Case Study: Perform CRUD Operations on Employee Data in ASP.NET WebForm Application using 3-Tier Architecture

Requirements

Create an ASP.NET Web Forms application to manage employee data with the following fields:

- `empno` (int, primary key)
- `ename` (varchar)
- `job` (varchar)
- `sal` (decimal)
- `deptno` (int)

Use **GridView** for displaying data and provide **Insert**, **Update**, **Delete**, and **Select** functionality.

Architecture Overview: 3-Tier Architecture

1. **Presentation Layer** (UI): ASP.NET Web Forms (.aspx)
2. **Business Logic Layer (BLL)**: Class that handles application logic
3. **Data Access Layer (DAL)**: Interacts with the database using ADO.NET

Database Script (SQL Server)

```
CREATE TABLE Employee (
    empno INT PRIMARY KEY,
    ename VARCHAR(50),
    job VARCHAR(50),
    sal DECIMAL(10, 2),
    deptno INT
);
```

File/Folder Structure

```
- App_Code
  - DAL
    - EmployeeDAL.cs
  - BLL
    - EmployeeBLL.cs
- Default.aspx
- Default.aspx.cs
- Web.config
```

Data Access Layer (DAL): EmployeeDAL.cs

```
using System;
using System.Data;
using System.Data.SqlClient;
using System.Configuration;

public class EmployeeDAL
{
    private string conStr =
ConfigurationManager.ConnectionStrings["DBCS"].ConnectionString;

    public DataTable GetEmployees()
    {
        using (SqlConnection con = new SqlConnection(conStr))
        {
            SqlDataAdapter da = new SqlDataAdapter("SELECT * FROM Employee", con);
            DataTable dt = new DataTable();
            da.Fill(dt);
            return dt;
        }
    }

    public void InsertEmployee(int empno, string ename, string job, decimal sal,
int deptno)
    {
        using (SqlConnection con = new SqlConnection(conStr))
        {
            SqlCommand cmd = new SqlCommand("INSERT INTO Employee VALUES(@empno,
@ename, @job, @sal, @deptno)", con);
            cmd.Parameters.AddWithValue("@empno", empno);
            cmd.Parameters.AddWithValue("@ename", ename);
            cmd.Parameters.AddWithValue("@job", job);
            cmd.Parameters.AddWithValue("@sal", sal);
            cmd.Parameters.AddWithValue("@deptno", deptno);
            con.Open();
            cmd.ExecuteNonQuery();
        }
    }

    public void UpdateEmployee(int empno, string ename, string job, decimal sal,
int deptno)
    {
        using (SqlConnection con = new SqlConnection(conStr))
        {
            SqlCommand cmd = new SqlCommand("UPDATE Employee SET ename=@ename,
job=@job, sal=@sal, deptno=@deptno WHERE empno=@empno", con);
            cmd.Parameters.AddWithValue("@empno", empno);
            cmd.Parameters.AddWithValue("@ename", ename);
            cmd.Parameters.AddWithValue("@job", job);
            cmd.Parameters.AddWithValue("@sal", sal);
            cmd.Parameters.AddWithValue("@deptno", deptno);
```

```
            con.Open();
            cmd.ExecuteNonQuery();
        }
    }

    public void DeleteEmployee(int empno)
    {
        using (SqlConnection con = new SqlConnection(conStr))
        {
            SqlCommand cmd = new SqlCommand("DELETE FROM Employee WHERE
empno=@empno", con);
            cmd.Parameters.AddWithValue("@empno", empno);
            con.Open();
            cmd.ExecuteNonQuery();
        }
    }
}
```

Business Logic Layer (BLL): `EmployeeBLL.cs`

```
using System.Data;

public class EmployeeBLL
{
    EmployeeDAL dal = new EmployeeDAL();

    public DataTable GetEmployees()
    {
        return dal.GetEmployees();
    }

    public void AddEmployee(int empno, string ename, string job, decimal sal, int
deptno)
    {
        dal.InsertEmployee(empno, ename, job, sal, deptno);
    }

    public void ModifyEmployee(int empno, string ename, string job, decimal sal,
int deptno)
    {
        dal.UpdateEmployee(empno, ename, job, sal, deptno);
    }

    public void RemoveEmployee(int empno)
    {
        dal.DeleteEmployee(empno);
    }
}
```

Presentation Layer: Default.aspx

```
<asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="False"
    DataKeyNames="empno" OnRowEditing="GridView1_RowEditing"
    OnRowUpdating="GridView1_RowUpdating"
OnRowCancelingEdit="GridView1_RowCancelingEdit"
    OnRowDeleting="GridView1_RowDeleting">
    <Columns>
        <asp:BoundField DataField="empno" HeaderText="Emp No" ReadOnly="True" />
        <asp:BoundField DataField="ename" HeaderText="Name" />
        <asp:BoundField DataField="job" HeaderText="Job" />
        <asp:BoundField DataField="sal" HeaderText="Salary" />
        <asp:BoundField DataField="deptno" HeaderText="Dept No" />
        <asp:CommandField ShowEditButton="True" ShowDeleteButton="True" />
    </Columns>
</asp:GridView>

<br />

<asp:Label ID="lblMessage" runat="server" ForeColor="Red" />
<br />
<h4>Add New Employee</h4>
<asp:TextBox ID="txtEmpno" runat="server" Placeholder="Emp No" />
<asp:TextBox ID="txtEname" runat="server" Placeholder="Name" />
<asp:TextBox ID="txtJob" runat="server" Placeholder="Job" />
<asp:TextBox ID="txtSal" runat="server" Placeholder="Salary" />
<asp:TextBox ID="txtDeptno" runat="server" Placeholder="Dept No" />
<asp:Button ID="btnAdd" runat="server" Text="Add Employee" OnClick="btnAdd_Click"
/>
```

Code-behind: Default.aspx.cs

```
public partial class _Default : Page
{
    EmployeeBLL bll = new EmployeeBLL();

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

    private void BindGrid()
    {
        GridView1.DataSource = bll.GetEmployees();
        GridView1.DataBind();
    }

    protected void btnAdd_Click(object sender, EventArgs e)
```

```
    {
        try
        {
            bll.AddEmployee(
                int.Parse(txtEmpno.Text),
                txtEname.Text,
                txtJob.Text,
                decimal.Parse(txtSal.Text),
                int.Parse(txtDeptno.Text)
            );
            lblMessage.Text = "Employee Added Successfully!";
            BindGrid();
        }
        catch (Exception ex)
        {
            lblMessage.Text = ex.Message;
        }
    }

    protected void GridView1_RowEditing(object sender, GridViewEditEventArgs e)
    {
        GridView1.EditIndex = e.NewEditIndex;
        BindGrid();
    }

    protected void GridView1_RowUpdating(object sender, GridViewUpdateEventArgs e)
    {
        GridViewRow row = GridView1.Rows[e.RowIndex];
        int empno = Convert.ToInt32(GridView1.DataKeys[e.RowIndex].Value);
        string ename = ((TextBox)row.Cells[1].Controls[0]).Text;
        string job = ((TextBox)row.Cells[2].Controls[0]).Text;
        decimal sal = Convert.ToDecimal(((TextBox)row.Cells[3].Controls[0]).Text);
        int deptno = Convert.ToInt32(((TextBox)row.Cells[4].Controls[0]).Text);

        bll.ModifyEmployee(empno, ename, job, sal, deptno);
        GridView1.EditIndex = -1;
        BindGrid();
    }

    protected void GridView1_RowCancelingEdit(object sender,
GridViewCancelEditEventArgs e)
    {
        GridView1.EditIndex = -1;
        BindGrid();
    }

    protected void GridView1_RowDeleting(object sender, GridViewDeleteEventArgs e)
    {
        int empno = Convert.ToInt32(GridView1.DataKeys[e.RowIndex].Value);
        bll.RemoveEmployee(empno);
        BindGrid();
    }
}
```

Web.config (Add connection string)

```
<configuration>
  <connectionStrings>
    <add name="DBCS" connectionString="Data Source=.;Initial
Catalog=YourDBName;Integrated Security=True" />
  </connectionStrings>
</configuration>
```

Conclusion

This example shows how to implement:

- 3-tier architecture (DAL, BLL, UI)
- ADO.NET for DB operations
- GridView for CRUD
- Web Forms for UI