**design asp.net web apllication to generate hotel bill.accept custer id,customer name,address,menu items,quality and price per item.Apply 5% GST on total.Display bill on another web form .use appropriate validation controls**

**WebForm1.aspx**

<%@ Page Language="C#" AutoEventWireup="true" CodeFile="WebForm1.aspx.cs" Inherits="WebForm1" %>

<!DOCTYPE html>

<html>

<head runat="server">

<title>Hotel Bill Generator</title>

</head>

<body>

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

<div style="width: 400px; margin: auto;">

<h2>Hotel Bill Input</h2>

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

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

<asp:RequiredFieldValidator ControlToValidate="txtCustomerId" ErrorMessage="\* Required" runat="server" ForeColor="Red" /><br />

<asp:Label Text="Customer Name:" runat="server" />

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

<asp:RequiredFieldValidator ControlToValidate="txtName" ErrorMessage="\* Required" runat="server" ForeColor="Red" /><br />

<asp:Label Text="Address:" runat="server" />

<asp:TextBox ID="txtAddress" runat="server" TextMode="MultiLine" Rows="3" /><br />

<asp:Label Text="Menu Item:" runat="server" />

<asp:DropDownList ID="ddlMenu" runat="server">

<asp:ListItem Text="-- Select Item --" Value="" />

<asp:ListItem Text="Pizza" Value="Pizza" />

<asp:ListItem Text="Burger" Value="Burger" />

<asp:ListItem Text="Pasta" Value="Pasta" />

</asp:DropDownList>

<asp:RequiredFieldValidator ControlToValidate="ddlMenu" InitialValue="" ErrorMessage="\* Required" runat="server" ForeColor="Red" /><br />

<asp:Label Text="Quantity:" runat="server" />

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

<asp:RequiredFieldValidator ControlToValidate="txtQuantity" ErrorMessage="\* Required" runat="server" ForeColor="Red" />

<asp:RangeValidator ControlToValidate="txtQuantity" MinimumValue="1" MaximumValue="100" Type="Integer" ErrorMessage="\* Enter valid quantity" runat="server" ForeColor="Red" /><br />

<asp:Label Text="Price per item:" runat="server" />

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

<asp:RequiredFieldValidator ControlToValidate="txtPrice" ErrorMessage="\* Required" runat="server" ForeColor="Red" />

<asp:RangeValidator ControlToValidate="txtPrice" MinimumValue="1" MaximumValue="10000" Type="Double" ErrorMessage="\* Enter valid price" runat="server" ForeColor="Red" /><br /><br />

<asp:Button ID="btnGenerate" runat="server" Text="Generate Bill" OnClick="btnGenerate\_Click" />

</div>

</form>

</body>

</html>

**WebForm1.aspx.cs**

using System;

public partial class WebForm1 : System.Web.UI.Page

{

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

{

// Store data in Session

Session["CustomerId"] = txtCustomerId.Text;

Session["Name"] = txtName.Text;

Session["Address"] = txtAddress.Text;

Session["Menu"] = ddlMenu.SelectedItem.Text;

Session["Quantity"] = int.Parse(txtQuantity.Text);

Session["Price"] = double.Parse(txtPrice.Text);

// Redirect to output page

Response.Redirect("WebForm2.aspx");

}}

**WebForm2.aspx**

<%@ Page Language="C#" AutoEventWireup="true" CodeFile="WebForm2.aspx.cs" Inherits="WebForm2" %>

<!DOCTYPE html>

<html>

<head runat="server">

<title>Bill</title>

</head>

<body>

<form id="form2" runat="server">

<div style="width: 500px; margin: auto;">

<h2>Hotel Bill</h2>

<asp:Label ID="lblOutput" runat="server" Font-Names="Consolas" Font-Size="Large"></asp:Label>

</div>

</form>

</body>

</html>

**WebForm2.aspx.cs**

using System;

public partial class WebForm2 : System.Web.UI.Page

{

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

{

string id = Session["CustomerId"].ToString();

string name = Session["Name"].ToString();

string address = Session["Address"].ToString();

string menu = Session["Menu"].ToString();

int quantity = (int)Session["Quantity"];

double price = (double)Session["Price"];

double total = quantity \* price;

double gst = total \* 0.05;

double grandTotal = total + gst;

lblOutput.Text = $@"

Customer ID : {id}<br/>

Name : {name}<br/>

Address : {address}<br/><br/>

Menu Item : {menu}<br/>

Quantity : {quantity}<br/>

Price per item: ${price:F2}<br/>

---------------------------<br/>

Total : ${total:F2}<br/>

GST (5%) : ${gst:F2}<br/>

Grand Total : ${grandTotal:F2}

";

}

**display asp.net web application to perform update and delete operations on subject details table using connected architecture**

**WebForm1.aspx**

<%@ Page Language="C#" AutoEventWireup="true" CodeFile="WebForm1.aspx.cs" Inherits="WebForm1" %>

<!DOCTYPE html>

<html>

<head runat="server">

<title>Subject Details</title>

</head>

<body>

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

<h2>Subject Details (Update/Delete)</h2>

<asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="False" DataKeyNames="SubjectID"

OnRowEditing="GridView1\_RowEditing"

OnRowCancelingEdit="GridView1\_RowCancelingEdit"

OnRowUpdating="GridView1\_RowUpdating"

OnRowDeleting="GridView1\_RowDeleting">

<Columns>

<asp:BoundField DataField="SubjectID" HeaderText="Subject ID" ReadOnly="True" />

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

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

<asp:CommandField ShowEditButton="True" ShowDeleteButton="True" />

</Columns>

</asp:GridView>

</form>

</body>

</html>

**WebForm1.aspx.cs**

using System;

using System.Data;

using System.Data.SqlClient;

using System.Configuration;

public partial class WebForm1 : System.Web.UI.Page

{

string connStr = ConfigurationManager.ConnectionStrings["conn"].ConnectionString;

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

{

if (!IsPostBack)

{

BindGrid();

}

}

private void BindGrid()

{

using (SqlConnection con = new SqlConnection(connStr))

{

string query = "SELECT \* FROM SubjectDetails";

SqlDataAdapter da = new SqlDataAdapter(query, con);

DataTable dt = new DataTable();

da.Fill(dt);

GridView1.DataSource = dt;

GridView1.DataBind();

}

}

protected void GridView1\_RowEditing(object sender, System.Web.UI.WebControls.GridViewEditEventArgs e)

{

GridView1.EditIndex = e.NewEditIndex;

BindGrid();

}

protected void GridView1\_RowCancelingEdit(object sender, System.Web.UI.WebControls.GridViewCancelEditEventArgs e)

{

GridView1.EditIndex = -1;

BindGrid();

}

protected void GridView1\_RowUpdating(object sender, System.Web.UI.WebControls.GridViewUpdateEventArgs e)

{

int subjectId = Convert.ToInt32(GridView1.DataKeys[e.RowIndex].Value);

string subjectName = ((System.Web.UI.WebControls.TextBox)GridView1.Rows[e.RowIndex].Cells[1].Controls[0]).Text;

int credits = Convert.ToInt32(((System.Web.UI.WebControls.TextBox)GridView1.Rows[e.RowIndex].Cells[2].Controls[0]).Text);

using (SqlConnection con = new SqlConnection(connStr))

{

string query = "UPDATE SubjectDetails SET SubjectName=@name, Credits=@credits WHERE SubjectID=@id";

SqlCommand cmd = new SqlCommand(query, con);

cmd.Parameters.AddWithValue("@id", subjectId);

cmd.Parameters.AddWithValue("@name", subjectName);

cmd.Parameters.AddWithValue("@credits", credits);

con.Open();

cmd.ExecuteNonQuery();

con.Close();

}

GridView1.EditIndex = -1;

BindGrid();

}

protected void GridView1\_RowDeleting(object sender, System.Web.UI.WebControls.GridViewDeleteEventArgs e)

{

int subjectId = Convert.ToInt32(GridView1.DataKeys[e.RowIndex].Value);

using (SqlConnection con = new SqlConnection(connStr))

{

string query = "DELETE FROM SubjectDetails WHERE SubjectID=@id";

SqlCommand cmd = new SqlCommand(query, con);

cmd.Parameters.AddWithValue("@id", subjectId);

con.Open();

cmd.ExecuteNonQuery();

con.Close();

}

BindGrid();

}

}

**Create the SubjectDetails Table**

CREATE TABLE SubjectDetails (

SubjectID INT PRIMARY KEY,

SubjectName NVARCHAR(100),

Credits INT

);

INSERT INTO SubjectDetails VALUES (101, 'Mathematics', 4);

INSERT INTO SubjectDetails VALUES (102, 'Physics', 3);

INSERT INTO SubjectDetails VALUES (103, 'Chemistry', 4);

**Web.config**

<configuration>

<connectionStrings>

<add name="conn" connectionString="Data Source=YOUR\_SERVER\_NAME;Initial Catalog=YOUR\_DB\_NAME;Integrated Security=True" providerName="System.Data.SqlClient"/>

</connectionStrings>

</configuration>

design WCF web service to test whether the given number is prime number or not and consume it with client application

 Open **Visual Studio**

 Create a new project: **WCF Service Library**

 Name it: PrimeNumberService

**IService1.cs**

csharp

Copy code

using System.ServiceModel;

[ServiceContract]

public interface IPrimeService

{

[OperationContract]

bool IsPrime(int number);

}

**Service1.cs**

csharp

Copy code

public class PrimeService : IPrimeService

{

public bool IsPrime(int number)

{

if (number <= 1) return false;

if (number == 2) return true;

for (int i = 2; i <= Math.Sqrt(number); i++)

{

if (number % i == 0) return false;

}

return true;

}

}

**App.config**

xml

Copy code

<configuration>

<system.serviceModel>

<services>

<service name="PrimeService">

<endpoint address="" binding="wsHttpBinding" contract="IPrimeService"/>

<host>

<baseAddresses>

<add baseAddress="http://localhost:8733/PrimeService/"/>

</baseAddresses>

</host>

</service>

</services>

<behaviors>

<serviceBehaviors>

<behavior>

<serviceMetadata httpGetEnabled="True"/>

<serviceDebug includeExceptionDetailInFaults="True"/>

</behavior>

</serviceBehaviors>

</behaviors>

</system.serviceModel>

</configuration>

Create a **WCF Service Application** or **self-host in Console App**. Example for **self-host**:

csharp

Copy code

using System;

using System.ServiceModel;

class Program

{

static void Main()

{

using (ServiceHost host = new ServiceHost(typeof(PrimeService)))

{

host.Open();

Console.WriteLine("Prime Service is running...");

Console.ReadLine();

}

}

}

 Create a new **Console Application** (e.g., PrimeClient)

 Right-click the project > **Add Service Reference**

 Enter address: http://localhost:8733/PrimeService/

 Name the namespace: PrimeReference

**Client Code (Program.cs)**

csharp

Copy code

using System;

class Program

{

static void Main()

{

var client = new PrimeReference.PrimeServiceClient();

Console.Write("Enter a number: ");

int number = int.Parse(Console.ReadLine());

bool isPrime = client.IsPrime(number);

Console.WriteLine($"{number} is {(isPrime ? "a Prime Number" : "not a Prime Number")}");

client.Close();

}

}

**Design asp.net application to display random advertisements using ADRotator control**

**Ads.xml**

xml

Copy code

<Advertisements>

<Ad>

<ImageUrl>~/Images/ad1.jpg</ImageUrl>

<NavigateUrl>https://www.example.com/ad1</NavigateUrl>

<AlternateText>Visit Ad 1</AlternateText>

<Impressions>50</Impressions>

</Ad>

<Ad>

<ImageUrl>~/Images/ad2.jpg</ImageUrl>

<NavigateUrl>https://www.example.com/ad2</NavigateUrl>

<AlternateText>Visit Ad 2</AlternateText>

<Impressions>30</Impressions>

</Ad>

<Ad>

<ImageUrl>~/Images/ad3.jpg</ImageUrl>

<NavigateUrl>https://www.example.com/ad3</NavigateUrl>

<AlternateText>Visit Ad 3</AlternateText>

<Impressions>20</Impressions>

</Ad>

</Advertisements>

* **Create a folder: Images**
* **Add images: ad1.jpg, ad2.jpg, ad3.jpg (or any image files)**

**Default.aspx**

aspx

Copy code

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

<!DOCTYPE html>

<html>

<head runat="server">

<title>Ad Rotator Example</title>

</head>

<body>

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

<div style="text-align:center;">

<h2>Random Advertisement</h2>

<asp:AdRotator

ID="AdRotator1"

runat="server"

AdvertisementFile="~/Ads.xml"

Height="250px"

Width="300px" />

</div>

</form>

</body>

</html>

 **Use a Timer and UpdatePanel from ScriptManager (ASP.NET AJAX)**

** Put AdRotator inside the UpdatePanel**