Weather App - Radius Mobile Assignment

Objective:

- The assignment's main objective is to create a weather app based on the provided Figma design and use the WeatherAPI to fetch the weather data.
- **Empty State**
    - When the app is opened for the first time, show the start screen with an empty state as per the design.
- **Search for a location**:
    - User should be able to search for a City or a region
    - The user should be able to select the current location option. When the user taps on the current location, use the CoreLocation (for iOS) or Android Location API (for Android) to get the latitude and longitude. Use the latitude and longitude to fetch the weather data (WeatherAPI supports lat, long)
    - To get the location suggestions, use Search Suggestions API from the WeatherAPI.
- **Weather Screen**
    - Once the desired location is selected, show the detailed weather of the location by fetching the weather data from the WeatherAPI. The weather data should be presented as per the design. **Explore the WeatherAPI documentation and use the appropriate API(s)** to fetch and display the data**.**
    - **Persist the location**: When the "+ Add" button is tapped in the Weather detailed screen, **persist the data locally** and show them as a list in the start screen. Users should be able to tap on the location the next time when they visit the app and go to the detailed weather screen. Users can use the "Remove" button to remove that location from the locally persisted location list.
- **Attention to detail**: Replication of exact designs during the development.

API

- Go to https://weatherapi.com and signup to get the free API Key.
- Detailed documentation at https://www.weatherapi.com/docs/ for the all the APIs. Go through the documentation and use the appropriate APIs.

Assets

- Assignment UI reference: Figma design link

- For all weather forecast-related icons, either use the URLs of the image directly from the API response or download the icons from here and use weather conditions API data to map the local icons with the API (Suitable for Light / Dark mode differentiation)
- For icons used in the app, use SF Symbols (for iOS) / Material Icons (for Android). Here is the list of icons and system name of the icons that were used in the Figma designs:

Screen	Icon	SF Symbols	Material Icon
Start screen	Empty state cloud icon	cloud.sun.rain.fill	weather_mix
Search Screen	Search icon	magnifyingglass	search
	Search Cancel icon	xmark.circle.fill	close
	Current Location icon	location	near_me
Weather Screen	Back button icon	arrow.backward	arrow_back
	Delete button icon	trash	delete
	Add - Plus icon	plus	add
	Highest Temperature: Arrow up	arrow.up	arrow_upward
	Lowest Temperature: Arrow down	arrow.down	arrow_downward
	5-day forecast icon	calendar	calendar_month
	Precipitation (Rain) icon	cloud.rain	rainy
	Wind icon	wind	air
	UV index icon	sun.min	wb_sunny
	Sun icon	sun.max	sunny
	Sunrise icon	sunrise	clear_day
	Sunset icon	sunset	night
	Alerts bell icon	bell	notifications

Expectations:

iOS:

- Use SwiftUI / UIKit to develop the UI of the app.
- Use MVVM architecture pattern to bind the data with the UI.
- Use of SOLID principles and highly scalable code.
- Free to use any third-party library, and justify your usage.
- **Bonus points:**
    - Handling Light / Dark mode. (For dark weather icons, refer Assets section)
    - Showing the data in the Metric / Imperial units based on the user's device settings.
    - Error handling at all suitable places.

Android:

- Using kotlin and its properties.
- MVVM with CLEAN architecture.
- Using appropriate use of coroutines and threads wherever it is necessary for asynchronous programming.
- Using flows / live-data.
- Use as many components of Android (activity, fragment, services, etc.) as possible.
- Make use of Data Binding with views / Compose screen, Realm / RoomsDB, Retrofit, Dagger / Hilt for the respective usage in code.
- **Bonus Points :**
    - Using Jetpack compose with jetpack navigation.
    - Theming app (light and dark theme) with material 3 guidelines.
    - Screen rotation compatibility. (with orientation-specific UI changes)
    - Complete use of kotlin (including DSL).
    - Error handling.

Submission:

Once the assignment is completed, make a zip of the file and email the same to hiring@radiusagent.com. Don't upload the code or assignment anywhere in the public domain.

For questions regarding the assignment, email mobile@radiusagent.com **cc'ing** hiring@radiusagent.com