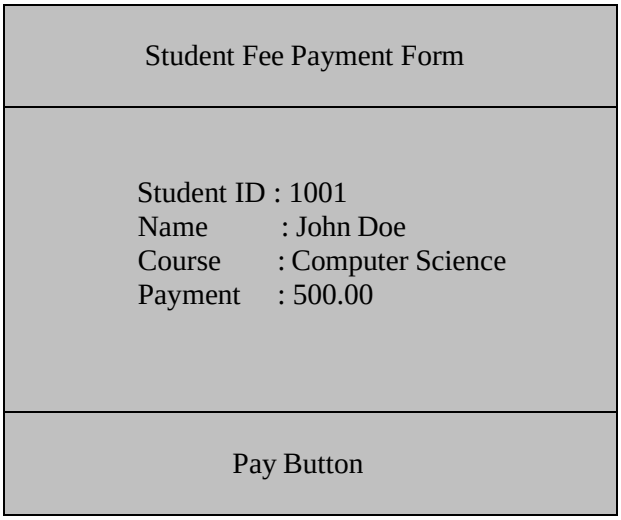
```
            SqlCommand cmdStudent = new SqlCommand(insertStudent, conn);

            cmdStudent.Parameters.AddWithValue("@StudentID", studentID);

            cmdStudent.Parameters.AddWithValue("@Name", name);

            cmdStudent.Parameters.AddWithValue("@Course", course);

            cmdStudent.ExecuteNonQuery();

            string insertPayment = "INSERT INTO Payments (StudentID, Amount,
PaymentDate) VALUES (@StudentID, @Amount, @PaymentDate); " +

                                   "SELECT SCOPE_IDENTITY();";

            SqlCommand cmdPayment = new SqlCommand(insertPayment, conn);

            cmdPayment.Parameters.AddWithValue("@StudentID", studentID);

            cmdPayment.Parameters.AddWithValue("@Amount", amount);

            cmdPayment.Parameters.AddWithValue("@PaymentDate", paymentDate);

            int paymentID = Convert.ToInt32(cmdPayment.ExecuteScalar());

            MessageBox.Show("Payment successful. Bill number will be generated.");

            string getBillNumber = "SELECT BillNumber FROM Payments WHERE
PaymentID = @PaymentID";

            SqlCommand cmdBill = new SqlCommand(getBillNumber, conn);

            cmdBill.Parameters.AddWithValue("@PaymentID", paymentID);

            Guid billNumber = (Guid)cmdBill.ExecuteScalar();

            MessageBox.Show($"Bill Generated Successfully!\n\nBill Number:
{billNumber}\nStudent ID: {studentID}\nName: {name}\nCourse: {course}\nAmount Paid:
{amount:C}\nDate: {paymentDate}");

        }

    }

  }

}
```

INPUT:

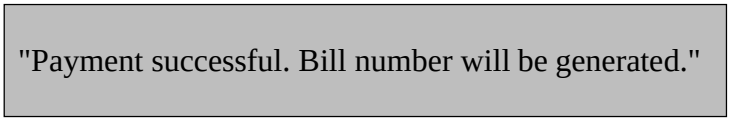
- **Student ID**: 1001
- **Name**: John Doe
- **Course**: Computer Science
- **Fee Amount**: 500.00

OUTPUT:

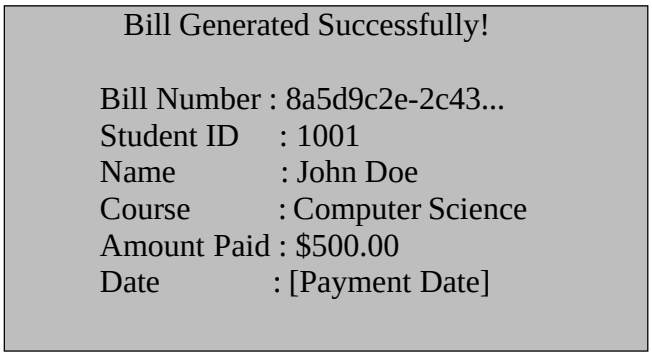
1. Form Layout:



2. Confirmation Message:
Message Box:



3. Bill Display:
Message Box:



3. DesignandimplementaWebServiceusingC#andASP.NETtoexposefunctionalityfor client applicationsto consume. Theweb serviceshould providea specificset ofoperations, such as retrieving data or performing a calculation.

AIM:

To design and implement a Web Service using C# and ASP.NET that exposes specific functionality for client applications to consume, such as retrieving data or performing calculations.

PROCEDURE:

- **Create an ASP.NET Web Service Project**: Open Visual Studio, create a new ASP.NET Web Application project, and select the Web API template.
- **Define the Web Service Operations**: Decide on a set of operations. For example, we'll create an operation that retrieves student data and calculates the average grade.
- **Implement Web Service Methods**:
  - Define a method to retrieve a list of students.
  - Define a method to calculate the average grade of a student.
- **Create a Data Model**: Define models for `Student` and `Grade`.
- **Test the Web Service**: Use tools like Postman or Swagger to test the Web API endpoints and verify that data is returned correctly.

PROGRAM:

1. **Project Structure**: Create an ASP.NET Web API project with controllers and models.
2. **Models**: Define `Student` and `Grade` models.

```
namespace StudentWebService.Models
{
    public class Student
    {
        public int StudentID { get; set; }
        public string Name { get; set; }
        public List<int> Grades { get; set; }
    }
}
```

3. **Controller**: Create a `StudentController` to define the Web API endpoints.

```
using Microsoft.AspNetCore.Mvc;
using StudentWebService.Models;
using System.Collections.Generic;
using System.Linq;
namespace StudentWebService.Controllers
{
```

```
    [ApiController]
    [Route("api/[controller]")]
    public class StudentController : ControllerBase
    {

        private static readonly List<Student> students = new List<Student>
        {
            new Student { StudentID = 1, Name = "John Doe", Grades = new List<int> { 85, 90, 78 } },
            new Student { StudentID = 2, Name = "Jane Smith", Grades = new List<int> { 92, 88, 94 } }
        };


        [HttpGet]
        public ActionResult<IEnumerable<Student>> GetStudents()
        {
            return Ok(students);
        }


        [HttpGet("{id}/average")]
        public ActionResult<double> GetAverageGrade(int id)
        {
            var student = students.FirstOrDefault(s => s.StudentID == id);
            if (student == null)
            {
                return NotFound("Student not found");
            }
            double averageGrade = student.Grades.Average();
            return Ok(averageGrade);
        }
    }
}
```

4.Startup Configuration: Configure the API services in `Startup.cs`.

```
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;

namespace StudentWebService
{
    public class Startup
    {
        public void ConfigureServices(IServiceCollection services)
        {
            services.AddControllers();
        }
```

```
        public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
        {
            if (env.IsDevelopment())
            {
                app.UseDeveloperExceptionPage();
            }

            app.UseRouting();

            app.UseEndpoints(endpoints =>
            {
                endpoints.MapControllers();
            });
        }
    }
}
```

INPUT:

1. **Get All Students**:
    - URL: `http://localhost:5000/api/student`
    - Method: GET

2. **Get Average Grade for a Student**:
    - URL: `http://localhost:5000/api/student/1/average`
    - Method: GET

OUTPUT:

Get All Students:

- **Request**: `GET http://localhost:5000/api/student`
- **Response** (JSON):


```
[
  {
    "StudentID": 1,
    "Name": "John Doe",
    "Grades": [85, 90, 78]
  },
  {
    "StudentID": 2,
    "Name": "Jane Smith",
    "Grades": [92, 88, 94]
  }
```


Get Average Grade for a Student:

- **Request**: `GET http://localhost:5000/api/student/1/average`
- **Response** (JSON):


```
{
  "averageGrade":
```

4.Our college is organizing an Alumni Meet on May 5, 2024. The alumni cell is in the process of creating a database to store a list of registered alumni who will attend the event. You are tasked with designing a registration form and implementing it using ADO.NET.

Requirements:

1. **Design the Registration Form**:
    - Create a Windows Forms application that includes the following controls:
        - **TextBox** for entering the **Alumni Name**
        - **TextBox** for entering the **Email**
        - **TextBox** for entering the **Phone Number**
        - **ComboBox** for selecting the **Department** (e.g., Computer Science, Business, Arts)
        - **Button** to **Register** alumni
        - **Button** to **Display** registered alumni
        - **DataGridView** control to display the list of registered alumni from the selected department
2. **Implement Functionality Using ADO.NET**:
    - **Register Button**:
        - When the **Register** button is clicked, validate the input fields.
        - If the inputs are valid, insert the entered details into the database using ADO.NET. Handle any database exceptions that may occur.
    - **Display Button**:
        - When the **Display** button is clicked, retrieve all registered alumni for the selected department from the ComboBox.
        - Display the results in the **DataGridView** control.

AIM:

To design a Windows Forms application for alumni registration for the Alumni Meet and implement functionality to store and display registered alumni details using ADO.NET.

PROCEDURE:

1. **Create the Database**: Set up an SQL Server database named `AlumniDB` with an `Alumni` table containing columns for `AlumniID`, `Name`, `Email`, `PhoneNumber`, and `Department`.
2. **Design the Windows Form**:
    - Add text boxes for Alumni Name, Email, and Phone Number.
    - Add a ComboBox for selecting Department.
    - Add a **Register** button to save data, a **Display** button to show data, and a `DataGridView` to display registered alumni.
3. **Implement Register and Display Functionality Using ADO.NET**:
    - **Register Button**: Validate input fields and insert data into the `Alumni` table using an ADO.NET `SqlCommand`.
    - **Display Button**: Retrieve and display alumni data for the selected department in the `DataGridView` using an ADO.NET `SqlDataAdapter`.
4. **Test the Application**: Run the form to ensure alumni data is saved correctly in the database and displays in the `DataGridView`.

PROGRAM:

Database Setup:

```
-- Create the AlumniDB database
CREATE DATABASE AlumniDB;

USE AlumniDB;

-- Create the Alumni table
CREATE TABLE Alumni (
    AlumniID INT PRIMARY KEY IDENTITY,
    Name NVARCHAR(50),
    Email NVARCHAR(50),
    PhoneNumber NVARCHAR(15),
    Department NVARCHAR(50)
);
```

C# Code for Alumni Registration Form:

1. **Designing the Form**:
    - Add controls for **Alumni Name**, **Email**, **Phone Number**, **Department** (`ComboBox`), **Register** and **Display** buttons, and a `DataGridView`.
2. **C# Code for Form**:

```
using System;
using System.Data;
using System.Data.SqlClient;
using System.Windows.Forms;
namespace AlumniRegistrationApp
{
    public partial class AlumniForm : Form
    {
        private string connectionString = "Data Source=YourServerName;Initial
Catalog=AlumniDB;Integrated Security=True";

        public AlumniForm()
        {
            InitializeComponent();
            LoadDepartments();
        }
        private void LoadDepartments()
        {
            comboBoxDepartment.Items.AddRange(new string[] { "Computer Science", "Business",
"Arts" });
        }
```

```
        private void btnRegister_Click(object sender, EventArgs e)
        {
            string name = txtName.Text;
            string email = txtEmail.Text;
            string phoneNumber = txtPhoneNumber.Text;
            string department = comboBoxDepartment.SelectedItem?.ToString();

            if (string.IsNullOrWhiteSpace(name) || string.IsNullOrWhiteSpace(email) ||
                string.IsNullOrWhiteSpace(phoneNumber) || string.IsNullOrWhiteSpace(department))
            {
                MessageBox.Show("All fields are required.");
                return;
            }

            using (SqlConnection conn = new SqlConnection(connectionString))
            {
                try
                {
                    conn.Open();
                    string query = "INSERT INTO Alumni (Name, Email, PhoneNumber, Department)
VALUES (@Name, @Email, @PhoneNumber, @Department)";
                    using (SqlCommand cmd = new SqlCommand(query, conn))
                    {
                        cmd.Parameters.AddWithValue("@Name", name);
                        cmd.Parameters.AddWithValue("@Email", email);
                        cmd.Parameters.AddWithValue("@PhoneNumber", phoneNumber);
                        cmd.Parameters.AddWithValue("@Department", department);
                        cmd.ExecuteNonQuery();
                    }
                    MessageBox.Show("Alumni registered successfully.");
                }
                catch (Exception ex)
                {
                    MessageBox.Show($"Error: {ex.Message}");
                }
            }
        }

        private void btnDisplay_Click(object sender, EventArgs e)
        {
            string department = comboBoxDepartment.SelectedItem?.ToString();
            if (string.IsNullOrWhiteSpace(department))
            {
                MessageBox.Show("Please select a department.");
                return;
            }
            using (SqlConnection conn = new SqlConnection(connectionString))
            {
                try
                {
                    conn.Open();
```

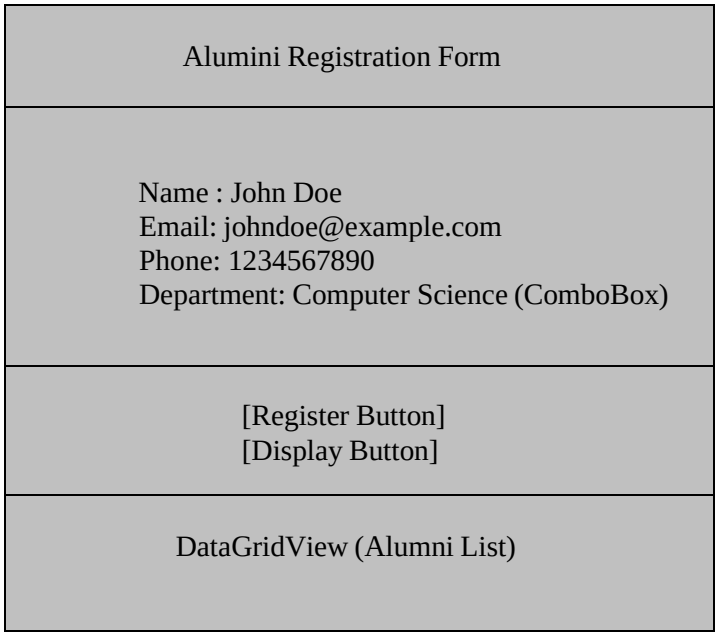
```
                string query = "SELECT AlumniID, Name, Email, PhoneNumber, Department FROM
Alumni WHERE Department = @Department";
                using (SqlCommand cmd = new SqlCommand(query, conn))
                {
                    cmd.Parameters.AddWithValue("@Department", department);
                    using (SqlDataAdapter adapter = new SqlDataAdapter(cmd))
                    {
                        DataTable dt = new DataTable();
                        adapter.Fill(dt);
                        dataGridViewAlumni.DataSource = dt;
                    }
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Error: {ex.Message}");
            }
        }
    }
}
```

INPUT:

- **Alumni Name**: John Doe
- **Email**: johndoe@example.com
- **Phone Number**: 1234567890
- **Department**: Computer Science

DISPLAY:

Form Layout:



OUTPUT:

After Registering Alumni:

- Message Box:

"Alumni registered successfully."

Displaying Registered Alumni for Selected Department:

- On clicking **Display** with "Computer Science" selected, the `DataGridView` displays all registered alumni in the Computer Science department:

DataGridView (Alumni List):

Alumini Id	Name	Email	Phone	Dept
1	John Doe	johndoe@example.com	1234567890	CS

K S PRABA(73772214176)

BE-CSE-B-3rd yr