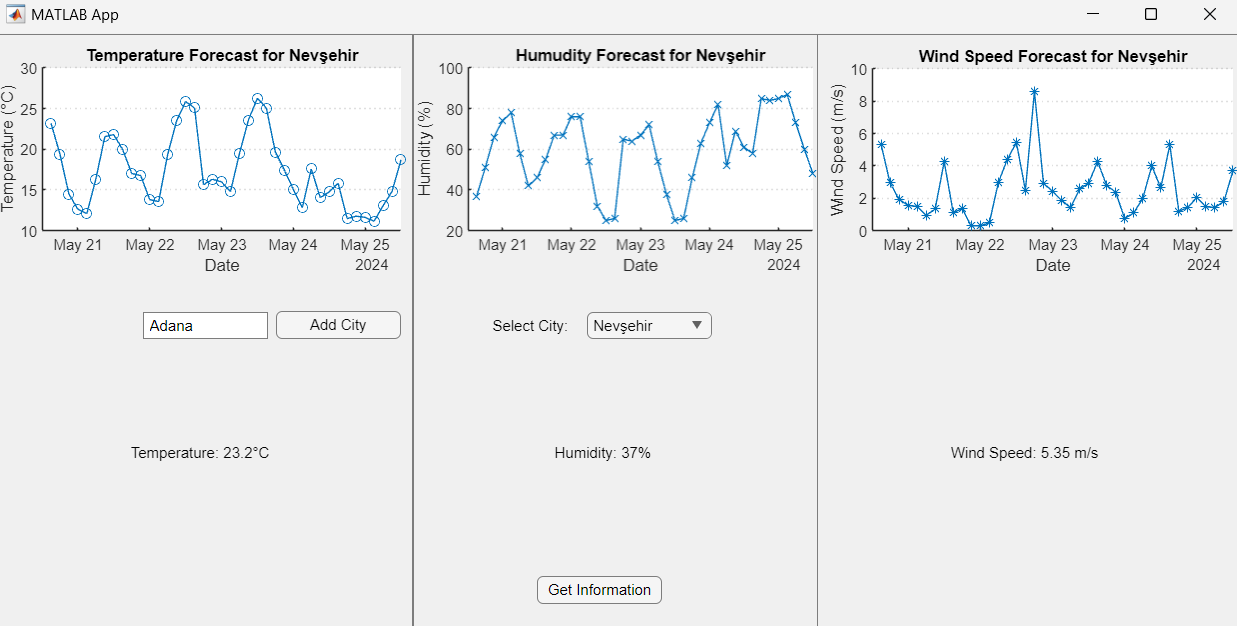
**Project Report:** “WeatherScope”: City-Based Weather Forecasting and Visualization Application

**Overview:**

****

**Description of the Steps within the Project:**

1. **Create axes, buttons, dropdown menus and labels for GUI.**

This code automatically generated by matlab. Not necessary to share.

1. **Write a function for fetching the forecast data through API.**

Code:

function forecastData = fetchFiveDayForecast(app, cityName)

apiKey = '023a7640a0b0fb3b619c06ab9af9ee66';

% Convert string to character array if necessary

if isstring(cityName)

cityName = char(cityName);

end

% Create the API URL

apiUrl = ['http://api.openweathermap.org/data/2.5/forecast?q=', cityName, '&appid=', apiKey];

response = NaN;

% Use try-catch to see city name is valid or not

try

% Send the HTTP GET request

response = webread(apiUrl);

catch

warning = sprintf('There is no city named "%s"', cityName);

uialert(app.UIFigure, warning, 'City Input Error');

end

% Since the response is already a JSON structure, assign it directly

forecastData = response;

end

1. **Write functions for extracting datas like temperatures, humidity and wind speed from forecast data, and prepare for use.**

Code:

% Helper function to extract temperature data

function temperatures = extractTemperatures(app, forecastData)

% Initialize arrays to hold the dates and temperature values

dates = datetime([], 'ConvertFrom', 'posixtime');

values = [];

% Loop through the forecast data to extract temperatures

for i = 1:numel(forecastData.list)

% Use try-catch to avoid brace errors

try

% Convert the date from Unix time to datetime

date = datetime(forecastData.list{i}.dt, 'ConvertFrom', 'posixtime');

dates(end+1) = date;

% Extract the temperature value in Kelvin

tempValueKelvin = forecastData.list{i}.main.temp;

% Convert the temperature from Kelvin to Celsius

tempValueCelsius = tempValueKelvin - 273.15;

catch

% Convert the date from Unix time to datetime

date = datetime(forecastData.list(i).dt, 'ConvertFrom', 'posixtime');

dates(end+1) = date;

% Extract the temperature value in Kelvin

tempValueKelvin = forecastData.list(i).main.temp;

% Convert the temperature from Kelvin to Celsius

tempValueCelsius = tempValueKelvin - 273.15;

end

values(end+1) = tempValueCelsius;

end

% Return a structure with the dates and values

temperatures.dates = dates;

temperatures.values = values;

end

% Helper function to extract humidity data

function humidity = extractHumidity(app, forecastData)

% Initialize arrays to hold the dates and humidity values

dates = datetime([], 'ConvertFrom', 'posixtime');

values = [];

% Loop through the forecast data to extract humidity

for i = 1:numel(forecastData.list)

% Use try-catch to avoid brace errors

try

% Convert the date from Unix time to datetime

date = datetime(forecastData.list{i}.dt, 'ConvertFrom', 'posixtime');

dates(end+1) = date;

% Extract the humidity value

humidityValue = forecastData.list{i}.main.humidity;

catch

% Convert the date from Unix time to datetime

date = datetime(forecastData.list(i).dt, 'ConvertFrom', 'posixtime');

dates(end+1) = date;

% Extract the humidity value

humidityValue = forecastData.list(i).main.humidity;

end

values(end+1) = humidityValue;

end

% Return a structure with the dates and values

humidity.dates = dates;

humidity.values = values;

end

% Helper function to extract wind speed data

function windSpeed = extractWindSpeed(app, forecastData)

% Initialize arrays to hold the dates and wind speed values

dates = datetime([], 'ConvertFrom', 'posixtime');

values = [];

% Loop through the forecast data to extract wind speeds

for i = 1:numel(forecastData.list)

% Use try-catch to avoid brace errors

try

% Convert the date from Unix time to datetime

date = datetime(forecastData.list{i}.dt, 'ConvertFrom', 'posixtime');

dates(end+1) = date;

% Extract the wind speed value

windSpeedValue = forecastData.list{i}.wind.speed;

catch

% Convert the date from Unix time to datetime

date = datetime(forecastData.list(i).dt, 'ConvertFrom', 'posixtime');

dates(end+1) = date;

% Extract the wind speed value

windSpeedValue = forecastData.list(i).wind.speed;

end

values(end+1) = windSpeedValue;

end

% Return a structure with the dates and values

windSpeed.dates = dates;

windSpeed.values = values;

end

1. **Write a function for dropdown menu thus entered city name can be selected.**

Code:

% Button pushed function: AddCityButton

function onAddCityButtonPressed(app, event)

% Get the new city name from the Edit Field

newCity = app.CityEditField.Value;

% Check if the new city is not empty and not already in the dropdown

if ~isempty(newCity) && ~any(strcmp(app.SelectCityDropDown.Items, newCity))

% Check if the city name is valid

forecastData = fetchFiveDayForecast(app, newCity);

if ~isempty(forecastData) && isfield(forecastData, 'list') && isfield(forecastData, 'city')

% Add the new city to the dropdown list

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

end

else

% Display an error message if the city is empty or already exists

uialert(app.UIFigure, 'Please enter a valid city name that is not already in the list.', 'City Input Error');

end

end

1. **Write a function for button to get information about the selected city.**

Code:

% Button pushed function: GetInformationButton

function onGetInformationButtonPressed(app, event)

% Select the new city in the dropdown

city = app.SelectCityDropDown.Value;

if ~isempty(city)

% Fetch and display the forecast data for the new city

forecastData = fetchFiveDayForecast(app, city);

updateGraphs(app, forecastData);

displayCurrentConditions(app, forecastData);

else

uialert(app.UIFigure, 'Selected city can not be blank!', 'Selected City Error');

end

end

1. **Write a function for updating graphs. Show 5 days forecast for temperature, humudity and wind speed for selected city.**

Code:

function updateGraphs(app, forecastData)

% Extract temperature, humidity, and wind speed data

temperatures = extractTemperatures(app, forecastData);

humidity = extractHumidity(app, forecastData);

windSpeed = extractWindSpeed(app, forecastData);

% Update the Temperature graph

plot(app.TemperatureAxesUI, temperatures.dates, temperatures.values, '-o');

xlabel(app.TemperatureAxesUI, 'Date');

ylabel(app.TemperatureAxesUI, 'Temperature (°C)');

% Update the Humidity graph

plot(app.HumidityAxesUI, humidity.dates, humidity.values, '-x');

xlabel(app.HumidityAxesUI, 'Date');

ylabel(app.HumidityAxesUI, 'Humidity (%)');

% Update the Wind Speed graph

plot(app.WindSpeedAxesUI, windSpeed.dates, windSpeed.values, '-\*');

xlabel(app.WindSpeedAxesUI, 'Date');

ylabel(app.WindSpeedAxesUI, 'Wind Speed (m/s)');

end

1. **Write a function for displaying current conditions of the selected city.**

Code:

% Fuction to display current conditions

function displayCurrentConditions(app, forecastData)

% Use try-catch to avoid brace errors

try

% Extract the current conditions from the first entry of the forecast data

currentConditions = forecastData.list{1}.main;

currentWind = forecastData.list{1}.wind;

catch

% Extract the current conditions from the first entry of the forecast data

currentConditions = forecastData.list(1).main;

currentWind = forecastData.list(1).wind;

end

% Substract 273.15 to convert kelvin to celsius

currentTemperature = currentConditions.temp - 273.15;

currentHumidity = currentConditions.humidity;

currentWindSpeed = currentWind.speed;

% Update the labels to display the current conditions

app.CurrentTemperatureLabel.Text = sprintf('Temperature: %.1f°C', currentTemperature);

app.CurrentHumidityLabel.Text = sprintf('Humidity: %d%%', currentHumidity);

app.CurrentWindSpeedLabel.Text = sprintf('Wind Speed: %.2f m/s', currentWindSpeed);

% Update Graph titles

city = app.SelectCityDropDown.Value;

app.TemperatureAxesUI.Title.String = sprintf('Temperature Forecast for %s', city);

app.HumidityAxesUI.Title.String = sprintf('Humudity Forecast for %s', city);

app.WindSpeedAxesUI.Title.String = sprintf('Wind Speed Forecast for %s', city);

end

1. **Make sure to all potential errors are handled. Be careful about empty strings, invalid strings etc.**

This code embedded into other functions.

**Discussion of Potential Improvements within the Project:**

Future improvements for this project

* We can use more variables about weather forecasting like the felt temperature.
* We can use more readable graphs. We can use more colors in our graphs,
* We can store the previous temperatures to create a archive.
* We can develop a different GUI and view two or more cities at the same time.