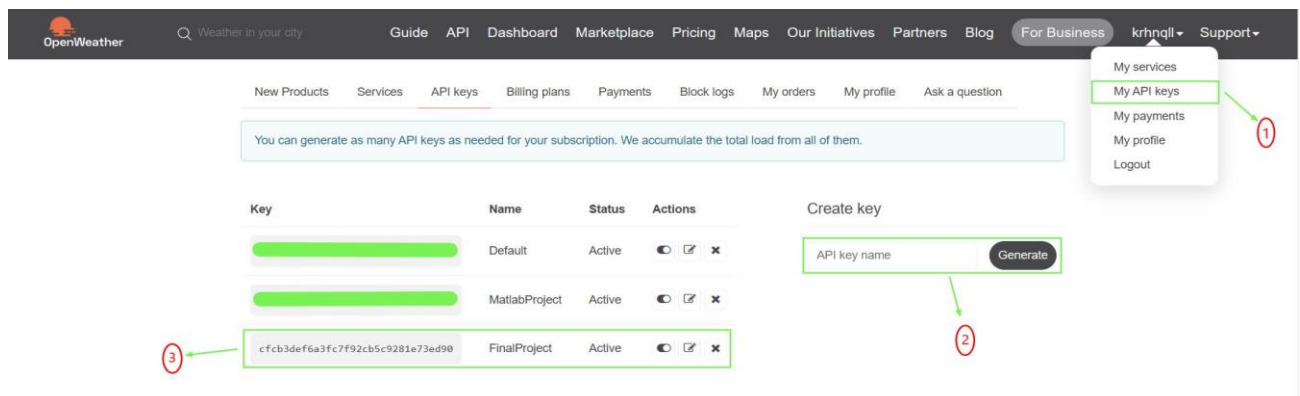


1.0 | Getting Started: My Application Steps

In this project, we are required to create a city-based weather forecast and visualization application. At the beginning of the project, I worked with the information needed about Matlab's AppDesigner extension. First, I created a new project in AppDesigner. Then, I placed the necessary tools on the application to provide the required data. I added three graphs, two buttons, one dropdown, one edit field, and three labels. The graphs display the 5-day forecast of temperature, humidity, and wind speed for the selected city. By using the edit field and the add city button, I assigned the city data to the select city dropdown, allowing the user to list their desired cities. Then, using the second button and the API provided by openweathermap.org, I fetched real-time data. Of course, I wrote the necessary code in the background to make the buttons work and to retrieve the data.

1.1 | How to Provide an API? Using OpenWeatherMap:



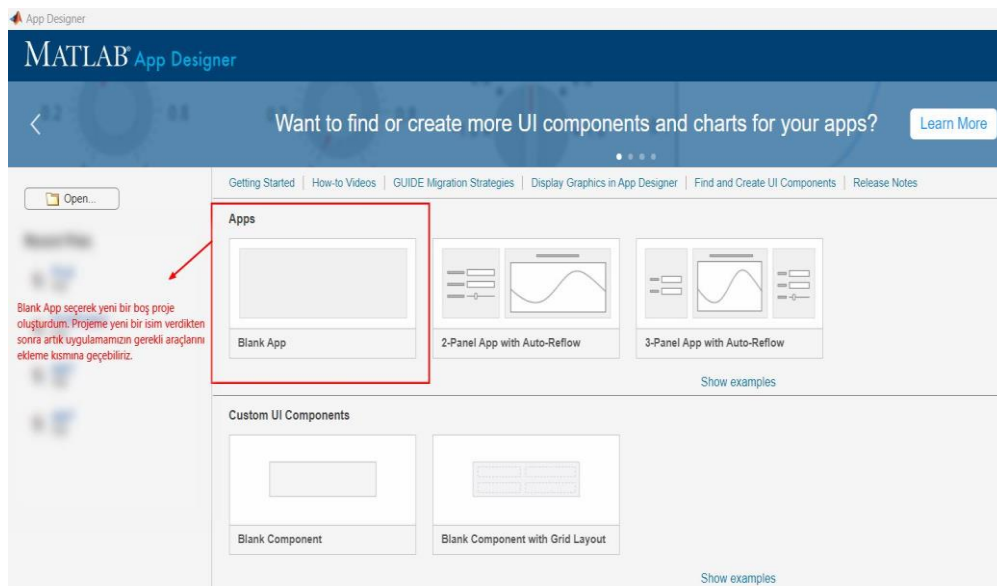
First, you need to visit openweathermap.org and sign up. Then:

1. Complete the registration and email verification process**. After that, navigate to the **My API KEYS** section under your username.
2. Create the necessary API Key**: In the **Create Key** section, name your API key and generate it.
3. After completing steps (1) and (2), your usable API Key will be created. You will also have the ability to activate or deactivate the generated Key.

1.2 | Getting Started with AppDesigner:



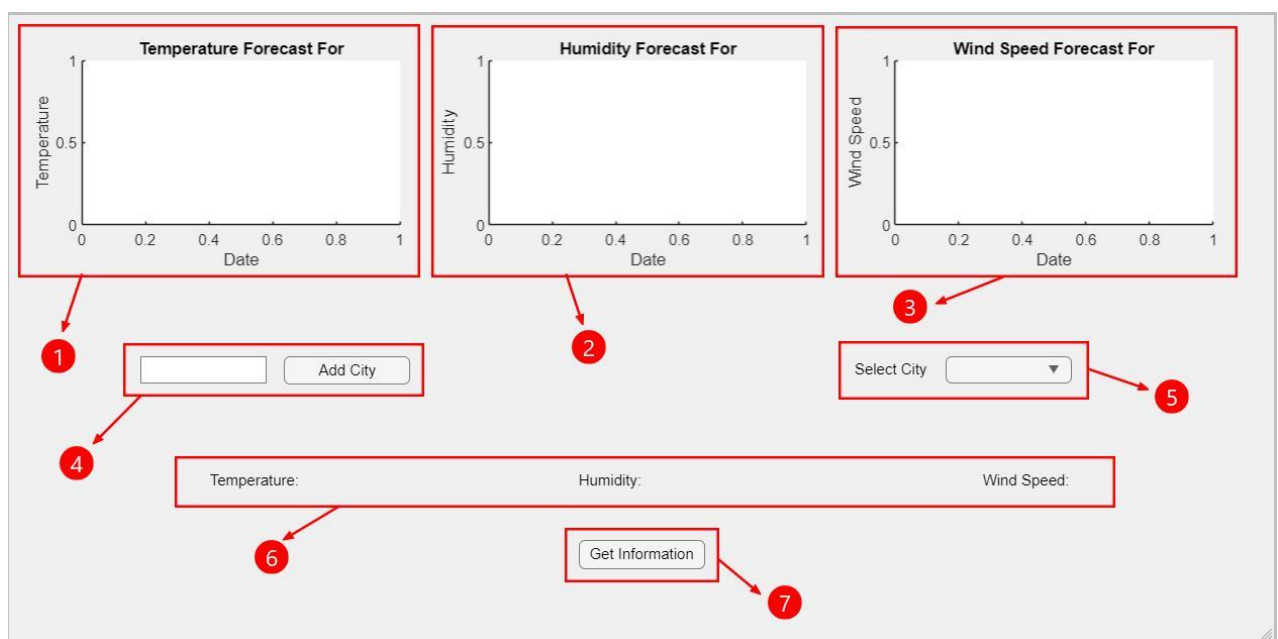
(1) To open App Designer, first, you need to type `appdesigner` in the COMMAND WINDOW.



(2)- "I created a new project by selecting Blank App. After giving a new name to my project, now I can move on to adding the necessary tools for our application."

2.0 | Steps of my project: How did I do it?

2.1 | Placement of necessary tools:



- (1)- The graph I added to my project to display the 5-day temperature values of the city selected by the user.
- (2)- The graph I added to my project to display the 5-day humidity values of the city selected by the user.
- (3)- The graph I added to my project to display the 5-day wind speed of the city selected by the user.
- (4)- Button to add the city that the user wants to add to (5) or view its values.
- (5)- Tool I added for the list of cities that the user wants to see the data for.
- (6)- Labels I added to display the instantaneous temperature, humidity, and wind speed data of the city selected by the user.
- (7)- Button used by the user to fetch the data of the selected city. Button to help run the functions.

3.0 | Interface background:

Firstly, I wrote the necessary function codes for the 'add city' button. Then, I wrote the codes connecting the 'fetchFiveDayForecast' function, required for fetching data, and the 'Get Information' button. Lastly, within 'GetInformationButtonPushed', I've written code to process the data I've received into graphs and labels.

3.1 | Add City Button: Callbacks

```
% Button pushed function: AddCityButton
function AddCityButtonPushed(app, event)

    1 AddnewCity = app.EditField.Value; %Figür üzerinde bulun
    2 if ~isempty(AddnewCity) % Dönüştürdüğümüz string'in boş
    3 app.SelectCityDropDown.Items{end+1} = AddnewCity; %Gere

        end
    end
end
```

```
function AddCityButtonPushed(app, event)

    AddnewCity = app.EditField.Value;

    if ~isempty(AddnewCity)

        app.SelectCityDropDown.Items{end+1} = AddnewCity;

    end

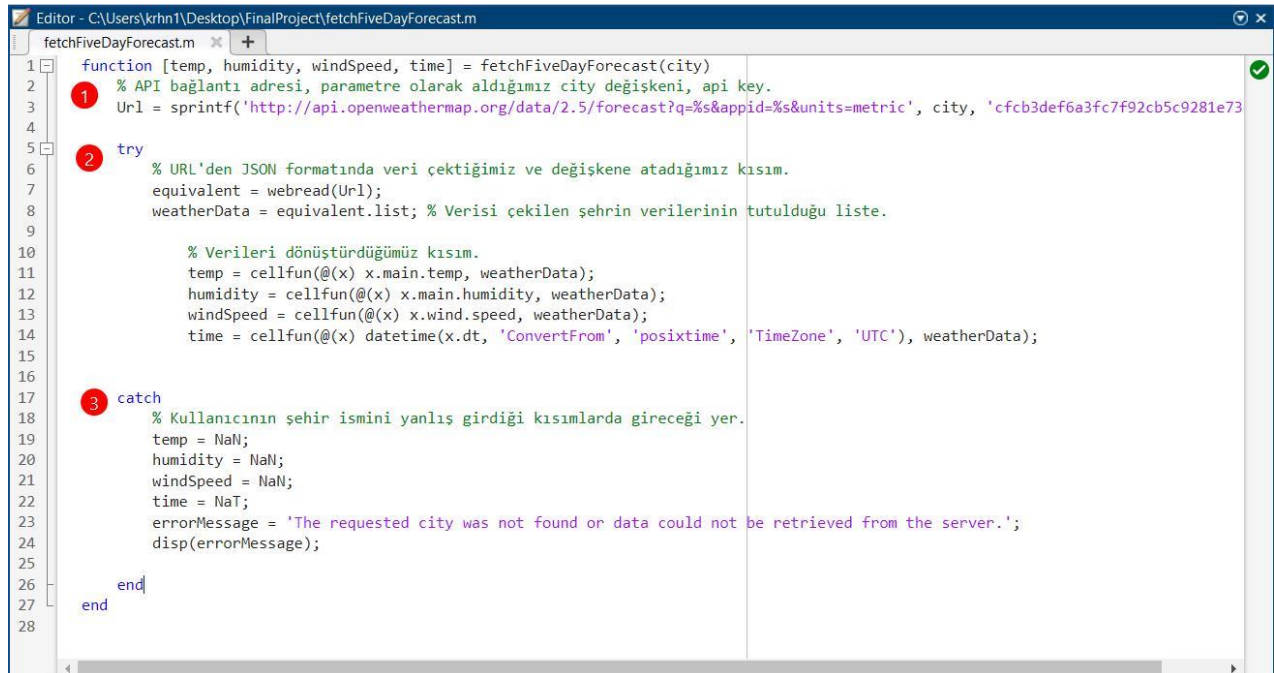
end
```

(1) - It takes the value we write in the app.EditField section on the figure and converts it into a string.

(2) - It checks if the converted string is empty.

(3) - If the necessary conditions are met, it adds the entry to the last row in app.SelectCityDropDown.

3.2 | fetchFiveDayForecast Function:



```
1 function [temp, humidity, windSpeed, time] = fetchFiveDayForecast(city)
2 % API bağlantı adresi, parametre olarak aldığımız city değişkeni, api key.
3 Url = sprintf('http://api.openweathermap.org/data/2.5/forecast?q=%s&appid=%s&units=metric', city, 'cfc3def6a3fc7f92cb5c9281e73');
4
5 try
6 % URL'den JSON formatında veri çektiğimiz ve değişkene atadığımız kısım.
7 equivalent = webread(Url);
8 weatherData = equivalent.list; % Verisi çekilen şehrin verilerinin tutulduğu liste.
9
10 % Verileri dönüştürdüğümüz kısım.
11 temp = cellfun(@(x) x.main.temp, weatherData);
12 humidity = cellfun(@(x) x.main.humidity, weatherData);
13 windSpeed = cellfun(@(x) x.wind.speed, weatherData);
14 time = cellfun(@(x) datetime(x.dt, 'ConvertFrom', 'posixtime', 'TimeZone', 'UTC'), weatherData);
15
16 catch
17 % Kullanıcının şehir ismini yanlış girdiği kısımlarda gireceği yer.
18 temp = NaN;
19 humidity = NaN;
20 windSpeed = NaN;
21 time = NaT;
22 errorMessage = 'The requested city was not found or data could not be retrieved from the server.';
23 disp(errorMessage);
24
25 end
26 end
27
28
```

The first line defines the title of the function. The function named `fetchFiveDayForecast` takes two input parameters: `app` and `city`, and returns four outputs: `temp`, `humidity`, `windSpeed`, and `time`.

(1)- On the 3rd line, data such as temperature, humidity, etc., is fetched from the specified web address using the city selected and the API KEY we wrote.

Firstly, it initiates the try-catch block. The code inside the try block is attempted to be executed, and if an error occurs, the catch block comes into play.

(2)- On the 7th line, we retrieve data from the URL in JSON format and assign it to the `equivalent` variable. We use the `webread` function to fetch data from the API and assign the response in JSON format to the `equivalent` variable. On line 8, we assign the retrieved data for the relevant city to the `weatherData` variable. Lines 11, 12, 13, 14 perform the extraction of data. The `Cellfun` method extracts temperature, humidity, wind speed, and time for each forecast in the `weatherData` list and assigns them to the variables `temp`, `humidity`, `windSpeed`, and `time`, respectively. The `'ConvertFrom'` and `'posixtime'` methods are crucial in the `Time` variable. In this function, for each item in the `weatherData` list, `x.dt` gets the POSIX timestamp and converts it to a `datetime` object using the `datetime` function. This process is done for each item, and the results are stored in the `time` variable.

The catch block runs if any error occurs in the try block.

(3)- When an error occurs, `NaN` and `NaT` values are assigned to the `temp`, `humidity`, `windSpeed`, and `time` variables accordingly. Additionally, an error message is displayed in the `ErrorMessage` label on the interface, and the error message is printed in the MATLAB command window.

```

function [temp, humidity, windSpeed, time] = fetchFiveDayForecast(city)
    % API bağlantı adresi, parametre olarak aldığımız city değişkeni, api key.
    Url = sprintf('http://api.openweathermap.org/data/2.5/forecast?q=%s&appid=%s&units=metric', city,
'cfcb3def6a3fc7f92cb5c9281e73ed90');

    try
        % URL'den JSON formatında veri çektiğimiz ve değişkene atadığımız kısım.
        equivalent = webread(Url);
        weatherData = equivalent.list; % Verisi çekilen şehrin verilerinin tutulduğu liste.

        % Verileri dönüştürdüğümüz kısım.
        temp = cellfun(@(x) x.main.temp, weatherData);
        humidity = cellfun(@(x) x.main.humidity, weatherData);
        windSpeed = cellfun(@(x) x.wind.speed, weatherData);
        time = cellfun(@(x) datetime(x.dt, 'ConvertFrom', 'posixtime', 'TimeZone', 'UTC'),
weatherData);

    catch
        % Kullanıcının şehir ismini yanlış girdiği kısımlarda gireceği yer.
        temp = NaN;
        humidity = NaN;
        windSpeed = NaN;
        time = NaN;
        errorMessage = 'The requested city was not found or data could not be retrieved from the
server.';
        disp(errorMessage);

    end
end

```

3.3 | Get Information Button: Callbacks

The image shows the MATLAB App Designer interface with the 'Code View' tab selected. The code is for a function named 'GetInformationButtonPushed' which is triggered when the 'Get Information' button is pushed. The code is annotated with red circles and brackets indicating different sections of the function:

- 1** [Line 27-29]: FetchFiveDayForecast fonksiyonunu çağırma satırları /28 /29
- 2** [Line 31-36]: Sıcaklık grafiğinin alınan verilere göre çizildiği kısım. Seçilen şehri grafik üzerine formatlı şekilde yazdırdığımız kısım.
- 3** [Line 38-43]: Nem grafiğinin alınan verilere göre çizildiği kısım. Seçilen şehri grafik üzerine formatlı şekilde yazdırdığımız kısım.
- 4** [Line 45-50]: Rüzgar hızı grafiğinin alınan verilere göre çizildiği kısım. Seçilen şehri grafik üzerine formatlı şekilde yazdırdığımız kısım.
- 5** [Line 52-59]: Anlık sıcaklık değerini yazdırdığımız kısım.Formatlı kullanım. Anlık nem değerini yazdırdığımız kısım.Formatlı kullanım. Anlık Rüzgar Hızını yazdırdığımız kısım.Formatlı kullanım.

(1)- It takes the city selected by the user from the SelectCityDropDown component and assigns it to the 'city' variable. This represents the selection in the dropdown menu. After this assignment, the fetchFiveDayForecast function is called, and 'city' is sent as a parameter.

The called fetchFiveDayForecast function returns temperature (temp), humidity (humidity), wind speed (windSpeed), and time information.

(2)- On TemperatureAxes: the plot function draws the required graph using the temp and time data and marks the intersection of the x and y axes with (-o). The Title function displays the title of the graph along with the selected city name. The xlabel and ylabel functions set the labels of the x and y axes, respectively.

(3)- On HumidityAxes: the plot function draws the required graph using the humidity and time data and marks the intersection of the x and y axes with (-x). The Title function displays the title of the graph along with the selected city name. The xlabel and ylabel functions set the labels of the x and y axes, respectively.

(4)- On WindspeedAxes: the plot function draws the required graph using the windspeed and time data and marks the intersection of the x and y axes with (-*)The Title function displays the title of the graph along with the selected city name. The xlabel and ylabel functions set the labels of the x and y axes, respectively.

(5)- It prints the first temperature value to the tempLabel, the first humidity value to the HumidityLabel, and the first wind speed value to the WindSpeedLabel. The sprintf function formats the incoming values into text.

- The %.1f format displays the value with one digit after the decimal point.
- The %.2f format displays the value with two digits after the decimal point.

```
function GetInformationButtonPushed(app, event)
    %Seçilen şehri string olarak alma ve
    %FetchFiveDayForecast fonksiyonunu çağırma satırları /28 /29
    city = app.SelectCityDropDown.Value;
    [temp,humidity,windSpeed,time] = fetchFiveDayForecast(city);

    %Sıcaklık grafiğinin alınan verilere göre çizildiği kısım.
    plot(app.TemperatureAxes, time, temp, '-o');
    %Seçilen şehri grafik üzerine formatlı şekilde yazdırdığımız kısım.
    title(app.TemperatureAxes, sprintf('Temperature (°C) Forecast for %s',city));
    xlabel(app.TemperatureAxes, 'Date');
    ylabel(app.TemperatureAxes, 'Temperature (°C)');

    %Nem grafiğinin alınan verilere göre çizildiği kısım.
    plot(app.HumidityAxes, time, humidity, '-x');
    %Seçilen şehri grafik üzerine formatlı şekilde yazdırdığımız kısım.
    title(app.HumidityAxes, sprintf('Humidity Forecast for %s', city));
    xlabel(app.HumidityAxes, 'Date');
    ylabel(app.HumidityAxes, 'Humidity (%)');

    %Rüzgar hızı grafiğinin alınan verilere göre çizildiği kısım.
    plot(app.WindSpeedAxes, time, windSpeed, '-*');
    %Seçilen şehri grafik üzerine formatlı şekilde yazdırdığımız kısım.
    title(app.WindSpeedAxes, sprintf('Wind Speed Forecast for %s ', city));
    xlabel(app.WindSpeedAxes, 'Date');
    ylabel(app.WindSpeedAxes, 'Wind Speed (m/s)');

    %Anlık sıcaklık değerini yazdırdığımız kısım.Formatlı kullanım.
    app.tempLabel.Text = sprintf('Temperature: %.1f °C', temp(1));

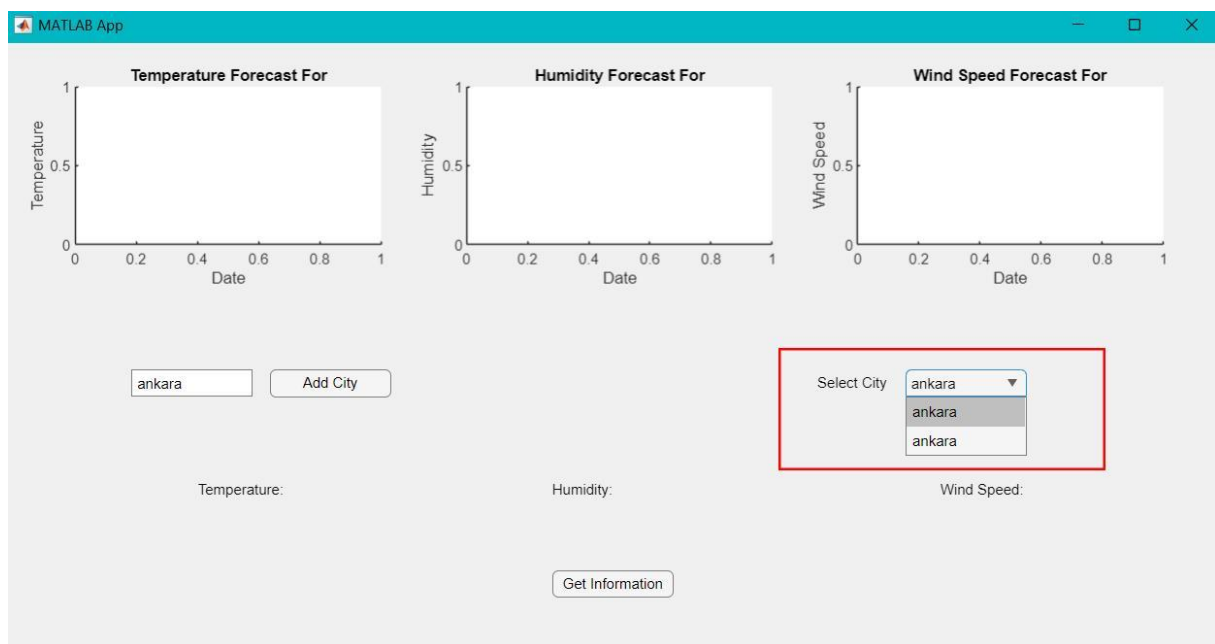
    %Anlık nem değerini yazdırdığımız kısım.Formatlı kullanım.
    app.HumidityLabel.Text = sprintf('Humidity: %.1f%%', humidity(1));

    %Anlık Rüzgar Hızını yazdırdığımız kısım.Formatlı kullanım.
    app.WindSpeedLabel.Text = sprintf('Wind Speed: %.2f m/s', windSpeed(1));
end
```

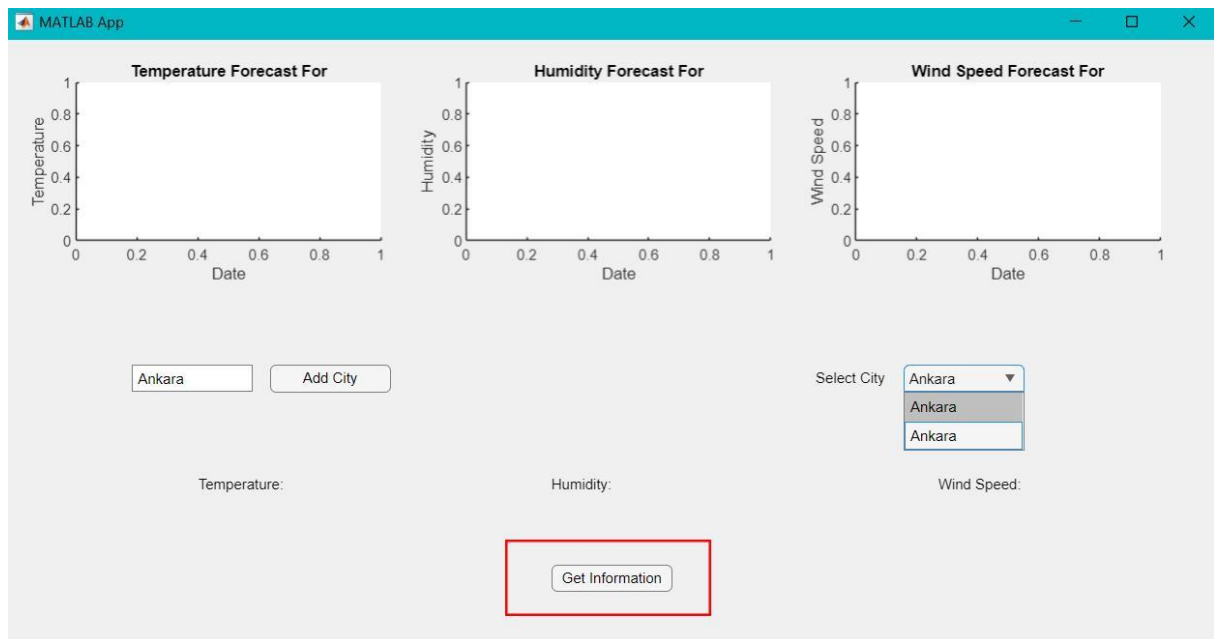
4.0 | Operation Screenshots of the Application:



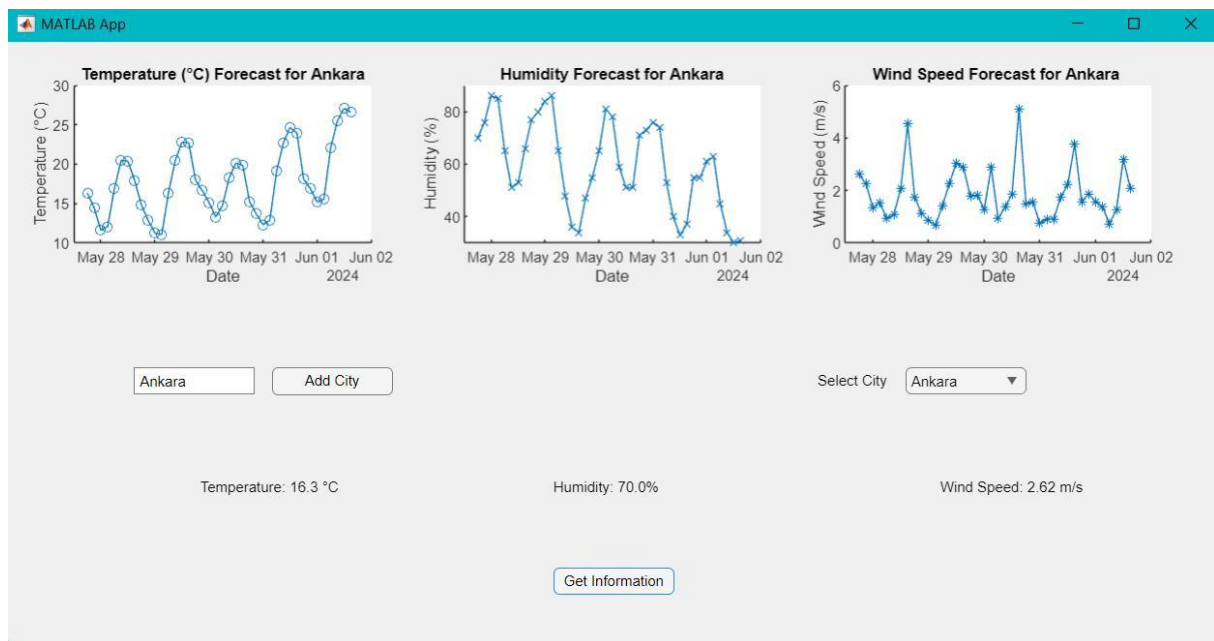
"First, we use the 'Add City' button. We type the city whose data we want to see and click the 'Add City' button."



"With the 'Add City' button, we can view and select the cities that the user has added in the 'Select City' section. We select Ankara, which we added in the previous step."



"After completing the previous steps, we can now press the 'Get Information' button to fetch the data."



"After pressing the 'Get Information' button, the result looks like this. The data has been successfully fetched and processed without any issues."



The requested city was not found or data could not be retrieved from the server.
fx >>

"If the user enters an incorrect city name, the application returns an error message in the console, and the graphs remain empty. Additionally, the real-time temperature, humidity, and wind speed data return NaN values."