LGC Android Tech Task

For this Task you will be creating the LetsGetWeather application. Below is an overview of features/requirements that this app will need to have implemented to be considered complete.

API Access:

For this sample application you should use APIs from the following source - https://www.weatherapi.com/. Upon signing up you will be granted a free api key (valid for 2 weeks).

- An example of one such API call for grabbing weather data would be https://api.weatherapi.com/v1/current.json?Key={API-KEY}&q=Dublin. This HTTP GET request returns data about Dublin's weather.
- Another example, this one fetches data for Dublin on a specified date https://api.weatherapi.com/v1/forecast.json?Key={API-KEY}&q=Dublin&dt={yyyy-MM-dd}

For a full breakdown of API's you can utilize with your trial account see here.

LetsGetWeather Requirements:

The LGC weather app should allow a user to **input/select** the name of a given location in Ireland. Upon selecting a location an action should be executed (button click etc) fetching weather information for the provided location.

Should the API call be successful, and data is returned. This data should be rendered on screen giving our users a valuable overview of weather conditions in the given location. (Not all data needs to be rendered - **minimum of 3 view elements**).

The user should be able to search as **many times as they like**, and the UI should reflect these changes.

On the same screen we should also provide our users with an **additional button**, this button should only appear if the API call was successful and data was returned. This button should **redirect** the user to a new screen.

Upon arrival to this screen **another API call** should be executed to get additional information about the **provided location** in the previous call. (Feel free to choose the API you hit at this point from the swagger documentation (https://app.swaggerhub.com/apis-docs/WeatherAPI.com/WeatherAPI/1.0.2#/APIs/realtime-weather))

Additional:

- Application should be written in Kotlin.
- The application should handle API call failures & unexpected results gracefully.
- At least one unit test should be written.
- If external dependencies are added, please include a description of why & where it has been used.
- A document should also be prepared explaining the architecture & design decisions made during development.
- On submission of this tech task, a functional APK should also be provided.
- Access to the repo of your choosing should be granted to myself (ploughran@letsgetchecked.com)

This is the minimum required. Feel free to have fun & impress.