**PART I- CONSOLE APPLICATION**

**Console Application:**

A Console Application, in the context of C#, is an application that takes inputs and displays output at a command-line console with access to three basic streams: standard input, standard output, and standard error**.**

**1)Aim :** Write a program to create a class department which has department information and also create an employee class which inherits the department and displays all the information**.**

**Objective:** To perform inheritance.

**Theory:**

Inheritance enables you to create new classes that reuse, extend, and modify the behavior defined in other classes. The class whose members are inherited is called the base class, and the class that inherits those members is called the derived class. A derived class can have only one direct base class. However, inheritance is transitive.If ClassC is derived from ClassB, and ClassB is derived from ClassA, ClassC inherits the members declared in ClassB and ClassA.

**Source code:-**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp1

{

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Console\_App1

{

public abstract class Department

{

public virtual void Depart(string s)

{

}

}

public class Employee : Department

{

int id;

string name;

string depart;

public void insertData(int i, string n)

{

id = i;

name = n;

Console.WriteLine("Employee ID is: " + id);

Console.WriteLine("Employee name is: " + name);

}

override

public void Depart(string d)

{

depart = d;

Console.WriteLine("Department is: " + depart);

}

}

class App

{

static void Main(string[] args)

{

Employee emp = new Employee();

emp.insertData(1, "Sairaj Yende");

emp.Depart("Web Development");

Console.WriteLine("\n");

emp.insertData(2, "Kaushal Satam");

emp.Depart("Android Development");

Console.WriteLine("\n");

emp.insertData(3, "Pratik Mestry");

emp.Depart("Business Analytics");

Console.WriteLine("\n");

Console.ReadKey();

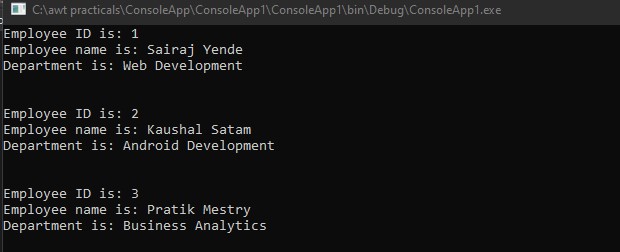
}

}

}

}

**Output:-**



2**) Aim** : Write a program to create an abstract class named employee which has an abstract method salary and having 2 subclasses developer and non-developer. Calculate salary of both (developer and non-developer.)

**Objective:** To demonstrate Abstract Class and Abstract Method

**Theory**:

**Abstract Class:**

If a class is defined as abstract then we can&#39;t create an instance of that class. By thecreation of the derived class object where an abstract class is inherit from, we can call the method of the abstract class.

**Abstract Method:**

An Abstract method is a method without a body. The implementation of an abstract method is done by a derived class. When the derived class inherits the abstract method from the abstract class, it must override the abstract method. This requirment is enforced at compile time and is also called dynamic polymorphism.

**The syntax of using the abstract method is as follows**:

&lt;access-modifier&gt;abstract&lt;return-type&gt;method name (parameter)

**source code:-**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Xml.Schema;

namespace Ass1Q1ka2

{

public abstract class Employee

{

public virtual void salary()

{

}

}

public class Developer : Employee

{

int basicsal;

int ca;

int sum;

public void basicsaldev(int b)

{

basicsal = b;

Console.WriteLine("The basic salary of developer is:-" + basicsal);

}

override

public void salary()

{

Console.WriteLine(" company allowance is:- 3000");

}

public void totalsal(int t,int b)

{

ca = t;

basicsal = b;

sum = ca + basicsal;

Console.WriteLine(sum);

}

}

public class Nondeveloper : Employee

{

int basicsal1;

int ca;

int sum;

public void basicsalnondev(int c)

{

basicsal1 = c;

Console.WriteLine("The basic salary of Nondeveloper is:-" + basicsal1);

}

override

public void salary()

{

Console.WriteLine(" company allowance is :-3000");

}

public void totalsal1(int t, int d)

{

ca = t;

basicsal1 = d;

sum = ca + basicsal1;

Console.WriteLine(sum);

}

}

class program

{

static void Main(string[] args)

{

Developer dev = new Developer();

Console.WriteLine("total salary of developer is:");

dev.basicsaldev(47000);

dev.salary();

Console.WriteLine("The total salary of developer is:-");

dev.totalsal(3000, 47000);

Nondeveloper dev2 = new Nondeveloper();

Console.WriteLine("total salary of Nondeveloper is:");

dev2.basicsalnondev(27000);

dev2.salary();

dev2.totalsal1(3000, 27000);

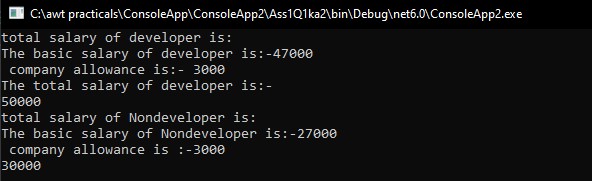
Console.ReadKey();

}

}

}

**Output:-**



3) **Aim :** Write a program for interface.

**Objective**: To demonstrate Interface

**Theory:**

**Interface**:

In C#, an interface can be defined using the interface keyword. An interface can contain declarations of methods, properties, indexers, and events. However, it cannot contain fields, auto-implemented properties.

**Source code:-**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ASS1part1q3

{

interface ILeague

{

void trophywin(int a)

{

}

}

class Ipl : ILeague

{

int fplay;

public void finalplayed(int w)

{

fplay = w;

Console.WriteLine("The finals played by this team is:-" +fplay);

}

public void trophywin(int a)

{

int winn=a;

Console.WriteLine("The trophy win by this team is:-" + winn);

}

}

class Program

{

static void Main(string[] args)

{

Ipl t = new Ipl();

Console.WriteLine(" IPlteams trophy stat is:-");

Console.WriteLine(" MI Ipl team stat is:-");

t.finalplayed(6);

t.trophywin(5);

Console.WriteLine(" Csk IPl team trophy stat is:-");

t.finalplayed(9);

t.trophywin(4);

Console.WriteLine(" Rcb team trophy stat is:-");

t.finalplayed(3);

t.trophywin(0);

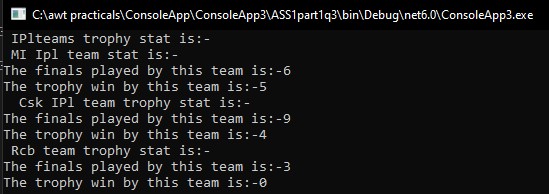
Console.ReadKey();

}

}

}

**Output:-**



**PART II- WINDOWS APPLICATION**

##### WINDOWS APPLICATION

Windows Forms is a UI framework for building Windows desktop apps. It provides one of the most productive ways to create desktop apps based on the visual designer provided in Visual Studio. Functionality such as drag-and-drop placement of visual controls makes it easy to build desktop apps.

**1)Aim :** Write a program to design a form that contains employee information (Id, Name, department, city). By clicking view data is displayed in the list box. Create an employee class which has different properties**.**

**Objective:** To demonstrate windows application by creating form and displaying its data in the listbox.

**Theory:**

A Windows forms application is one that runs on the desktop computer. A Windows forms application will normally have a collection of controls such as labels, textboxes, list boxes, etc.

2)**Aim** : Write a program to design and perform following operations.

* 1. Create multiple(at least 2 form) inherited form
  2. First form for general information of MCA course.
  3. Second form for information of first year.

**Objective:** To demonstrate form Inheritance

**Theory**:

In Asp.Net web form can also be inherited which means property of the parent form can be transferred to the child form.

**PART III- WEB APPLICATION**

ASP.NET is an open-source, server-side web-application framework designed for web development to produce dynamic web pages. It was developed by Microsoft to allow programmers to build dynamic web sites, applications and services. NET Framework and is the successor to Microsoft's Active Server Pages (ASP) technology.

**1)Aim :** Write a program to demonstrate navigation controls in ASP.NET.

**Objective:** To demonstrate navigation control in ASP.NET

**Theory:**

Basically ASP.NET 2.0 has three navigation controls:

* 1. Dynamic menus
  2. Tree Views
  3. Site Map Path

1. Dynamic menus

It was the very difficult task to maintain the menu of a large website and time consuming. It is used to display the Menus. You can use it as easy as other Navigation controls. Menu can be stored in a file to make it easier to maintain. This file is normally called web. Sitemap, and is stored in the root directory of the web.

1. Tree Views

A Tree View control displays a hierarchical list of items using lines to connect related items in a hierarchy. Each item consists of a label and an optional bitmap. Windows Explorer uses a Tree View control to display directories. You can use the Tree View control in any situation in which you need to display hierarchical data.

1. Site Map Path

Use of this control is very simple. You can add this control to your page then view your page in browser. The Sitemap Path control displays the navigation path of the current page. The path acts as click able links to previous pages.

The Sitemap Path control uses the web. Sitemap file by default.

**Source Code:**

Site.Master

<%@ Master Language="C#" AutoEventWireup="true" CodeBehind="Site.master.cs" Inherits="Web\_Application\_I.SiteMaster" %>

<!DOCTYPE html>

<html lang="en">

<head runat="server">

<meta charset="utf-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<title><%: Page.Title %> - My ASP.NET Application</title>

<asp:PlaceHolder runat="server">

<%: Scripts.Render("~/bundles/modernizr") %>

</asp:PlaceHolder>

<webopt:bundlereference runat="server" path="~/Content/css" />

<link href="~/favicon.ico" rel="shortcut icon" type="image/x-icon" />

</head>

<body>

<form runat="server">

<asp:ScriptManager runat="server">

<Scripts>

<%--To learn more about bundling scripts in ScriptManager see https://go.microsoft.com/fwlink/?LinkID=301884 --%>

<%--Framework Scripts--%>

<asp:ScriptReference Name="MsAjaxBundle" />

<asp:ScriptReference Name="jquery" />

<asp:ScriptReference Name="WebForms.js" Assembly="System.Web" Path="~/Scripts/WebForms/WebForms.js" />

<asp:ScriptReference Name="WebUIValidation.js" Assembly="System.Web" Path="~/Scripts/WebForms/WebUIValidation.js" />

<asp:ScriptReference Name="MenuStandards.js" Assembly="System.Web" Path="~/Scripts/WebForms/MenuStandards.js" />

<asp:ScriptReference Name="GridView.js" Assembly="System.Web" Path="~/Scripts/WebForms/GridView.js" />

<asp:ScriptReference Name="DetailsView.js" Assembly="System.Web" Path="~/Scripts/WebForms/DetailsView.js" />

<asp:ScriptReference Name="TreeView.js" Assembly="System.Web" Path="~/Scripts/WebForms/TreeView.js" />

<asp:ScriptReference Name="WebParts.js" Assembly="System.Web" Path="~/Scripts/WebForms/WebParts.js" />

<asp:ScriptReference Name="Focus.js" Assembly="System.Web" Path="~/Scripts/WebForms/Focus.js" />

<asp:ScriptReference Name="WebFormsBundle" />

<%--Site Scripts--%>

</Scripts>

</asp:ScriptManager>

<nav class="navbar navbar-expand-sm navbar-toggleable-sm navbar-dark bg-dark">

<div class="container">

<a class="navbar-brand" runat="server" href="~/">Zoro.to</a>

<button type="button" class="navbar-toggler" data-bs-toggle="collapse" data-bs-target=".navbar-collapse" title="Toggle navigation" aria-controls="navbarSupportedContent"

aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<div class="collapse navbar-collapse d-sm-inline-flex justify-content-between">

<ul class="navbar-nav flex-grow-1">

<li class="nav-item"><a class="nav-link" runat="server" href="~/Home">Home</a></li>

<li class="nav-item"><a class="nav-link" runat="server" href="~/About">About</a></li>

<li class="nav-item"><a class="nav-link" runat="server" href="~/Contact">Contact</a></li>

</ul>

</div>

</div>

</nav>

<div class="container body-content">

<asp:ContentPlaceHolder ID="MainContent" runat="server">

</asp:ContentPlaceHolder>

<hr />

<footer>

<p>&copy; <%: DateTime.Now.Year %> - My ASP.NET Application</p>

</footer>

</div>

</form>

<asp:PlaceHolder runat="server">

<%: Scripts.Render("~/Scripts/bootstrap.js") %>

</asp:PlaceHolder>

</body>

</html>

Default.aspx

<%@ Page Title="Home Page" Language="C#" MasterPageFile="~/Site.Master" AutoEventWireup="true" CodeBehind="Default.aspx.cs" Inherits="Web\_Application\_I.\_Default" %>

<asp:Content ID="BodyContent" ContentPlaceHolderID="MainContent" runat="server">

<main>

<section class="row" aria-labelledby="aspnetTitle">

<h1 id="aspnetTitle">Anime Streaming Platform</h1>

<p class="lead">You will find n numbers of anime here, and even japanese animated movies too.</p>

<p><a href="http://www.asp.net" class="btn btn-primary btn-md">Watch more &raquo;</a></p>

</section>

</main>

</asp:Content>

Home.aspx

<%@ Page Title="" Language="C#" MasterPageFile="~/Site.Master" AutoEventWireup="true" CodeBehind="Home.aspx.cs" Inherits="Web\_Application\_I.Home" %>

<asp:Content ID="Content1" ContentPlaceHolderID="MainContent" runat="server">

<div>

<h2>Let's watch anime together and learn some jutsu from it.</h2>

</div>

</asp:Content>

About.aspx

<%@ Page Title="About" Language="C#" MasterPageFile="~/Site.Master" AutoEventWireup="true" CodeBehind="About.aspx.cs" Inherits="Web\_Application\_I.About" %>

<asp:Content ID="BodyContent" ContentPlaceHolderID="MainContent" runat="server">

<main aria-labelledby="title">

<!-- <h2 id="title"><%: Title %>.</h2> -->

<h3>Its Zoro.to website free anime streaming platform</h3>

</main>

</asp:Content>

Contact.aspx

<%@ Page Title="Contact" Language="C#" MasterPageFile="~/Site.Master" AutoEventWireup="true" CodeBehind="Contact.aspx.cs" Inherits="Web\_Application\_I.Contact" %>

<asp:Content ID="BodyContent" ContentPlaceHolderID="MainContent" runat="server">

<main aria-labelledby="title">

<h2 id="title"><%: Title %>.</h2>

<h3>Your contact page.</h3>

<address>

One Microsoft Way<br />

Redmond, WA 98052-6399<br />

<abbr title="Phone">P:</abbr>

425.555.0100

</address>

<address>

<strong>Support:</strong> <a href="mailto:Support@example.com">Support@example.com</a><br />

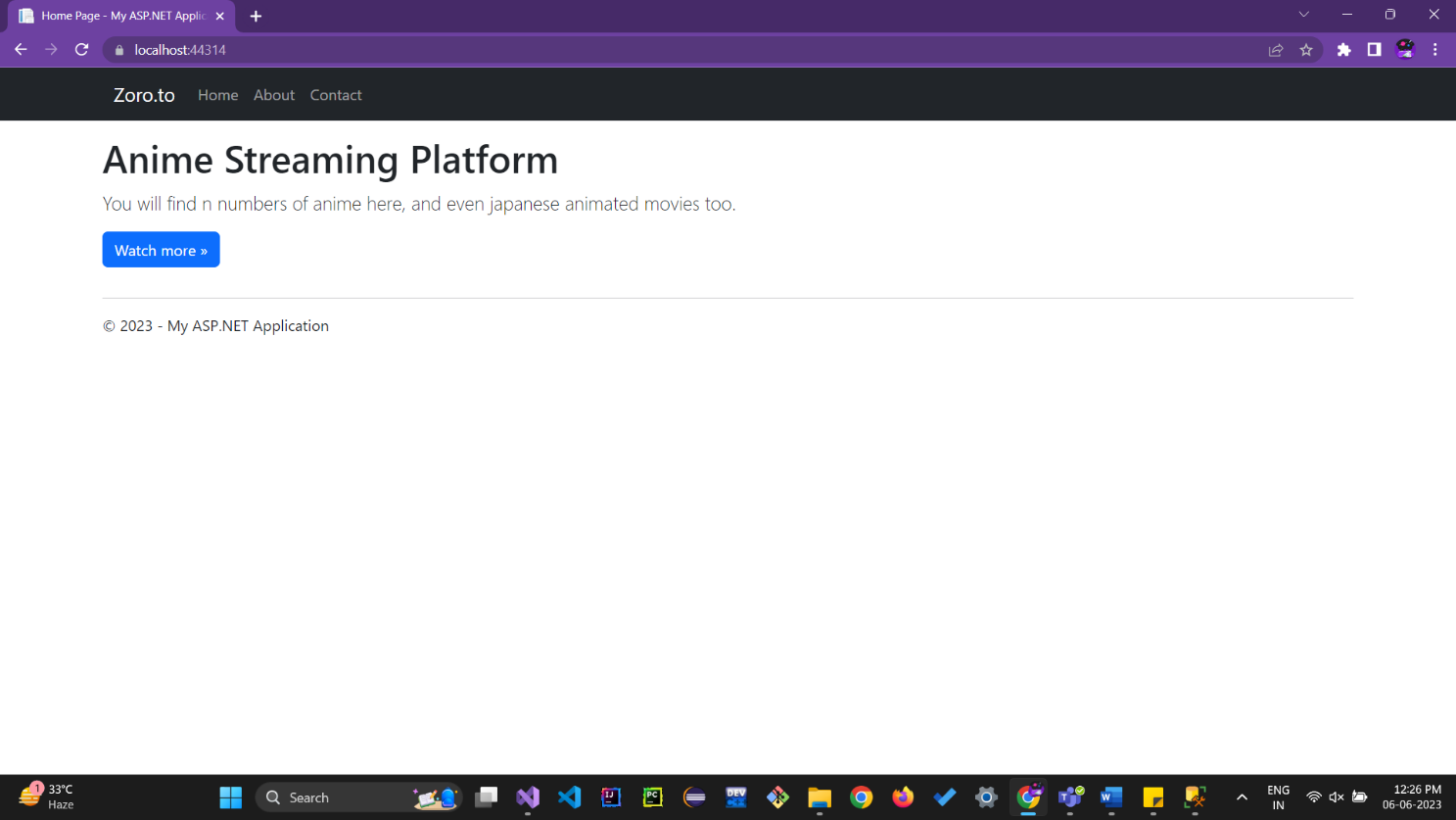
<strong>Marketing:</strong> <a href="mailto:Marketing@example.com">Marketing@example.com</a>

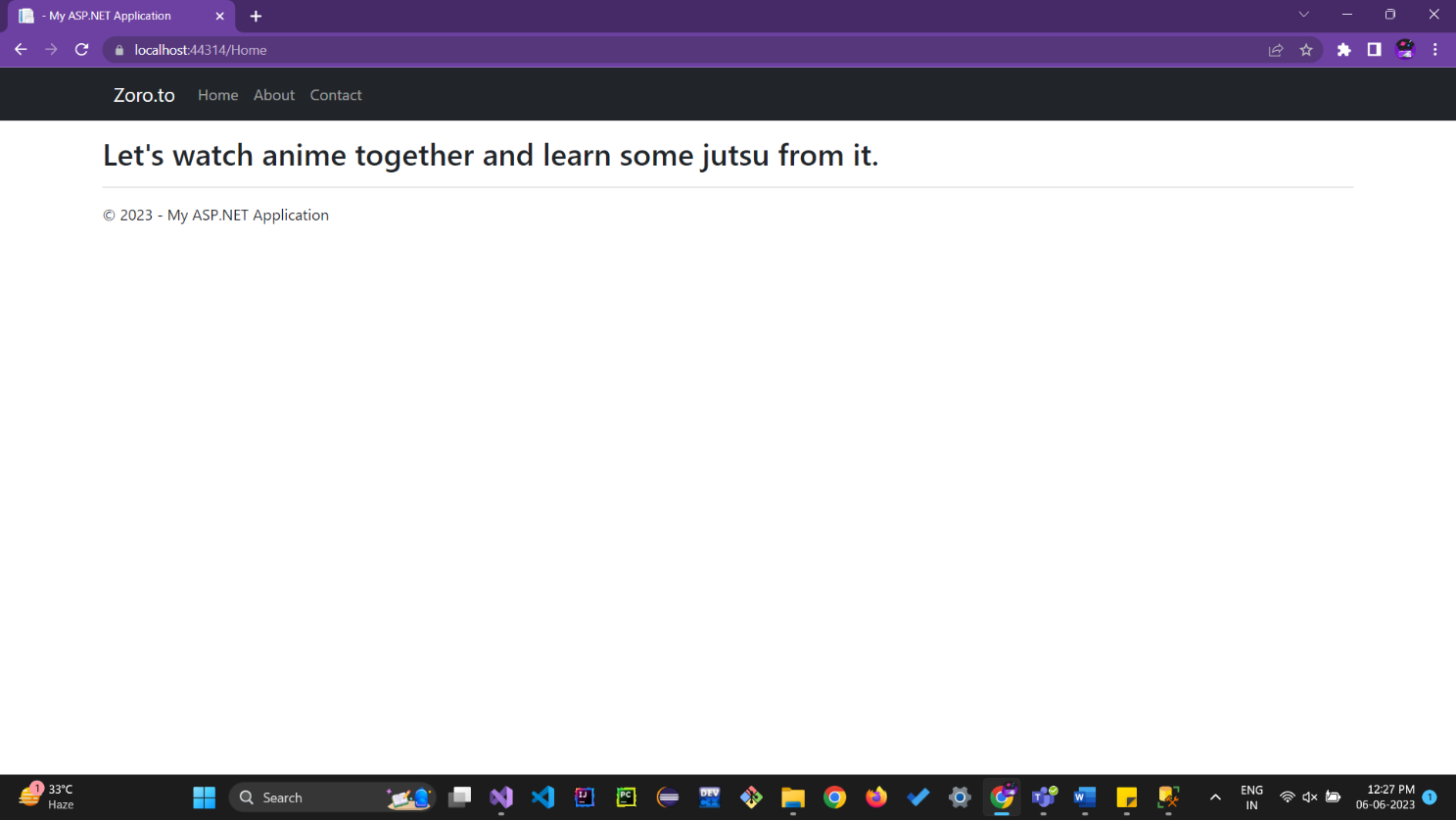
</address>

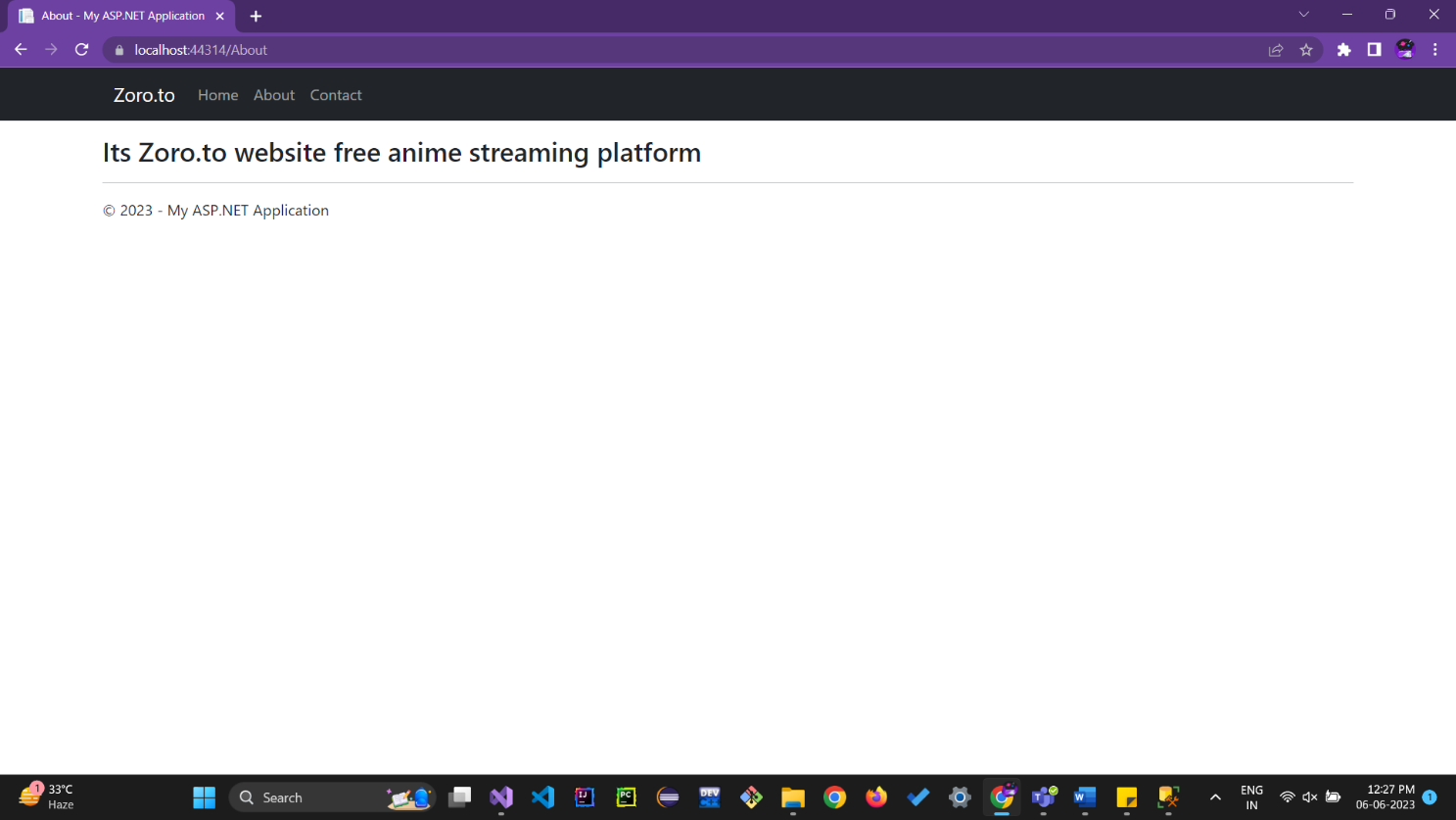
</main>

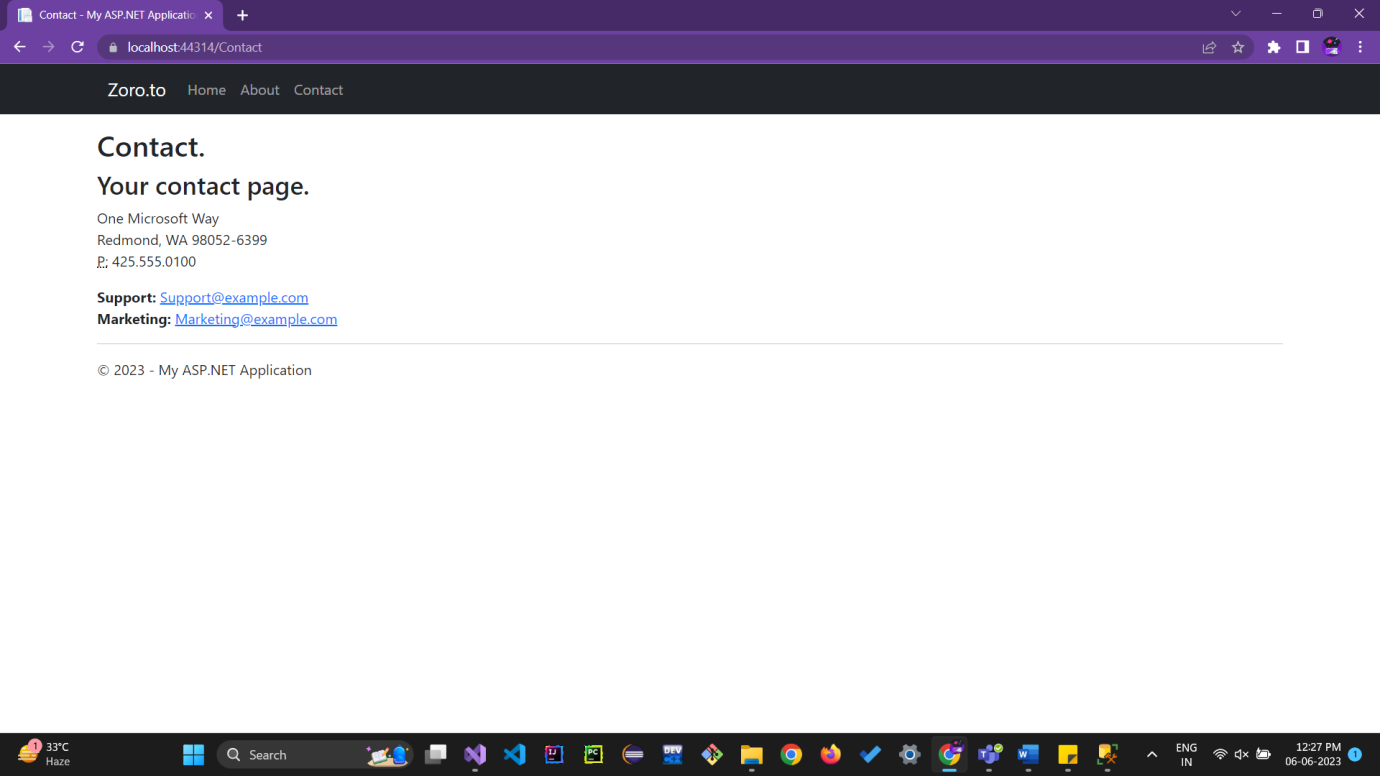
</asp:Content>

**Output:-**









2) **Aim:**Create a website using the master page concept and content pages also use navigation control on that.

**Objective:** To demonstrate Master Pages Concept.

**Theory:**

**MasterPage:**

ASP.NET master pages allow you to create a consistent layout for the pages in your application. A single master page defines the look and feel and standard behavior that you want for all of the pages (or a group of pages) in your application.

**Nested MasterPage:**

In a nested Master page, the child master pages define the attribute MasterPageFile set in the @ Master declaration. This attribute points to the parent master page.

**Source code:-**

navigationControl.aspx

<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="navigationControl.aspx.cs" Inherits="Web\_Application\_I.navigationControl" %>

<!DOCTYPE html>

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

<head runat="server">

<title></title>

</head>

<body>

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

<div>

<h1>Navigation Controls Application</h1>

<asp:Menu ID="menu1" runat="server" CssClass="menu">

<Items>

<asp:MenuItem Text="Home" NavigateUrl="~/Default.aspx" />

<asp:MenuItem Text="About" NavigateUrl="~/About.aspx" />

<asp:MenuItem Text="Contact" NavigateUrl="~/Contact.aspx" />

</Items>

</asp:Menu>

<asp:MultiView ID="multiView1" runat="server">

<asp:View ID="viewHome" runat="server">

<h2>Welcome to the Home Page!</h2>

<p>This is the home page content.</p>

</asp:View>

<asp:View ID="viewAbout" runat="server">

<h2>About Us</h2>

<p>About page content goes here.</p>

</asp:View>

<asp:View ID="viewContact" runat="server">

<h2>Contact Us</h2>

<p>Contact information goes here.</p>

</asp:View>

</asp:MultiView>

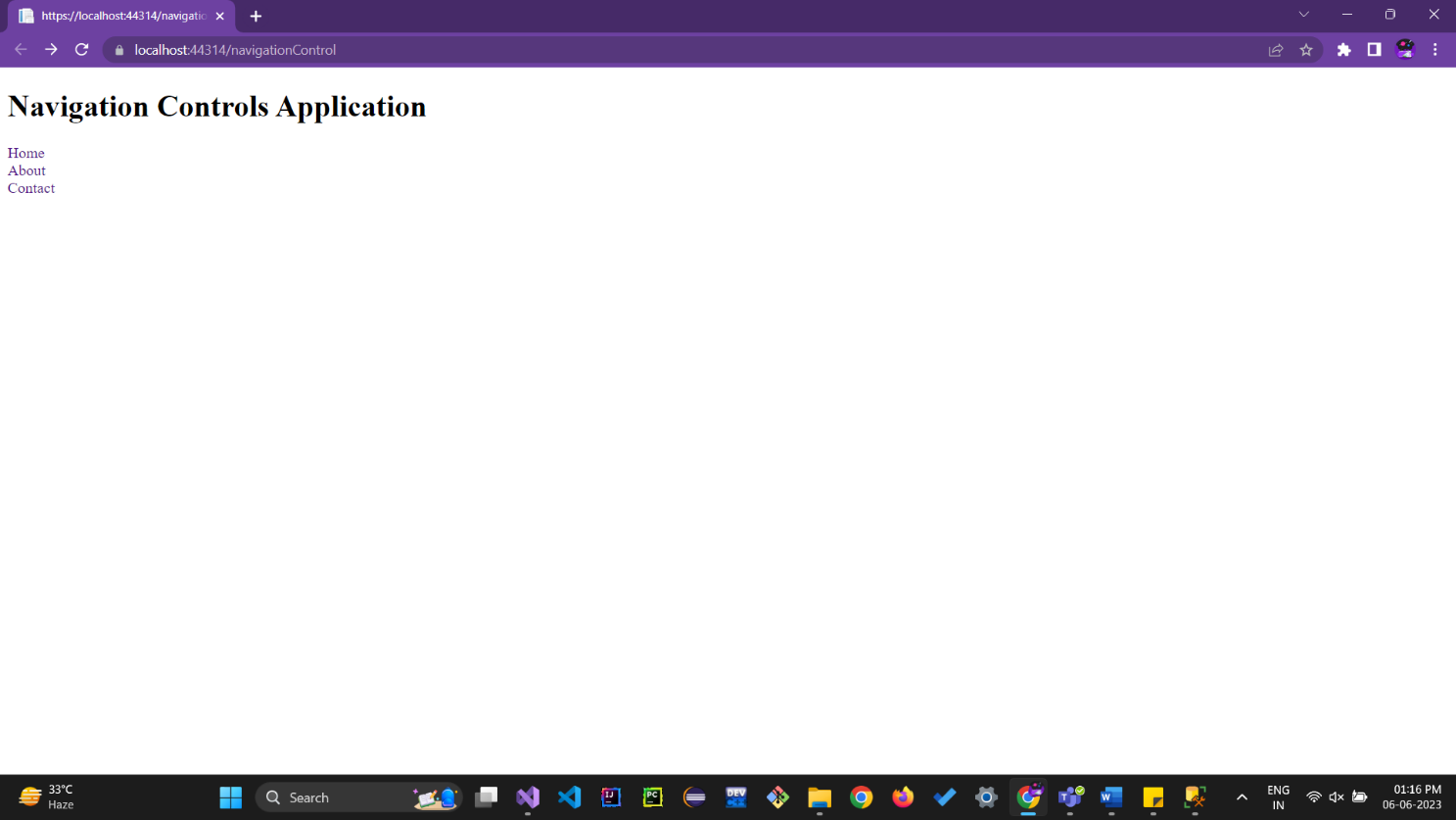
</div>

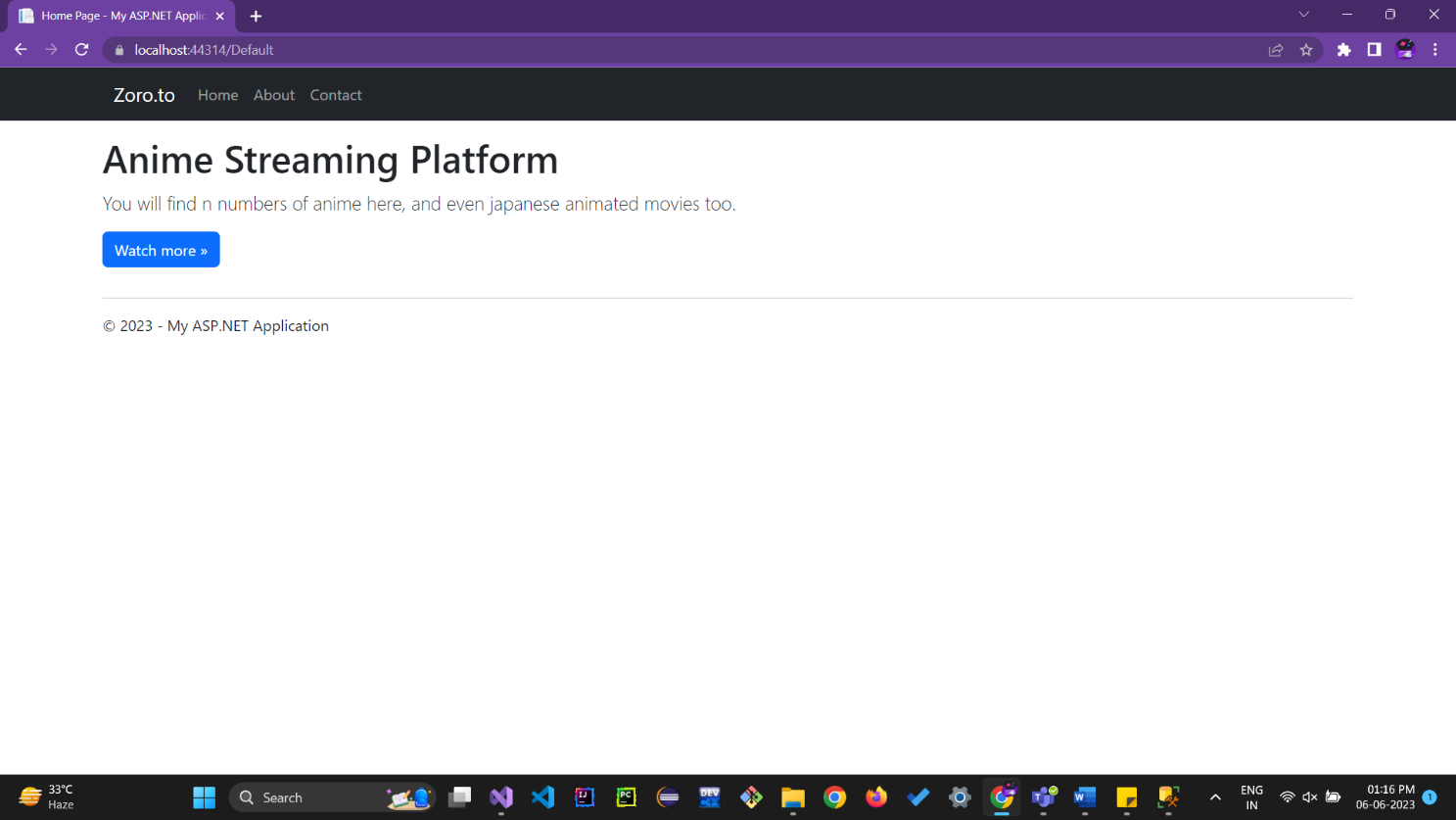
</form>

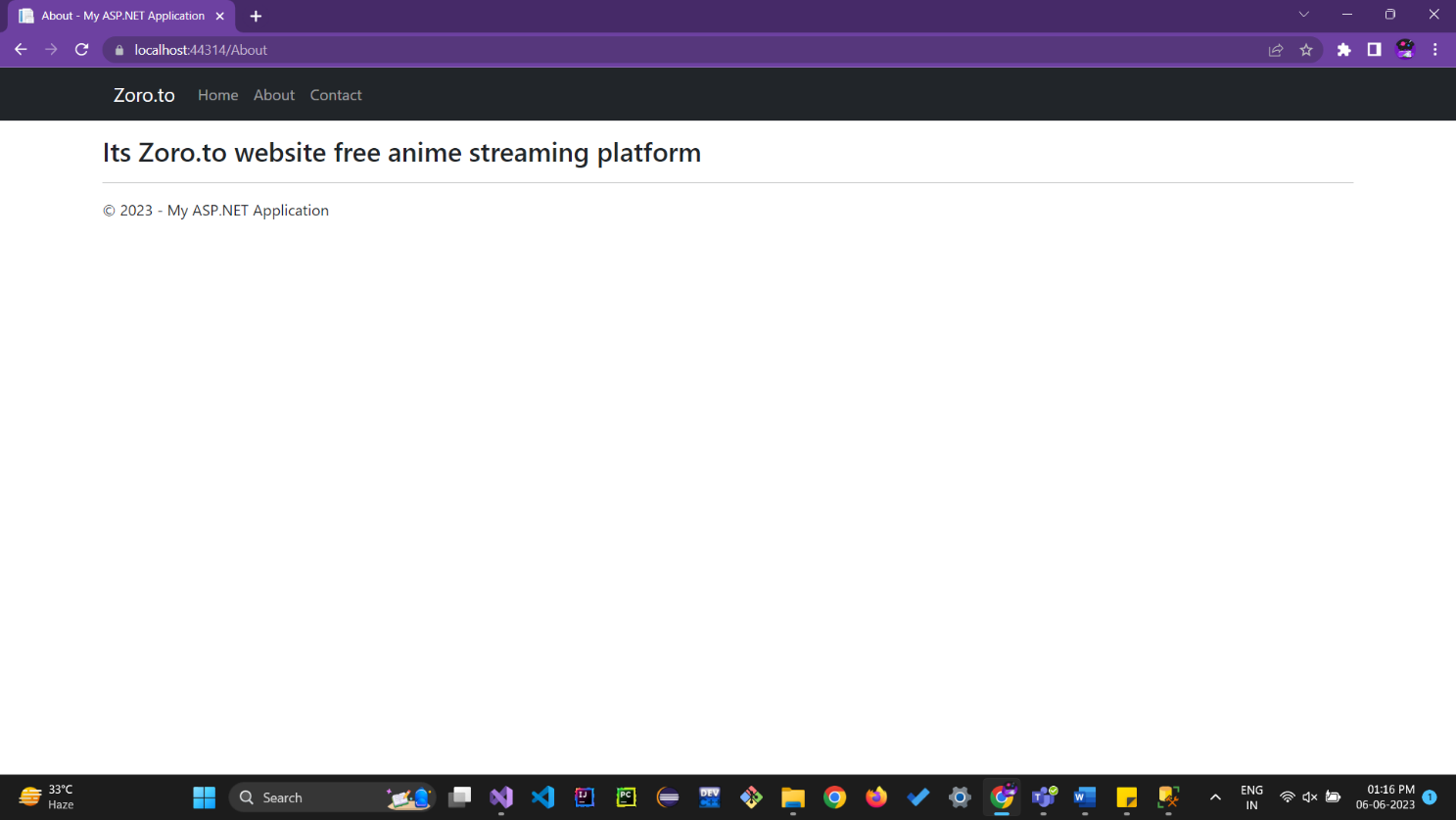
</body>

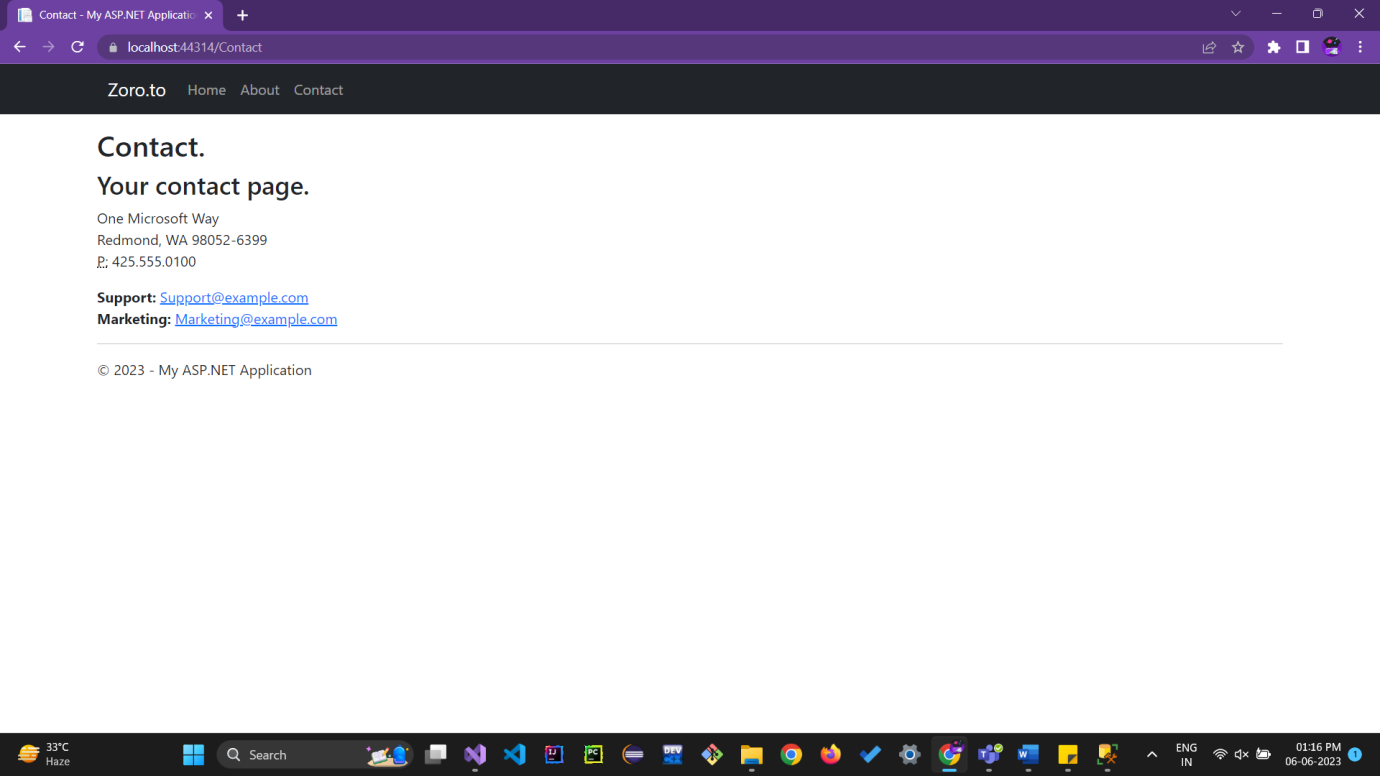
</html>

**Output:-**









3) **Aim:** Design an ASP.NET Application to Display Random Advertisements using ADRotator Control. Use XML DataSource and image folder.

**Objective:** To demonstrate AdRotator Control

**Theory:**

**AdRotator:**

The AdRotator control randomly selects banner graphics from a list, which is specified in an external XML schedule file. This external XML schedule file is called the advertisement file.

The AdRotator control allows you to specify the advertisement file and the type of window that the link should follow in the AdvertisementFile and the Target property respectively.

**Source Code:**

AdForm.aspx

<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="AdForm.aspx.cs" Inherits="Web\_Application\_III.AdForm" %>

<!DOCTYPE html>

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

<head runat="server">

<title>AdRotator</title>

</head>

<body>

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

<div style="height: 312px">

<asp:Label ID="Label1" runat="server" Font-Size="Medium" Text="Advertisements"></asp:Label>

<br />

<br />

<asp:ScriptManager ID="ScriptManager1" runat="server"></asp:ScriptManager>

<br />

<br />

<asp:UpdatePanel ID="UpdatePanel1" runat="server">

<ContentTemplate>

<asp:Timer ID="Timer1" runat="server" Interval="1000"></asp:Timer> &nbsp;&nbsp;&nbsp;

<asp:AdRotator ID="AdRotator1" runat="server" ForeColor="#333300" AdvertisementFile="~/XMLFile1.xml"/>

</ContentTemplate>

</asp:UpdatePanel>

</div>

</form>

</body>

</html>

XMLFile1.aspx

<?xml version="1.0" encoding="utf-8" ?>

<Advertisements>

<Ad>

<ImageUrl>~/Imgs/react.png</ImageUrl>

<NavigateUrl>https://react.dev/</NavigateUrl>

<AlternateText>Read React Documentation</AlternateText>

<Impressions>80</Impressions>

<Keyword>Search</Keyword>

</Ad>

<Ad>

<ImageUrl>~/Imgs/node.png</ImageUrl>

<NavigateUrl>https://nodejs.org/en</NavigateUrl>

<AlternateText>Read Node Documentation</AlternateText>

<Impressions>80</Impressions>

<Keyword>Search</Keyword>

</Ad>

<Ad>

<ImageUrl>~/Imgs/gatsby.png</ImageUrl>

<NavigateUrl>https://www.gatsbyjs.com/</NavigateUrl>

<AlternateText>Read Gatsby Documentation</AlternateText>

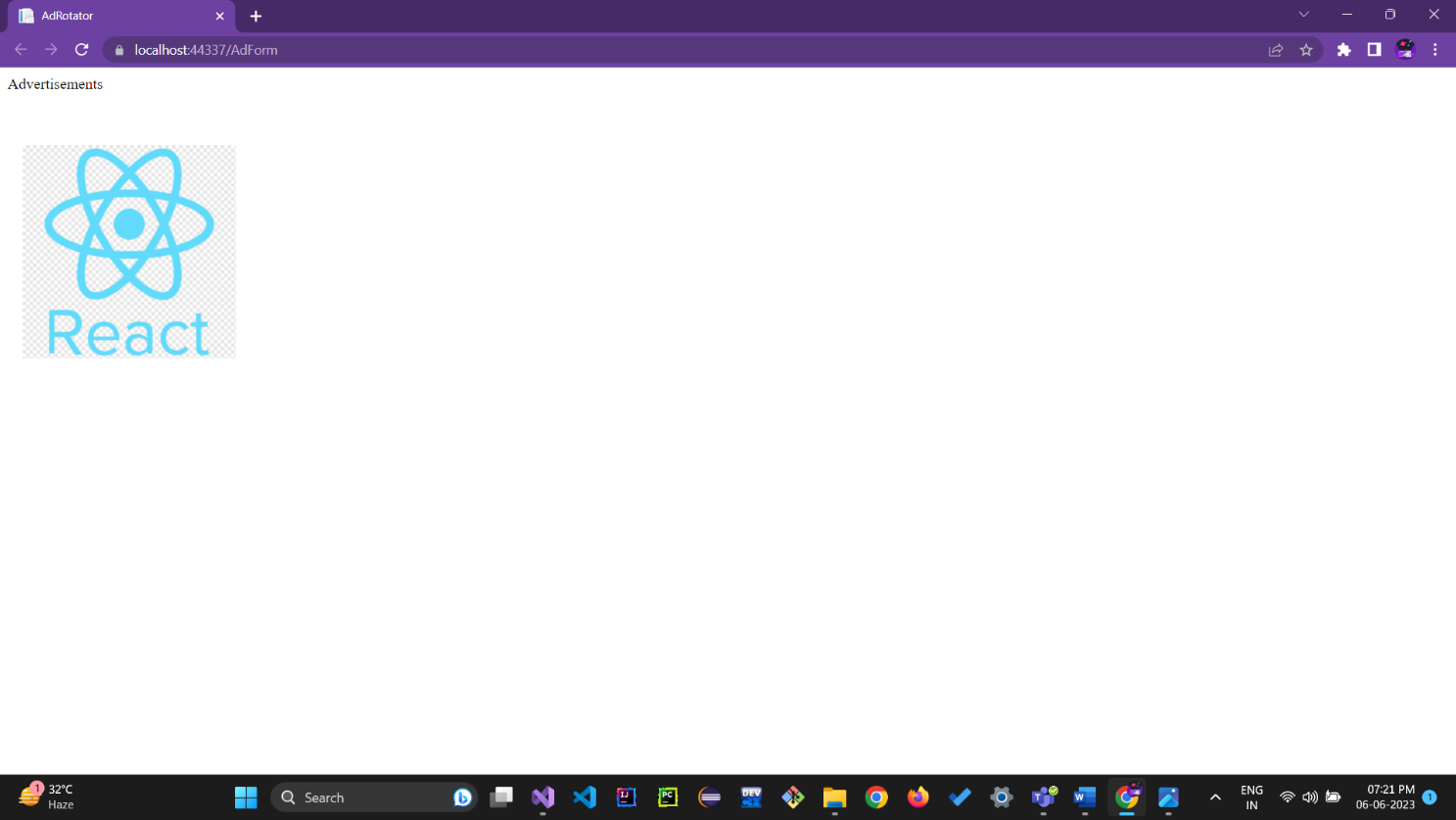
<Impressions>80</Impressions>

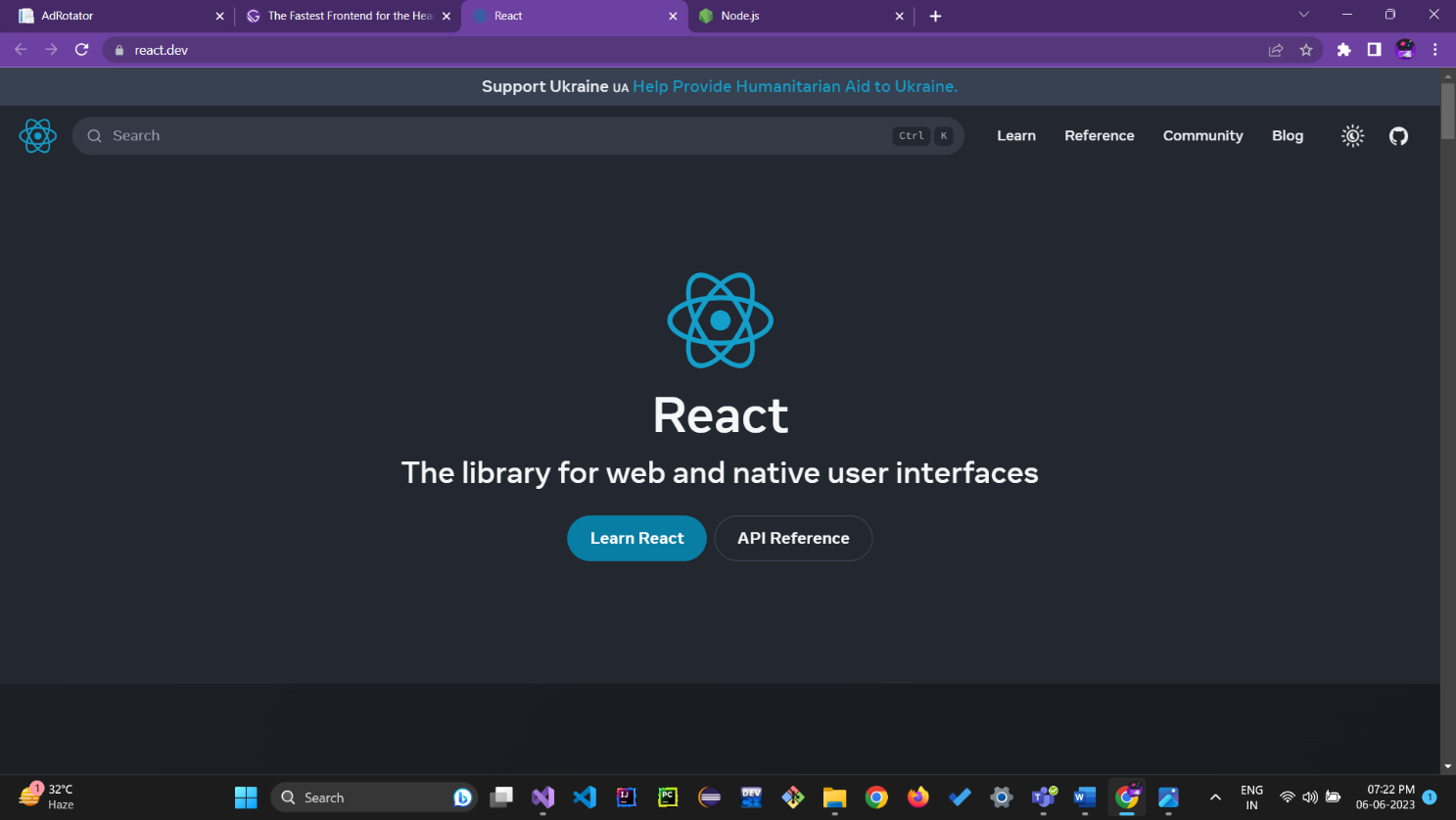
<Keyword>Search</Keyword>

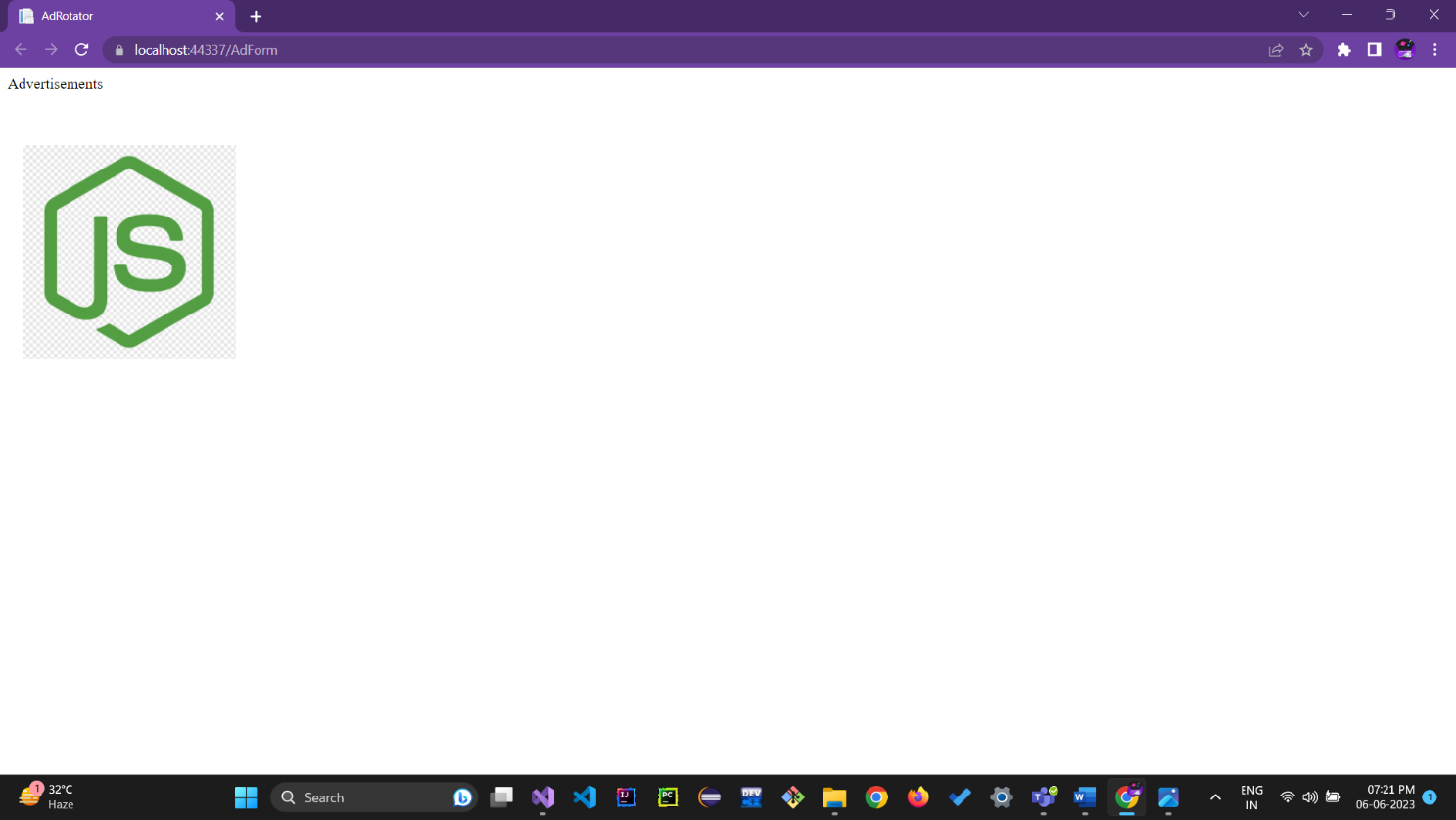
</Ad>

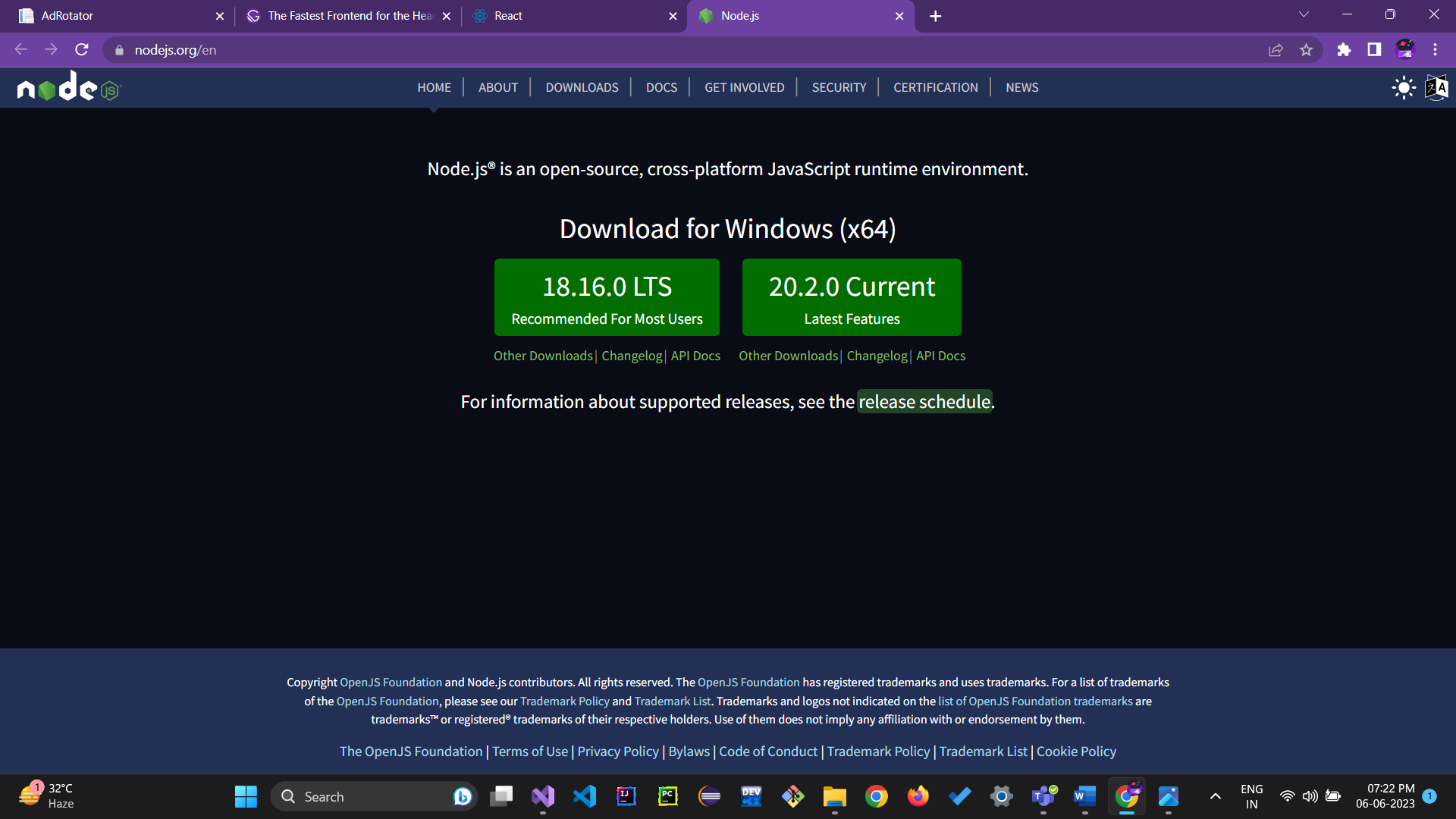
</Advertisements>

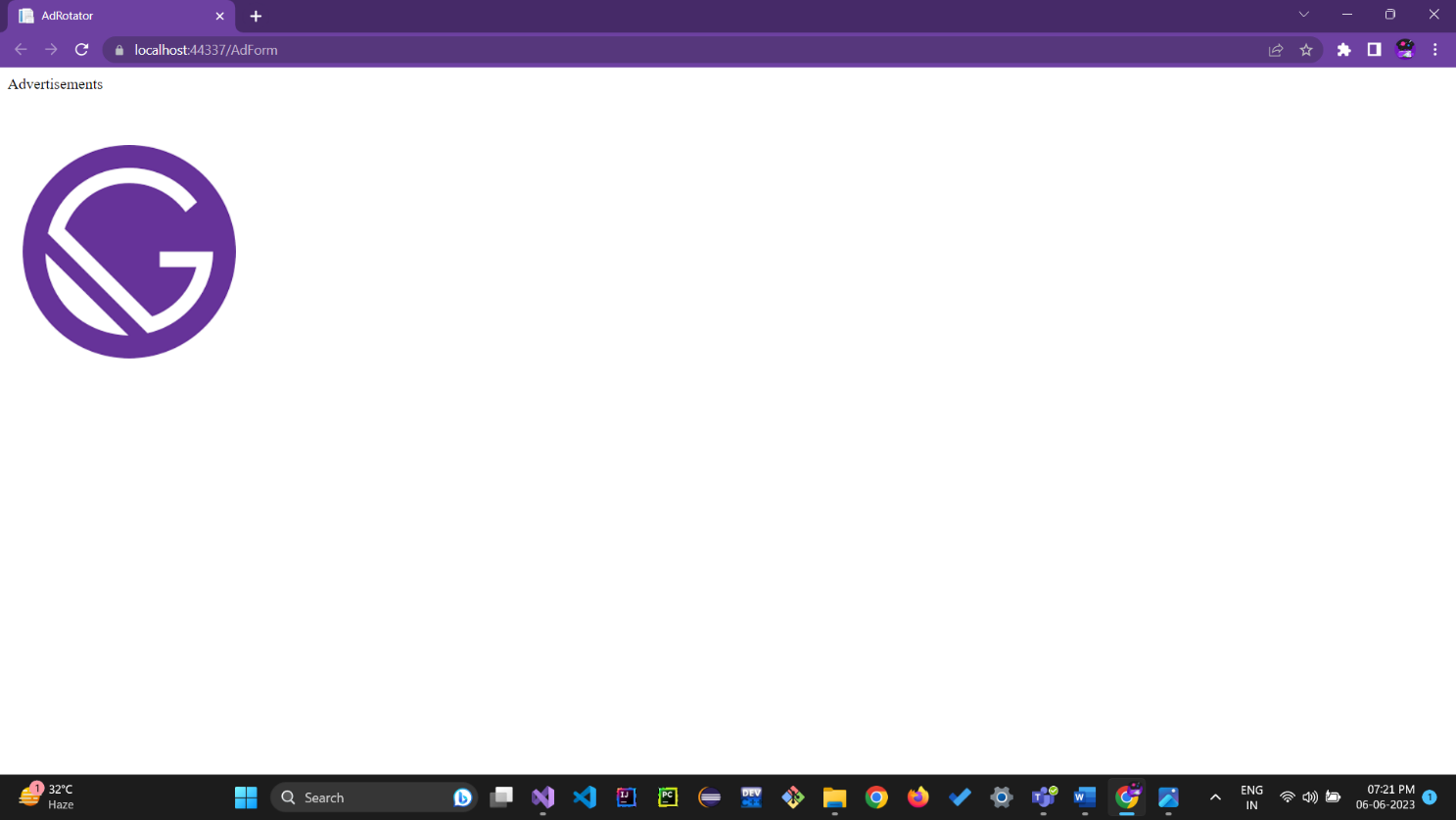
**Output:-**











**Aim 4):** Design an ASP.NET application to Display Current Month’s Calendar. Render the calendar to week days and current months days only.

**Objective:** To demonstrate Calendar Control

**Theory:**

ASP.NET provides a Calendar control that is used to display a calendar on a Web page. It can be used to display or take birthdays, anniversaries, appointments, holidays, bill payments, and project deadlines. ASP.NET Calendar control displays a month calendar that allows user to select dates and move to the next and previous months.

**Source Code:**

Form1.cs

namespace DateTimeForm

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

int days;

DateTime d = new DateTime();

d = DateTime.Now;

DateTime newyear = new DateTime(d.Year, 12, 31);

textBox1.Text = d.ToString();

textBox2.Text = d.ToShortDateString();

textBox3.Text = d.ToString("D");

textBox4.Text = d.ToString("F");

textBox5.Text = d.ToLongTimeString();

days = newyear.DayOfYear - d.DayOfYear;

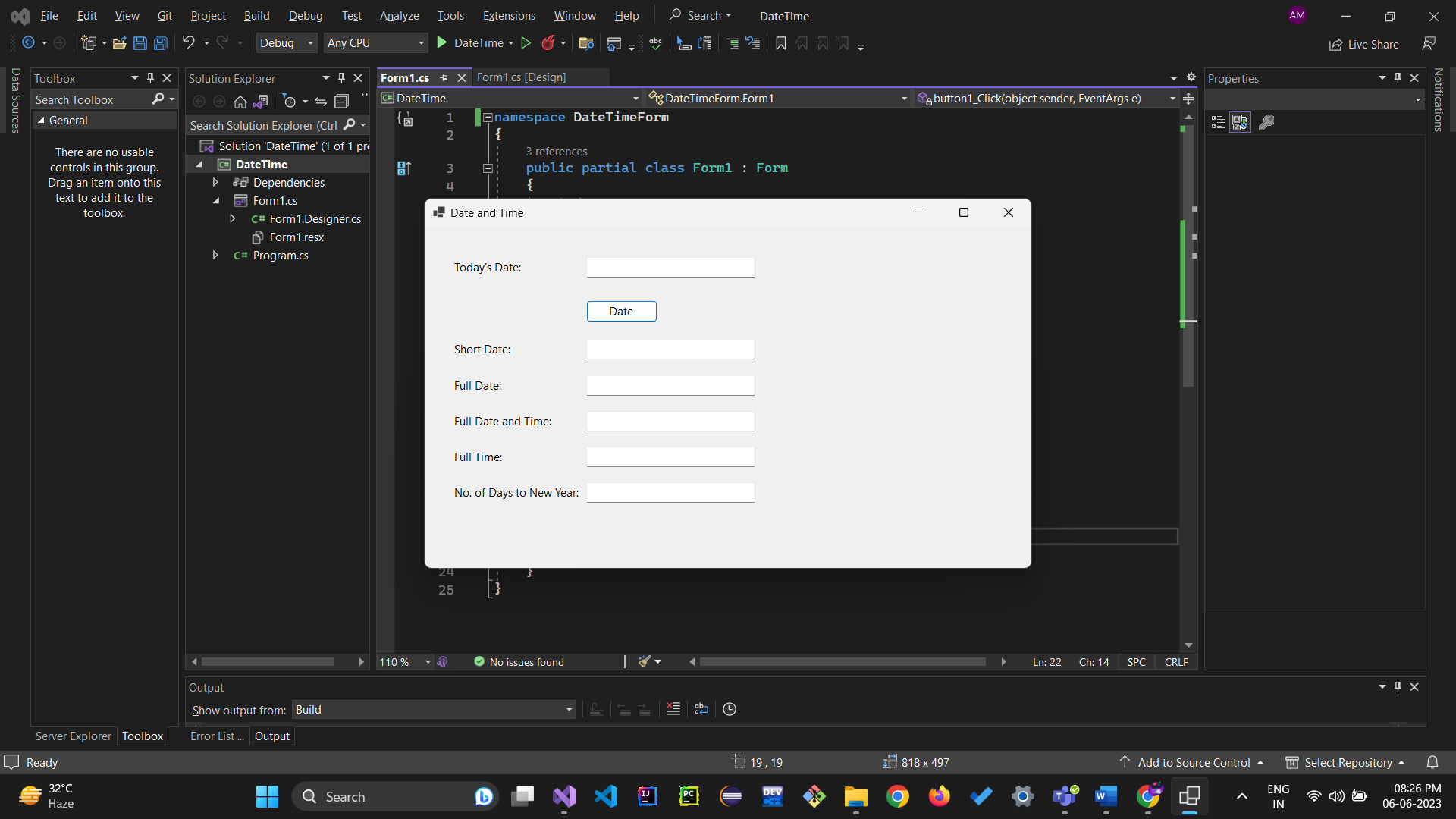
textBox6.Text = days.ToString();

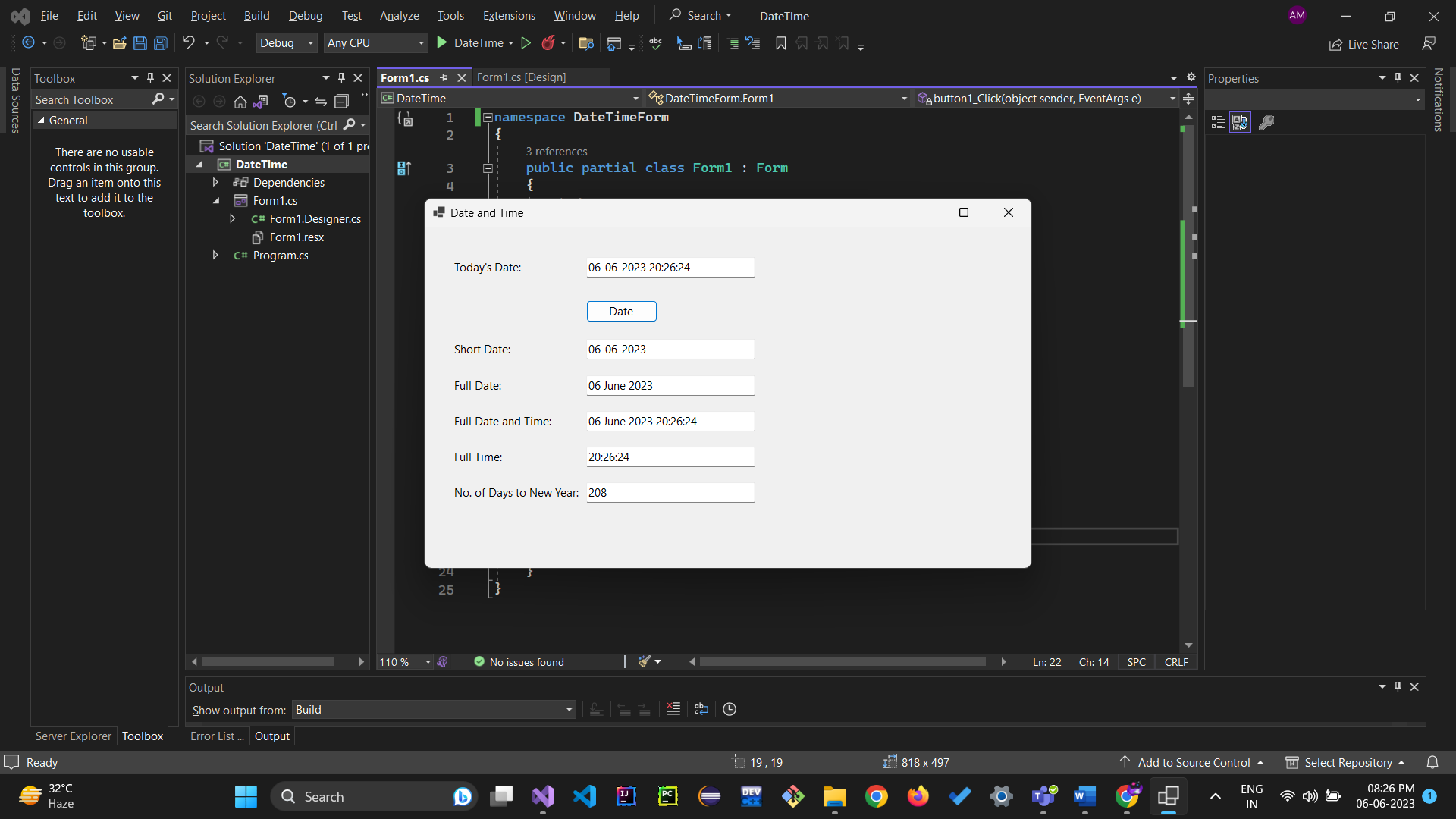
}

}

}

**Output:-**





5) **Aim:** Design a Web Application for an Organization with Registration forms and advanced controls

**a)Registration form must i) Emp\_Name ii) Emp\_ID iii) DOB iv) photograph**

**v) Password vi) new password**

**b)Apply validation control on form field**

**c)Apply calendar on DOB to select date**

**d)file upload on photograph.**

**Objective:** To demonstrate form Validation

**Theory:**

**Validation Control:**

There are Six Servers as well as client-side validation controls in ASP.Net that are as follows:

* + 1. RequiredFieldValidator control:

This control ensures that the control it is used for validation is not empty when the form is submitted. In other words suppose their is one Text Box control and you have used a RequiredFieldValidator to validate that text box; then before submitting the data on the server it checks if the text box is not empty.

* + 1. RangeValidator:

Checks that the value of the associated control is within a specified range. The value and the range can be numerical, a date or a string. In other words suppose their is one text box and you want to allow only 10 digits or any strings with a specified range using RangeValidator then before submitting the data on the server it ensure that the value is within a specified range.

* + 1. CompareValidator:

Checks that the value of the associated control matches a specified comparison (less than, greater than, and so on) against another constant value or control.

* + 1. RegularExpressionValidator

Checks if the value of the control it has to validate matches the specified regular expression.

* + 1. CustomValidator

Allows specification of any client-side JavaScript validation routine and its server-side counterpart to perform your own custom validation logic.

6.ValidationSummary

Shows a summary with the error messages for each failed validator on the page (or in a pop- up message box).

Source Code:

Emp\_Regs.aspx

<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Emp\_Regs.aspx.cs" Inherits="Web\_Application\_V.Emp\_Regs" %>

<!DOCTYPE html>

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

<head runat="server">

<title></title>

<link href="~/Content/bootstrap.min.css" rel="stylesheet" />

</head>

<body>

<form id="form1" runat="server" class="p-5">

<div style="width:75%; text-align:center;">

<h2>Employee Registration Form</h2>

</div>

<div style="width: 75%;" class="border border-primary">

<div class="d-flex justify-content-between align-items-center m-3">

<div style="width: 40%; margin-left: 40px;">

Name: <br />

<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>

<asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server" ErrorMessage="Required Field" ControlToValidate="TextBox1" ForeColor="Red"></asp:RequiredFieldValidator>

</div>

<div style="width: 40%">

ID: <br />

<asp:TextBox ID="TextBox2" runat="server"></asp:TextBox>

<asp:RequiredFieldValidator ID="RequiredFieldValidator2" runat="server" ErrorMessage="Required Field" ControlToValidate="TextBox2" ForeColor="Red"></asp:RequiredFieldValidator>

</div>

</div>

<div class="d-flex justify-content-between align-items-center m-3">

<div style="width: 40%; margin-left: 40px;">

DOB: <br />

<asp:TextBox ID="TextBox3" runat="server" TextMode="Date"></asp:TextBox>

</div>

<div style="width: 40%">

photograph: <br />

<input id="Text2" type="file" />

</div>

</div>

<div class="d-flex justify-content-between align-items-center m-3">

<div style="width: 40%; margin-left: 40px;">

Password: <br />

<asp:TextBox ID="TextBox4" runat="server" TextMode="Password"></asp:TextBox>

</div>

<div style="width: 40%">

Confirm Password: <br />

<asp:TextBox ID="TextBox5" runat="server" TextMode="Password"></asp:TextBox>

<asp:CompareValidator ID="CompareValidator1" runat="server" ErrorMessage="Passwords do not match." ControlToValidate="TextBox5" ControlToCompare="TextBox4" ForeColor="Red"></asp:CompareValidator>

</div>

</div>

<div class="d-flex justify-content-center align-items-center m-3">

<asp:Button ID="Button1" runat="server" Text="Submit" class="btn btn-success" />

</div>

</div>

</form>

</body>

</html>

**Output:-**

