SKILLS INTERNATIONAL

School management system

FINAL REPORT

A.S.DINUSHKA THARIDU

AVI 00182521

Acknowledgement

I would like to extend my gratitude for the successful completion of this project, which served as the final assignment of the DiTEC Diploma program. Special thanks are due to Mr. Isuru V Samarasinghe, the lecturer, whose guidance and support were invaluable throughout this endeavor, as well as for his teaching throughout the course.

Furthermore, I express my appreciation to the staff of Esoft Metro Campus, Avissawella, for providing us with all the necessary facilities that facilitated our work.

About Me

Name: A.S. Dinushka Tharidu

Registration No: 00182521

Course: DiTEC

Batch: 113

Branch: Avissawella

Contents

01.Introduction

1.1 Project Overview

The School Management System (SMS) is a software application designed to streamline the administrative processes of a school. It offers functionalities for Registering students, Registering teachers and other school-related activities.

1.2 Problem Statement

Traditional paper-based systems for managing student data, attendance, and other school activities can be time-consuming, prone to errors, and difficult to maintain. This project aims to develop a user-friendly and efficient software solution to address these challenges.

1.3 Objectives

- Develop a user-friendly system for managing student data, including registration and profiles.
- Provide functionalities for managing Teacher's information.
- Generate reports for student bio data, and other relevant data.

1.4 Scope

This project focuses on developing the core functionalities for managing student data, attendance, and grades. Additional features like online fee payment or integrated communication modules can be considered for future enhancements.

02.System Analysis

2.1 System Requirements

There are two main categories of requirements: functional and non-functional.

2.1.1 Functional Requirements

Functional requirements define the specific tasks the system should be able to perform.

- **Student Management:**
  - Add, edit, and delete student information (name, registration number, grade level, etc.)
- **Teachers Management**
  - Manage Teachers data (name, designation, contact information)

2.1.2 Non-Functional Requirements

Non-functional requirements define the overall qualities of the system, such as performance, usability, security, and reliability.

- **Performance**: The system should be responsive and provide quick loading times.
- **Usability:** The user interface (UI) should be intuitive and easy to navigate for users with varying technical skills.
- **Security:** The system should have robust security measures to protect sensitive data (student information, grades, etc.). Regular data backups and recovery procedures are crucial.
- **Reliability:** The system should be reliable and minimize downtime or errors.
- **Scalability:** The system should be scalable to accommodate a growing number of users and data.

03.System Design

The system design phase translates the system requirements from the analysis phase into a technical blueprint. Here's an overview of the design for a School Management System (SMS)

3.1 System Architecture

The system architecture defines the overall structure and components of the SMS. A common architecture for this type of application is a three-tier architecture:

- **Presentation Layer:** This layer consists of the user interface (UI) elements like web forms or mobile app screens that users interact with to access functionalities.
- **Business Logic Layer:** This layer handles the core functionalities of the system, such as processing data, performing calculations, and interacting with the database.
- **Data Access Layer:** This layer manages communication with the database, storing, retrieving, and manipulating data.

This layered approach promotes separation of concerns, making the system more modular, maintainable, and scalable.

04.System Implementation

The system implementation phase translates the system design into a working application. Here's a breakdown of the implementation process for a School Management System (SMS)

4.1 Technology Stack

The technology stack refers to the hardware, software, and programming languages used to develop the SMS. Here's a possible technology stack for this project

Backend : C#

Database : Microsoft SQL Server Management Studio

4.2 Development Tools and Languages

These tools can significantly improve development efficiency and maintainability.

- **Development IDE:** Microsoft Visual Studio 2022
- **Database Management Tool**: SQL Server Management Studio

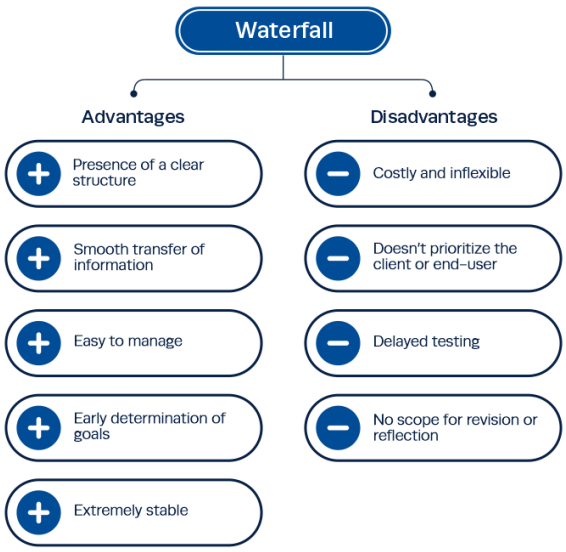
4.3 Development Methodology

The development methodology defines the approach used to build the SMS. Here are two common methodologies:

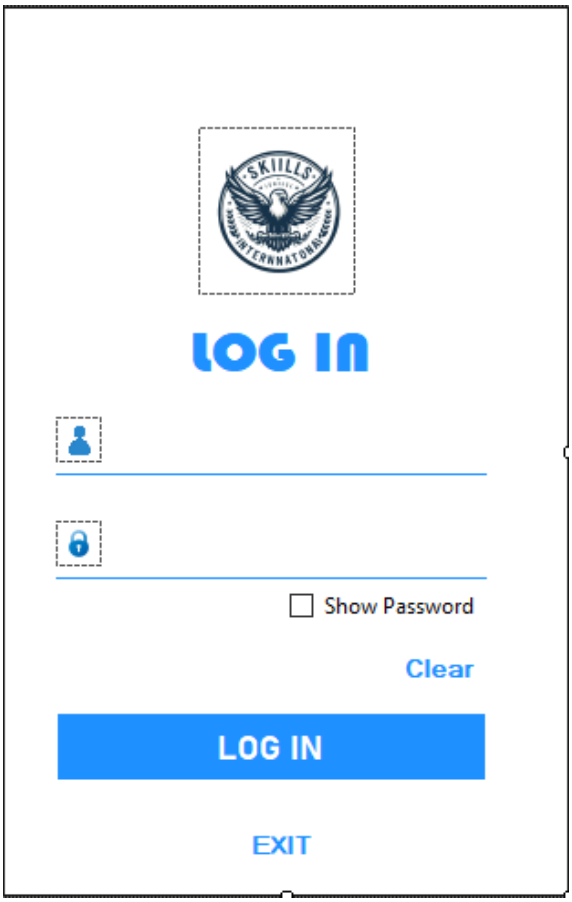
Agile: This iterative approach involves delivering features in short cycles (sprints) with continuous feedback and adaptation based on user needs. It's suitable for projects with evolving requirements.

Waterfall: This traditional approach follows a sequential development process (planning, design, development, testing, deployment). It requires well-defined requirements upfront but may be less flexible for changing needs.

I chose the Waterfall model to develop this software because it is use to build simple Software.



4.4 Login Form Codes



- **Login Butten Codes**

```
1 reference
private void button1_Click(object sender, EventArgs e)
{
    if (textUsername.Text == "Admin" && textpassword.Text == "Skills@123")
    {
        new Form2().Show();
        this.Hide();
    }
    else
    {
        MessageBox.Show("Invalid Login credentials, Please check Username And Password and Try again.");
        textUsername.Clear();
        textpassword.Clear();
        textUsername.Focus();


    }


}
```

- **Clear Butten Codes**

```
1 reference
private void label2_Click(object sender, EventArgs e)
{
    textUsername.Clear();
    textpassword.Clear();
    textUsername.Focus();
}
```

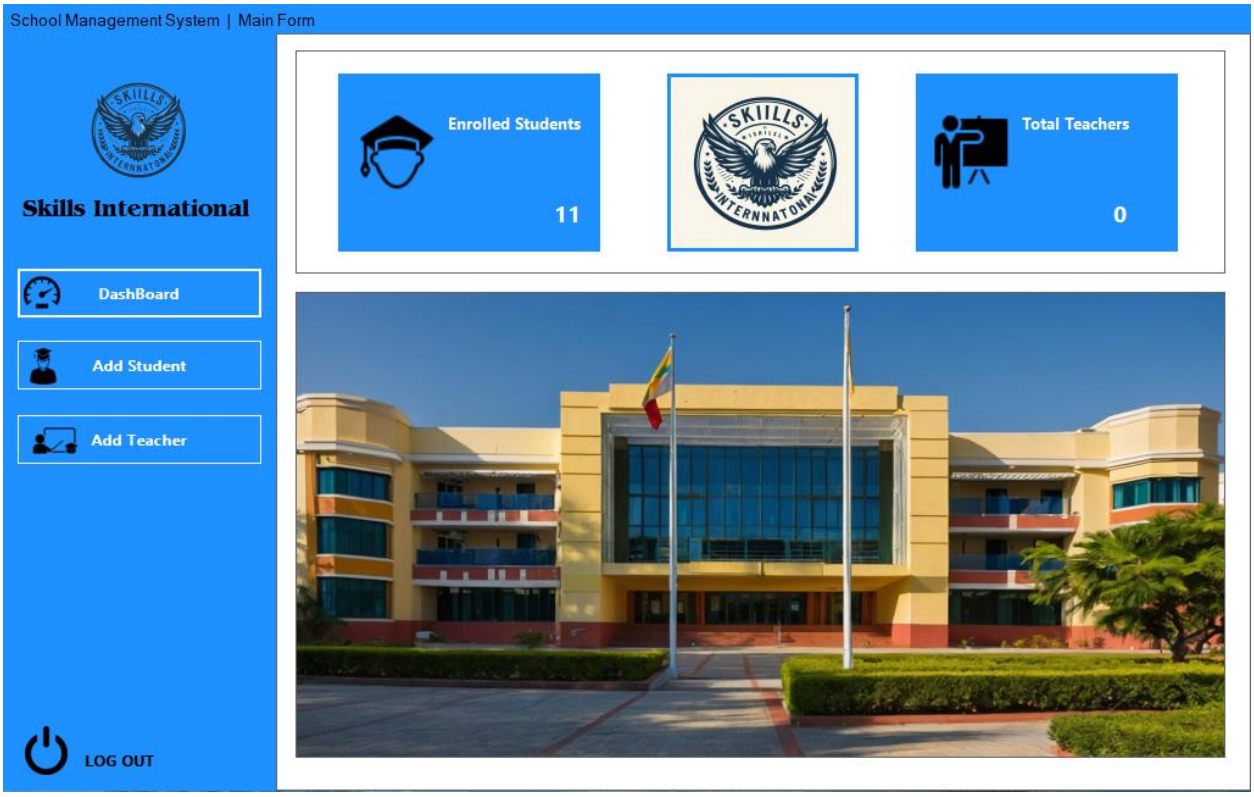
- **Exit Butten Codes**

```
1 reference
private void label3_Click(object sender, EventArgs e)
{
    DialogResult result = MessageBox.Show("Do you really want to exit?", "Confirmation", MessageBoxButtons.YesNo, M

    // Check the user's response
    if (result == DialogResult.Yes)
    {
        // Close the application
        Application.Exit();
    }
}
```

- **Show Password Check Box Codes**

```
1 reference
private void checkBox1_CheckedChanged(object sender, EventArgs e)
{
    if (checkBox1.Checked)
    {
        textpassword.PasswordChar = '\0';
    }
    else
    {
        textpassword.PasswordChar = '*';
    }
}
```

4.5 Main Form Codes



- **Add Student Button Codes**

```
1 reference
private void button2_Click(object sender, EventArgs e)
{
    dashboardForm1.Visible = false;
    addStudentForm1.Visible = true;
    addTeachersForm1.Visible = false;
}

1 reference
private void button3_Click(object sender, EventArgs e)
```

- **Add Teacher Button Codes**

```
1 reference
private void button3_Click(object sender, EventArgs e)
{
    dashboardForm1.Visible = false;
    addStudentForm1.Visible = false;
    addTeachersForm1.Visible = true;
}
```

- **Dashboard Button Codes**

```
1 reference
private void button1_Click(object sender, EventArgs e)
{
    dashboardForm1.Visible = true;
    addStudentForm1.Visible = false;
    addTeachersForm1.Visible = false;
}
```

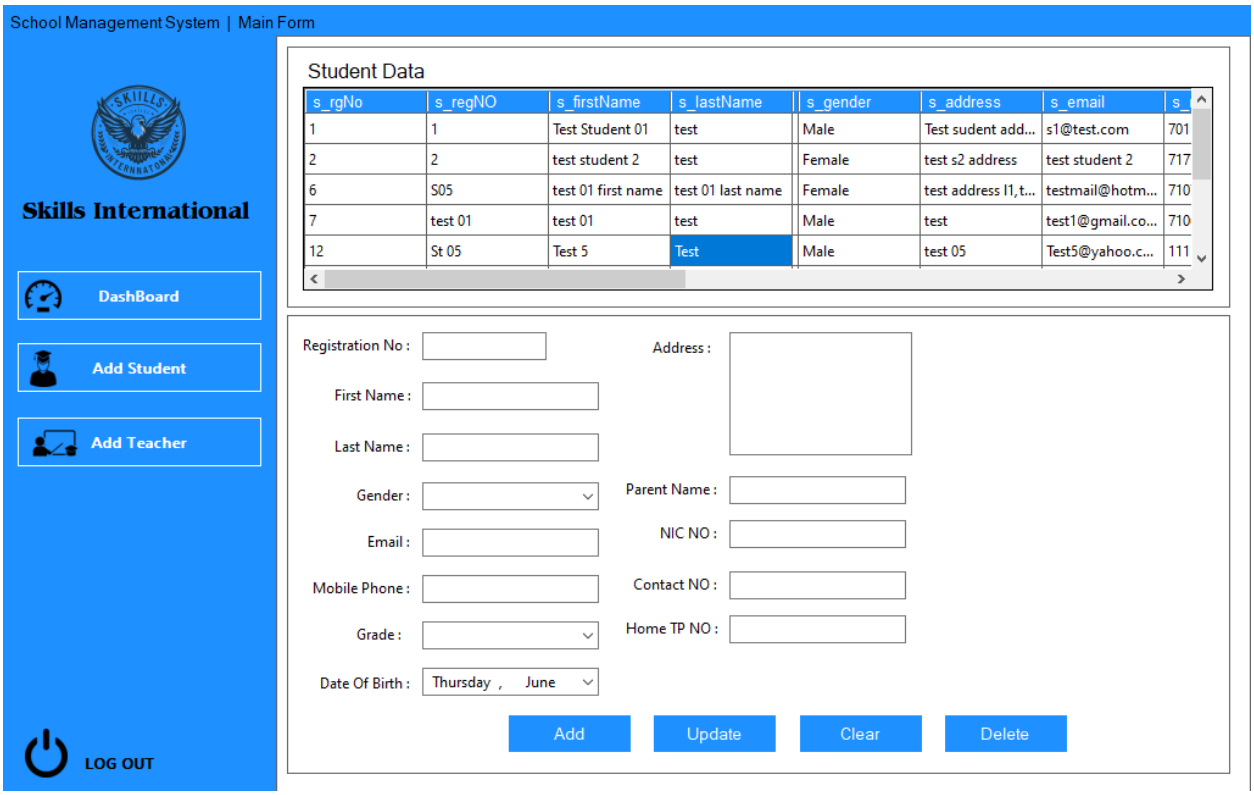
- **Logout Button Codes**

```
1 reference
private void button4_Click(object sender, EventArgs e)
{
    DialogResult check = MessageBox.Show("Are You Sure You Want To LogOut?", "Confirmation Message", MessageBoxBut
    if (check == DialogResult.Yes)
    {
        // Close the application
        new Form1().Show();
        this.Hide();

    }
```

- **Exit Button Codes**

```
1 reference
private void label4_Click(object sender, EventArgs e)
{

    DialogResult check = MessageBox.Show("Are You Sure You Want To Exit?", "Confirmation Message", MessageBoxButto
    if (check == DialogResult.Yes)
    {
        // Close the application
        Application.Exit();
    }
```

4.6 Register Student Form Codes



- **Add Button Codes**

```
1 reference
private void button1_Click(object sender, EventArgs e)
{
    if (string.IsNullOrWhiteSpace(s_regNO.Text) || string.IsNullOrWhiteSpace(s_firstName.Text) || string.IsNu
    {
        MessageBox.Show("Please fill all fields.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    else
    {
        try
        {
            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                connection.Open();
                string checkStudentID = "SELECT COUNT(*) FROM students WHERE s_regNO = @studentRg";
                using (SqlCommand command = new SqlCommand(checkStudentID, connection))
                {
                    command.Parameters.AddWithValue("@studentRg", s_regNO.Text.Trim());
                    int count = (int)command.ExecuteScalar();
                    if (count >= 1)
                    {
                        MessageBox.Show("Student with ID " + s_regNO.Text.Trim() + " already exists.", "Error
                    }
                    else
                    {
                        DateTime today = DateTime.Today;
                        string insertData = "INSERT INTO students (s_regNO, s_firstName, s_lastName, s_gender
                        using (SqlCommand cmd = new SqlCommand(insertData, connection))
                        {
                            cmd.Parameters.AddWithValue("@studentRg", s_regNO.Text.Trim());
                            cmd.Parameters.AddWithValue("@studentFName", s_firstName.Text.Trim());
                            cmd.Parameters.AddWithValue("@studentLName", s_lastName.Text.Trim());
                            cmd.Parameters.AddWithValue("@studentGender", s_gender.Text.Trim());
                            cmd.Parameters.AddWithValue("@studentAddress", s_address.Text.Trim());
                            cmd.Parameters.AddWithValue("@studentEmail", s_email.Text.Trim());
                            cmd.Parameters.AddWithValue("@studentPhone", s_mobilePhone.Text.Trim());
                            cmd.Parameters.AddWithValue("@studentDOB", s_dob.Value);
                            cmd.Parameters.AddWithValue("@studentHPhone", s_homePhone.Text.Trim());
```

```
                            cmd.Parameters.AddWithValue("@studentDOB", s_dob.Value);
                            cmd.Parameters.AddWithValue("@studentHPhone", s_homePhone.Text.Trim());
                            cmd.Parameters.AddWithValue("@studentPName", s_parentName.Text.Trim());
                            cmd.Parameters.AddWithValue("@studentPNic", s_Pnic.Text.Trim());
                            cmd.Parameters.AddWithValue("@studentPContactno", s_PcontactNo.Text.Trim());
                            cmd.Parameters.AddWithValue("@dateInsert", today);
                            cmd.ExecuteNonQuery();
                            DisplayStudentData();
                            MessageBox.Show("Student added successfully!", "Success", MessageBoxButtons.OK,
                            ClearFields();
                        }
                    }
                }
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show("Error: " + ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }
}
```

- **Clear Button Codes**

```
private void button4_Click_1(object sender, EventArgs e)
{
    // Call the ClearFields method to clear all input fields
    s_regNO.Text = "";
    s_firstName.Text = "";
    s_lastName.Text = "";
    s_gender.Text = "";
    s_email.Text = "";
    s_mobilePhone.Text = "";
    s_grade.Text = "";
    s_address.Text = "";
    s_parentName.Text = "";
    s_Pnic.Text = "";
    s_PcontactNo.Text = "";
    s_homePhone.Text = "";

}
```

- **Update Button Codes**

```
private void button2_Click(object sender, EventArgs e)
{
    // Your existing code for updating a student record
    try
    {
        // Establish connection to the database
        using (SqlConnection connection = new SqlConnection(connectionString))
        {
            connection.Open();

            // Define the SQL UPDATE query
            string updateQuery = @"
        UPDATE students
        SET
            s_firstName = @studentFName,
            s_lastName = @studentLName,
            s_gender = @studentGender,
            s_address = @studentAddress,
            s_email = @studentEmail,
            s_mobilePhone = @studentPhone,
            s_dob = @studentDOB,
            s_homePhone = @studentHPhone,
            s_parentName = @studentPName,
            s_Pnic = @studentPNic,
            s_PcontactNo = @studentPContactno
        WHERE
            s_regNO = @studentRg";

            // Create a command object
            using (SqlCommand cmd = new SqlCommand(updateQuery, connection))
            {
```

```
                // Create a command object
                using (SqlCommand cmd = new SqlCommand(updateQuery, connection))
                {
                    // Set parameter values with the modified data
                    cmd.Parameters.AddWithValue("@studentFName", s_firstName.Text.Trim());
                    cmd.Parameters.AddWithValue("@studentLName", s_lastName.Text.Trim());
                    cmd.Parameters.AddWithValue("@studentGender", s_gender.Text.Trim());
                    cmd.Parameters.AddWithValue("@studentAddress", s_address.Text.Trim());
                    cmd.Parameters.AddWithValue("@studentEmail", s_email.Text.Trim());
                    cmd.Parameters.AddWithValue("@studentPhone", s_mobilePhone.Text.Trim());
                    cmd.Parameters.AddWithValue("@studentDOB", s_dob.Value);
                    cmd.Parameters.AddWithValue("@studentHPhone", s_homePhone.Text.Trim());
                    cmd.Parameters.AddWithValue("@studentPName", s_parentName.Text.Trim());
                    cmd.Parameters.AddWithValue("@studentPNic", s_Pnic.Text.Trim());
                    cmd.Parameters.AddWithValue("@studentPContactno", s_PcontactNo.Text.Trim());
                    cmd.Parameters.AddWithValue("@studentRg", s_regNO.Text.Trim());

                    // Execute the UPDATE query
                    int rowsAffected = cmd.ExecuteNonQuery();

                    // Check if any rows were affected
                    if (rowsAffected > 0)
                    {
                        MessageBox.Show("Record updated successfully!", "Success", MessageBoxButtons.OK, MessageB

                        // Refresh the DataGridView to reflect the changes
                        DisplayStudentData();

                        // Clear the input fields
                        ClearFields();
                    }
```

```
                    else
                    {
                        MessageBox.Show("No record found with the provided Registration No.", "Error", MessageBox
                    }
                }
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show("Error: " + ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }

    }
```

- **Delete Button Codes**

```
1 reference
private void button3_Click(object sender, EventArgs e)
{
    // Check if any cell is selected
    if (student_studentData.SelectedCells.Count > 0)
    {
        // Ask for confirmation
        DialogResult result = MessageBox.Show("Do you really want to delete this record?", "Confirmation", M

        if (result == DialogResult.Yes)
        {
            // Get the ID of the selected record
            int selectedRowIndex = student_studentData.SelectedCells[0].RowIndex;
            DataGridViewRow selectedRow = student_studentData.Rows[selectedRowIndex];
            string studentRg = selectedRow.Cells["s_regNO"].Value.ToString(); // Assuming 's_regNO' is the na

            try
            {
                // Delete the record from the database
                using (SqlConnection connection = new SqlConnection(connectionString))
                {
                    connection.Open();
                    string deleteQuery = "DELETE FROM students WHERE s_regNO = @studentRg";
                    using (SqlCommand cmd = new SqlCommand(deleteQuery, connection))
                    {
                        cmd.Parameters.AddWithValue("@studentRg", studentRg);
                        int rowsAffected = cmd.ExecuteNonQuery();
                        if (rowsAffected > 0)
                        {
                            MessageBox.Show("Record deleted successfully!", "Success", MessageBoxButtons.OK,
                            // Refresh the DataGridView
                            DisplayStudentData();
                        }
                        else
                        {
                            MessageBox.Show("No record found with the provided Registration No.", "Error", M
                        }
                    }
```

```
                    }
                    else
                    {
                        MessageBox.Show("No record found with the provided Registration No.", "Error", Me
                    }
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show("Error: " + ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }
    }
    else
    {
        MessageBox.Show("Please select a record to delete.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Er
    }
}
}
}
```

- **“student_studentData” Data Grid Code**

```
2 references
private void student_studentData_CellClick(object sender, DataGridViewCellEventArgs e)
{
    // Your existing code for populating textboxes with selected row data
    // Check if the clicked cell is not the header row
    if (e.RowIndex != -1)
    {
        // Get the selected row
        DataGridViewRow row = student_studentData.Rows[e.RowIndex];

        // Populate the text boxes and input boxes with the data from the selected row
        s_regNO.Text = row.Cells["s_regNO"].Value.ToString();
        s_firstName.Text = row.Cells["s_firstName"].Value.ToString();
        s_lastName.Text = row.Cells["s_lastName"].Value.ToString();
        s_gender.Text = row.Cells["s_gender"].Value.ToString();
        s_address.Text = row.Cells["s_address"].Value.ToString();
        s_dob.Value = Convert.ToDateTime(row.Cells["s_dob"].Value);
        s_email.Text = row.Cells["s_email"].Value.ToString();
        s_mobilePhone.Text = row.Cells["s_mobilePhone"].Value.ToString();
        s_homePhone.Text = row.Cells["s_homePhone"].Value.ToString();
        s_parentName.Text = row.Cells["s_parentName"].Value.ToString();
        s_Pnic.Text = row.Cells["s_Pnic"].Value.ToString();
        s_PcontactNo.Text = row.Cells["s_PcontactNo"].Value.ToString();
    }
}
```

- **“AddStudentData” Class code**

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Globalization;
using Microsoft.Data.SqlClient;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data.SqlClient;

namespace School_Management_System
{
    7 references
    class AddStudentData
    {
        private readonly SqlConnection connect = new SqlConnection(@"Data Source=(LocalDB)\MSSQLLocalDB;Initial Catal

        1 reference
        public string? StudentRg { get; set; }
        1 reference
        public string? StudentFName { get; set; }
        0 references
        public string? StudentLName { get; set; }
        0 references
        public string? StudentGender { get; set; }
        0 references
        public string? StudentAddress { get; set; }
        0 references
        public string? StudentDOB { get; set; }
        0 references
        public string? StudentEmail { get; set; }
        0 references
        public string? StudentPhone { get; set; }
        0 references
        public string? StudentHPhone { get; set; }
        0 references
        public string? StudentPName { get; set; }
        0 references
        public string? StudentPNic { get; set; }
        0 references
        public string? StudentPContactno { get; set; }
        1 reference
```

```
        public DateTime? DateInsert { get; set; }

        1 reference
        public List<AddStudentData> StudentData()
        {
            List<AddStudentData> listData = new List<AddStudentData>();

            try
            {
                if (connect.State != ConnectionState.Open)
                {
                    connect.Open();
                }

                DateTime today = DateTime.Today;
                string sql = "SELECT * FROM students WHERE date_insert = @dateInsert AND date_delete IS NULL";

                using (SqlCommand cmd = new SqlCommand(sql, connect))
                {
                    cmd.Parameters.AddWithValue("@dateInsert", today);
                    SqlDataReader reader = cmd.ExecuteReader();

                    while (reader.Read())
                    {
                        AddStudentData addSD = new AddStudentData();

                        string? dateString = reader["date_insert"]?.ToString();
                        if (dateString != null)
                        {
                            try
                            {
                                addSD.DateInsert = DateTime.Parse(dateString, CultureInfo.InvariantCulture);
                            }
                            catch (FormatException)
                            {
                                Console.WriteLine("Failed to parse date format: " + dateString);
                                // Log the error or handle it differently (e.g., set a default value)
                            }
```

```
                        }
                        }

                        addSD.StudentRg = reader["s_regNO"]?.ToString();
                        addSD.StudentFName = reader["s_firstName"]?.ToString();

                        listData.Add(addSD);
                    }
                    reader.Close();
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine("Error: " + ex);
            }
            finally
            {
                if (connect.State == ConnectionState.Open)
                {
                    connect.Close();
                }
            }

            return listData;
        }

    }
}
```

- **AddStudentForm Full codes**

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

namespace School_Management_System
{
    4 references
    public partial class AddStudentForm : UserControl
    {
        private readonly string connectionString = @"Data Source=(LocalDB)\MSSQLLocalDB;Initial Catalog=C:\USERS\DINU

        1 reference
        public AddStudentForm()
        {
            InitializeComponent();
            DisplayStudentData();
        }

        4 references
        public void DisplayStudentData()
        {
            try
            {
                using (SqlConnection connection = new SqlConnection(connectionString))
                {
                    connection.Open();
                    string query = "SELECT * FROM students";
                    using (SqlCommand command = new SqlCommand(query, connection))
                    {
                        SqlDataAdapter adapter = new SqlDataAdapter(command);
                        DataTable dataTable = new DataTable();
                        adapter.Fill(dataTable);
                        student_studentData.DataSource = dataTable;
                    }
                }
            }
```

```
            catch (Exception ex)
            {
                MessageBox.Show("Error: " + ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }

        1 reference
        private void button1_Click(object sender, EventArgs e)
        {
            if (string.IsNullOrWhiteSpace(s_regNO.Text) || string.IsNullOrWhiteSpace(s_firstName.Text) || string.IsNu
            {
                MessageBox.Show("Please fill all fields.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
            else
            {
                try
                {
                    using (SqlConnection connection = new SqlConnection(connectionString))
                    {
                        connection.Open();
                        string checkStudentID = "SELECT COUNT(*) FROM students WHERE s_regNO = @studentRg";
                        using (SqlCommand command = new SqlCommand(checkStudentID, connection))
                        {
                            command.Parameters.AddWithValue("@studentRg", s_regNO.Text.Trim());
                            int count = (int)command.ExecuteScalar();
                            if (count >= 1)
                            {
                                MessageBox.Show("Student with ID " + s_regNO.Text.Trim() + " already exists.", "Error
                            }
                            else
                            {
                                DateTime today = DateTime.Today;
                                string insertData = "INSERT INTO students (s_regNO, s_firstName, s_lastName, s_gender
                                using (SqlCommand cmd = new SqlCommand(insertData, connection))
                                {
```

```
68                                  using (SqlCommand cmd = new SqlCommand(insertData, connection))
69                                  {
70                                      cmd.Parameters.AddWithValue("@studentRg", s_regNO.Text.Trim());
71                                      cmd.Parameters.AddWithValue("@studentFName", s_firstName.Text.Trim());
72                                      cmd.Parameters.AddWithValue("@studentLName", s_lastName.Text.Trim());
73                                      cmd.Parameters.AddWithValue("@studentGender", s_gender.Text.Trim());
74                                      cmd.Parameters.AddWithValue("@studentAddress", s_address.Text.Trim());
75                                      cmd.Parameters.AddWithValue("@studentEmail", s_email.Text.Trim());
76                                      cmd.Parameters.AddWithValue("@studentPhone", s_mobilePhone.Text.Trim());
77                                      cmd.Parameters.AddWithValue("@studentDOB", s_dob.Value);
78                                      cmd.Parameters.AddWithValue("@studentHPhone", s_homePhone.Text.Trim());
79                                      cmd.Parameters.AddWithValue("@studentPName", s_parentName.Text.Trim());
80                                      cmd.Parameters.AddWithValue("@studentPNic", s_Pnic.Text.Trim());
81                                      cmd.Parameters.AddWithValue("@studentPContactno", s_PcontactNo.Text.Trim());
82                                      cmd.Parameters.AddWithValue("@dateInsert", today);
83                                      cmd.ExecuteNonQuery();
84                                      DisplayStudentData();
85                                      MessageBox.Show("Student added successfully!", "Success", MessageBoxButtons.OK, M
86                                      ClearFields();
87                                  }
88                              }
89                          }
90                      }
91                  }
92                  catch (Exception ex)
93                  {
94                      MessageBox.Show("Error: " + ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
95                  }
96              }
97          }
98
```

```
3 references
99          private void ClearFields()
100         {
101             // Clear all textboxes
102             foreach (Control control in Controls)
103             {
104                 if (control is TextBox textBox)
105                 {
106                     textBox.Clear();
107                 }
108             }
109         }
110
2 references
111         private void student_studentData_CellClick(object sender, DataGridViewCellEventArgs e)
112         {
113             // Your existing code for populating textboxes with selected row data
114             // Check if the clicked cell is not the header row
115             if (e.RowIndex != -1)
116             {
117                 // Get the selected row
118                 DataGridViewRow row = student_studentData.Rows[e.RowIndex];
119
120                 // Populate the text boxes and input boxes with the data from the selected row
121                 s_regNO.Text = row.Cells["s_regNO"].Value.ToString();
122                 s_firstName.Text = row.Cells["s_firstName"].Value.ToString();
123                 s_lastName.Text = row.Cells["s_lastName"].Value.ToString();
124                 s_gender.Text = row.Cells["s_gender"].Value.ToString();
125                 s_address.Text = row.Cells["s_address"].Value.ToString();
126                 s_dob.Value = Convert.ToDateTime(row.Cells["s_dob"].Value);
127                 s_email.Text = row.Cells["s_email"].Value.ToString();
128                 s_mobilePhone.Text = row.Cells["s_mobilePhone"].Value.ToString();
129                 s_homePhone.Text = row.Cells["s_homePhone"].Value.ToString();
130                 s_parentName.Text = row.Cells["s_parentName"].Value.ToString();
131                 s_Pnic.Text = row.Cells["s_Pnic"].Value.ToString();
132                 s_PcontactNo.Text = row.Cells["s_PcontactNo"].Value.ToString();
133             }
134         }
135
```

```
1 reference
private void button2_Click(object sender, EventArgs e)
{
    // Your existing code for updating a student record
    try
    {
        // Establish connection to the database
        using (SqlConnection connection = new SqlConnection(connectionString))
        {
            connection.Open();

            // Define the SQL UPDATE query
            string updateQuery = @"
        UPDATE students
        SET
            s_firstName = @studentFName,
            s_lastName = @studentLName,
            s_gender = @studentGender,
            s_address = @studentAddress,
            s_email = @studentEmail,
            s_mobilePhone = @studentPhone,
            s_dob = @studentDOB,
            s_homePhone = @studentHPhone,
            s_parentName = @studentPName,
            s_Pnic = @studentPNic,
            s_PcontactNo = @studentPContactno
        WHERE
            s_regNO = @studentRg";

            // Create a command object
            using (SqlCommand cmd = new SqlCommand(updateQuery, connection))
            {
```

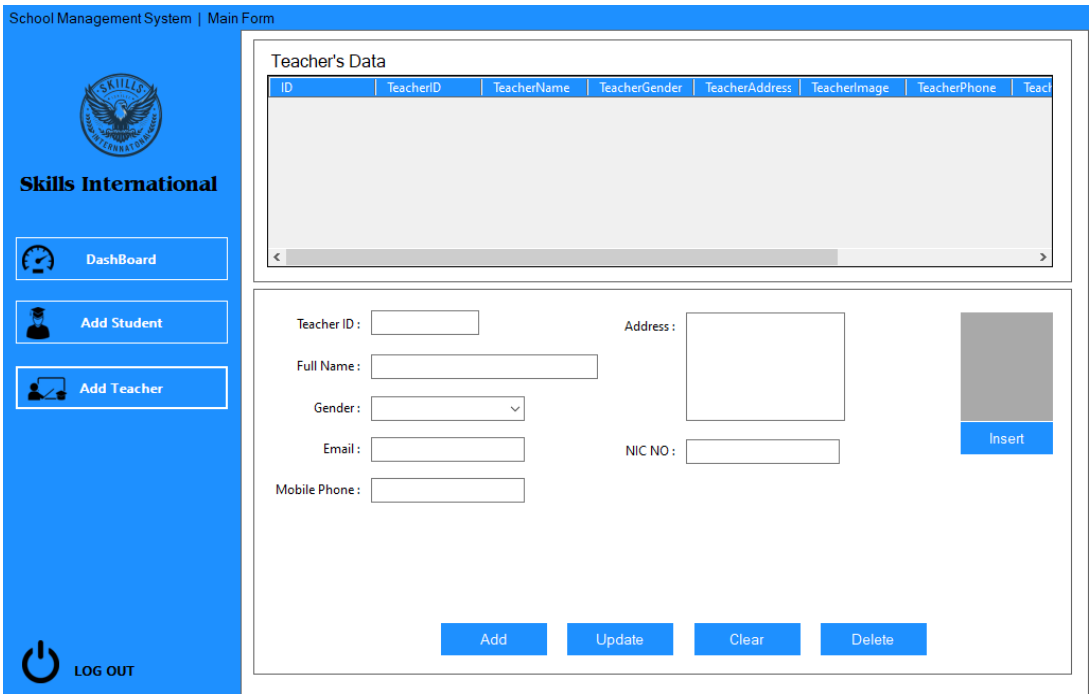
```
            // Create a command object
            using (SqlCommand cmd = new SqlCommand(updateQuery, connection))
            {
                // Set parameter values with the modified data
                cmd.Parameters.AddWithValue("@studentFName", s_firstName.Text.Trim());
                cmd.Parameters.AddWithValue("@studentLName", s_lastName.Text.Trim());
                cmd.Parameters.AddWithValue("@studentGender", s_gender.Text.Trim());
                cmd.Parameters.AddWithValue("@studentAddress", s_address.Text.Trim());
                cmd.Parameters.AddWithValue("@studentEmail", s_email.Text.Trim());
                cmd.Parameters.AddWithValue("@studentPhone", s_mobilePhone.Text.Trim());
                cmd.Parameters.AddWithValue("@studentDOB", s_dob.Value);
                cmd.Parameters.AddWithValue("@studentHPhone", s_homePhone.Text.Trim());
                cmd.Parameters.AddWithValue("@studentPName", s_parentName.Text.Trim());
                cmd.Parameters.AddWithValue("@studentPNic", s_Pnic.Text.Trim());
                cmd.Parameters.AddWithValue("@studentPContactno", s_PcontactNo.Text.Trim());
                cmd.Parameters.AddWithValue("@studentRg", s_regNO.Text.Trim());

                // Execute the UPDATE query
                int rowsAffected = cmd.ExecuteNonQuery();

                // Check if any rows were affected
                if (rowsAffected > 0)
                {
                    MessageBox.Show("Record updated successfully!", "Success", MessageBoxButtons.OK, MessageB

                    // Refresh the DataGridView to reflect the changes
                    DisplayStudentData();

                    // Clear the input fields
                    ClearFields();
                }
                else
                {
                    MessageBox.Show("No record found with the provided Registration No.", "Error", MessageBox
                }
            }
```

```
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show("Error: " + ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }

    }

    1 reference
    private void panel2_Paint(object sender, PaintEventArgs e)
    {
        // Your existing painting logic here
    }

    2 references
    private void dateTimePicker1_ValueChanged(object sender, EventArgs e)
    {
        // Your existing handling logic here
    }


    0 references
    private void button4_Click(object sender, EventArgs e)
    {
        ClearFields();
    }

    1 reference
    private void s_regNO_TextChanged(object sender, EventArgs e)
    {

    }

    1 reference
    private void button4_Click_1(object sender, EventArgs e)
    {
        // Call the ClearFields method to clear all input fields
        s_regNO.Text = "";
        s_firstName.Text = "";
```

```
    1 reference
    private void button4_Click_1(object sender, EventArgs e)
    {
        // Call the ClearFields method to clear all input fields
        s_regNO.Text = "";
        s_firstName.Text = "";
        s_lastName.Text = "";
        s_gender.Text = "";
        s_email.Text = "";
        s_mobilePhone.Text = "";
        s_grade.Text = "";
        s_address.Text = "";
        s_parentName.Text = "";
        s_Pnic.Text = "";
        s_PcontactNo.Text = "";
        s_homePhone.Text = "";

    }

    1 reference
    private void button3_Click(object sender, EventArgs e)
    {
        // Check if any cell is selected
        if (student_studentData.SelectedCells.Count > 0)
        {
            // Ask for confirmation
            DialogResult result = MessageBox.Show("Do you really want to delete this record?", "Confirmation", Me

            if (result == DialogResult.Yes)
            {
                // Get the ID of the selected record
                int selectedRowIndex = student_studentData.SelectedCells[0].RowIndex;
                DataGridViewRow selectedRow = student_studentData.Rows[selectedRowIndex];
                string studentRg = selectedRow.Cells["s_regNO"].Value.ToString(); // Assuming 's_regNO' is the na
```

```
263                 try
264                 {
265                     // Delete the record from the database
266                     using (SqlConnection connection = new SqlConnection(connectionString))
267                     {
268                         connection.Open();
269                         string deleteQuery = "DELETE FROM students WHERE s_regNO = @studentRg";
270                         using (SqlCommand cmd = new SqlCommand(deleteQuery, connection))
271                         {
272                             cmd.Parameters.AddWithValue("@studentRg", studentRg);
273                             int rowsAffected = cmd.ExecuteNonQuery();
274                             if (rowsAffected > 0)
275                             {
276                                 MessageBox.Show("Record deleted successfully!", "Success", MessageBoxButtons.OK,
277                                 // Refresh the DataGridView
278                                 DisplayStudentData();
279                             }
280                             else
281                             {
282                                 MessageBox.Show("No record found with the provided Registration No.", "Error", Me
283                             }
284                         }
285                     }
286                 }
287                 catch (Exception ex)
288                 {
289                     MessageBox.Show("Error: " + ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
290                 }
291             }
292         }
293         else
294         {
295             MessageBox.Show("Please select a record to delete.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Er
296         }
297     }
298 }
299 }
```

4.7 Register Teachers Form Codes



- **Add Teachers Button**

```
45          private void teacher_AddBtn_Click(object sender, EventArgs e)
46          {
47              if (teacher_id.Text == ""
48                  || teacher_name.Text == ""
49                  || teacher_nic.Text == ""
50                  || teacher_address.Text == ""
51                  || teacher_email.Text == ""
52                  || teacher_gender.Text == ""
53                  || teacher_phone.Text == ""
54                  || teacher_image == null
55                  || imagePath == null)
56              {
57                  MessageBox.Show("Please Fill All the Blank Field", "Error Message", MessageBoxButtons.OK, MessageBoxI
58
59              }
60              else
61              {
62                  if (connect.State != ConnectionState.Open)
63                  {
64                      try
65                      {
66                          connect.Open();
67
68                          // Stop Duplicating data
69
70                          String checkTeacherID = "SELECT COUNT(*) FROM teachers WHERE teacher_id=@teacherID";
71                          using (SqlCommand checkTID = new SqlCommand(checkTeacherID, connect))
72                          {
73                              checkTID.Parameters.AddWithValue("@teacherID", teacher_id.Text.Trim());
74                              int count = (int)checkTID.ExecuteScalar();
75
76                              if (count >= 1)
77                              {
78                                  MessageBox.Show("teacher ID:  " + teacher_id.Text.Trim() + " is Already exist", "Erro
79                              }
80                              else
81                              {
```

```
81                              {
82                                  DateTime today = DateTime.Today;
83                                  string insertData = "INSERT INTO teachers " +
84                                  "(teacher_id,teacher_name,teacher_gender,teacher_address," +
85                                  "teacher_nic,teacher_email,teacher_image,teacher_phone,date_insert,)" +
86                                  "VALUES(@teacherID, @teacherName, @teacherGender, @teacherAddress," +
87                                  "@teacherNIC, @teacherEmail,@teacherImage,@teacherPhone, @dateInsert)";
88
89                                  // TO SAVE TO YOUR DIRECTORY
90                                  string path = Path.Combine(@"C:\Users\DINUSHKA_THARIDU-AS\source\repos\School Managem
91
92                                  string directoryPath = Path.GetDirectoryName(path);
93
94                                  if (!Directory.Exists(directoryPath))
95                                  {
96                                      Directory.CreateDirectory(directoryPath);
97                                  }
98
99                                  File.Copy(imagePath, path, true);
100
101
102
103                                 using (SqlCommand cmd = new SqlCommand(insertData, connect))
104                                 {
105                                     cmd.Parameters.AddWithValue("@teacherID", teacher_id.Text.Trim());
106                                     cmd.Parameters.AddWithValue("@teacher_name", teacher_name.Text.Trim());
107                                     cmd.Parameters.AddWithValue("@teacher_gender", teacher_gender.Text.Trim());
108                                     cmd.Parameters.AddWithValue("@teacher_address", teacher_address.Text.Trim());
109                                     cmd.Parameters.AddWithValue("@teacher_nic", teacher_nic.Text.Trim());
110                                     cmd.Parameters.AddWithValue("@teacher_email", teacher_email.Text.Trim());
111                                     cmd.Parameters.AddWithValue("@teacher_image", path);
112                                     cmd.Parameters.AddWithValue("@teacher_phone", teacher_phone.Text.Trim());
113                                     cmd.Parameters.AddWithValue("@dateInsert", today.ToString());
114
115                                     cmd.ExecuteNonQuery();
116
117                                     teacherDisplayData();
118
```

```
                                teacherDisplayData();

                                MessageBox.Show("Added successfully!", "Information Message", MessageBoxButtons.O

                                clearFields();


                            }


                        }


                    }



                }
                catch (Exception ex)
                {
                    MessageBox.Show("Error Connecting Database: " + ex, "Error Message", MessageBoxButtons.OK, Me
                }
                finally
                {
                    connect.Close();
                }
            }
        }


    }
    2 references
```

- **Clear Teachers Button**

```
    1 reference
    private void teacher_ClearBtn_Click(object sender, EventArgs e)
    {
        clearFields();
    }
```

- **“teacher_gridData” Data Grid Codes**

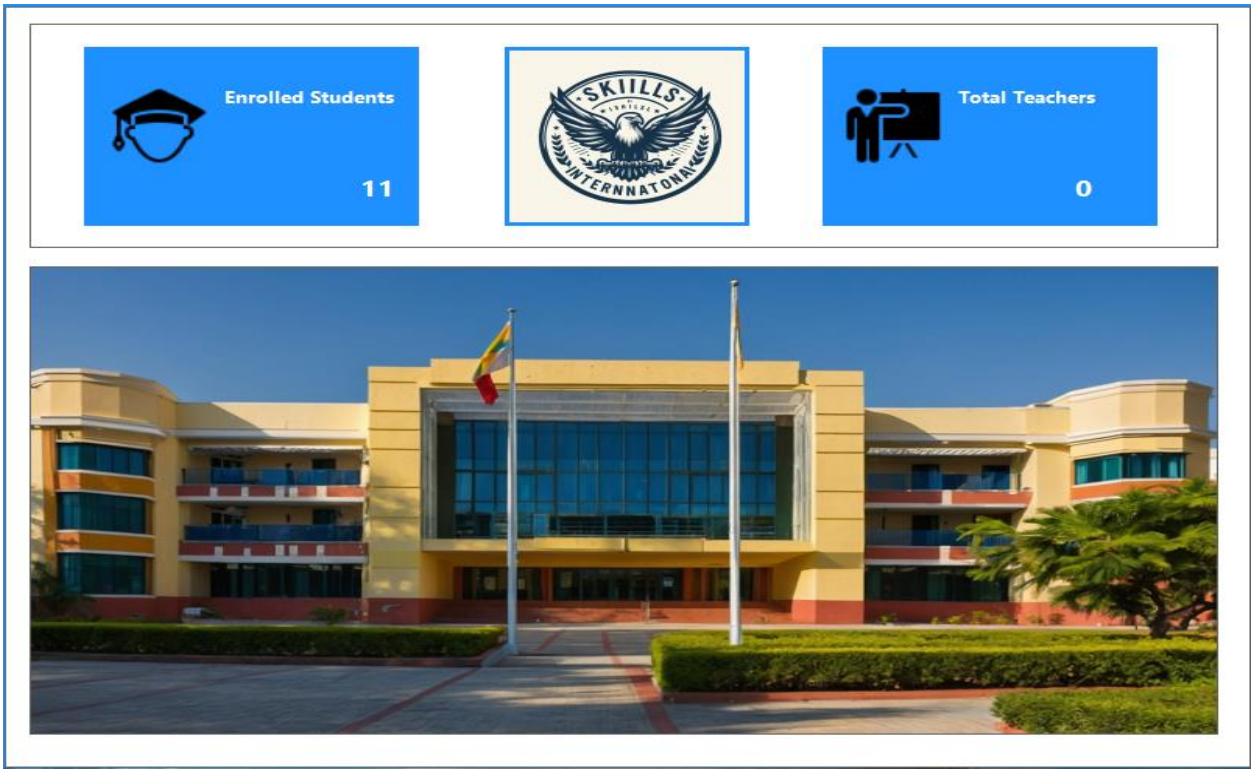
```
1 reference
private void teacher_gridData_CellClick(object sender, DataGridViewCellEventArgs e)
{
    if (e.RowIndex != -1)
    {
        DataGridViewRow row = teacher_gridData.Rows[e.RowIndex];
        teacher_id.Text = row.Cells[1].Value.ToString();
        teacher_name.Text = row.Cells[2].Value.ToString();
        teacher_gender.Text = row.Cells[3].Value.ToString();
        teacher_address.Text = row.Cells[4].Value.ToString();
        teacher_email.Text = row.Cells[5].Value.ToString();
        teacher_phone.Text = row.Cells[6].Value.ToString();
        teacher_nic.Text = row.Cells[7].Value.ToString();


        imagePath = row.Cells[8].Value.ToString();

        string imageData = row.Cells[5].Value.ToString();

        if (imageData != null && imageData.Length > 0)
        {
            teacher_image.Image = Image.FromFile(imageData);
        }
        else
        {
            teacher_image.Image = null;
        }



    }
}
```

- **“AddTeachersData” Class**

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data;
using System.Data.SqlClient;
using System.IO;
using Microsoft.Data.SqlClient;


namespace School_Management_System
{
    7 references
    internal class AddTeachersData
    {
        SqlConnection connect = new SqlConnection(@"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=C:\Users\DINU
        1 reference
        public int ID { set; get; }
        1 reference
        public string TeacherID { set; get; }
        1 reference
        public string TeacherName { set; get; }
        1 reference
        public string TeacherGender{ set; get; }
        1 reference
        public string TeacherAddress{ set; get; }
        1 reference
        public string TeacherImage { set; get;}
        1 reference
        public string TeacherPhone { set; get; }
        1 reference
        public string TeacherEmail { set; get; }
        1 reference
        public string TeacherNIC { set; get; }
        1 reference
        public string DateInsert { set; get; }



        1 reference
        public List<AddTeachersData> teacherData()
```

```
        1 reference
        public List<AddTeachersData> teacherData()
        {
            List<AddTeachersData> listData = new List<AddTeachersData>();
            if (connect.State != ConnectionState.Open)
            {
                try
                {
                    connect.Open();
                    string sql = "SELECT * FROM teachers WHERE data_delete IS NULL";
                    using (SqlCommand cmd = new SqlCommand(sql, connect))
                    {
                        SqlDataReader reader = cmd.ExecuteReader();
                        while (reader.Read())
                        {
                            AddTeachersData addTD=new AddTeachersData();
                            addTD.ID = (int)reader["id"];
                            addTD.TeacherID = reader["teacher_id"].ToString();
                            addTD.TeacherName = reader["teacher_name"].ToString();
                            addTD.TeacherGender = reader["teacher_gender"].ToString();
                            addTD.TeacherAddress = reader["teacher_address"].ToString();
                            addTD.TeacherEmail = reader["teacher_email"].ToString();
                            addTD.TeacherPhone = reader["teacher_phone"].ToString();
                            addTD.TeacherNIC = reader["teacher_nic"].ToString();
                            addTD.DateInsert = reader["date_insert"].ToString();
                            addTD.TeacherImage= reader["teacher_image"].ToString();

                            listData.Add(addTD);
                        }
                        reader.Close();
                    }
                }
                catch (Exception ex)
                {
                    Console.WriteLine("Error Connecting Database: " + ex);
                }
                finally
```

4.8 Dashboard Codes



- **Dashboard Form Codes**

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;
using Microsoft.Data.SqlClient;



namespace School_Management_System
{
    5 references
    public partial class DashboardForm : UserControl
    {
        private readonly string connectionString = @"Data Source=(LocalDB)\MSSQLLocalDB;Initial Catalog=C:\USERS\DINU
        1 reference
        public DashboardForm()
        {
            InitializeComponent();
            UpdateStudentCount();
        }

        2 references
        public void UpdateStudentCount()
        {
            try
            {
                using (SqlConnection connection = new SqlConnection(connectionString))
                {
                    connection.Open();
                    string countQuery = "SELECT COUNT(*) FROM students";
                    using (SqlCommand command = new SqlCommand(countQuery, connection))
                    {
```

```
            }
            catch (Exception ex)
            {
                MessageBox.Show("Error: " + ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }

        1 reference
        private void panel2_Paint(object sender, PaintEventArgs e)
        {

        }

        1 reference
        public void DashboardForm_Load(object sender, EventArgs e)
        {
            UpdateStudentCount();
        }
    }
}
```

4.9 Data Base Codes

- **Student table**

```
CREATE TABLE students
(
s_rgNo INT PRIMARY KEY IDENTITY(1,1),
s_regNO  VARCHAR (50) ,
s_firstName  VARCHAR (50) NULL,
s_lastName VARCHAR (50) NULL,
s_dateOfBirth DATE NULL,
s_gender VARCHAR (50) NULL,
s_address VARCHAR (
s_email VARCHAR (50) ,
s_mobilePhone INT ,
s_dob DATE NULL,
s_homePhone INT ,
s_parentName VARCHAR (50) NULL,
s_Pnic VARCHAR (50) NULL,
s_PcontactNo INT NULL,
s_date_insert DATE NULL,
s_date_update DATE NULL,
s_date_delete DATE NULL,
s_student_grade VARCHAR (50) NULL

);

SELECT * FROM students ;
```

- **Teachers table**

```sql
CREATE TABLE teachers
(
    id INT PRIMARY KEY IDENTITY (1,1),
    teacher_id VARCHAR (MAX) NULL,
    teacher_name VARCHAR (MAX) NULL,
    teacher_gender VARCHAR (MAX) NULL,
    teacher_address VARCHAR (MAX) NULL,
    teacher_email VARCHAR (MAX) NULL,
    teacher_phone VARCHAR (MAX) NULL,
    teacher_nic VARCHAR (MAX) NULL,
    teacher_image VARCHAR (MAX) NULL,
    date_insert DATE,
    date_update DATE,
    date_delete DATE,
);
SELECT * FROM teachers ;
```

05.Testing the System

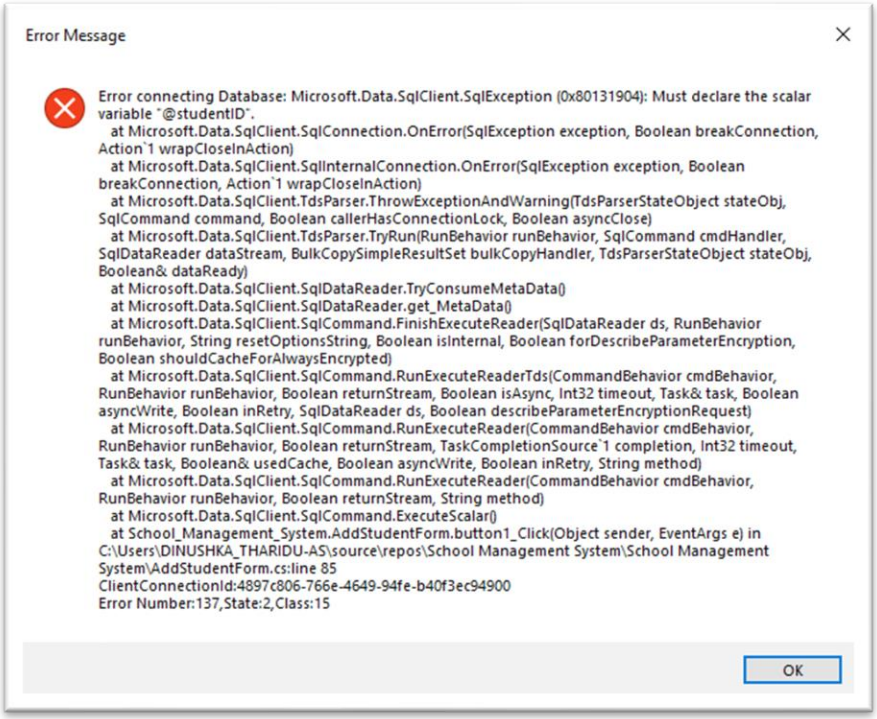
5.1 Testing Strategies

There are different testing strategies to ensure comprehensive testing:

- **Unit Testing:** Individual modules or units of code are tested in isolation to verify they function correctly.
- **Integration Testing:** Different modules are integrated and tested together to ensure they interact seamlessly.
- **System Testing:** The entire system is tested as a whole with various use cases and scenarios to identify any functional or non-functional issues.
- **User Acceptance Testing (UAT):** End-users test the system to ensure it meets their needs and is user-friendly.

A combination of these strategies provides a thorough evaluation of the SMS.

5.2 Bugs Encountered during System creating



Error Message

Error connecting Database: Microsoft.Data.SqlClient.SqlException (0x80131904): Must declare the scalar variable "@studentID".
at Microsoft.Data.SqlClient.SqlConnection.OnError(SqlException exception, Boolean breakConnection, Action`1 wrapCloseInAction)
at Microsoft.Data.SqlClient.SqlInternalConnection.OnError(SqlException exception, Boolean breakConnection, Action`1 wrapCloseInAction)
at Microsoft.Data.SqlClient.TdsParser.ThrowExceptionAndWarning(TdsParserStateObject stateObj, SqlCommand command, Boolean callerHasConnectionLock, Boolean asyncClose)
at Microsoft.Data.SqlClient.TdsParser.TryRun(RunBehavior runBehavior, SqlCommand cmdHandler, SqlDataReader dataStream, BulkCopySimpleResultSet bulkCopyHandler, TdsParserStateObject stateObj, Boolean& dataReady)
at Microsoft.Data.SqlClient.SqlDataReader.TryConsumeMetaData()
at Microsoft.Data.SqlClient.SqlDataReader.get_MetaData()
at Microsoft.Data.SqlClient.SqlCommand.FinishExecuteReader(SqlDataReader ds, RunBehavior runBehavior, String resetOptionsString, Boolean isInternal, Boolean forDescribeParameterEncryption, Boolean shouldCacheForAlwaysEncrypted)
at Microsoft.Data.SqlClient.SqlCommand.RunExecuteReaderTds(CommandBehavior cmdBehavior, RunBehavior runBehavior, Boolean returnStream, Boolean isAsync, Int32 timeout, Task& task, Boolean asyncWrite, Boolean inRetry, SqlDataReader ds, Boolean describeParameterEncryptionRequest)
at Microsoft.Data.SqlClient.SqlCommand.RunExecuteReader(CommandBehavior cmdBehavior, RunBehavior runBehavior, Boolean returnStream, TaskCompletionSource`1 completion, Int32 timeout, Task& task, Boolean& usedCache, Boolean asyncWrite, Boolean inRetry, String method)
at Microsoft.Data.SqlClient.SqlCommand.RunExecuteReader(CommandBehavior cmdBehavior, RunBehavior runBehavior, Boolean returnStream, String method)
at Microsoft.Data.SqlClient.SqlCommand.ExecuteScalar()
at School_Management_System.AddStudentForm.button1_Click(Object sender, EventArgs e) in C:\Users\DINUSHKA_THARIDU-AS\source\repos\School Management System\School Management System\AddStudentForm.cs:line 85
ClientConnectionId:4897c806-766e-4649-94fe-b40f3ec94900
Error Number:137,State:2,Class:15

OK

06.System Deployment

6.1 Installation Instructions

Providing clear installation instructions is crucial for a smooth deployment process. Here's what to include:

- **System Requirements:** Specify the minimum hardware and software requirements (operating system, RAM, storage) necessary to run the SMS.
    - **Operating System:** Windows 10
    - **Ram**: 4GB (minimum – 8GB recommended)
    - **Storage :** 100MB
- **Installation Steps:** Detail the steps involved in running the installer and configuring the software (e.g., database connection details, user creation).
- **Troubleshooting Guide (Optional):** Include a basic troubleshooting guide for common installation issues users might encounter.

These instructions will guide users through the installation process and ensure they can start using the SMS effectively.

07.System Maintenance

7.1 Maintenance Plan

A well-defined maintenance plan outlines the activities required to keep the SMS running smoothly and securely. The plan should address:

- **Bug Fixes:** Address any bugs or errors reported by users through a ticketing system or feedback mechanism.
- **System Updates:** Implement critical security updates and bug fixes released for the underlying software libraries or frameworks used in the SMS.
- **Performance Monitoring:** Monitor system performance (response times, resource utilization) and proactively identify potential bottlenecks. Regular database maintenance like optimizing queries and indexes can also be included.
- **User Training:** Provide ongoing user training to ensure users are familiar with new functionalities or updates to the SMS. This can be achieved through user manuals, online tutorials, or workshops.

A comprehensive maintenance plan ensures the SMS remains reliable, secure, and up-to-date over time.

08.Conclusion

This School Management System (SMS) project has the potential to streamline administrative processes and improve efficiency within a school environment.

8.1 Summary of Achievements

- Developed a user-friendly system
- Can Register Students
- Can Register Teachers
- It has a Loging Form with username and password

8.2 Future Enhancements

- Facilitate access to the System for Student and Teachers
- Develop UI more User Friendly
- Connect database to Remote Server

8.3 Overall Impact

The successful implementation of the SMS can significantly impact the school by:

- Reducing administrative workload for staff.
- Improving data accuracy and accessibility.
- Enhancing communication and collaboration between school stakeholders.
- Facilitating data-driven decision making for school leadership.
- Creating a more efficient and organized learning environment.

09.References

- ChatGPT (Open AI)
- Google Gemini AI
- Microsoft Copilot
- Internet
- YouTube
- W3 School - C# Tutorial (C Sharp) (w3schools.com)
- Elms
- DiTEC Book
- Draw.io - draw.io (diagrams.net)

The End..!