Acquiring Real-time Air Quality Information at your location using PRAISE-HK Web Service

[***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 app user's detected location.

The general plan of acquiring this information is mainly consisted of four main parts:

1. Get the current location
2. Get the current time
3. Through the "PRAISE_HK_web" extension to send the current location and time obtained from the previous 2 steps to the PRAISE-HK Web Service for extracting corresponding air quality information which includes:
    a. Air Quality Health Index (AQHI)
    b. % added short-term health risk (%AR)
    c. Concentration of air pollutants - Nitrogen Dioxide (NO_2), Ozone (O_3), Respirable Suspended Particulates (PM_{10}), Fine suspended particulates ($PM_{2.5}$), and Sulphur Dioxide (SO_2)
4. After receiving the returned data, again use the "PRAISE_HK_web extension" to extract the preferred piece of air quality information to the App Inventor platform.

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

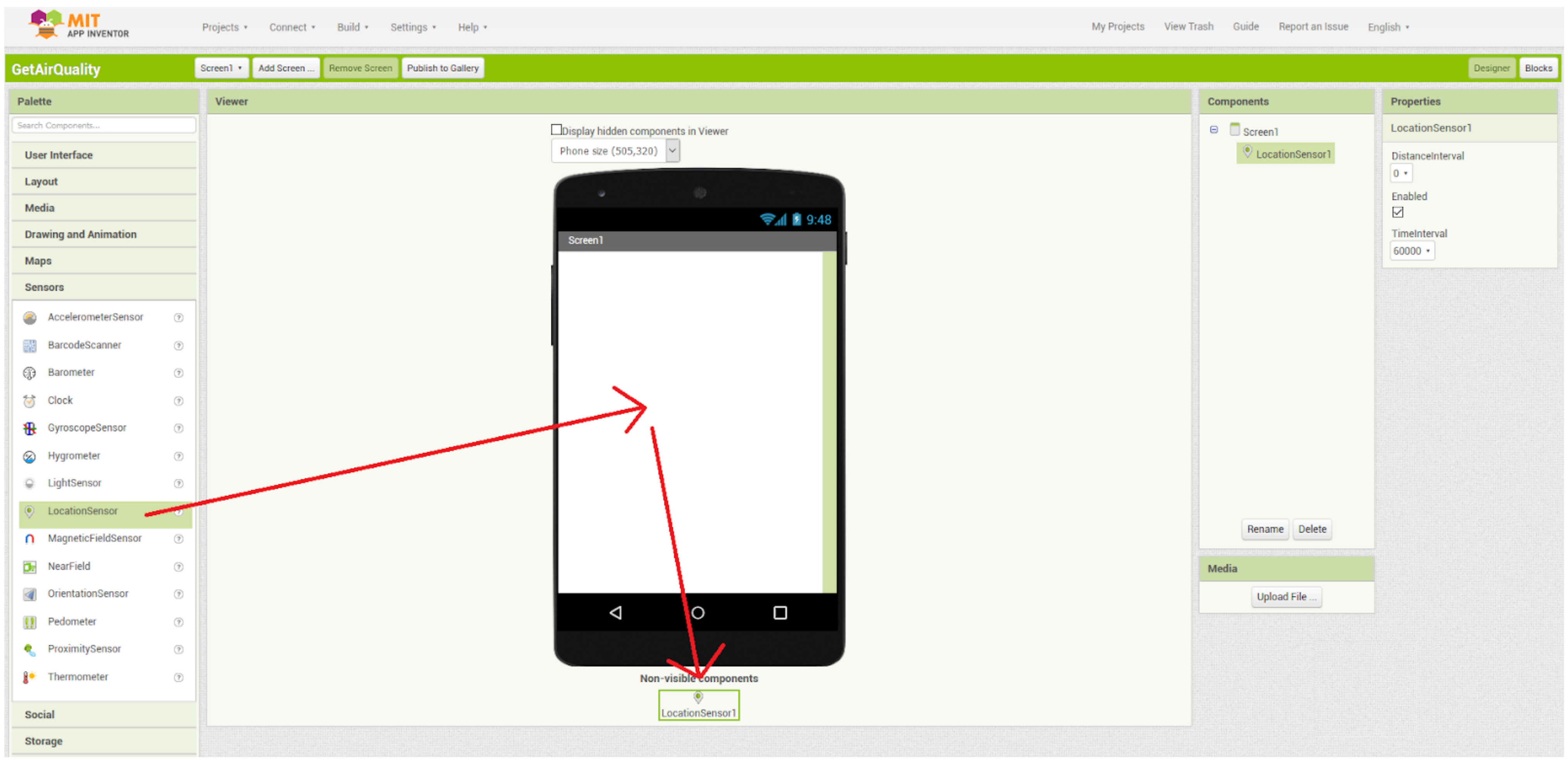
Step 1. Get the current location

Every location on earth can be described and communicated in latitude and longitude. And as nowadays smartphones mostly have location sensors, we will use App Inventor's "LocationSensor" component to access latitudes and longitudes from our mobile phones.

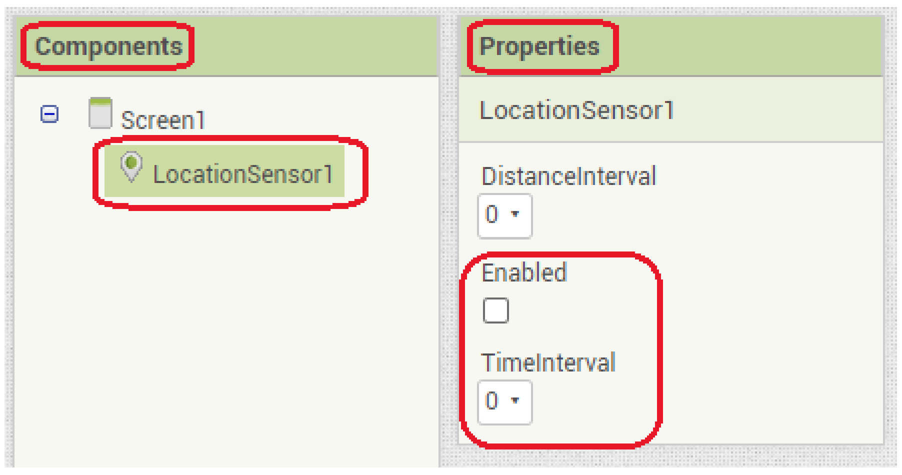
1.1. User interface(UI) design:

1. To start with, let's go to the "Designer" tab by clicking the "Designer" button at the right top corner, for creating the app's outlook
2. Then, in the "Palette" column, search "LocationSensor" inside the "Sensors" folder, and
3. Drag the "LocationSensor" item onto the app screen in the middle

[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 informaiton
2 **High data-density:** the current project is able to provide air quality (and associated health risk) information up to 2-meter solution.
[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 article for details.



4. As it is a non-visible component, it will be automatically placed outside of the screen, rather than staying on the screen. Next, we have to customize it. Select “LocationSensor1” in the “Components” column. Then, in the “Properties” column, uncheck the “Enabled” checkbox and set “TimeInterval” to zero.



5. For the remaining part, we will deal with visible components. First, similar to the above, select “Screen1” in the “Components” column. Then, in the “Properties” column, set both “AlignHorizontal” and “AlignVertical” to “center”. Optionally, we can also give it a nicer “AppName”, by changing the original one to “Get Real-time Air Quality”.
6. Next, drag and drop the various visual components onto “Screen1”, and set the “Properties” as suggested in the following table:

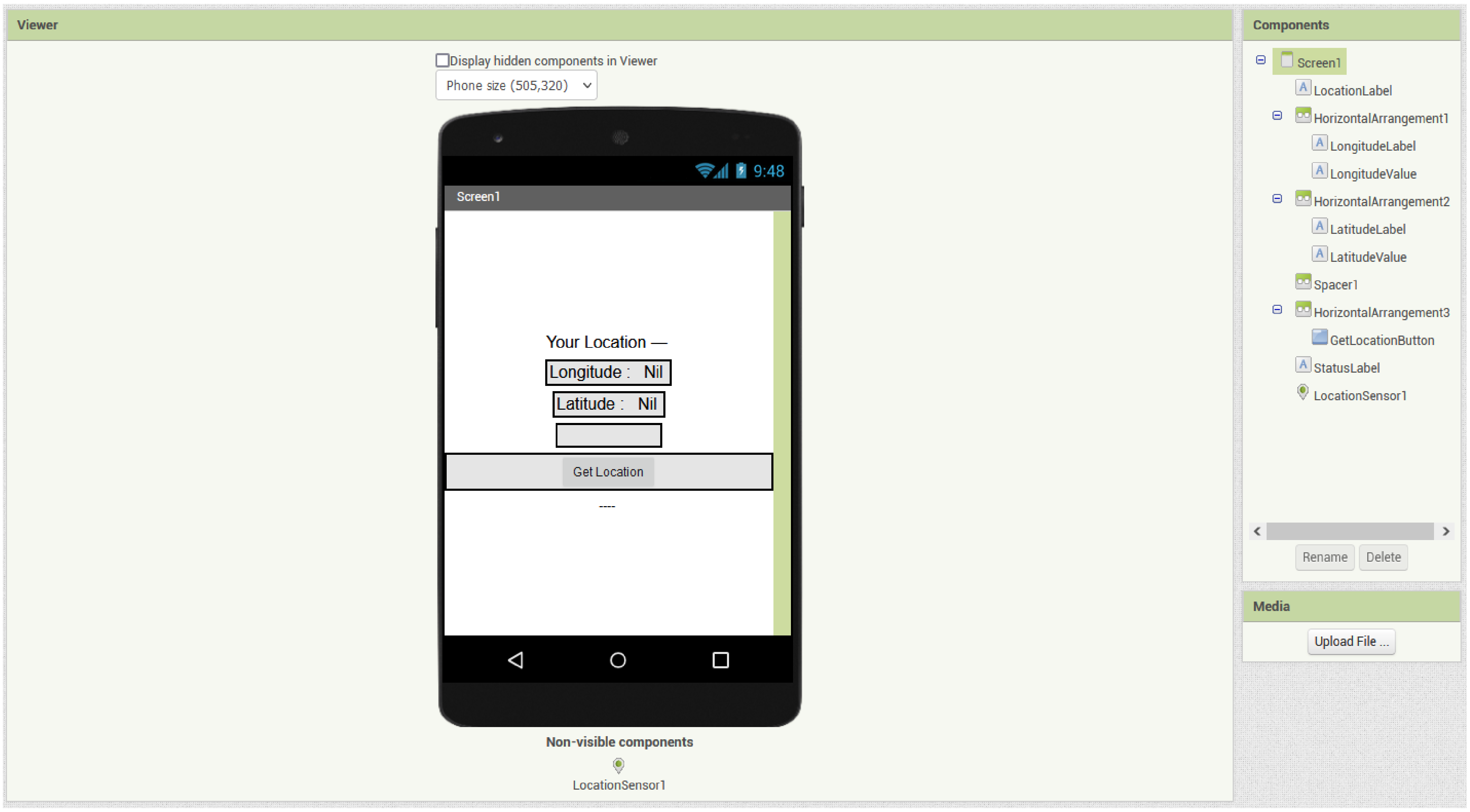
Component Type[1]	***Inside Which Component***	***Follow Which Component***	***Component Name***[2]	***Change Which Properties of the Component***
User Interface / Label	Screen1		LocationLabel	**FontSize:** 18 **Text:** Your Location —
Layout / HorizontalArrangement	Screen1	LocationLabel	HorizontalArrangement1	
User Interface / Label	HorizontalArrangement1		LongitudeLabel	**FontSize:** 18 **Text:** Longitude :
User Interface / Label	HorizontalArrangement1	LongitudeLabel	LongitudeValue	**FontSize:** 18 **Text:** Nil
Layout / HorizontalArrangement	Screen1	HorizontalArrangement1	HorizontalArrangement2	
User Interface / Label	HorizontalArrangement2		LatitudeLabel	**FontSize:** 18 **Text:** Latitude :
User Interface / Label	HorizontalArrangement2	LatitudeLabel	LatitudeValue	**FontSize:** 18 **Text:** Nil
Layout / HorizontalArrangement	Screen1	HorizontalArrangement2	Spacer1	**Height:** 20pixels
Layout / HorizontalArrangement	Screen1	Spacer1	HorizontalArrangement3	**AlignHorizontal:** Center **AlignVertical:** Center **Width:** Fill parent
User Interface / Button	HorizontalArrangement3		GetLocationButton	**Text:** Get Location

User Interface / Label	Screen1	HorizontalArrangement3	StatusLabel	**FontItalic:** checked **Text:** ----

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

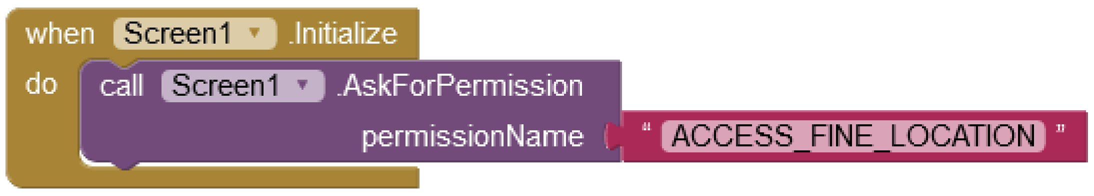
If everything is alright, the appearance would look like this:



After defining its appearance, we can now click the "Blocks" button (at the top-right corner) to enter the "Blocks" Editor tab to define the App's behaviour (by coding).

1.2. Coding - Permission Issue

In order to obtain the user's location information, apps have to seek prior permissions from the users. To do that, we should add a coding "blocks" structure as follows: (You can drag "blocks" from the "Blocks" column [the left side of the tab], to the "Viewer" column in the centre, for completing the structure.)



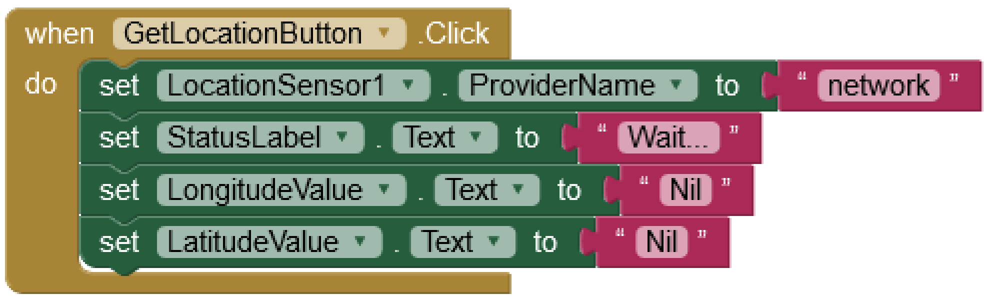
```
when Screen1 .Initialize
do  call Screen1 .AskForPermission
                 permissionName " ACCESS_FINE_LOCATION "
```

Code Explanation:

When the App is initialized (when the Screen1.Initialize event is occured), we will ask the user for this particular permission : ACCESS_FINE_LOCATION. If the permission was already granted during previous uses, this block is simply ignored.

1.3. Coding: Getting Current Location

For getting the location information, we assign the following block structure to the "Click" event handler of the "GetLocationButton":



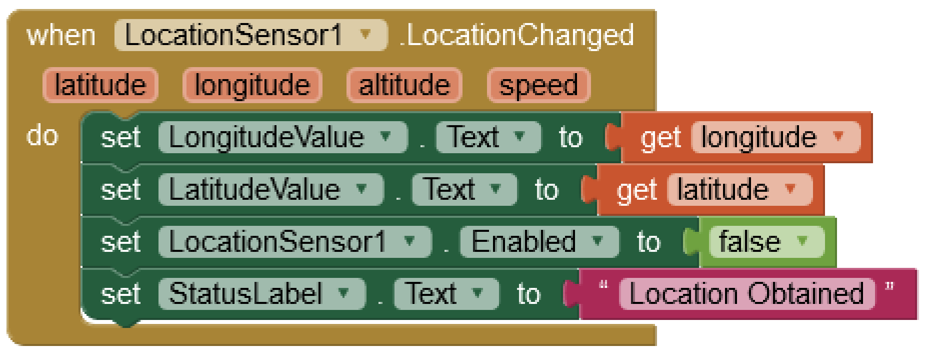
```
when GetLocationButton .Click
do  set LocationSensor1 . ProviderName to " network "
    set StatusLabel . Text to " Wait... "
    set LongitudeValue . Text to " Nil "
    set LatitudeValue . Text to " Nil "
```

Code Explanation:

When this button is clicked, the ‘GetLocationButton.Click’ event handler will run. Inside this handler, we just need to set the ‘LocationSensor1.ProviderName’ to “network” (actually there are two ways to get location: network or GPS, but we use network in this demo as it is faster and less restriction to use) in order to enable and then instruct the ‘LocationSensor1’ component to get location information for us in one go. Actually we should use this property rather than setting the “LocationSensor1.Enabled” to “true”. It is because setting this property always forces the location service to report the current location, even if the location has not been changed since reporting last time. The info might not be available immediately, it might take some time to get it. We will handle this in the next section. The other blocks’ values are the values we use to indicate to the user that the app is now trying to get the location information, but not yet there.

1.4. Coding: When the location information is available

We use the following block structure to display the location information when it is available:



```
when LocationSensor1 .LocationChanged
  latitude  longitude  altitude  speed
do  set LongitudeValue . Text to  get longitude
    set LatitudeValue . Text to  get latitude
    set LocationSensor1 . Enabled to  false
    set StatusLabel . Text to  " Location Obtained "
```

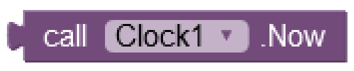
Code Explanation:

When the information is arrived, the ‘LocationSensor1.LocationChanged’ will run. In the handler, various labels are set to indicate the result (such as longitude and latitude values obtained). ‘LocationSensor1.Enabled’ is also set to false to prevent further fetching of location info.

Now, run the codes in AI Companion or build a separate apk for testing. You can also check this against an online map (such as Google Map) to see if the result is correct or not.

Step 2. Get the current time

It is straightforward to get the current time. First, go to the “Designer” tab and drag the “Clock” component (non-visible) inside the “Sensor” folder in the “Palette” column to the phone. Then, go to the “Blocks” editor tab, drag the following method of the “Clock” out from the ‘Blocks’ column:



```
call Clock1 .Now
```

That’s it! The above method is the one we use to get the current time. You will be shown how to use it a bit later.

Step 3. Send the current location and time obtained above to the PRAISE-HK Web Service

Now, it is the time to send those things we obtained in step 1 & 2 to PRAISE-HK Web Service to get a particular piece of air quality info we need, with the help of the PRAISE_HK_web extension.

Designing the UI

In order to do this, we first need to change the interface. So go to the Designer again, and change the interface as instructed in the following tables:

Components to be changed

Component Name	***Change its Name to***	***Change Which Properties of the Component***
GetLocationButton	GetAirQualityButton	**FontSize:** 16 **Text:** Get

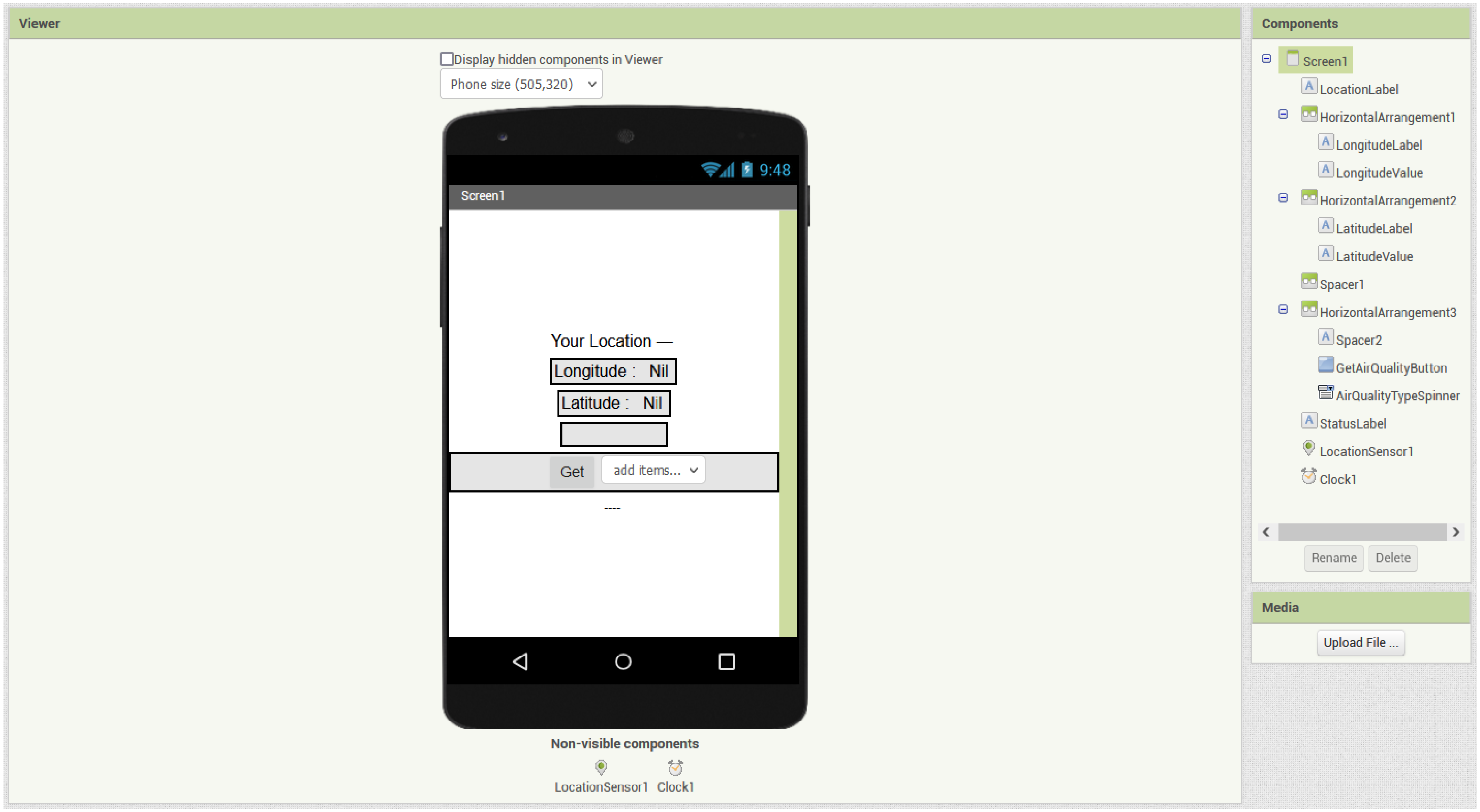
Components to be added

Component Type	***Inside Which Component***	***Follow Which Component***	***Component Name***	***Change Which Properties of the Component***
User Interface / Label	HorizontalArrangement3		Spacer2	**Width:** 20 pixels **Text:** *{(empty)}*[1]
User Interface / Spinner	HorizontalArrangement3	GetAirQualityButton	AirQualityTypeSpinner	**ElementsFromString:** AQHI,%AR,PM10,PM2.5,NO2,O3,SO2 **Selection:** AQHI

Annotation --

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

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

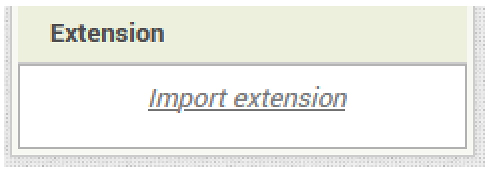


The button "GetAirQualityButton" is used to first get the current time and location, then get the air quality info. The spinner is used to select which piece of air quality info should get.

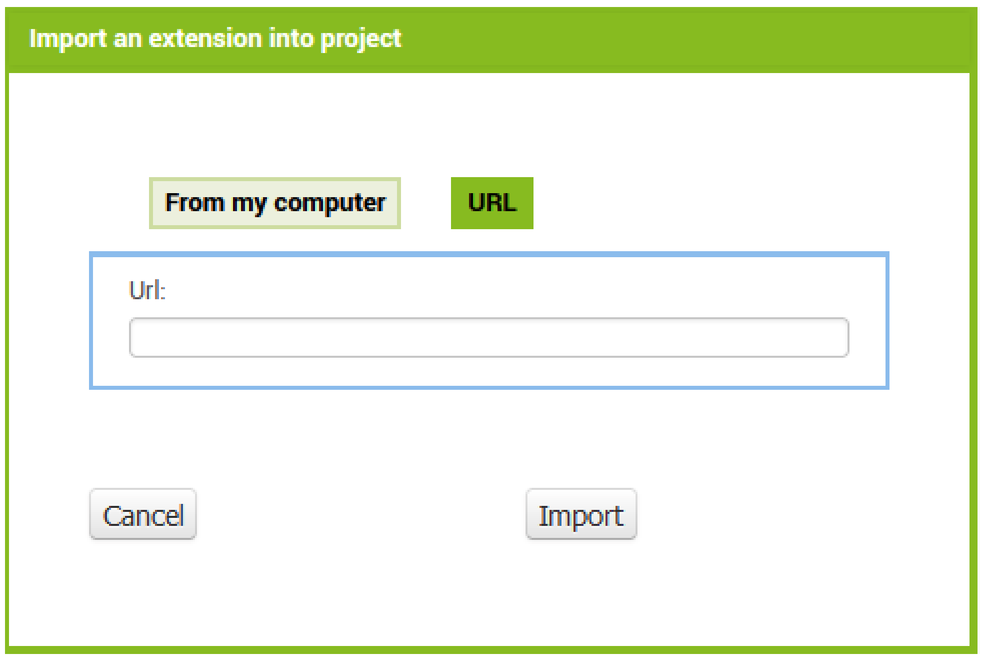
In order to send or receive data through the web, we have to use a built-in component called 'Web'. You can find it at 'Palette' ➔ 'Connectivity' section. After locating it, drag it to the phone so that we can use it.

Last but not least, we have to import the PRAISE_HK_web extension.

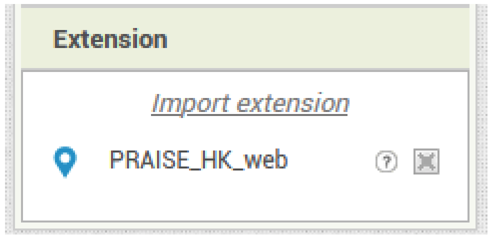
First, go to the 'Palette' ➔ 'Extension' and click 'import extension'



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_v0.9.aix into the 'Url' textbox to import the extension.

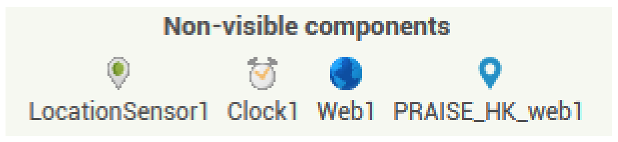


If the import is successful, the 'Extension' section will turn into this:



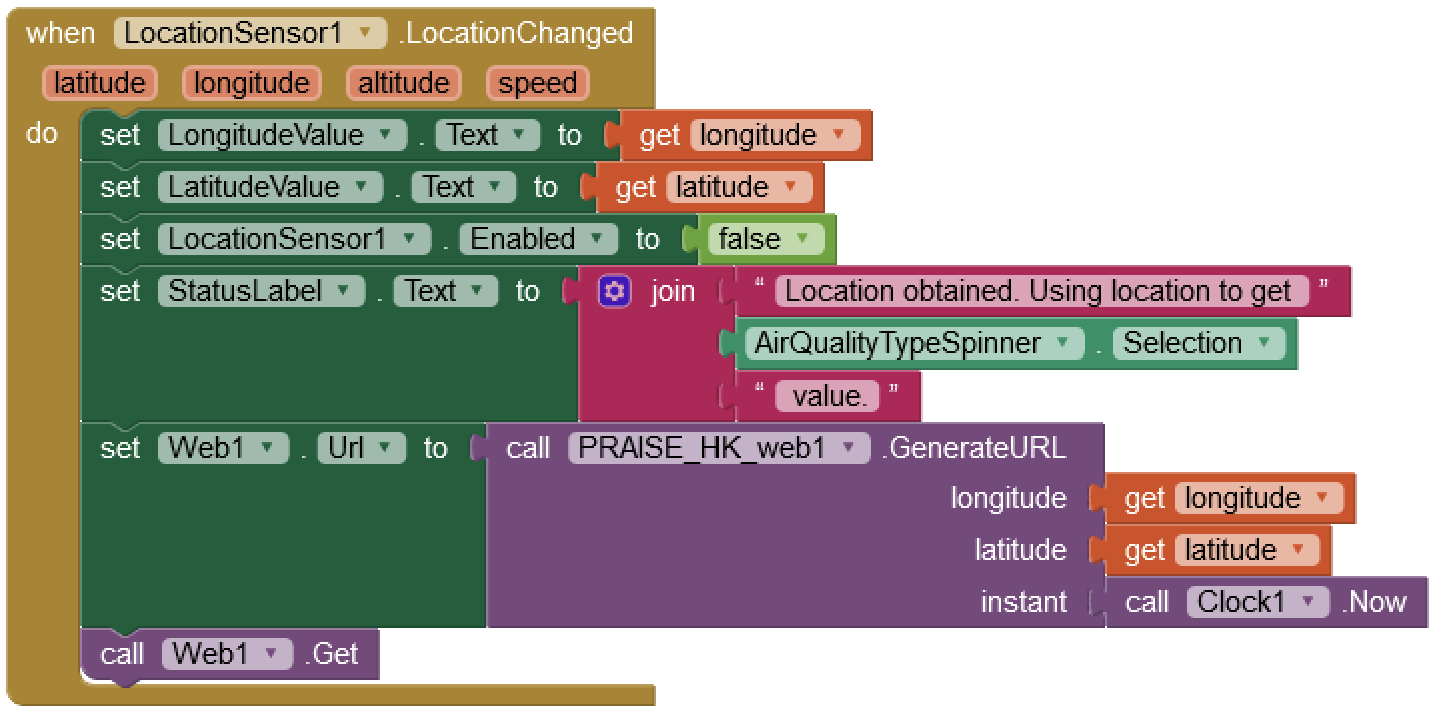
Now, you should drag this onto the phone (like other components). And the extension is ready for use.

The Non-visible components (at the bottom of the phone) should look like the following after adding the PRAISE_HK_web.



Add the behaviour

Next, go to the Blocks Editor. And from the 'Blocks' menu, select corresponding blocks to change the "LocationSensor1.LocationChanged" event handler to the following structure:



```
when LocationSensor1 .LocationChanged
  latitude  longitude  altitude  speed
do  set LongitudeValue . Text to  get longitude
    set LatitudeValue . Text to  get latitude
    set LocationSensor1 . Enabled to  false
    set StatusLabel . Text to  join  " Location obtained. Using location to get "
                                     AirQualityTypeSpinner . Selection
                                     " value. "
    set Web1 . Url to  call PRAISE_HK_web1 .GenerateURL
                                   longitude  get longitude
                                   latitude   get latitude
                                   instant    call Clock1 .Now
    call Web1 .Get
```

Code Explanation:

It is natural to add codes in the 'LocationSensor1.LocationChanged' to send the web request as it is here we obtain the location. In this handler, first set StatusLabel.Text to another wording, indicating trying to get the corresponding air quality value, with location obtained. Sending web requests to PRAISE-HK Web Service is very similar to requesting a web page in a browser, you have to give it a URL. You can think of the Web1 component as a browser in this case. So, we set the Web1.Url to the URL provided by calling 'PRAISE_HK_web1.GenerateURL', which needs three parameters. Please be noted that the last one is the time (called instant in App Inventor), here we provide the current time using 'Clock1.Now' which we prepared in step 2. After setting the URL, we use 'Web1.Get' to send out the request.

As the response won't come back immediately, we will use another event handler to process the returned data.

Step 4. Processing the data after receiving it from the PRAISE-HK Web Service

Finally, after receiving the data, we again use the PRAISE_HK_web extension to help us to get a particular piece of air quality info we need.

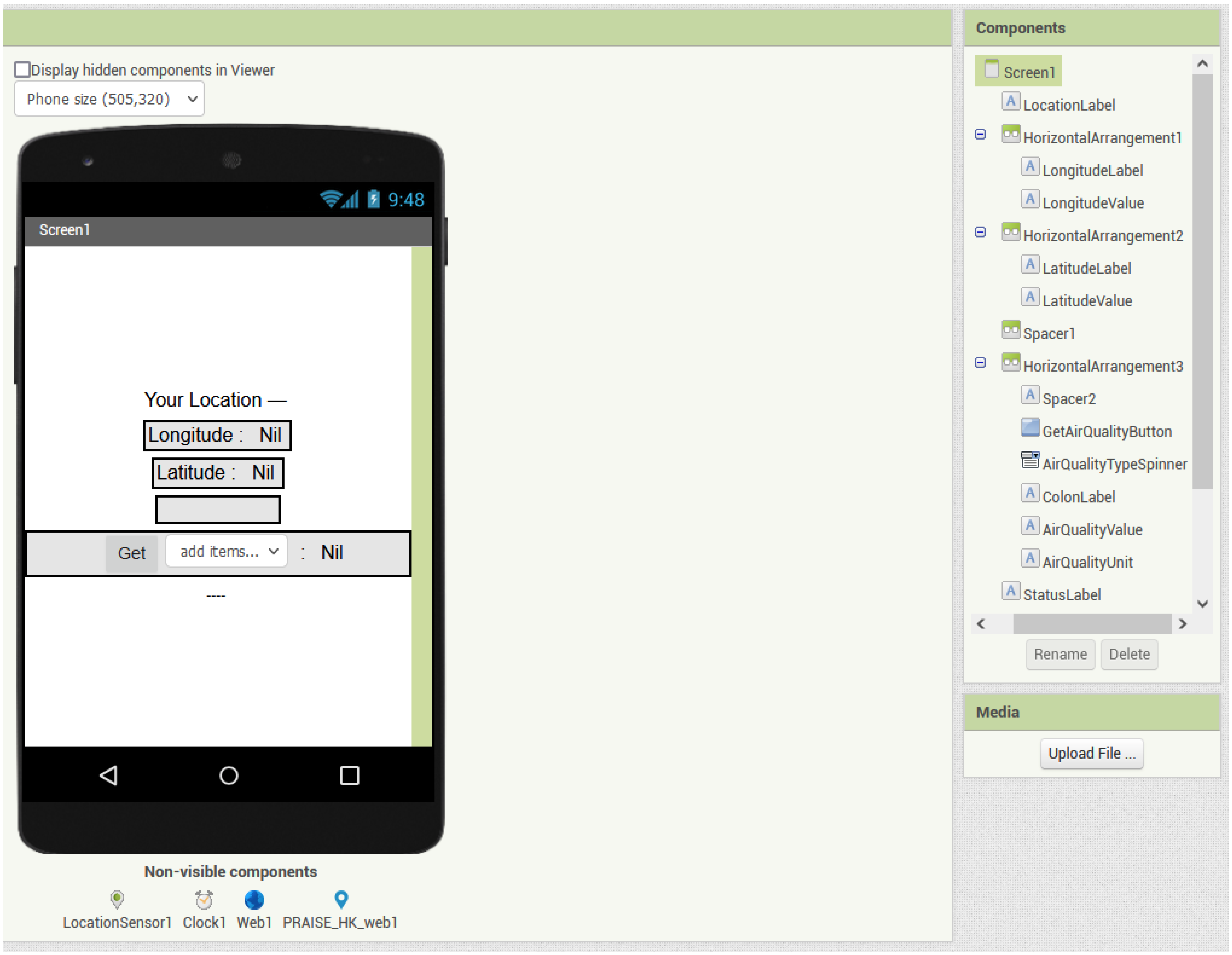
Designing the UI

In order to display the info, we first need to change the interface. So go to the Designer again, and change the interface as instructed in the following table:

Components to be added

Component Type	***Inside Which Component***	***Follow Which Component***	***Component Name***	***Change Which Properties of the Component***
User Interface / Label	HorizontalArrangement3	AirQualityTypeSpinner	ColonLabel	**FontSize:** 18 **Text:** :
User Interface / Label	HorizontalArrangement3	ColonLabel	AirQualityValue	**FontSize:** 18 **Text:** Nil
User Interface / Label	HorizontalArrangement3	AirQualityValue	AirQualityUnit	**FontSize:** 16 **Text:** *{(empty)}*

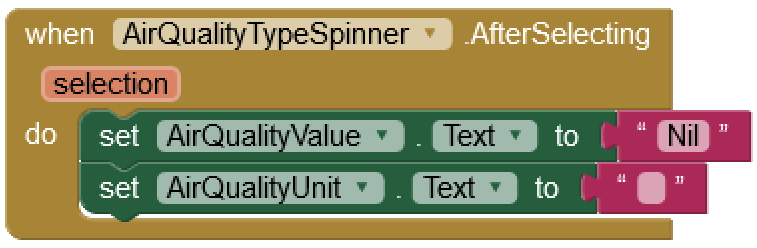
If everything is correct, the interface should look like this:



Add the behaviour

Now, go to the Blocks Editor. From the ‘Blocks’ menu, we have to add three more structures to finish the task --

1st structure:

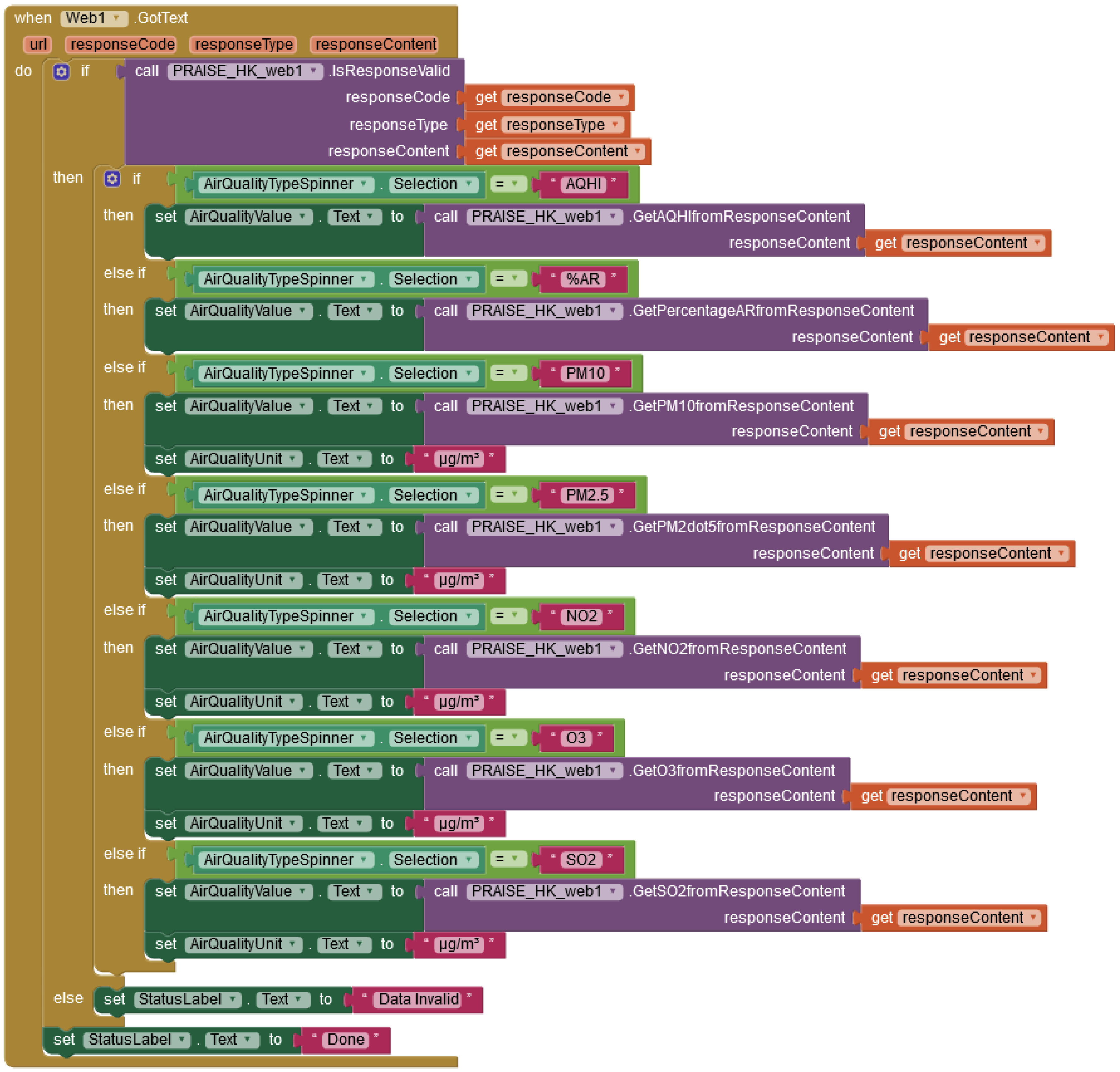


```
when AirQualityTypeSpinner .AfterSelecting
  selection
do  set AirQualityValue . Text to " Nil "
    set AirQualityUnit . Text to " "
```

Code Explanation:

“AirQualityValue.Text” and “AirQualityUnit.Text” are needed to be reset when an item of the “AirQualityTypeSpinner” is selected. This is because the existing air quality value is not valid anymore when a new air quality type is selected.

2nd structure:



```
when Web1 .GotText
  url  responseCode  responseType  responseContent
do  if  call PRAISE_HK_web1 .IsResponseValid
                       responseCode  get responseCode
                       responseType  get responseType
                    responseContent  get responseContent
    then  if  AirQualityTypeSpinner . Selection  =  " AQHI "
          then  set AirQualityValue . Text to  call PRAISE_HK_web1 .GetAQHIfromResponseContent
                                                               responseContent  get responseContent
          else if  AirQualityTypeSpinner . Selection  =  " %AR "
          then  set AirQualityValue . Text to  call PRAISE_HK_web1 .GetPercentageARfromResponseContent
                                                               responseContent  get responseContent
          else if  AirQualityTypeSpinner . Selection  =  " PM10 "
          then  set AirQualityValue . Text to  call PRAISE_HK_web1 .GetPM10fromResponseContent
                                                               responseContent  get responseContent
                set AirQualityUnit . Text to  " μg/m³ "
          else if  AirQualityTypeSpinner . Selection  =  " PM2.5 "
          then  set AirQualityValue . Text to  call PRAISE_HK_web1 .GetPM2dot5fromResponseContent
                                                               responseContent  get responseContent
                set AirQualityUnit . Text to  " μg/m³ "
          else if  AirQualityTypeSpinner . Selection  =  " NO2 "
          then  set AirQualityValue . Text to  call PRAISE_HK_web1 .GetNO2fromResponseContent
                                                               responseContent  get responseContent
                set AirQualityUnit . Text to  " μg/m³ "
          else if  AirQualityTypeSpinner . Selection  =  " O3 "
          then  set AirQualityValue . Text to  call PRAISE_HK_web1 .GetO3fromResponseContent
                                                               responseContent  get responseContent
                set AirQualityUnit . Text to  " μg/m³ "
          else if  AirQualityTypeSpinner . Selection  =  " SO2 "
          then  set AirQualityValue . Text to  call PRAISE_HK_web1 .GetSO2fromResponseContent
                                                               responseContent  get responseContent
                set AirQualityUnit . Text to  " μg/m³ "
    else  set StatusLabel . Text to  " Data Invalid "
    set StatusLabel . Text to  " Done "
```

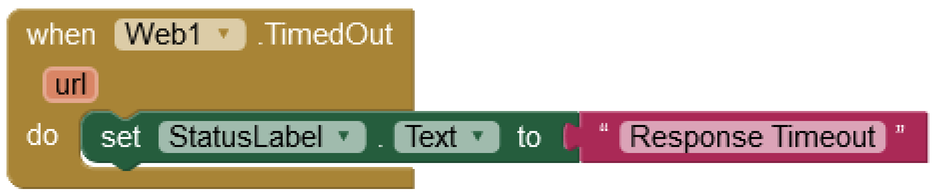
Code Explanation:

The ‘Web1.GotText’ event handler will be called if there is a response from the web service. 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). Next, set the “AirQualityValue.Text" property to display this value, and then follow it with a unit if one has (by setting the “AirQualityUnit.Text”).

The ‘StatusLabel.Text’ is also refreshed to reflect the updated situation (success or error).

3rd structure:

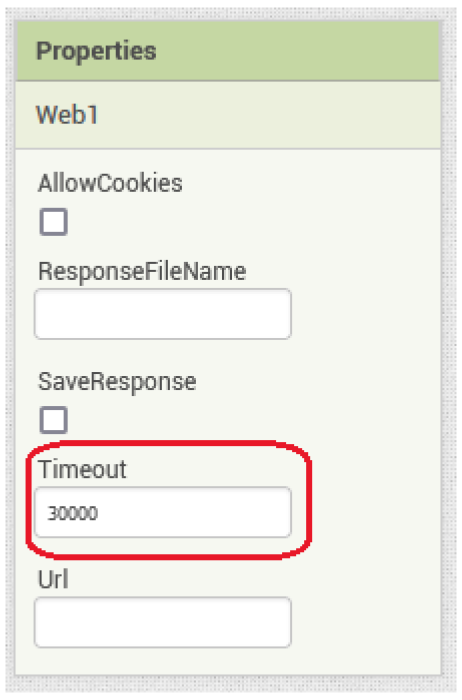


```
when Web1 .TimedOut
  url
do  set StatusLabel . Text to  " Response Timeout "
```

Code Explanation:

The internet is not a very reliable place. In rare situations, for whatever reason, we get no response from the web. The “Web1.TimedOut” event handler is used to handle this situation. Inside it, we inform the user about this by setting the “StatusLabel.Text”.

In order for this handler to work, we have to do one more thing. Go back to the “Designer” tab, select the “Web1” component, and set the “Timeout” value in the “Properties” column to 30000. This value means we will wait for 30 seconds at most before giving up, and running the “TimedOut” event handler.


Properties

Web1

AllowCookies ☐

ResponseFileName

SaveResponse ☐

Timeout 30000

Url

That’s it! 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!