Create an endpoint GET /weather/{city} that fetches weather data from a public weather API (e.g., OpenWeatherMap). Parse the response and return only the necessary details (temperature, humidity, conditions, etc.). Implement error handling for invalid city names or API errors.

2. Folder Structure

Here's the structure we'll create:

```
markdown
Copy code
/WeatherApp
    /Controllers
    /Models
    /Services
    /Program.cs
```

---

3. Step-by-Step Implementation

Step 1: Create Models

1. **Create the `Models` folder**:
   - Right-click the project > **Add** > **New Folder**.
   - Name it **Models**.
2. **Add `WeatherResponse.cs`**:
   - Right-click the `Models` folder > **Add** > **Class**.
   - Name it `WeatherResponse.cs`.

WeatherResponse.cs:

```
csharp
Copy code
namespace WeatherApp.Models
{
    public class WeatherResponse
    {
        public string City { get; set; }
        public double Temperature { get; set; }
        public int Humidity { get; set; }
        public string Conditions { get; set; }
    }
}
```

Step 2: Create Weather Service

1. **Create the `Services` folder**:
   - Right-click the project > **Add** > **New Folder**.
   - Name it **Services**.
2. **Add `WeatherService.cs`**:
   - Right-click the `Services` folder > **Add** > **Class**.
   - Name it `WeatherService.cs`.

WeatherService.cs:

```
csharp
Copy code
using System.Net.Http;
using System.Text.Json;
using System.Threading.Tasks;
using WeatherApp.Models;

public class WeatherService
{
    private readonly HttpClient _httpClient;
    private const string ApiKey = "YOUR_API_KEY"; // Replace with
your OpenWeatherMap API key
    private const string BaseUrl =
"https://api.openweathermap.org/data/2.5/weather";

    public WeatherService(HttpClient httpClient)
    {
        _httpClient = httpClient;
    }

    public async Task<WeatherResponse> GetWeatherAsync(string city)
    {
        var url = $"{BaseUrl}?q={city}&appid={ApiKey}&units=metric";

        var response = await _httpClient.GetAsync(url);

        if (!response.IsSuccessStatusCode)
        {
            throw new HttpRequestException($"Weather API returned
status code {response.StatusCode}");
        }

        var jsonResponse = await
response.Content.ReadAsStringAsync();
        var weatherData = JsonDocument.Parse(jsonResponse);

        return new WeatherResponse
        {
            City =
weatherData.RootElement.GetProperty("name").GetString(),
            Temperature =
weatherData.RootElement.GetProperty("main").GetProperty("temp").GetDo
uble(),
            Humidity =
weatherData.RootElement.GetProperty("main").GetProperty("humidity").G
etInt32(),
            Conditions =
weatherData.RootElement.GetProperty("weather")[0].GetProperty("descri
ption").GetString()
        };
    }
}
```

Step 3: Create Controller

1. **Create the `Controllers` folder**:

- Right-click the project > **Add** > **New Folder**.
- Name it **Controllers**.

2. **Add `WeatherController.cs`**:
   - Right-click the `Controllers` folder > **Add** > **Controller**.
   - Select **API Controller - Empty**.
   - Name it `WeatherController`.

WeatherController.cs:

```
csharp
Copy code
using Microsoft.AspNetCore.Mvc;
using System.Threading.Tasks;
using WeatherApp.Models;
using WeatherApp.Services;

[ApiController]
[Route("api/[controller]")]
public class WeatherController : ControllerBase
{
    private readonly WeatherService _weatherService;

    public WeatherController(WeatherService weatherService)
    {
        _weatherService = weatherService;
    }

    [HttpGet("{city}")]
    public async Task<IActionResult> GetWeather(string city)
    {
        try
        {
            var weather = await
_weatherService.GetWeatherAsync(city);
            return Ok(weather);
        }
        catch (HttpRequestException ex)
        {
            return StatusCode(500, $"External API error:
{ex.Message}");
        }
        catch (KeyNotFoundException)
        {
            return NotFound($"City '{city}' not found.");
        }
        catch
        {
            return StatusCode(500, "An unexpected error occurred.");
        }
    }
}
```

Step 4: Configure Dependency Injection

1. **Edit `Program.cs`**: Open `Program.cs` and register the `WeatherService`.

```
csharp
```

```
Copy code
var builder = WebApplication.CreateBuilder(args);

// Add services to the container.
builder.Services.AddControllers();
builder.Services.AddHttpClient<WeatherService>();

var app = builder.Build();

// Configure the HTTP request pipeline.
app.UseHttpsRedirection();
app.UseAuthorization();
app.MapControllers();

app.Run();
```

---

4. Test the Application

1. **Run the application**.
2. Use a tool like **Postman** or **Swagger** to test:
   - `GET /api/weather/{city}`

   Example:

   ```
   bash
   Copy code
   GET /api/weather/London
   ```

3. **Expected Response**:

   ```
   json
   Copy code
   {
       "city": "London",
       "temperature": 15.6,
       "humidity": 72,
       "conditions": "clear sky"
   }
   ```

Create ASP.Net MVC Web application for Online Shopping with Master Page and minimum 4 Pages.

2. Folder and File Structure

The project structure will look like this:

```
vbnet
Copy code
/OnlineShopping
    /Controllers
    /Models
    /Views
        /Shared
        /Home
```

```
    /Products
    /Cart
    /Contact
/Content
/Scripts
/App_Start
/Global.asax
```

---

3. Step-by-Step Implementation

Step 1: Create the Master Page (Layout)

1. **Right-click the `Views/Shared` folder > Add > New Item.**
   - Select **MVC Layout Page**.
   - Name it `Layout.cshtml`.
2. **Edit `Layout.cshtml`:**

```
html
Copy code
<!DOCTYPE html>
<html>
<head>
    <title>@ViewData["Title"] - Online Shopping</title>
    <link href="~/Content/site.css" rel="stylesheet" />
</head>
<body>
    <header>
        <h1>Online Shopping Platform</h1>
        <nav>
            <ul>
                <li>@Html.ActionLink("Home", "Index", "Home")</li>
                <li>@Html.ActionLink("Products", "Index",
"Products")</li>
                <li>@Html.ActionLink("Cart", "Index", "Cart")</li>
                <li>@Html.ActionLink("Contact", "Contact",
"Home")</li>
            </ul>
        </nav>
    </header>
    <main>
        @RenderBody()
    </main>
    <footer>
        <p>© 2024 Online Shopping</p>
    </footer>
</body>
</html>
```

Step 2: Create Controllers

1. HomeController

1. **Right-click the `Controllers` folder > Add > Controller.**
   - Choose **MVC 5 Controller - Empty**.

- Name it `HomeController`.

2. **Edit `HomeController.cs`:**

```
csharp
Copy code
using System.Web.Mvc;

namespace OnlineShopping.Controllers
{
    public class HomeController : Controller
    {
        public ActionResult Index()
        {
            return View();
        }

        public ActionResult Contact()
        {
            ViewBag.Message = "Contact us for support.";
            return View();
        }
    }
}
```

2. ProductsController

1. **Right-click the `Controllers` folder > Add > Controller.**
   - Choose **MVC 5 Controller - Empty**.
   - Name it `ProductsController`.
2. **Edit `ProductsController.cs`:**

```
csharp
Copy code
using System.Collections.Generic;
using System.Web.Mvc;
using OnlineShopping.Models;

namespace OnlineShopping.Controllers
{
    public class ProductsController : Controller
    {
        public ActionResult Index()
        {
            var products = new List<Product>
            {
                new Product { Id = 1, Name = "Laptop", Price = 800,
Category = "Electronics" },
                new Product { Id = 2, Name = "Smartphone", Price =
600, Category = "Electronics" },
                new Product { Id = 3, Name = "Headphones", Price =
50, Category = "Accessories" }
            };

            return View(products);
        }
    }
}
```

3. CartController

1. **Right-click the `Controllers` folder > Add > Controller.**
   - Choose **MVC 5 Controller - Empty**.
   - Name it `CartController`.
2. **Edit `CartController.cs`:**

```
csharp
Copy code
using System.Web.Mvc;

namespace OnlineShopping.Controllers
{
    public class CartController : Controller
    {
        public ActionResult Index()
        {
            ViewBag.Message = "Your shopping cart is empty.";
            return View();
        }
    }
}
```

Step 3: Create Models

1. **Right-click the `Models` folder > Add > Class.**
   - Name it `Product.cs`.
2. **Edit `Product.cs`:**

```
csharp
Copy code
namespace OnlineShopping.Models
{
    public class Product
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public decimal Price { get; set; }
        public string Category { get; set; }
    }
}
```

Step 4: Create Views

1. Home Views

1. **Right-click the `Views/Home` folder > Add > View.**
   - Name it `Index.cshtml`.
2. **Edit `Index.cshtml`:**

```
html
Copy code
@{
```

```
    ViewData["Title"] = "Home";
}

<h2>Welcome to Online Shopping</h2>
<p>Discover the best products at unbeatable prices!</p>
```

3. **Right-click the `Views/Home` folder > Add > View.**
   - Name it `Contact.cshtml`.
4. **Edit `Contact.cshtml`:**

```
html
Copy code
@{
    ViewData["Title"] = "Contact";
}

<h2>Contact Us</h2>
<p>@ViewBag.Message</p>
```

2. Products View

1. **Right-click the `Views/Products` folder > Add > View.**
   - Name it `Index.cshtml`.
2. **Edit `Index.cshtml`:**

```
html
Copy code
@{
    ViewData["Title"] = "Products";
}

<h2>Our Products</h2>
<table class="table">
    <thead>
        <tr>
            <th>Name</th>
            <th>Price</th>
            <th>Category</th>
        </tr>
    </thead>
    <tbody>
        @foreach (var product in Model)
        {
            <tr>
                <td>@product.Name</td>
                <td>@product.Price</td>
                <td>@product.Category</td>
            </tr>
        }
    </tbody>
</table>
```

3. Cart View

1. **Right-click the `Views/Cart` folder > Add > View.**
   - Name it `Index.cshtml`.
2. **Edit `Index.cshtml`:**

```
html
```

```
Copy code
@{
    ViewData["Title"] = "Cart";
}

<h2>Shopping Cart</h2>
<p>@ViewBag.Message</p>
```

Step 5: Configure Routing

1. Open **`RouteConfig.cs`** in the **App_Start** folder:

```
csharp
Copy code
public class RouteConfig
{
    public static void RegisterRoutes(RouteCollection routes)
    {
        routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

        routes.MapRoute(
            name: "Default",
            url: "{controller}/{action}/{id}",
            defaults: new { controller = "Home", action = "Index", id
= UrlParameter.Optional }
        );
    }
}
```