Forecasting 48-hour Air Quality Information at any location on the Map* using PRAISE-HK Web Service

* *Map inside Hong Kong only*

[***Note: this is not a basic level App Inventor tutorial. Readers are advised to write one or two basic level apps before trying this one.***]

PRAISE-HK[1] API is a web service for providing air quality information with high data-density[2] (down to street level) and high accuracy[3], and is able to provide 48-hour forecasted air quality information.

To access PRAISE-HK data, our team has developed an "extension[4]" - namely "PRAISE_HK_web" for users to access our data. This tutorial aims to demonstrate how to call/acquire specific air quality information based on the location and time chosen by the app user.

Okay, if you are ready, let's get started!
First of all, let us start a new project by naming it "GetAirQuality_fromMap".

Step 1. User interface(UI) design

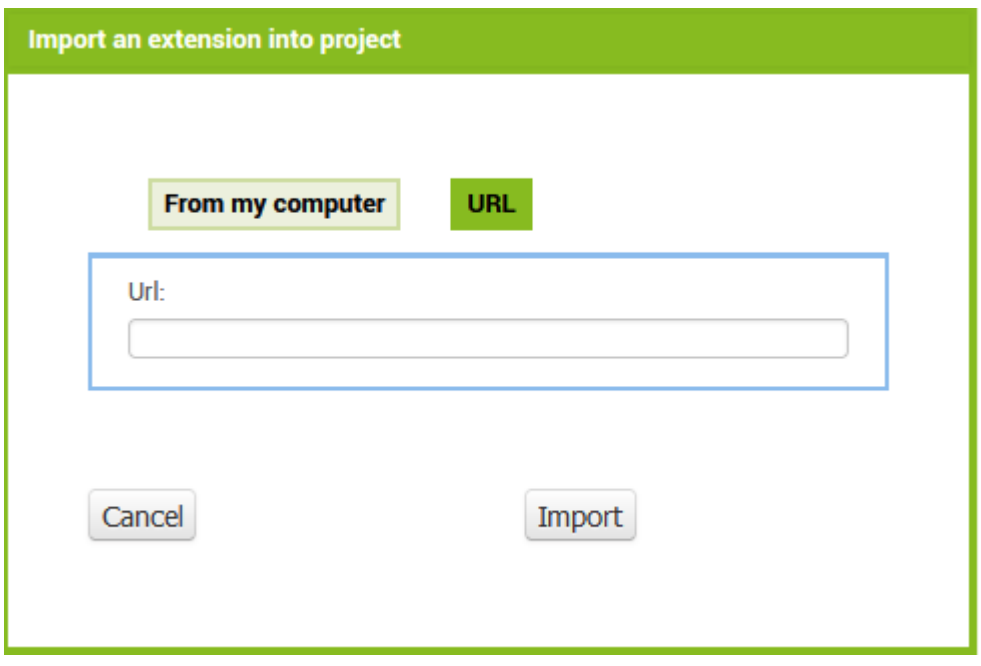
After creating the project, we are automatically in the "Designer" tab.

1.1. Non-visible components:

First we have to import the PRAISE_HK_web extension.

Go to the 'Palette' ➔ 'Extension' and click 'import extension'

Then, from the box popped up, click the 'URL' button, and then input the following URL -- https://envf.ust.hk/~stcheng/PRAISE_HK_web/hk.ust.praise.web.aix into the 'Url' textbox to import the extension.



If the import is successful, the 'Extension' section will turn into this:

Now, you should drag this onto the phone (like other components). And the extension is ready for use.

Next, we drag other non-visible components in the "Palette" column to the phone too.

---

[1] **PRAISE-HK** is a short form for "Personalized Real-time Air-quality Informatics System for Exposure in HK" with the project goal to empower the public with personalized air quality information.

[2] **High data-density:** PRAISE-HK is able to provide air quality (and associated health risk) information up to 2-meter resolution.

[3] **How accurate are PRAISE-HK predictions compared with data from Hong Kong's air quality monitoring stations?**

[4] **What is an "extension" here?** An "extension" provides app developers additional information/components for advanced and extended app features. Please refer to the following to the following article for details.

1. ‘User Interface’ ➔ ’Notifier’ (*this component displays various pop-up alert messages when the app needs to alert the user for some reason*)
2. ‘Connectivity’ ➔ ’Web’ (*this component enables the app to get/send data from/to the Web*)
3. ‘Sensors’ ➔ ‘Clock’ (*this component provides functionality of a clock*)

The Non-visible components (at the bottom of the phone) should look like the following when completed:

1.2. Visible components:

1.2.1. Top Area

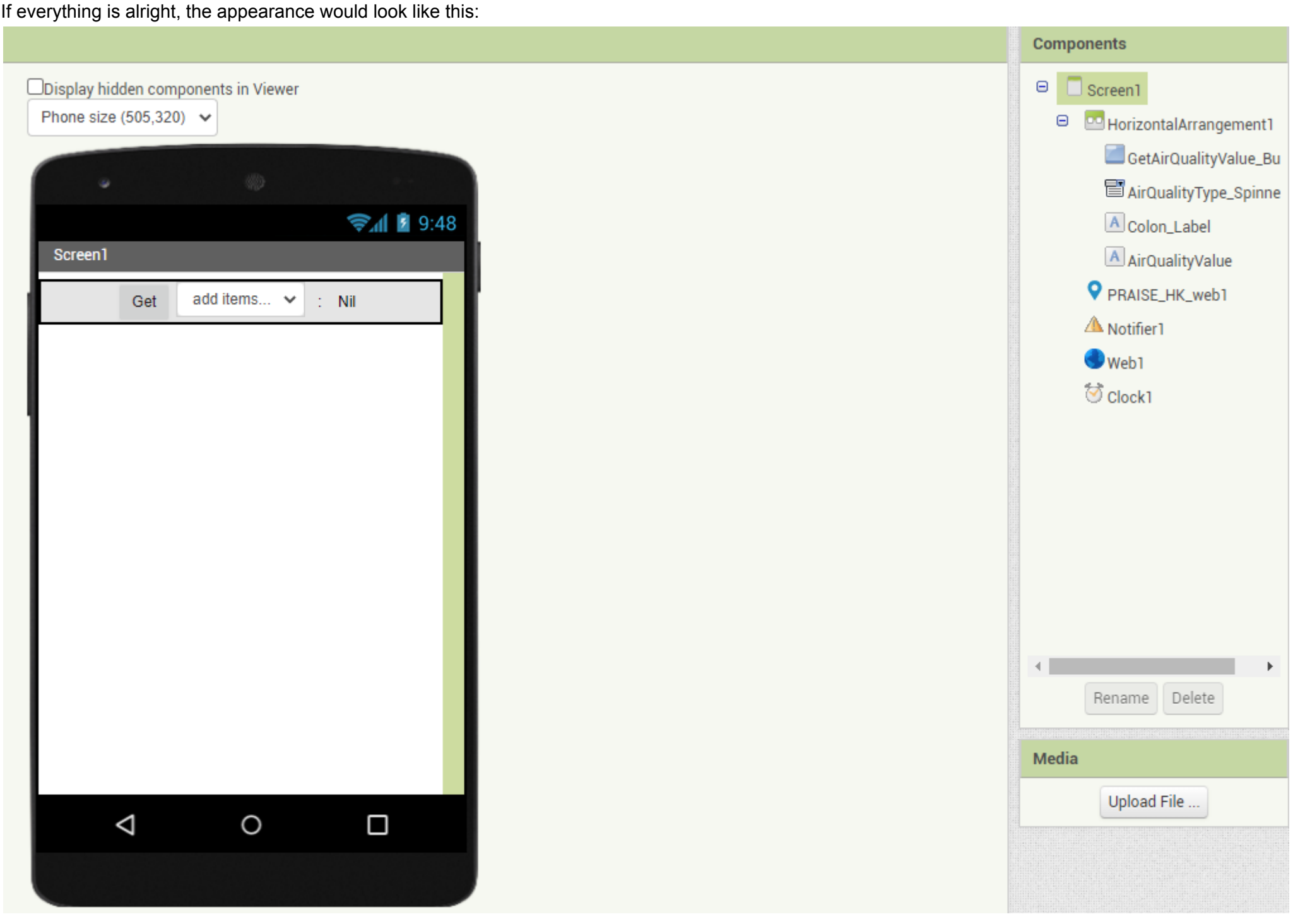
In this area, we drag and drop the various visual components onto “Screen1”, and set the “Properties” as suggested in the following table:

Component Type[1]	***Inside Which Component***	***Follow Which Component***	***Component Name[2]***	***Change Which Properties of the Component***
Layout / HorizontalArrangement	Screen1		HorizontalArrangement1	**AlignHorizontal:** Center **AlignVertical:** Center **Width:** Fill parent
User Interface / Button	HorizontalArrangement1		GetAirQualityValue_Button	**Text:** Get
User Interface / Spinner	HorizontalArrangement1	GetAirQualityValue_Button	AirQualityType_Spinner	**ElementsFromString:** AQHI,%AR,PM10,PM2.5,NO2,O3,SO2
User Interface / Label	HorizontalArrangement1	AirQualityType_Spinner	Colon_Label	**Text:** :
User Interface / Label	HorizontalArrangement1	Colon_Label	AirQualityValue	**Text:** Nil

Annotation —

1. Component Type are items in the “Palette” column
2. if needed, click the “Rename” button in the “Components” column to change the name of a particular component

If everything is alright, the appearance would look like this:

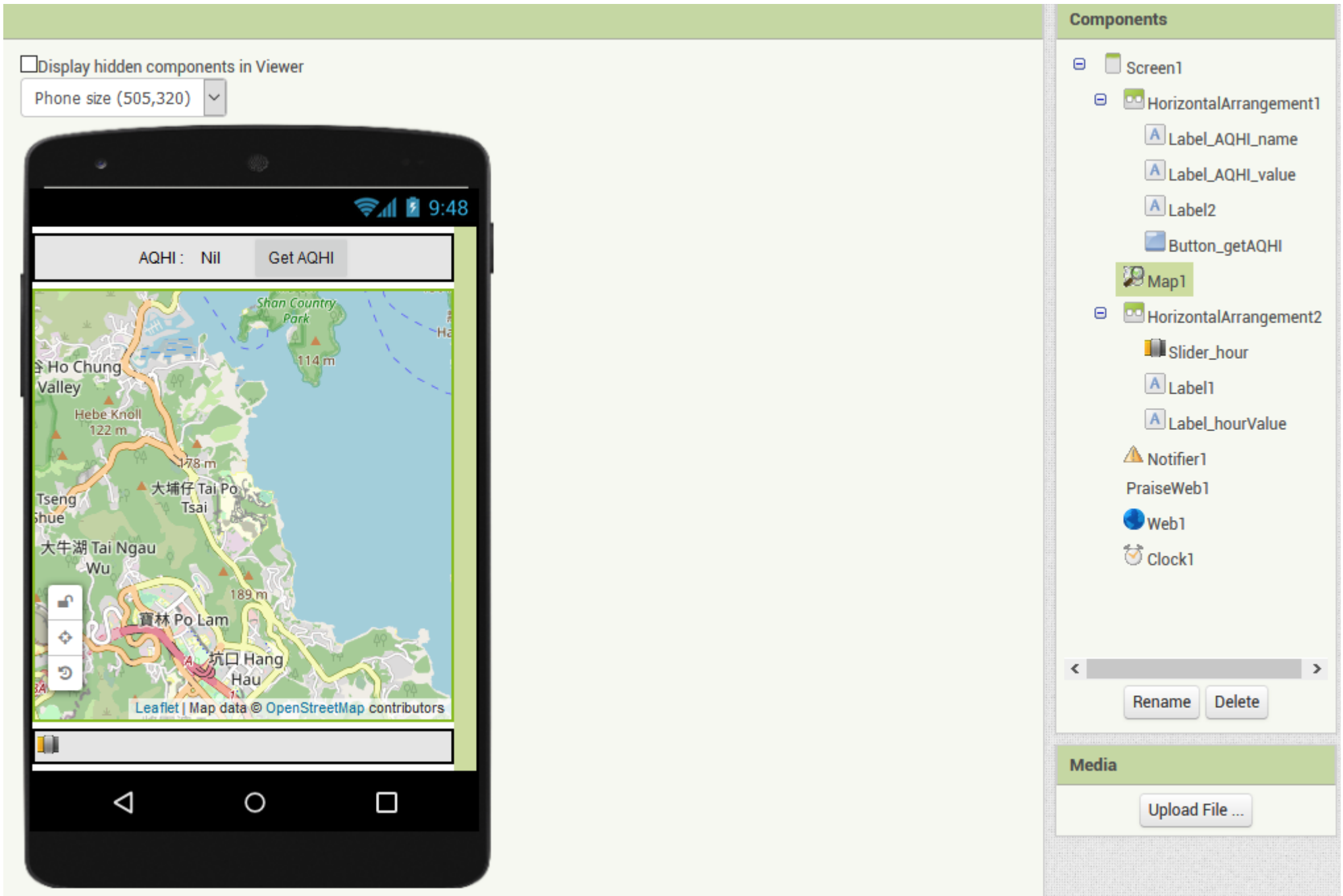


1.2.2. Middle Area

In this area, we just need to drag and drop a map component onto “Screen1”, under the top area, and set the “Properties” as suggested in the following table:

Component Type	***Inside Which Component***	***Follow Which Component***	***Component Name***	***Change Which Properties of the Component***
Maps / Map	Screen1	HorizontalArrangement1	Map1	**CenterFromString:** 22.336028,114.265461 **Height:** Fill parent **Width:** Fill parent

If everything is alright, the appearance would look like this:



1.2.3. Bottom Area

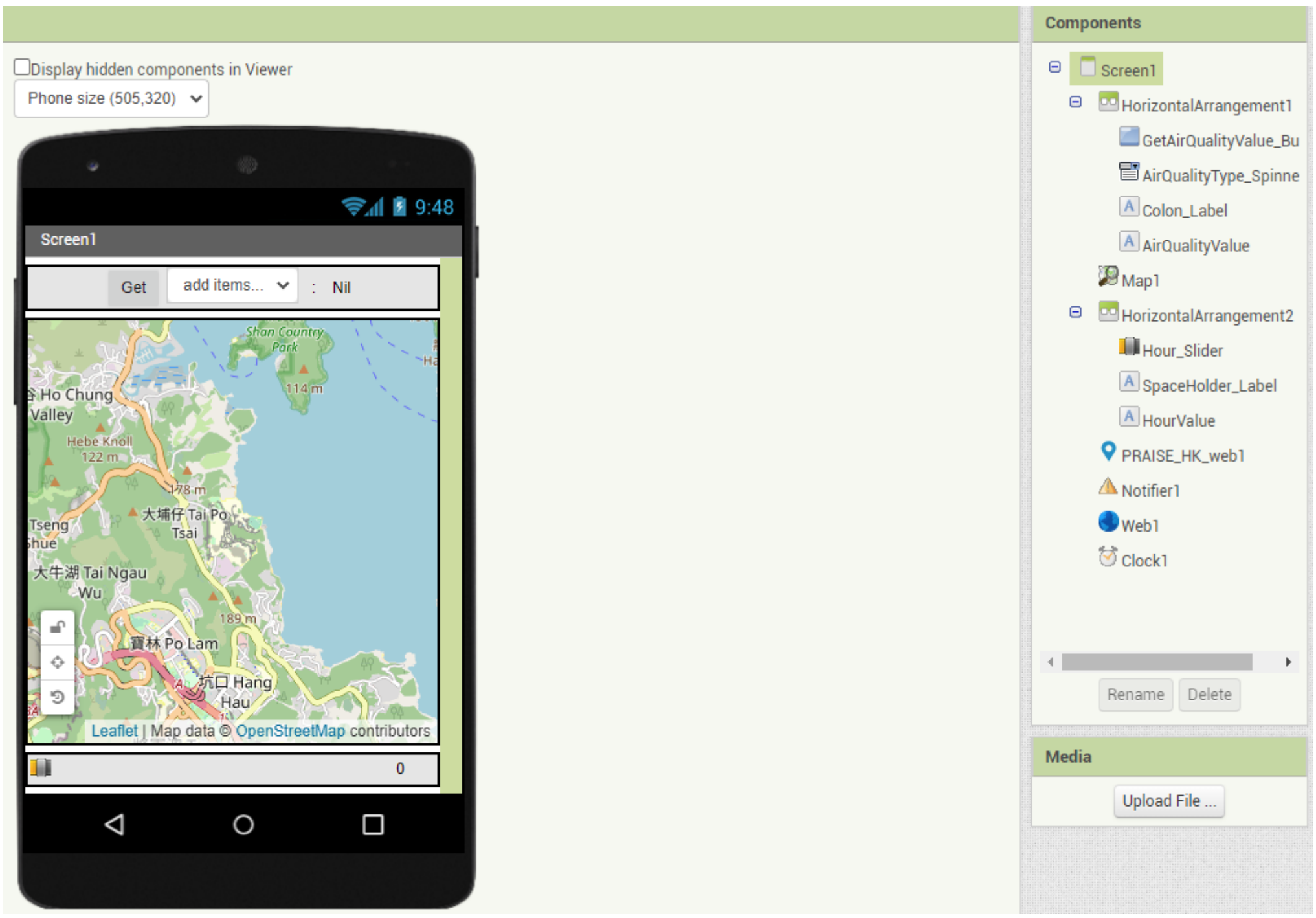
In this area, we drag and drop the various visual components onto “Screen1”, under the middle area, and set the “Properties” as suggested in the following table:

Component Type[1]	***Inside Which Component***	***Follow Which Component***	***Component Name***[2]	***Change Which Properties of the Component***
Layout / HorizontalArrangement	Screen1		HorizontalArrangement2	**AlignVertical:** Center **Width:** Fill parent
User Interface / Slider	HorizontalArrangement2		Hour_Slider	**Width:** Fill parent **MaxValue**: 48 **MinValue:** 0 **ThumbPosition:** 0
User Interface / Label	HorizontalArrangement2	Hour_Slider	SpaceHolder_Label	**Width:** 10 pixels **Text:** *{(empty)}*[1]
User Interface / Label	HorizontalArrangement2	SpaceHolder_Label	HourValue	**Width:** 30 pixels **Text:** 0

Annotation --

1. *{(empty)}* means the property is really empty, devoid of any content.

If everything is correct, the interface will look like this:



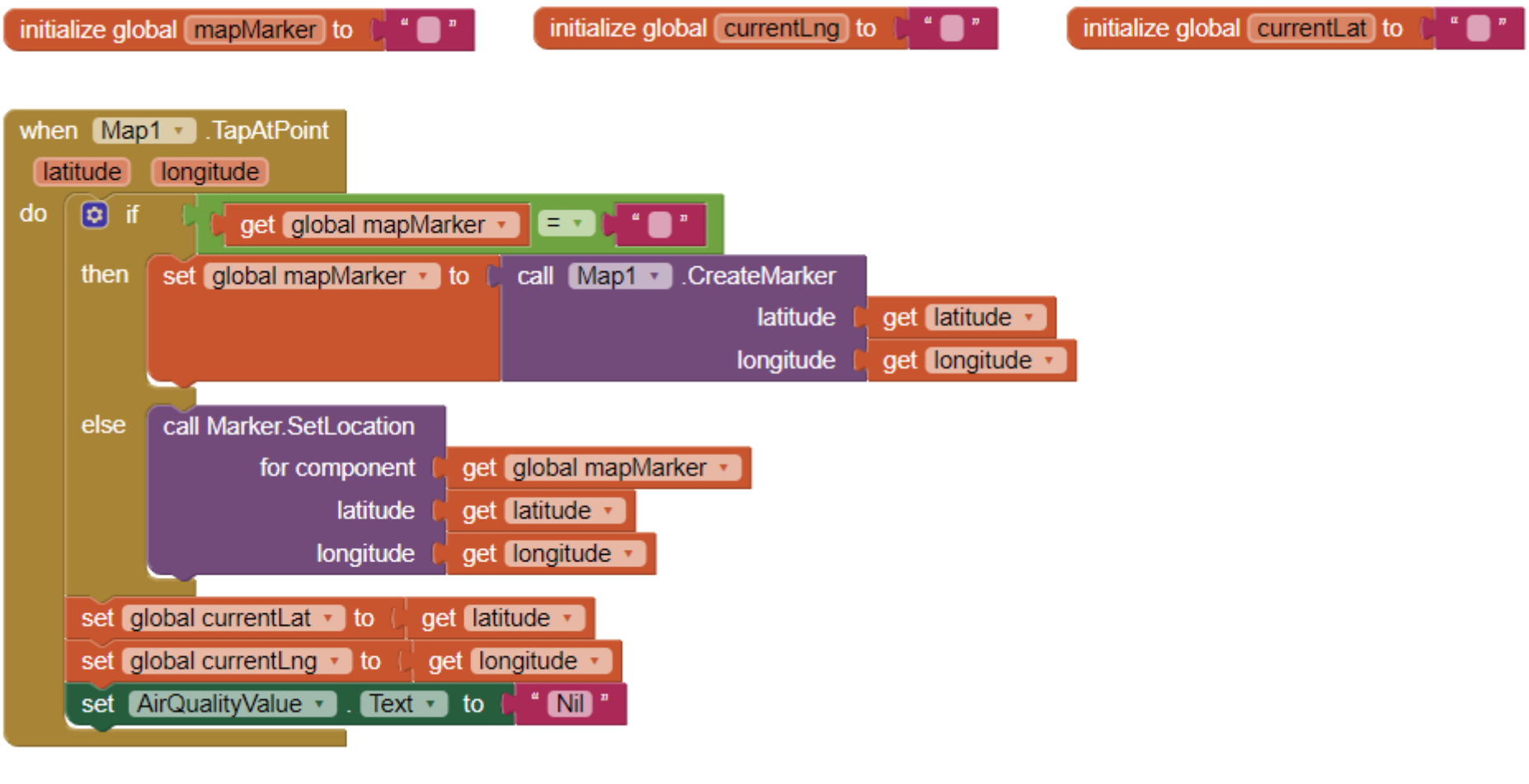
The UI is now completed. Next step will be the implementation of the behaviour.

Step 2. Behaviour Implementation (Blocks Building)

First switch to the Blocks Editor.

2.1. Map1 (Midde Area)

We use the map to get the location, so we assign the following block structure to the “TapAtPoint” event handler of the “Map1” (as there are global variables inside this structure, we thus need to initialize them first) :



Explanation:

The global variables are used as follows:

- “currentLat” is used to store the current Latitude chosen by the user
- “currentLng” is used to store the current Longitude chosen by the user
- “mapMarker” is used to store the map’s marker component

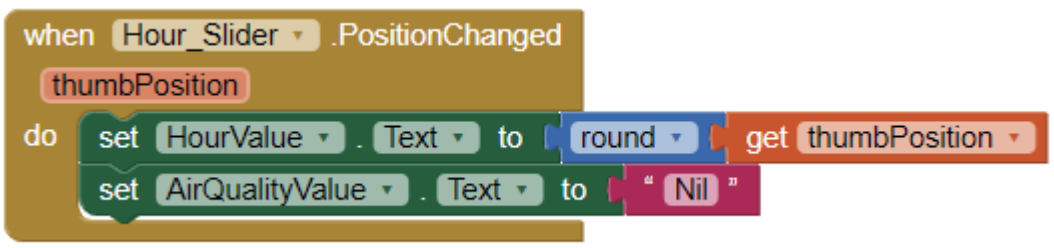
At the beginning, all are set to an empty string, meaning that nothing is stored in them.

If the user taps any point on the map, the above “Map1.TapAtPoint” event handler will be run, and passed in the latitude and the longitude value of that point. If currently there is no marker set, it will be created using the “Map1.CreateMarker” method and then assign it to the “mapMarker”. Otherwise, the marker will be relocated using the “Marker.SetLocation” method. So that a marker will always appear at the point the user taps.
Also inside the handler, the “currentLat” and “currentLng” are assigned the values of latitude and longitude respectively. These two global values will be used in the other part of the App.

Finally, reset “AirQualityValue.Text” back to “Nil”, as any new tap on the map will invalidate that value.

2.2. Hour_Slider (Bottom Area)

We use the slider to get the time (maximum 48 hours to the future). And the block structure is as follows:



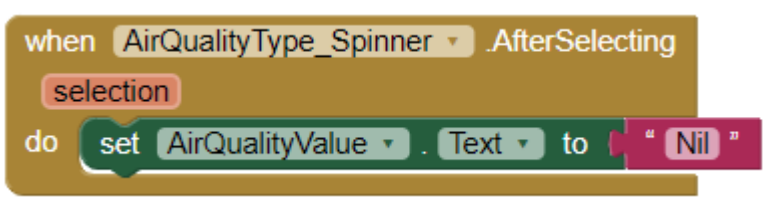
Explanation:

When the slider’s thumb is dragged to a new position, the slider’s “PositionChanged” event handler will be called. This event handler passes in a “thumbPosition”, telling where the thumb’s position is. The rounded (to an integer, because PRAISE-HK service only forecasts to integral hour value) value of the thumb's position is used to set the “HourValue.Text” property (noted: this property is what the “HourValue” component displays on the UI). This property will also be referenced when we send requests to the PRAISE-HK service.

Also, reset “AirQualityValue.Text” back to “Nil”, as any new position on the slider (i.e. a new time instant) will invalidate that value.

2.3. AirQualityType_Spinner (Top Area)

We use the spinner to choose the type of air quality information we want to view. Again, we store the selection to a global variable for use in another event handler. The block structure is as follows:

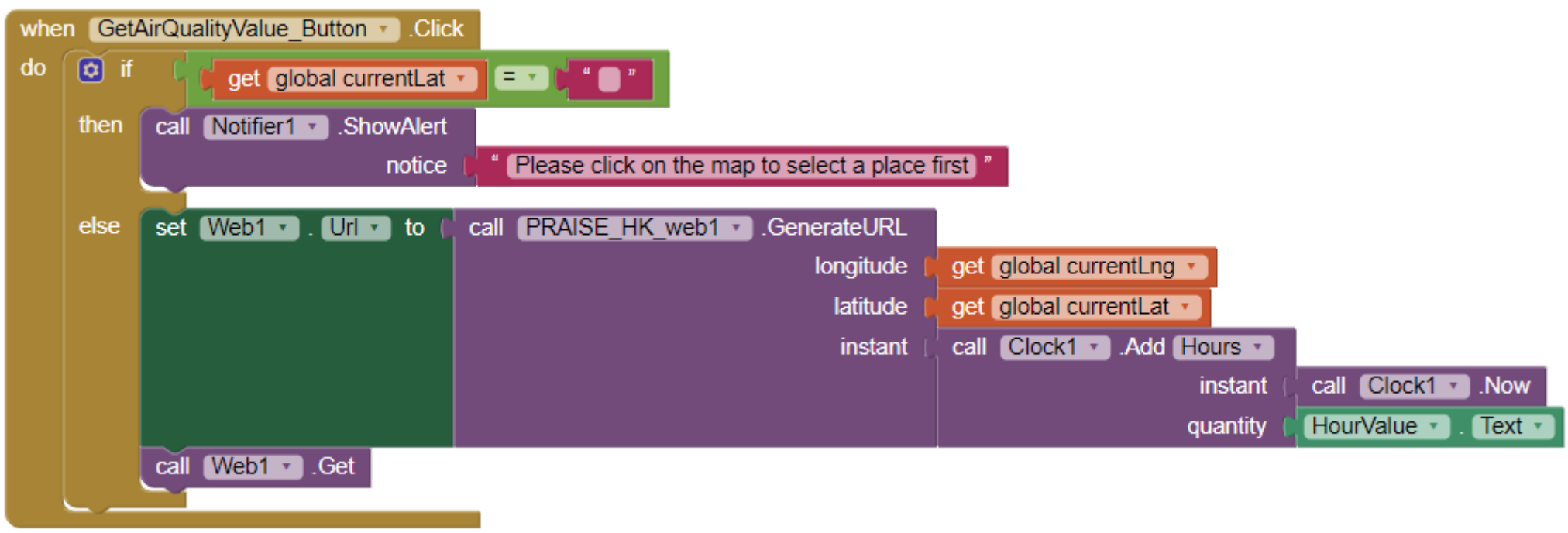


Explanation:

When an item inside the spinner is selected, the spinner's "AfterSelecting" event handler will be called. Within the handler, reset "AirQualityValue.Text" back to "Nil", as any new selection on the spinner will invalidate that value.

2.4. GetAirQualityValue_Button (Top Area)

As the name of the aforementioned button implies, we want to get the air quality information after clicking it. In order to do this, we have to send a web request to the PRAISE-HK web service. So, we build the block structure as follows:



Explanation:

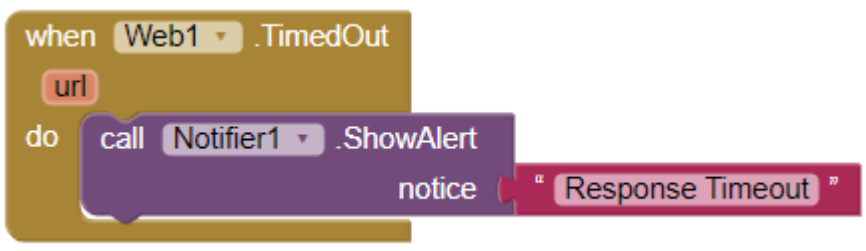
When "GetAirQualityValue_Button" is clicked, its "Click" event handler will be called. Inside the handler, we have to first beware that the user may not yet pick a point on the map. The quickest way is to check either the "currentLat" or "currentLng" is filled with a non-empty value (here we check the "currentLat"). If it is indeed an empty value, we will stop processing and show an alert using the "Notifier1.ShowAlert" method, prompting the user to pick a point first. If it is a non-empty value, we continue the process.

To send a web request, we first have to construct a URL. It is very much the same as trying to get a web page using a browser, such that we have to provide a URL in its address bar. Specifically, to send a web request to PRAISE-HK web service, we can use the "PRAISE_HK_web" extension's " "PRAISE_HK_web1.GenerateURL" to generate a correct URL. The parameters it needs are the longitude, latitude and instant(a time instant). The longitude and latitude are provided by "currentLng" and "currentLat" (these two global variables are the coordinates picked by the user on the map), while the instant is provided by using the Clock component's various methods (such as "Clock1.AddHours" & "Clock1.Now", details here) and the "HourValue" (hour value) selected by the user.

With the chosen location and time instant, the "PRAISE_HK_web1.GenerateURL" now has enough information to construct a right URL. We then use this URL to set the "Web1.Url". And with the right "Web1.Url", we just need to call "Web1.Get" in order to send the web request.

2.5. Handling Response from the Web Request

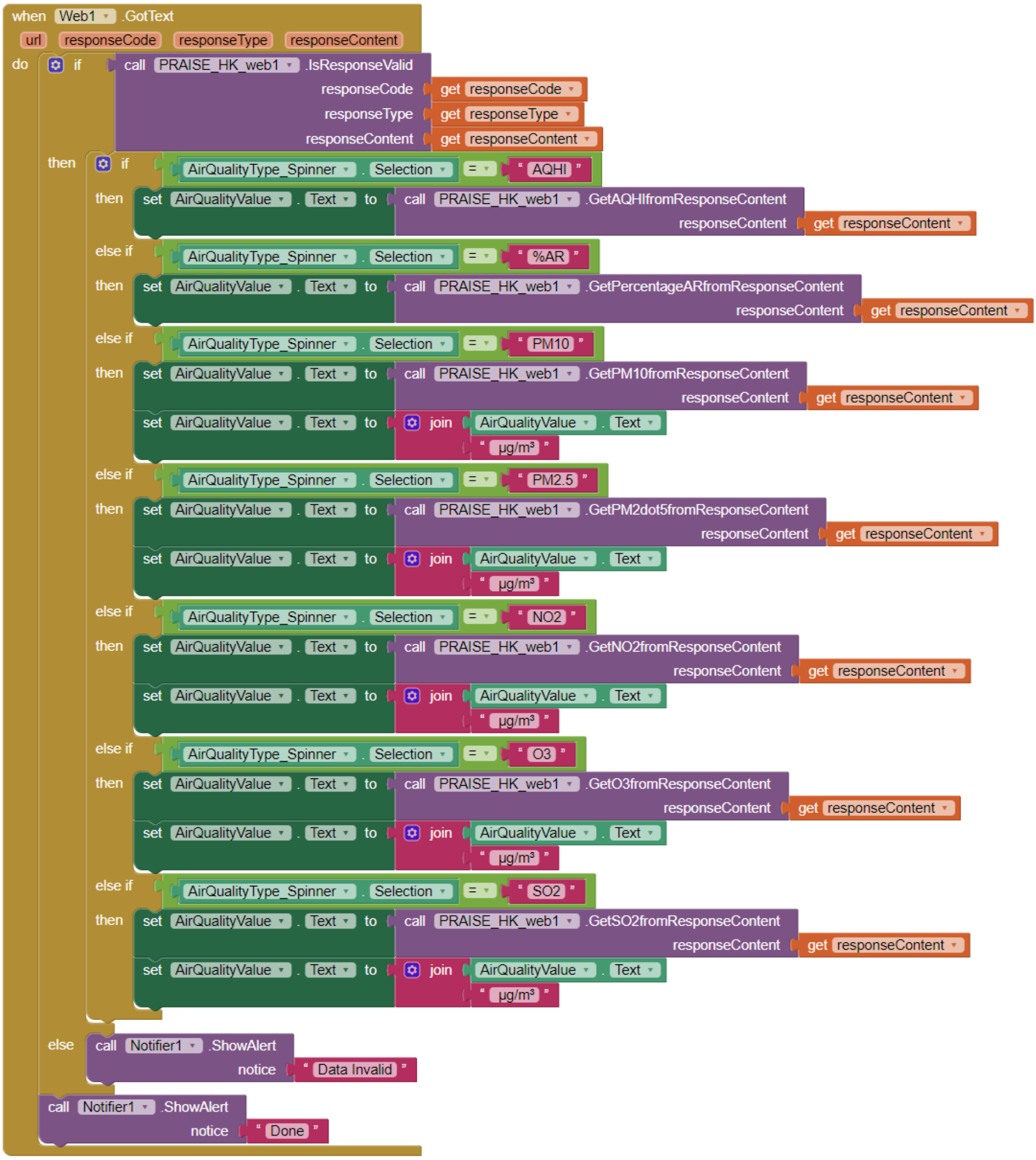
To handle the response from the web request, we use various event handlers. The first one is when there is no response (this is not extremely rare as the web is not a reliable place at all!) :



Explanation:

We can use the "Web1.TimedOut" event handler to handle the case of no response. The above blocks display a pop-up alert with a message "Response Timeout", when there is no response.

If there is a response, we handle it use the following block structure:



Explanation:

When there is a response, it will be handled by the “Web1.GotText” event handler. Inside it, we first need to test if the response is valid by calling “PRAISE_HK_web1.IsResponseValid”. Pass all response data (namely, responseCode, responseType & responseContent) into it, and let it decide whether the response is alright.

If the response is alright, extract the value you need by calling the corresponding “PRAISE_HK_web1” method (e.g. to extract NO_2 value, call “PRAISE_HK_web1.GetNO2fromResponseContent” method). Set the “AirQualityValue.Text" property to display this value (merge this value with its unit, if one has). And finally complete it by displaying a message “Done” using “Notifier1.ShowAlert”.

If the response is not alright, display a “Data Invalid” alert using “Notifier1.ShowAlert” instead.

After adding these final blocks, you can now test it using AI Companion or compiling it to an apk file.

Conclusion

This tutorial serves as a proof that PRAISE-HK service can be used with App Inventor. So, now even average secondary students or non-coders can produce air quality aware apps!