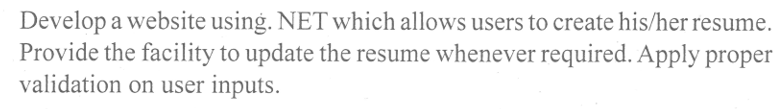
**[1].**

****

**Database:**

CREATE TABLE Resumes (

ResumeID INT IDENTITY(1,1) PRIMARY KEY,

Name NVARCHAR(100) NOT NULL,

Email NVARCHAR(100) NOT NULL,

Phone NVARCHAR(15) NOT NULL,

Education NVARCHAR(500) NOT NULL,

Skills NVARCHAR(500),

Experience NVARCHAR(500),

Summary NVARCHAR(1000),

LastUpdated DATETIME

);

Code:

**CreateResume.aspx**

<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="CreateResume.aspx.cs" Inherits="ResumeApp.CreateResume" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">

<title>Create or Update Resume</title>

</head>

<body>

<form id="form1" runat="server">

<div>

<h2>Create or Update Resume</h2>

<asp:ValidationSummary ID="ValidationSummary1" runat="server" ForeColor="Red" />

<div>

<label>Name:</label>

<asp:TextBox ID="txtName" runat="server"></asp:TextBox>

<asp:RequiredFieldValidator ID="rfvName" runat="server" ControlToValidate="txtName" ErrorMessage="Name is required" ForeColor="Red" />

</div>

<div>

<label>Email:</label>

<asp:TextBox ID="txtEmail" runat="server"></asp:TextBox>

<asp:RequiredFieldValidator ID="rfvEmail" runat="server" ControlToValidate="txtEmail" ErrorMessage="Email is required" ForeColor="Red" />

<asp:RegularExpressionValidator ID="revEmail" runat="server" ControlToValidate="txtEmail"

ErrorMessage="Invalid Email" ValidationExpression="^\S+@\S+\.\S+$" ForeColor="Red" />

</div>

<div>

<label>Phone:</label>

<asp:TextBox ID="txtPhone" runat="server"></asp:TextBox>

<asp:RequiredFieldValidator ID="rfvPhone" runat="server" ControlToValidate="txtPhone" ErrorMessage="Phone is required" ForeColor="Red" />

<asp:RegularExpressionValidator ID="revPhone" runat="server" ControlToValidate="txtPhone"

ErrorMessage="Invalid Phone Number" ValidationExpression="^\d{10}$" ForeColor="Red" />

</div>

<div>

<label>Education:</label>

<asp:TextBox ID="txtEducation" runat="server"></asp:TextBox>

<asp:RequiredFieldValidator ID="rfvEducation" runat="server" ControlToValidate="txtEducation" ErrorMessage="Education is required" ForeColor="Red" />

</div>

<div>

<label>Skills:</label>

<asp:TextBox ID="txtSkills" runat="server" TextMode="MultiLine"></asp:TextBox>

</div>

<div>

<label>Experience:</label>

<asp:TextBox ID="txtExperience" runat="server" TextMode="MultiLine"></asp:TextBox>

</div>

<div>

<label>Summary:</label>

<asp:TextBox ID="txtSummary" runat="server" TextMode="MultiLine"></asp:TextBox>

</div>

<asp:HiddenField ID="hfResumeID" runat="server" />

<asp:Button ID="btnSubmit" runat="server" Text="Save" OnClick="btnSubmit\_Click" />

</div>

</form>

</body>

</html>

**CreateResume.aspx.cs**

**using System;**

**using System.Data.SqlClient;**

**using System.Configuration;**

**namespace ResumeApp**

**{**

**public partial class CreateResume : System.Web.UI.Page**

**{**

**protected void Page\_Load(object sender, EventArgs e)**

**{**

**if (!IsPostBack)**

**{**

**// If this page is accessed for editing an existing resume**

**if (Request.QueryString["ResumeID"] != null)**

**{**

**int resumeID = int.Parse(Request.QueryString["ResumeID"]);**

**LoadResume(resumeID);**

**}**

**}**

**}**

**// Load resume data for editing**

**private void LoadResume(int resumeID)**

**{**

**string connectionString = ConfigurationManager.ConnectionStrings["DefaultConnection"].ConnectionString;**

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

**{**

**string query = "SELECT \* FROM Resumes WHERE ResumeID = @ResumeID";**

**SqlCommand cmd = new SqlCommand(query, conn);**

**cmd.Parameters.AddWithValue("@ResumeID", resumeID);**

**conn.Open();**

**SqlDataReader reader = cmd.ExecuteReader();**

**if (reader.Read())**

**{**

**txtName.Text = reader["Name"].ToString();**

**txtEmail.Text = reader["Email"].ToString();**

**txtPhone.Text = reader["Phone"].ToString();**

**txtEducation.Text = reader["Education"].ToString();**

**txtSkills.Text = reader["Skills"].ToString();**

**txtExperience.Text = reader["Experience"].ToString();**

**txtSummary.Text = reader["Summary"].ToString();**

**hfResumeID.Value = reader["ResumeID"].ToString();**

**}**

**reader.Close();**

**}**

**}**

**// Submit button event to insert/update resume**

**protected void btnSubmit\_Click(object sender, EventArgs e)**

**{**

**if (Page.IsValid)**

**{**

**string connectionString = ConfigurationManager.ConnectionStrings["DefaultConnection"].ConnectionString;**

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

**{**

**conn.Open();**

**SqlCommand cmd;**

**// If this is an edit operation (ResumeID exists)**

**if (!string.IsNullOrEmpty(hfResumeID.Value))**

**{**

**cmd = new SqlCommand("UPDATE Resumes SET Name = @Name, Email = @Email, Phone = @Phone, Education = @Education, Skills = @Skills, Experience = @Experience, Summary = @Summary, LastUpdated = @LastUpdated WHERE ResumeID = @ResumeID", conn);**

**cmd.Parameters.AddWithValue("@ResumeID", hfResumeID.Value);**

**}**

**else**

**{**

**// If this is a create operation**

**cmd = new SqlCommand("INSERT INTO Resumes (Name, Email, Phone, Education, Skills, Experience, Summary, LastUpdated) VALUES (@Name, @Email, @Phone, @Education, @Skills, @Experience, @Summary, @LastUpdated)", conn);**

**}**

**cmd.Parameters.AddWithValue("@Name", txtName.Text);**

**cmd.Parameters.AddWithValue("@Email", txtEmail.Text);**

**cmd.Parameters.AddWithValue("@Phone", txtPhone.Text);**

**cmd.Parameters.AddWithValue("@Education", txtEducation.Text);**

**cmd.Parameters.AddWithValue("@Skills", txtSkills.Text);**

**cmd.Parameters.AddWithValue("@Experience", txtExperience.Text);**

**cmd.Parameters.AddWithValue("@Summary", txtSummary.Text);**

**cmd.Parameters.AddWithValue("@LastUpdated", DateTime.Now);**

**cmd.ExecuteNonQuery();**

**}**

**// Redirect back to the listing or confirmation page**

**Response.Redirect("ResumeList.aspx");**

**}**

**}**

**}**

**}**

**Web.config**

**<configuration>**

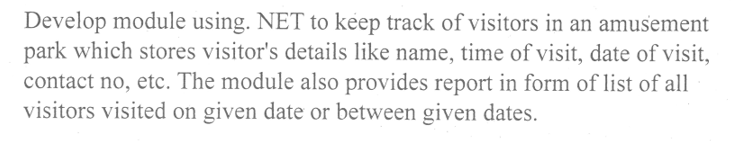
**<connectionStrings>**

**<add name="DefaultConnection" connectionString="Server=your\_server\_name;Database=ResumeDB;Trusted\_Connection=True;" providerName="System.Data.SqlClient" />**

**</connectionStrings>**

**</configuration>**

[2].



To create a .NET module for tracking visitors at an amusement park with features like storing visitor details and generating reports for specific dates or ranges, I'll provide an example using ASP.NET Web Forms, which uses `.aspx` for the front-end and `.aspx.cs` (C#) for the back-end.

### 1. \*\*ASP.NET Web Form: Visitor Tracking Module\*\*

Here’s an outline of the `.aspx` (HTML) page where users can enter visitor details and search for records:

#### \*\*Visitors.aspx\*\* (front-end)

```aspx

<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Visitors.aspx.cs" Inherits="Visitors" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">

<title>Visitor Tracking Module</title>

</head>

<body>

<form id="form1" runat="server">

<h2>Amusement Park Visitor Tracking</h2>

<!-- Input Form for Visitor Details -->

<div>

<label>Name:</label>

<asp:TextBox ID="txtName" runat="server"></asp:TextBox>

<br />

<label>Time of Visit:</label>

<asp:TextBox ID="txtTime" runat="server"></asp:TextBox>

<br />

<label>Date of Visit:</label>

<asp:TextBox ID="txtDate" runat="server"></asp:TextBox>

<asp:CalendarExtender ID="CalendarExtender1" runat="server" TargetControlID="txtDate" Format="yyyy-MM-dd" />

<br />

<label>Contact Number:</label>

<asp:TextBox ID="txtContact" runat="server"></asp:TextBox>

<br />

<asp:Button ID="btnAddVisitor" runat="server" Text="Add Visitor" OnClick="btnAddVisitor\_Click" />

</div>

<!-- Date Filter Section -->

<div>

<h3>Search Visitors by Date</h3>

<label>From Date:</label>

<asp:TextBox ID="txtFromDate" runat="server"></asp:TextBox>

<asp:CalendarExtender ID="CalendarExtender2" runat="server" TargetControlID="txtFromDate" Format="yyyy-MM-dd" />

<br />

<label>To Date:</label>

<asp:TextBox ID="txtToDate" runat="server"></asp:TextBox>

<asp:CalendarExtender ID="CalendarExtender3" runat="server" TargetControlID="txtToDate" Format="yyyy-MM-dd" />

<br />

<asp:Button ID="btnSearch" runat="server" Text="Search" OnClick="btnSearch\_Click" />

</div>

<!-- Display Visitor List -->

<div>

<asp:GridView ID="gvVisitors" runat="server" AutoGenerateColumns="false">

<Columns>

<asp:BoundField DataField="Name" HeaderText="Name" />

<asp:BoundField DataField="TimeOfVisit" HeaderText="Time of Visit" />

<asp:BoundField DataField="DateOfVisit" HeaderText="Date of Visit" />

<asp:BoundField DataField="ContactNo" HeaderText="Contact Number" />

</Columns>

</asp:GridView>

</div>

</form>

</body>

</html>

```

### 2. \*\*Code-behind (C#) Logic for the Visitor Module\*\*

The `.aspx.cs` file contains logic for handling the addition of visitors and fetching the records based on date filters.

#### \*\*Visitors.aspx.cs\*\* (back-end)

```csharp

using System;

using System.Data;

using System.Data.SqlClient;

using System.Configuration;

public partial class Visitors : System.Web.UI.Page

{

// Database connection string

string connectionString = ConfigurationManager.ConnectionStrings["AmusementParkDB"].ConnectionString;

protected void Page\_Load(object sender, EventArgs e)

{

if (!IsPostBack)

{

BindGrid(); // Load the visitor list on initial load

}

}

// Add Visitor to the Database

protected void btnAddVisitor\_Click(object sender, EventArgs e)

{

using (SqlConnection con = new SqlConnection(connectionString))

{

SqlCommand cmd = new SqlCommand("INSERT INTO Visitors (Name, TimeOfVisit, DateOfVisit, ContactNo) VALUES (@Name, @TimeOfVisit, @DateOfVisit, @ContactNo)", con);

cmd.Parameters.AddWithValue("@Name", txtName.Text);

cmd.Parameters.AddWithValue("@TimeOfVisit", txtTime.Text);

cmd.Parameters.AddWithValue("@DateOfVisit", txtDate.Text);

cmd.Parameters.AddWithValue("@ContactNo", txtContact.Text);

con.Open();

cmd.ExecuteNonQuery();

con.Close();

}

// Clear the form

txtName.Text = "";

txtTime.Text = "";

txtDate.Text = "";

txtContact.Text = "";

// Rebind the grid to show updated data

BindGrid();

}

// Fetch and bind the visitor data to GridView

private void BindGrid()

{

using (SqlConnection con = new SqlConnection(connectionString))

{

SqlCommand cmd = new SqlCommand("SELECT Name, TimeOfVisit, DateOfVisit, ContactNo FROM Visitors", con);

SqlDataAdapter da = new SqlDataAdapter(cmd);

DataTable dt = new DataTable();

da.Fill(dt);

gvVisitors.DataSource = dt;

gvVisitors.DataBind();

}

}

// Search Visitors by Date Range

protected void btnSearch\_Click(object sender, EventArgs e)

{

using (SqlConnection con = new SqlConnection(connectionString))

{

SqlCommand cmd = new SqlCommand("SELECT Name, TimeOfVisit, DateOfVisit, ContactNo FROM Visitors WHERE DateOfVisit BETWEEN @FromDate AND @ToDate", con);

cmd.Parameters.AddWithValue("@FromDate", txtFromDate.Text);

cmd.Parameters.AddWithValue("@ToDate", txtToDate.Text);

SqlDataAdapter da = new SqlDataAdapter(cmd);

DataTable dt = new DataTable();

da.Fill(dt);

gvVisitors.DataSource = dt;

gvVisitors.DataBind();

}

}

}

```

### 3. \*\*Database Structure\*\*

Assume the following SQL Server table for storing visitor information:

```sql

CREATE TABLE Visitors (

Id INT PRIMARY KEY IDENTITY,

Name NVARCHAR(100),

TimeOfVisit NVARCHAR(10),

DateOfVisit DATE,

ContactNo NVARCHAR(15)

);

```

### Key Features:

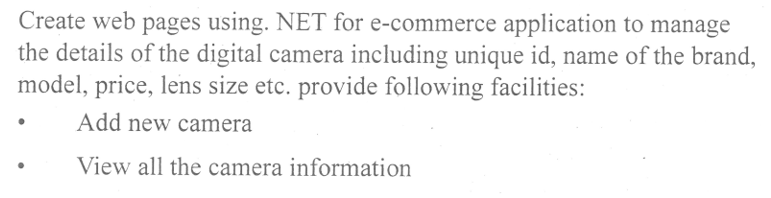
1. \*\*Adding Visitors\*\*: Input for visitor name, time, date, and contact number.

2. \*\*Date Search\*\*: Users can search visitors by a specific date range.

3. \*\*Display Visitors\*\*: A grid view displays the list of visitors.

This example provides a basic visitor management system in an amusement park using ASP.NET and C#. You may need to adjust your configuration settings, such as connection strings and database fields, to match your environment.

[3].



For creating an e-commerce application to manage digital camera details in ASP.NET, we will create two main functionalities:

1. \*\*Add a new camera\*\*

2. \*\*View all camera information\*\*

Here's an example implementation using ASP.NET Web Forms.

### 1. \*\*ASP.NET Web Form for Adding and Viewing Cameras\*\*

#### \*\*Cameras.aspx\*\* (front-end)

This page allows the user to add new camera details and view all the camera information in a GridView.

```aspx

<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Cameras.aspx.cs" Inherits="Cameras" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">

<title>Camera Management</title>

</head>

<body>

<form id="form1" runat="server">

<h2>Digital Camera Management</h2>

<!-- Form to Add a New Camera -->

<div>

<h3>Add New Camera</h3>

<label>Brand:</label>

<asp:TextBox ID="txtBrand" runat="server"></asp:TextBox>

<br />

<label>Model:</label>

<asp:TextBox ID="txtModel" runat="server"></asp:TextBox>

<br />

<label>Price:</label>

<asp:TextBox ID="txtPrice" runat="server"></asp:TextBox>

<br />

<label>Lens Size (mm):</label>

<asp:TextBox ID="txtLensSize" runat="server"></asp:TextBox>

<br />

<asp:Button ID="btnAddCamera" runat="server" Text="Add Camera" OnClick="btnAddCamera\_Click" />

</div>

<!-- Display Camera List -->

<div>

<h3>All Cameras</h3>

<asp:GridView ID="gvCameras" runat="server" AutoGenerateColumns="false">

<Columns>

<asp:BoundField DataField="UniqueId" HeaderText="ID" />

<asp:BoundField DataField="Brand" HeaderText="Brand" />

<asp:BoundField DataField="Model" HeaderText="Model" />

<asp:BoundField DataField="Price" HeaderText="Price" />

<asp:BoundField DataField="LensSize" HeaderText="Lens Size" />

</Columns>

</asp:GridView>

</div>

</form>

</body>

</html>

```

### 2. \*\*Code-behind (C#) Logic for Adding and Viewing Camera Details\*\*

The `.aspx.cs` file contains the logic to handle the camera addition and display all camera information.

#### \*\*Cameras.aspx.cs\*\* (back-end)

```csharp

using System;

using System.Data;

using System.Data.SqlClient;

using System.Configuration;

public partial class Cameras : System.Web.UI.Page

{

// Database connection string

string connectionString = ConfigurationManager.ConnectionStrings["EcommerceDB"].ConnectionString;

protected void Page\_Load(object sender, EventArgs e)

{

if (!IsPostBack)

{

BindGrid(); // Load camera list on initial load

}

}

// Add Camera to the Database

protected void btnAddCamera\_Click(object sender, EventArgs e)

{

using (SqlConnection con = new SqlConnection(connectionString))

{

SqlCommand cmd = new SqlCommand("INSERT INTO Cameras (Brand, Model, Price, LensSize) VALUES (@Brand, @Model, @Price, @LensSize)", con);

cmd.Parameters.AddWithValue("@Brand", txtBrand.Text);

cmd.Parameters.AddWithValue("@Model", txtModel.Text);

cmd.Parameters.AddWithValue("@Price", txtPrice.Text);

cmd.Parameters.AddWithValue("@LensSize", txtLensSize.Text);

con.Open();

cmd.ExecuteNonQuery();

con.Close();

}

// Clear the form fields

txtBrand.Text = "";

txtModel.Text = "";

txtPrice.Text = "";

txtLensSize.Text = "";

// Refresh the grid to show updated data

BindGrid();

}

// Fetch and bind the camera data to GridView

private void BindGrid()

{

using (SqlConnection con = new SqlConnection(connectionString))

{

SqlCommand cmd = new SqlCommand("SELECT UniqueId, Brand, Model, Price, LensSize FROM Cameras", con);

SqlDataAdapter da = new SqlDataAdapter(cmd);

DataTable dt = new DataTable();

da.Fill(dt);

gvCameras.DataSource = dt;

gvCameras.DataBind();

}

}

}

```

### 3. \*\*Database Structure for Camera Information\*\*

Here’s a simple SQL Server table structure to store camera details:

```sql

CREATE TABLE Cameras (

UniqueId INT PRIMARY KEY IDENTITY,

Brand NVARCHAR(100),

Model NVARCHAR(100),

Price DECIMAL(10, 2),

LensSize INT

);

```

### Key Features:

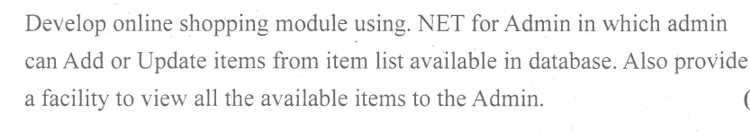
1. \*\*Add Camera\*\*: The form allows the user to add new camera details (brand, model, price, and lens size).

2. \*\*View Cameras\*\*: The `GridView` shows all the stored cameras with their information.

3. \*\*Database Operations\*\*: Uses SQL commands to insert new camera details and retrieve all records from the database.

This example provides a basic interface and functionality for managing camera details in an e-commerce application. You'll need to configure the connection string and ensure that the database is set up accordingly for the system to function correctly.

[4].



For creating an online shopping module where an admin can \*\*Add\*\*, \*\*Update\*\*, and \*\*View items\*\* in the item list from a database, we can implement this using ASP.NET Web Forms. Here's an example of how you can structure the code to achieve this functionality.

### 1. \*\*ASP.NET Web Form for Adding, Updating, and Viewing Items\*\*

#### \*\*ShoppingAdmin.aspx\*\* (front-end)

This page allows the admin to add or update items and view the list of all available items in a `GridView`.

```aspx

<%@ Page Language="C#" AutoEventWireup="true" CodeFile="ShoppingAdmin.aspx.cs" Inherits="ShoppingAdmin" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">

<title>Admin - Manage Items</title>

</head>

<body>

<form id="form1" runat="server">

<h2>Admin Item Management</h2>

<!-- Form to Add or Update Items -->

<div>

<h3>Add or Update Item</h3>

<label>Item Name:</label>

<asp:TextBox ID="txtItemName" runat="server"></asp:TextBox>

<br />

<label>Price:</label>

<asp:TextBox ID="txtPrice" runat="server"></asp:TextBox>

<br />

<label>Quantity:</label>

<asp:TextBox ID="txtQuantity" runat="server"></asp:TextBox>

<br />

<asp:Button ID="btnAddUpdateItem" runat="server" Text="Add/Update Item" OnClick="btnAddUpdateItem\_Click" />

</div>

<!-- Display Item List -->

<div>

<h3>All Items</h3>

<asp:GridView ID="gvItems" runat="server" AutoGenerateColumns="false" OnRowEditing="gvItems\_RowEditing" OnRowCancelingEdit="gvItems\_RowCancelingEdit" OnRowUpdating="gvItems\_RowUpdating">

<Columns>

<asp:BoundField DataField="ItemId" HeaderText="Item ID" ReadOnly="true" />

<asp:BoundField DataField="ItemName" HeaderText="Item Name" />

<asp:BoundField DataField="Price" HeaderText="Price" />

<asp:BoundField DataField="Quantity" HeaderText="Quantity" />

<asp:CommandField ShowEditButton="true" />

</Columns>

</asp:GridView>

</div>

</form>

</body>

</html>

```

### 2. \*\*Code-behind (C#) Logic for Adding, Updating, and Viewing Items\*\*

The `.aspx.cs` file will contain logic to handle adding/updating items and displaying all the items.

#### \*\*ShoppingAdmin.aspx.cs\*\* (back-end)

```csharp

using System;

using System.Data;

using System.Data.SqlClient;

using System.Configuration;

public partial class ShoppingAdmin : System.Web.UI.Page

{

// Database connection string

string connectionString = ConfigurationManager.ConnectionStrings["ShoppingDB"].ConnectionString;

protected void Page\_Load(object sender, EventArgs e)

{

if (!IsPostBack)

{

BindGrid(); // Load the item list on initial load

}

}

// Add or Update Item in the Database

protected void btnAddUpdateItem\_Click(object sender, EventArgs e)

{

using (SqlConnection con = new SqlConnection(connectionString))

{

SqlCommand cmd;

// Check if item exists by searching with its name

cmd = new SqlCommand("SELECT COUNT(\*) FROM Items WHERE ItemName = @ItemName", con);

cmd.Parameters.AddWithValue("@ItemName", txtItemName.Text);

con.Open();

int count = (int)cmd.ExecuteScalar();

con.Close();

if (count == 0) // If no item found, insert new item

{

cmd = new SqlCommand("INSERT INTO Items (ItemName, Price, Quantity) VALUES (@ItemName, @Price, @Quantity)", con);

}

else // If item exists, update its details

{

cmd = new SqlCommand("UPDATE Items SET Price = @Price, Quantity = @Quantity WHERE ItemName = @ItemName", con);

}

// Set the parameters

cmd.Parameters.AddWithValue("@Price", txtPrice.Text);

cmd.Parameters.AddWithValue("@Quantity", txtQuantity.Text);

// Execute the query

con.Open();

cmd.ExecuteNonQuery();

con.Close();

}

// Clear input fields after operation

txtItemName.Text = "";

txtPrice.Text = "";

txtQuantity.Text = "";

// Refresh the grid to show updated data

BindGrid();

}

// Fetch and bind the item data to GridView

private void BindGrid()

{

using (SqlConnection con = new SqlConnection(connectionString))

{

SqlCommand cmd = new SqlCommand("SELECT \* FROM Items", con);

SqlDataAdapter da = new SqlDataAdapter(cmd);

DataTable dt = new DataTable();

da.Fill(dt);

gvItems.DataSource = dt;

gvItems.DataBind();

}

}

// Edit Item

protected void gvItems\_RowEditing(object sender, System.Web.UI.WebControls.GridViewEditEventArgs e)

{

gvItems.EditIndex = e.NewEditIndex;

BindGrid();

}

// Cancel Edit

protected void gvItems\_RowCancelingEdit(object sender, System.Web.UI.WebControls.GridViewCancelEditEventArgs e)

{

gvItems.EditIndex = -1;

BindGrid();

}

// Update Item in GridView

protected void gvItems\_RowUpdating(object sender, System.Web.UI.WebControls.GridViewUpdateEventArgs e)

{

GridViewRow row = gvItems.Rows[e.RowIndex];

int itemId = Convert.ToInt32(gvItems.DataKeys[e.RowIndex].Value);

string itemName = (row.Cells[1].Controls[0] as TextBox).Text;

decimal price = Convert.ToDecimal((row.Cells[2].Controls[0] as TextBox).Text);

int quantity = Convert.ToInt32((row.Cells[3].Controls[0] as TextBox).Text);

using (SqlConnection con = new SqlConnection(connectionString))

{

SqlCommand cmd = new SqlCommand("UPDATE Items SET ItemName=@ItemName, Price=@Price, Quantity=@Quantity WHERE ItemId=@ItemId", con);

cmd.Parameters.AddWithValue("@ItemId", itemId);

cmd.Parameters.AddWithValue("@ItemName", itemName);

cmd.Parameters.AddWithValue("@Price", price);

cmd.Parameters.AddWithValue("@Quantity", quantity);

con.Open();

cmd.ExecuteNonQuery();

con.Close();

}

gvItems.EditIndex = -1; // Exit edit mode

BindGrid(); // Refresh grid with updated data

}

}

```

### 3. \*\*Database Structure for Item Information\*\*

Here’s a basic SQL Server table to store item details:

```sql

CREATE TABLE Items (

ItemId INT PRIMARY KEY IDENTITY,

ItemName NVARCHAR(100),

Price DECIMAL(10, 2),

Quantity INT

);

```

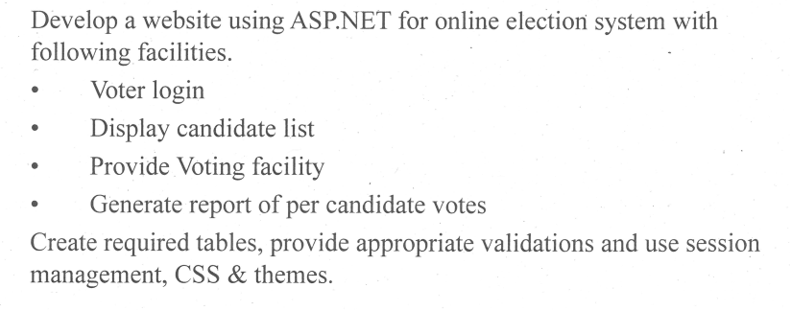
### Key Features:

1. \*\*Add or Update Items\*\*: Admin can enter new items or update existing ones by entering details in the form.

2. \*\*View Items\*\*: All items are displayed in a `GridView`, which includes the ability to edit any item directly from the grid.

3. \*\*Database Operations\*\*: The logic handles inserting new items, updating existing ones, and displaying all records from the database.

This example provides a basic shopping module for admin management of an item list. You’ll need to adjust your configuration settings, such as the connection string, to match your environment for this system to work correctly.



I see you are working on an online election system using ASP.NET. I'll provide you with a basic structure for both the front-end and back-end, as well as database creation and CSS integration.

### Front-End (ASP.NET Web Forms)

Here is a simple ASP.NET page (using `.aspx`):

```aspx

<%@ Page Language="C#" AutoEventWireup="true" CodeFile="ElectionSystem.aspx.cs" Inherits="ElectionSystem" %>

<!DOCTYPE html>

<html lang="en">

<head runat="server">

<meta charset="UTF-8">

<title>Online Election System</title>

<style>

body {

font-family: Arial, sans-serif;

margin: 20px;

}

.container {

width: 600px;

margin: auto;

text-align: center;

}

h1 {

color: #333;

}

table {

width: 100%;

border-collapse: collapse;

}

th, td {

padding: 10px;

border: 1px solid #ddd;

text-align: center;

}

th {

background-color: #f2f2f2;

}

.vote-button {

padding: 10px 20px;

background-color: #4CAF50;

color: white;

border: none;

cursor: pointer;

}

</style>

</head>

<body>

<div class="container">

<h1>Online Election System</h1>

<asp:Label ID="lblMessage" runat="server" Text=""></asp:Label>

<h3>Candidate List</h3>

<asp:GridView ID="gvCandidates" runat="server" AutoGenerateColumns="False">

<Columns>

<asp:BoundField DataField="CandidateName" HeaderText="Candidate Name" />

<asp:TemplateField>

<ItemTemplate>

<asp:Button ID="btnVote" runat="server" Text="Vote" CommandArgument='<%# Eval("CandidateID") %>' OnClick="Vote\_Click" CssClass="vote-button" />

</ItemTemplate>

</asp:TemplateField>

</Columns>

</asp:GridView>

<h3>Results</h3>

<asp:GridView ID="gvResults" runat="server" AutoGenerateColumns="False">

<Columns>

<asp:BoundField DataField="CandidateName" HeaderText="Candidate Name" />

<asp:BoundField DataField="Votes" HeaderText="Votes" />

</Columns>

</asp:GridView>

</div>

</body>

</html>

```

### Back-End (C# Code for Election System)

Here’s the code-behind file (`ElectionSystem.aspx.cs`):

```csharp

using System;

using System.Data;

using System.Data.SqlClient;

using System.Configuration;

public partial class ElectionSystem : System.Web.UI.Page

{

private string connStr = ConfigurationManager.ConnectionStrings["ElectionDB"].ConnectionString;

protected void Page\_Load(object sender, EventArgs e)

{

if (!IsPostBack)

{

LoadCandidates();

LoadResults();

}

}

protected void LoadCandidates()

{

using (SqlConnection conn = new SqlConnection(connStr))

{

SqlCommand cmd = new SqlCommand("SELECT CandidateID, CandidateName FROM Candidates", conn);

SqlDataAdapter da = new SqlDataAdapter(cmd);

DataTable dt = new DataTable();

da.Fill(dt);

gvCandidates.DataSource = dt;

gvCandidates.DataBind();

}

}

protected void LoadResults()

{

using (SqlConnection conn = new SqlConnection(connStr))

{

SqlCommand cmd = new SqlCommand("SELECT CandidateName, Votes FROM Candidates", conn);

SqlDataAdapter da = new SqlDataAdapter(cmd);

DataTable dt = new DataTable();

da.Fill(dt);

gvResults.DataSource = dt;

gvResults.DataBind();

}

}

protected void Vote\_Click(object sender, EventArgs e)

{

Button btn = (Button)sender;

int candidateID = Convert.ToInt32(btn.CommandArgument);

using (SqlConnection conn = new SqlConnection(connStr))

{

SqlCommand cmd = new SqlCommand("UPDATE Candidates SET Votes = Votes + 1 WHERE CandidateID = @CandidateID", conn);

cmd.Parameters.AddWithValue("@CandidateID", candidateID);

conn.Open();

cmd.ExecuteNonQuery();

}

lblMessage.Text = "Vote cast successfully!";

LoadResults();

}

}

```

### Database Creation

You can create your database and table using the following SQL script:

```sql

CREATE DATABASE ElectionDB;

GO

USE ElectionDB;

CREATE TABLE Candidates (

CandidateID INT PRIMARY KEY IDENTITY(1,1),

CandidateName NVARCHAR(100) NOT NULL,

Votes INT DEFAULT 0

);

INSERT INTO Candidates (CandidateName) VALUES ('John Doe'), ('Jane Smith'), ('Alex Johnson');

```

### Web.config Connection String

Ensure that you have the correct connection string set in your `Web.config`:

```xml

<configuration>

<connectionStrings>

<add name="ElectionDB" connectionString="Data Source=your\_server\_name;Initial Catalog=ElectionDB;Integrated Security=True" providerName="System.Data.SqlClient" />

</connectionStrings>

</configuration>

```

### CSS Styling

For a clean, simple design, I added CSS within the `<style>` tag of the front-end code. If you want external CSS, you can move the styles to a separate file and reference it in the `<head>` section:

```html

<link rel="stylesheet" type="text/css" href="styles.css" />

```

This should provide you with a basic setup for the online election system with voter login, candidate listing, voting, and vote count display.

Let me know if you need additional customization or clarification!