In this assignment exercise, you will redevelop the previous assignment (Weather App) with Services and Dependency Injection.

Requirement:

Imagine a weather application that shows weather details of the selected city. Create an Asp.Net Core Web Application that fulfils this requirement.

Consider the following hard-coded weather data of 3 cities.

```
CityUniqueCode = "LDN", CityName = "London", DateAndTime = "2030-01-01 8:00",  TemperatureFahrenheit = 33
CityUniqueCode = "NYC", CityName = "London", DateAndTime = "2030-01-01 3:00",  TemperatureFahrenheit = 60
CityUniqueCode = "PAR", CityName = "Paris", DateAndTime = "2030-01-01 9:00",  TemperatureFahrenheit = 82
```

Consider a model class called 'CityWeather' with following properties:

```
string CityUniqueCode
string CityName
DateTime DateAndTime
int TemperatureFahrenheit
```

Example #1:

If you receive a HTTP GET request at path "/", it has to generate a view with weather details of all cities with HTTP status code 200.

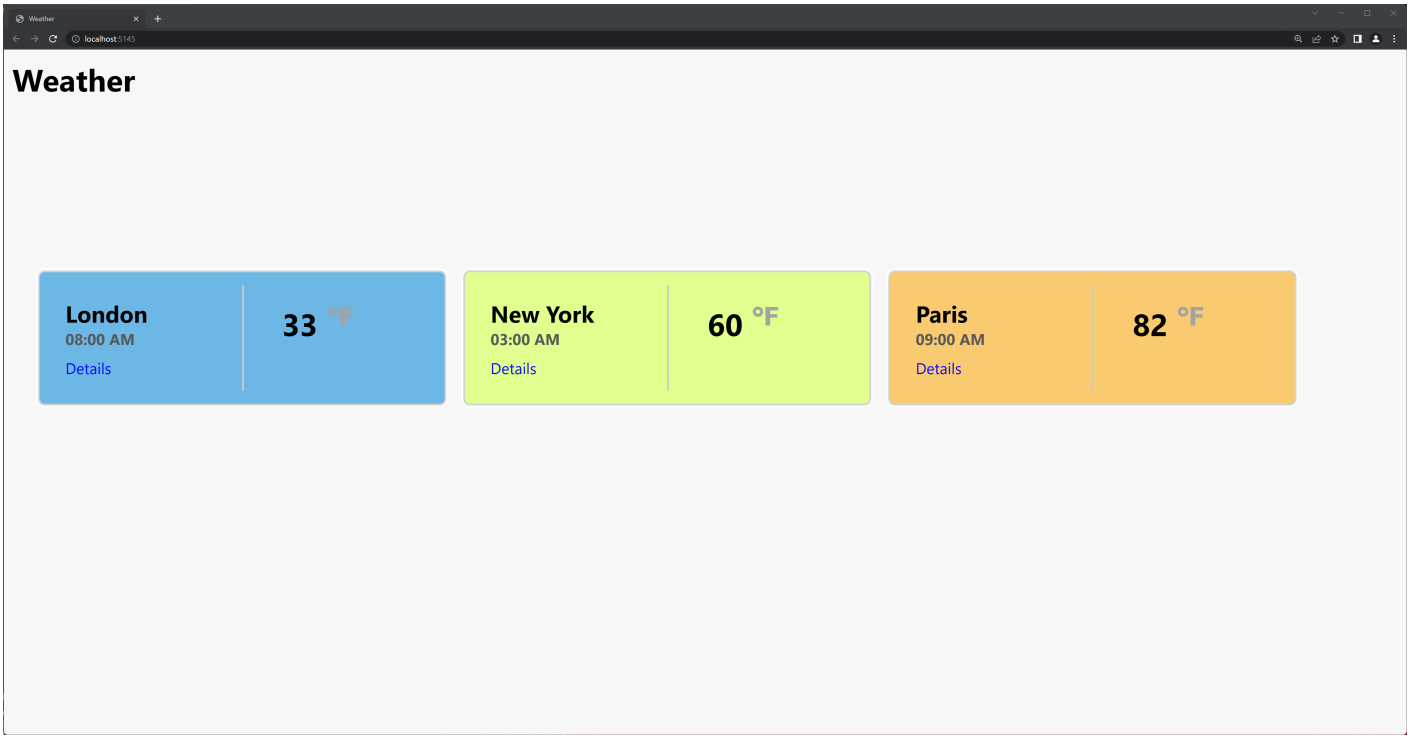
Request Url: /

Request Method: GET

Response Status Code: 200

Response body (output):

View as shown below.



Example #2:

If you receive a HTTP GET request at path "/weather/{cityCode}", it has to generate a view with weather details of the selected city s with HTTP status code 200.

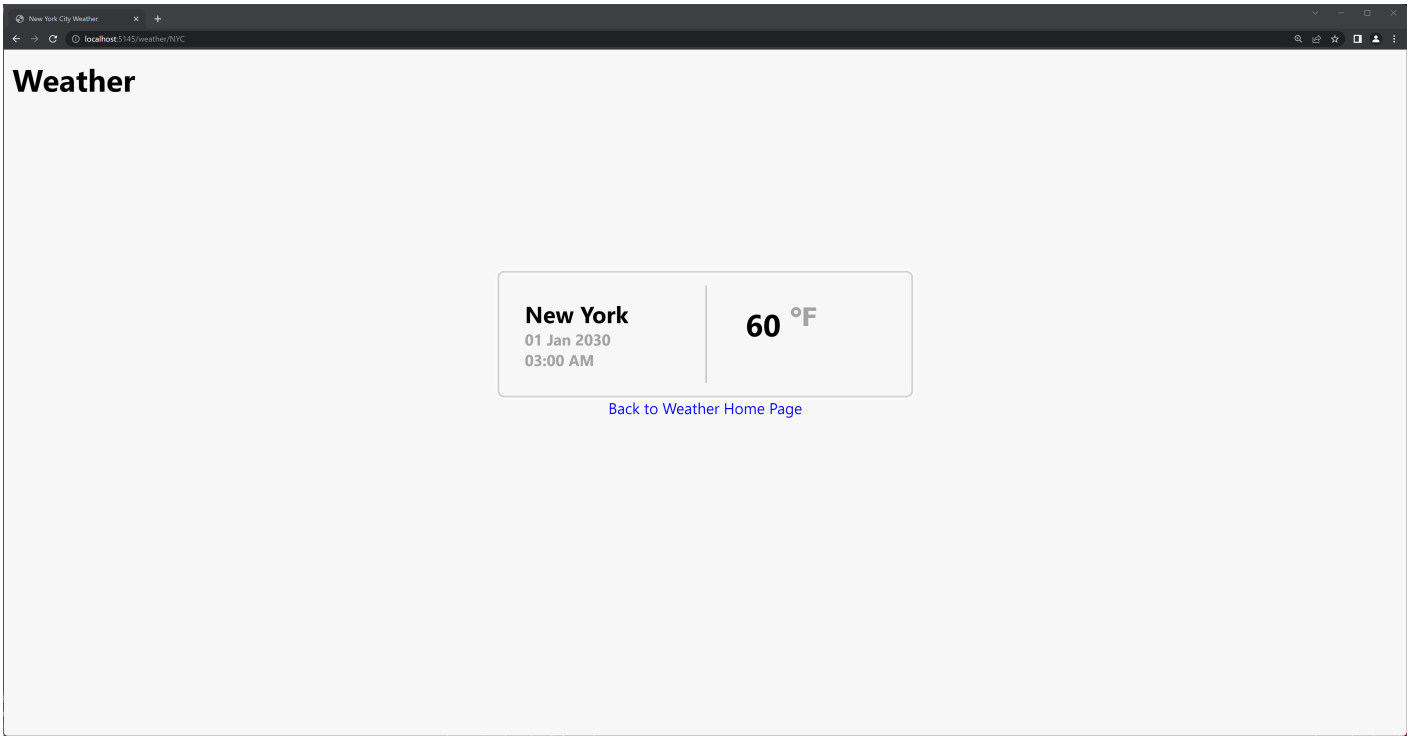
Request Url: /

Request Method: GET

Response Status Code: 200

Response body (output):

View as shown below.



Instructions:

- Create controller(s) with attribute routing.
- Create a service called "WeatherService" that implements the following service contract (interface).

```
IWeatherService
    - List<CityWeather> GetWeatherDetails() - Returns a list of
      CityWeather objects that contains weather details of
      cities
    - CityWeather? GetWeatherByCityCode(string CityCode) -
      Returns an object of CityWeather based on the given city
      code
```

- Inject the WeatherService into controller using constructor injection. Call the appropriate service methods in action methods, accordingly.
- Initialize the hard-coded data as collection of model objects in the service.
- Use strongly-typed views. Send model object(s) to view.
- If you supply an invalid city code as route parameter, it should show a page with proper error message, instead of throwing an exception.
- Use CSS styles, layout views, _ViewImports, _ViewStart, view components as per the necessity.
- The UI should be consistent and modern. It should minimum look like as shown in the screenshot. Optionally, you can try enhancing it based on your thoughts.
- You can create a view component that displays weather details of a single city; and invoke the same in "Index" view in foreach loop while reading city details.
- Apply background color for each box, based on the following categories of temperature value. Write essential code in view component, to determine the apppriate css class to apply background color.

```
Fahrenheit is less than 44 = blue background color
Fahrenheit is between 44 and 74 = blue background color
Fahrenheit is greater than 74 = blue background color
```

- The CSS file is provided as downlodable resource for applying essential styles. You can download and use it.