



Android

Application Development

Lab 2

Human-Computer Interaction, AUEB
Εαρινό εξάμηνο 2022-2023

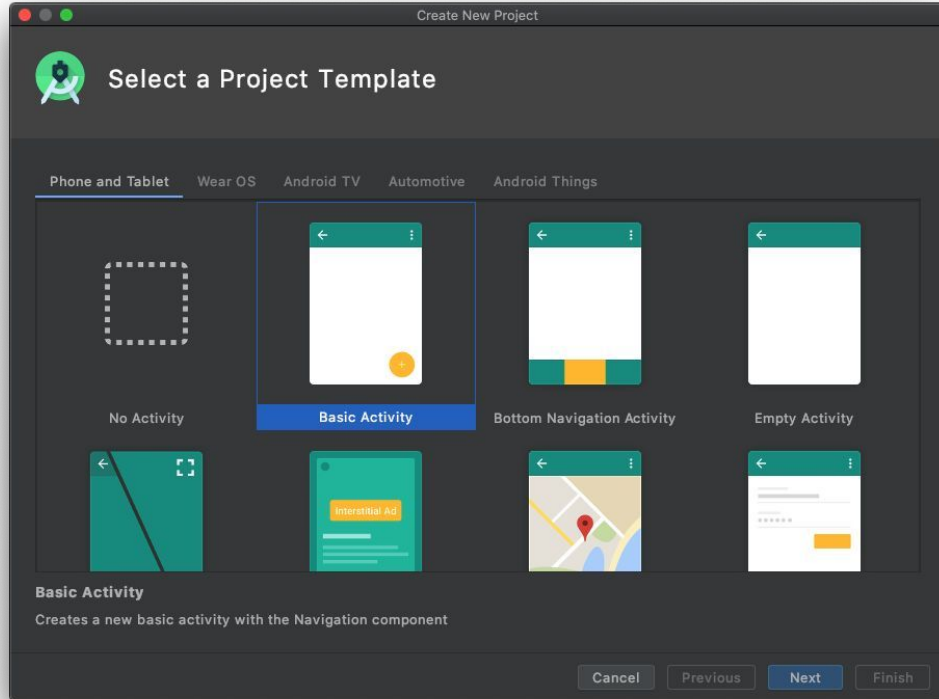
Lab Assistant: Sofia Eleftheriou



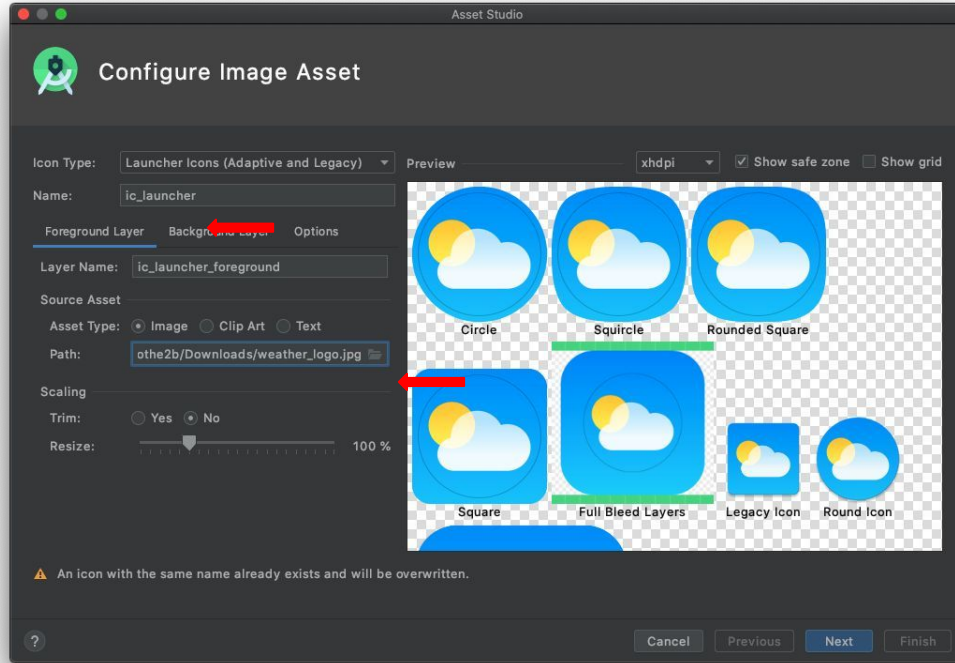
Android Development Advanced Use

- Catch up with Lab 1
- Refactor Project components
 - Rename Activity
 - Make property publicly available
- Use Public Forecast API
 - OpenWeatherMap API
 - REST call to API - JSON response
 - Create Asynchronous Task to call API
 - Add new Menu Option to refresh information
 - Update Business Logic to use the Asynchronous task - Involve the Refresh Option
 - Demonstrate updates

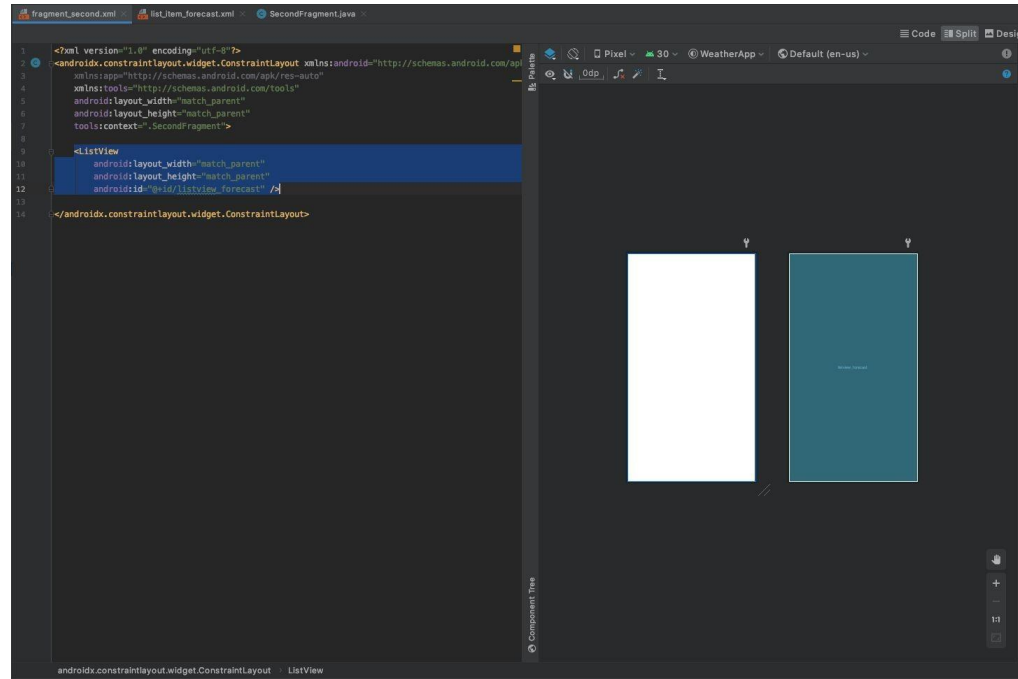
Catch up with Lab 1



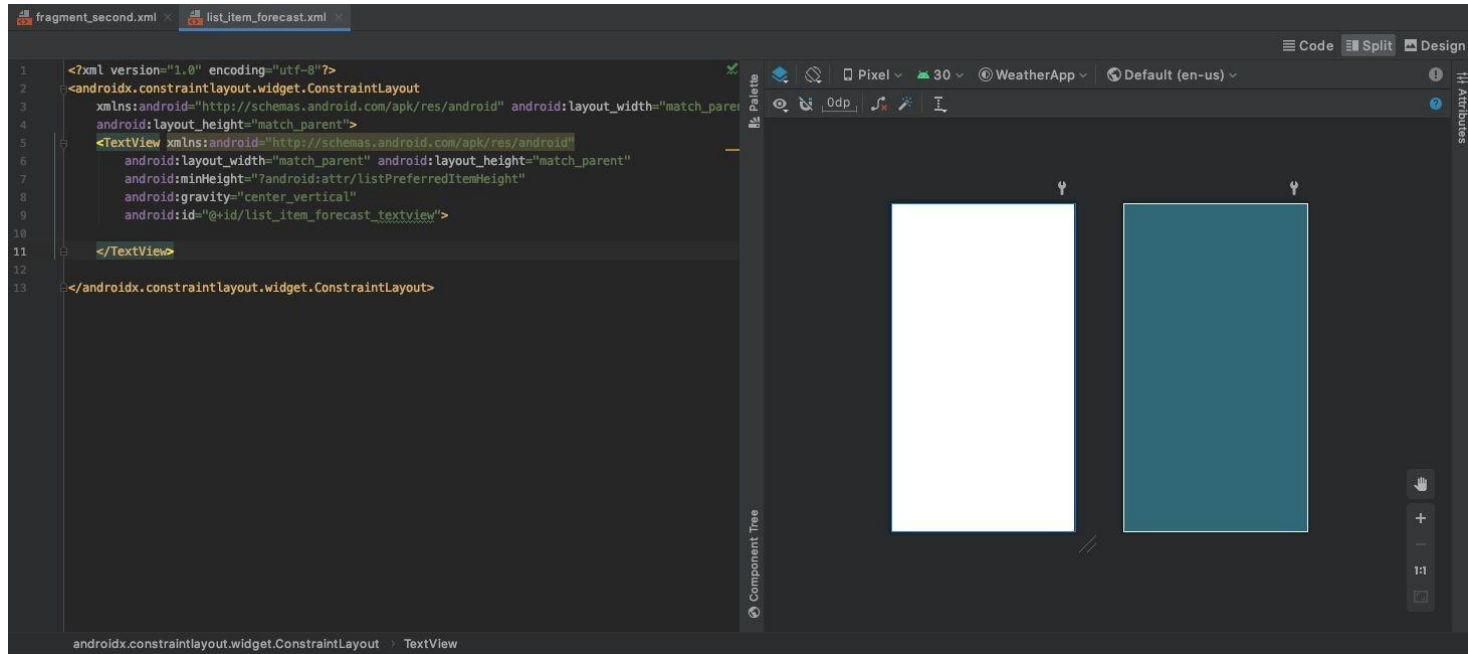
Create new project with Basic Activity



Change Application Icon



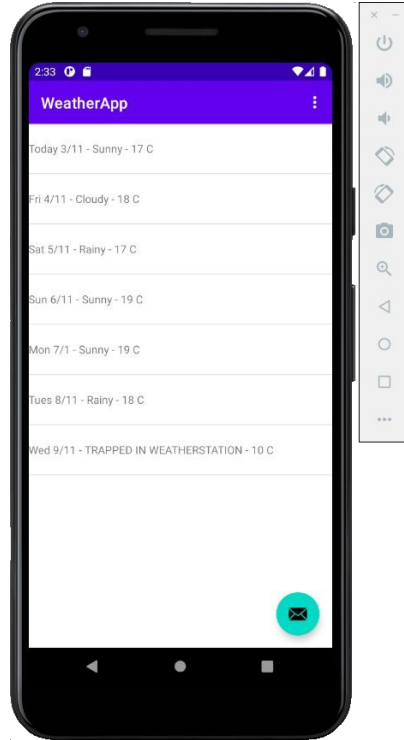
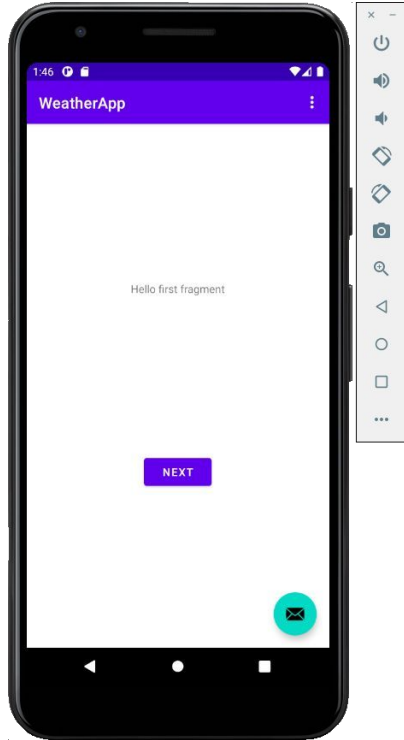
Add ListView to layout



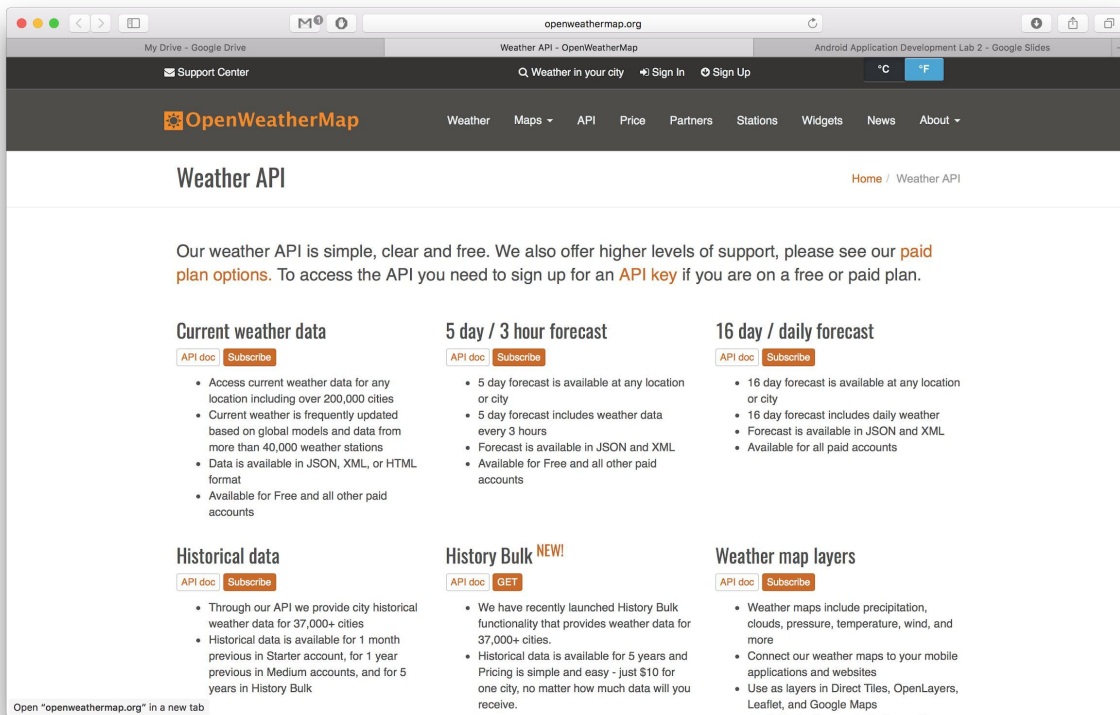
Add new layout with TextView for list items

```
SecondFragment.java
1 package com.example.weatherapp;
2
3 import android.os.Bundle;
4 import android.view.LayoutInflater;
5 import android.view.View;
6 import android.view.ViewGroup;
7
8 import androidx.fragment.app.Fragment;
9 import androidx.appcompat.widget.Toolbar;
10 import androidx.recyclerview.widget.RecyclerView;
11 import androidx.recyclerview.widget.RecyclerView.ViewHolder;
12
13 public class SecondFragment extends Fragment {
14
15     @Override
16     public View onCreateView(
17         LayoutInflater inflater, ViewGroup container,
18         Bundle savedInstanceState) {
19
20         // Inflate the layout for this fragment
21         RecyclerView recyclerView = (RecyclerView) inflater.inflate(
22             R.layout.fragment_second, container, false);
23
24         // Create a mock data list
25         String[] data = {
26             "Today 3/11 - Sunny - 17 C",
27             "Fri 4/11 - Cloudy - 18 C",
28             "Sat 5/11 - Rainy - 17 C",
29             "Sun 6/11 - Sunny - 19 C",
30             "Mon 7/1 - Sunny - 19 C",
31             "Tues 8/11 - Rainy - 18 C",
32             "Wed 9/11 - TRAPPED IN WEATHERSTATION - 10 C"
33         };
34
35         // Create a RecyclerView adapter
36         RecyclerView.Adapter<ViewHolder> weekForecast = new RecyclerView.Adapter<ViewHolder>() {
37             @Override
38             public int getItemCount() {
39                 return data.length;
40             }
41
42             @Override
43             public ViewHolder onCreateViewHolder(ViewGroup parent) {
44                 View itemView = LayoutInflater.from(parent.getContext())
45                     .inflate(R.layout.item_forecast, parent, false);
46                 return new ViewHolder(itemView);
47             }
48
49             @Override
50             public void onBindViewHolder(ViewHolder holder, int position) {
51                 holder.itemView.setText(data[position]);
52             }
53         };
54
55         // Attach the adapter to the RecyclerView
56         recyclerView.setAdapter(weekForecast);
57
58         return recyclerView;
59     }
60 }
```

New business logic / Populate list with mock (fake) data



OpenWeatherMap API



The screenshot shows a web browser window displaying the OpenWeatherMap website. The browser's address bar shows the URL openweathermap.org. The website's navigation bar includes the OpenWeatherMap logo and links for Weather, Maps, API, Price, Partners, Stations, Widgets, News, and About. The main heading is "Weather API", with a breadcrumb trail "Home / Weather API".

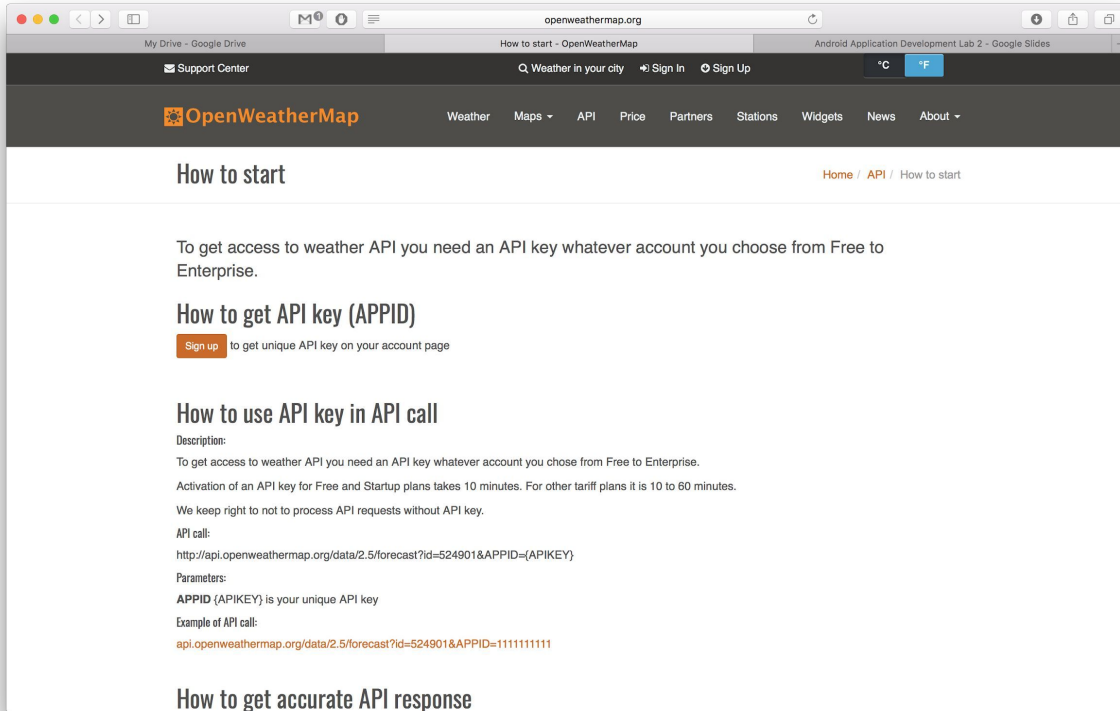
The main content area features a paragraph: "Our weather API is simple, clear and free. We also offer higher levels of support, please see our [paid plan options](#). To access the API you need to sign up for an [API key](#) if you are on a free or paid plan."

Below this are six feature sections, each with a title, a list of bullet points, and two buttons: "API doc" and "Subscribe".

- Current weather data**
 - Access current weather data for any location including over 200,000 cities
 - Current weather is frequently updated based on global models and data from more than 40,000 weather stations
 - Data is available in JSON, XML, or HTML format
 - Available for Free and all other paid accounts
- 5 day / 3 hour forecast**
 - 5 day forecast is available at any location or city
 - 5 day forecast includes weather data every 3 hours
 - Forecast is available in JSON and XML
 - Available for Free and all other paid accounts
- 16 day / daily forecast**
 - 16 day forecast is available at any location or city
 - 16 day forecast includes daily weather
 - Forecast is available in JSON and XML
 - Available for all paid accounts
- Historical data**
 - Through our API we provide city historical weather data for 37,000+ cities
 - Historical data is available for 1 month previous in Starter account, for 1 year previous in Medium accounts, and for 5 years in History Bulk
- History Bulk ^{NEW!}**
 - We have recently launched History Bulk functionality that provides weather data for 37,000+ cities.
 - Historical data is available for 5 years and Pricing is simple and easy - just \$10 for one city, no matter how much data you will receive.
- Weather map layers**
 - Weather maps include precipitation, clouds, pressure, temperature, wind, and more
 - Connect our weather maps to your mobile applications and websites
 - Use as layers in Direct Tiles, OpenLayers, Leaflet, and Google Maps

At the bottom left, a small note says: "Open 'openweathermap.org' in a new tab".

<http://openweathermap.org>



The screenshot shows a web browser window with the URL `openweathermap.org`. The page title is "How to start - OpenWeatherMap". The navigation bar includes "Support Center", "Weather in your city", "Sign In", "Sign Up", and temperature units "°C" and "°F". The main navigation menu contains "OpenWeatherMap", "Weather", "Maps", "API", "Price", "Partners", "Stations", "Widgets", "News", and "About".

How to start

[Home](#) / [API](#) / [How to start](#)

To get access to weather API you need an API key whatever account you choose from Free to Enterprise.

How to get API key (APPID)

[Sign up](#) to get unique API key on your account page

How to use API key in API call

Description:

To get access to weather API you need an API key whatever account you chose from Free to Enterprise.

Activation of an API key for Free and Startup plans takes 10 minutes. For other tariff plans it is 10 to 60 minutes.

We keep right to not to process API requests without API key.

API call:

```
http://api.openweathermap.org/data/2.5/forecast?id=524901&APPID={APIKEY}
```

Parameters:

APPID (APIKEY) is your unique API key

Example of API call:

```
api.openweathermap.org/data/2.5/forecast?id=524901&APPID=1111111111
```

How to get accurate API response

Register & Get API key

REST Call to API - JSON response



- **What we would like to know?**

We would like to have the weather forecast for 1 week

<http://api.openweathermap.org/data/2.5/forecast/daily>

- **Which parameter we need to specify?**

- City id (Athens - 264371)
- Format (JSON)
- Unit: (Metric system)
- APPID: 612ce9a4c7726a3a0a00a69b84b9a01a

<http://api.openweathermap.org/data/2.5/forecast/daily?id=264371&mode=json&units=metric&cnt=7&APPID=612ce9a4c7726a3a0a00a69b84b9a01a>




```
api.openweathermap.org
My Drive - Google Drive
api.openweathermap.org/data/2.5/forecast/daily?id=264371&mode=json&units...
Android Application Development Lab 2 - Google Slides

{"city":{"id":264371,"name":"Athens","coord":{"lon":23.7162,"lat":37.9795},"country":"GR","population":0},"cod":"200","message":0.0999551,"cnt":7,"list":[{"dt":1509357600,"temp":{"day":19.5,"min":15.77,"max":19.5,"night":15.77,"eve":18.51,"morn":19.5},"pressure":1019.74,"humidity":79,"weather":[{"id":800,"main":"Clear","description":"sky is clear","icon":"02d"}],"speed":5.2,"deg":22,"clouds":8},{"dt":1509444000,"temp":{"day":17.35,"min":14.74,"max":17.62,"night":14.74,"eve":16.86,"morn":16.47},"pressure":1028.4,"humidity":85,"weather":{"id":803,"main":"Clouds","description":"broken clouds","icon":"04d"},"speed":6.53,"deg":15,"clouds":80},{"dt":1509530400,"temp":{"day":16.4,"min":14.08,"max":16.4,"night":14.08,"eve":15.29,"morn":14.77},"pressure":1031.96,"humidity":90,"weather":{"id":801,"main":"Clouds","description":"few clouds","icon":"02d"},"speed":6.12,"deg":29,"clouds":20},{"dt":1509616800,"temp":{"day":16.39,"min":13.2,"max":17.64,"night":14.25,"eve":17.36,"morn":13.2},"pressure":1030.24,"humidity":92,"weather":{"id":800,"main":"Clear","description":"sky is clear","icon":"01d"},"speed":4.66,"deg":359,"clouds":0},{"dt":1509703200,"temp":{"day":18.87,"min":12.44,"max":18.87,"night":15.06,"eve":15.61,"morn":12.44},"pressure":1016.95,"humidity":0,"weather":{"id":800,"main":"Clear","description":"sky is clear","icon":"01d"},"speed":2.96,"deg":177,"clouds":9},{"dt":1509789600,"temp":{"day":20.24,"min":14.7,"max":20.24,"night":14.7,"eve":16.19,"morn":15.19},"pressure":1014.98,"humidity":0,"weather":{"id":500,"main":"Rain","description":"light rain","icon":"10d"},"speed":0.82,"deg":188,"clouds":24,"rain":1.45},{"dt":1509876000,"temp":{"day":16.25,"min":13.53,"max":16.25,"night":13.53,"eve":13.71,"morn":14.45},"pressure":1019.61,"humidity":0,"weather":{"id":500,"main":"Rain","description":"light rain","icon":"10d"},"speed":2.96,"deg":24,"clouds":76,"rain":2.15}]}
```

Refactor Project components



```
SecondFragment.java | FirstFragment.java
1 package com.example.weatherapp;
2
3 import ...
4
12
13 public class SecondFragment extends Fragment {
14
15     @Override
16     public View onCreateView(
17         LayoutInflater inflater, ViewGroup container,
18         Bundle savedInstanceState
19     ) {
20         ArrayAdapter<String> mForecastAdapter;
21         String[] data = {
22             "Today 3/11 - Sunny - 17 C",
23             "Fri 4/11 - Cloudy - 18 C",
24             "Sat 5/11 - Rainy - 17 C",
25             "Sun 6/11 - Sunny - 19 C",
26             "Mon 7/1 - Sunny - 19 C",
27             "Tues 8/11 - Rainy - 18 C",
28             "Wed 9/11 - TRAPPED IN WEATHERSTATION - 10 C"
29         };
30
31         List<String> weekForecast = new ArrayList<>(Arrays.asList(data));
32         mForecastAdapter = new ArrayAdapter<String>(getActivity(), // The current context (this activity)
33             R.layout.list_item_forecast, // The name of the layout ID,
34             R.id.list_item_forecast_textview, // The ID of the textview to populate.
35             weekForecast);
36         View rootView = inflater.inflate(R.layout.fragment_second, container, attachToRoot false);
37         ListView listView = (ListView) rootView.findViewById(R.id.listview_forecast);
38         listView.setAdapter(mForecastAdapter);
39         return rootView;
40     }
41 }
42
43 }
```

Array Adapter should be public to be amendable



```
12 import ...
13 public class SecondFragment extends Fragment {
14     public ArrayAdapter<String> mForecastAdapter;
15
16     @Override
17     public View onCreateView(
18         LayoutInflater inflater, ViewGroup container,
19         Bundle savedInstanceState
20     ){
21         String[] data = {
22             "Today 3/11 - Sunny - 17 C",
23             "Fri 4/11 - Cloudy - 18 C",
24             "Sat 5/11 - Rainy - 17 C",
25             "Sun 6/11 - Sunny - 19 C",
26             "Mon 7/1 - Sunny - 19 C",
27             "Tues 8/11 - Rainy - 18 C",
28             "Wed 9/11 - TRAPPED IN WEATHERSTATION - 10 C"
29         };
30
31         List<String> weekForecast = new ArrayList<>(Arrays.asList(data));
32         mForecastAdapter = new ArrayAdapter<String>(getActivity(), // The current context (this activity)
33             R.layout.list_item_forecast, // The name of the layout ID.
34             R.id.list_item_forecast_textview, // The ID of the textview to populate.
35             weekForecast);
36         View rootView = inflater.inflate(R.layout.fragment_second, container, attachToRoot: false);
37         ListView listView = (ListView) rootView.findViewById(R.id.listview_forecast);
38         listView.setAdapter(mForecastAdapter);
39         return rootView;
40     }
41 }
42 }
43 }
44 }
```

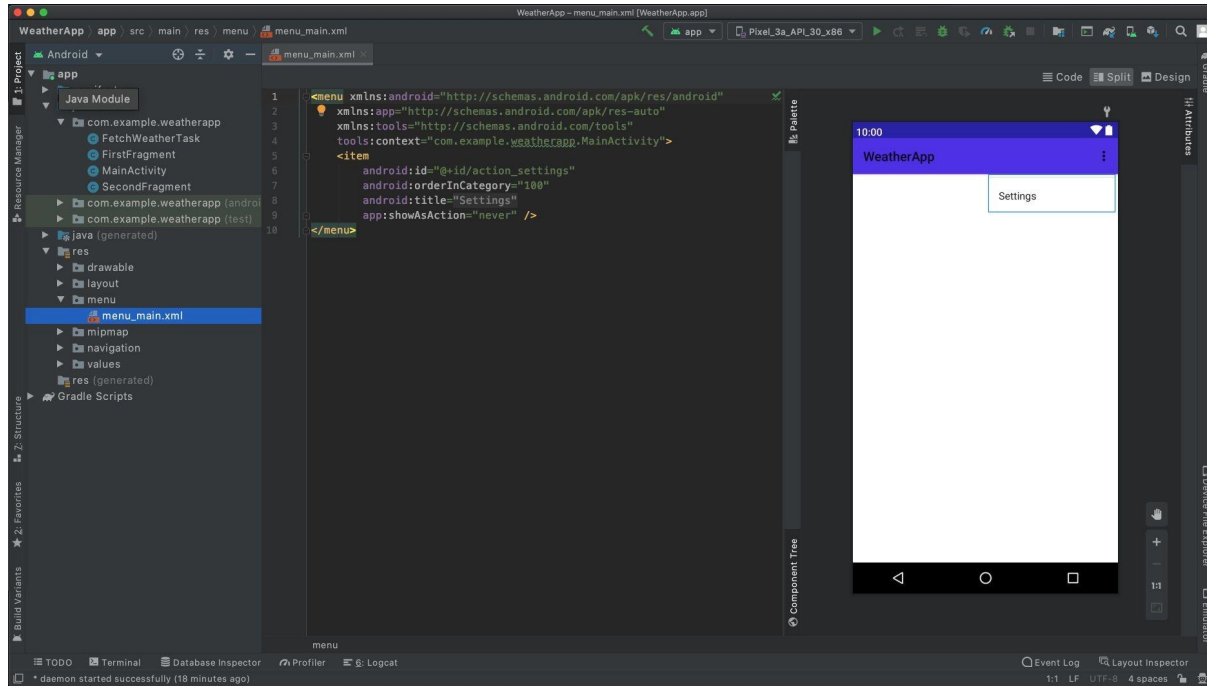
Turn Array Adapter should be public to be amendable



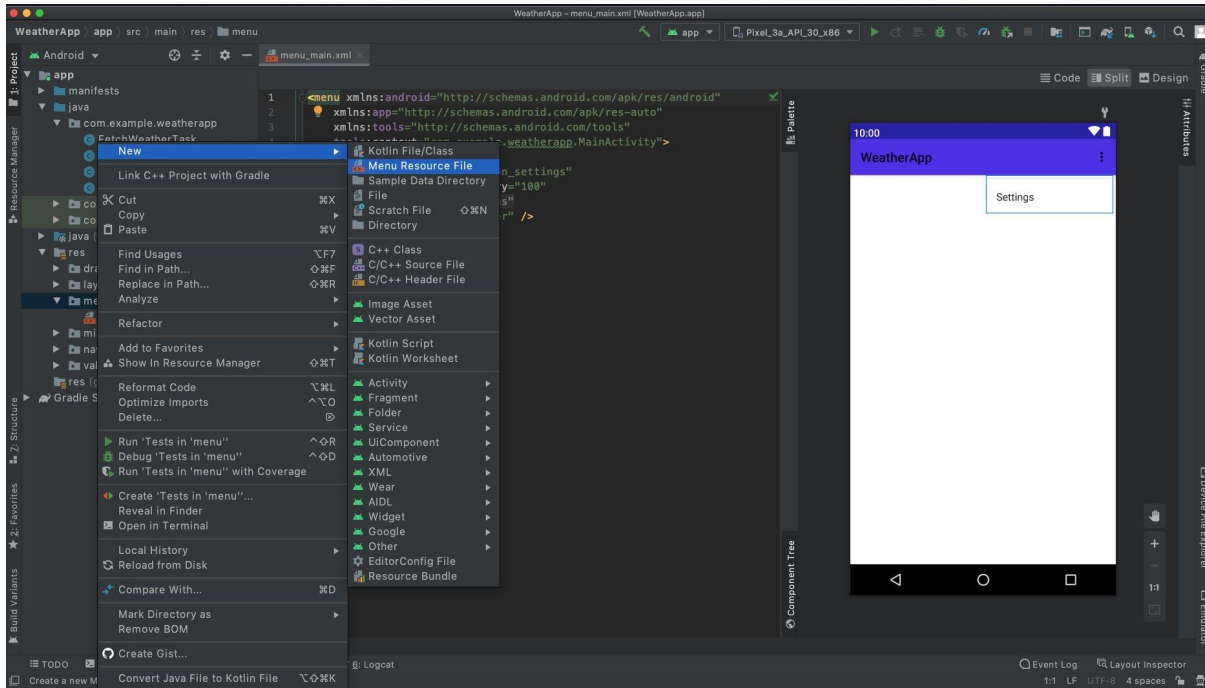
```
setHasOptionsMenu(true);
```

Inside onCreateView()

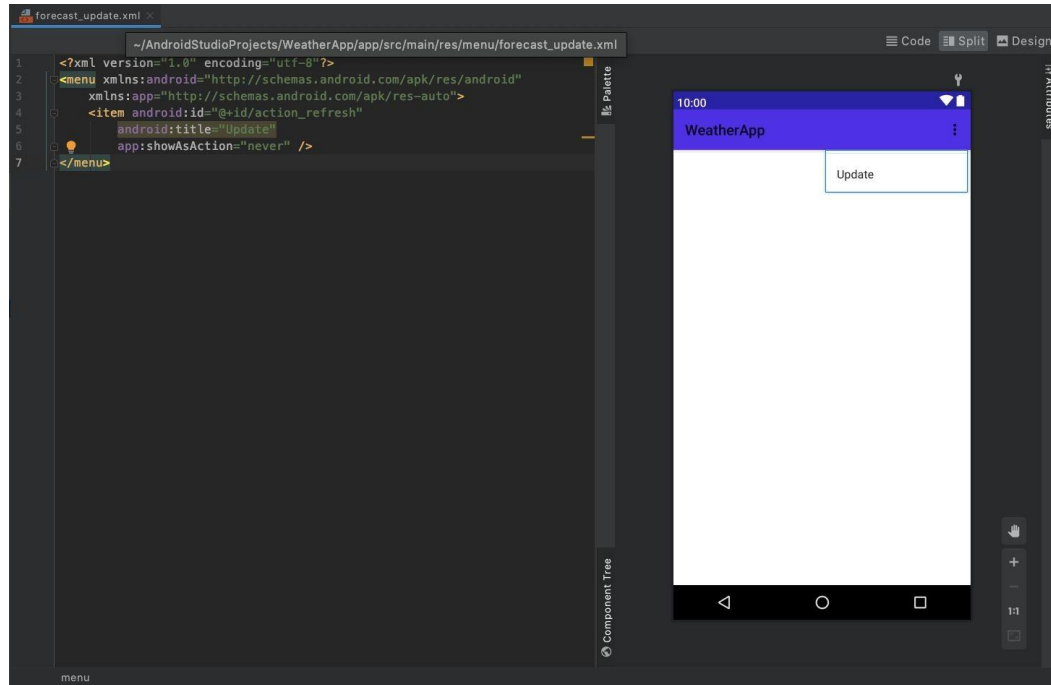
Create new Menu option to update Forecast



Current menu for first (home) fragment (Settings option)



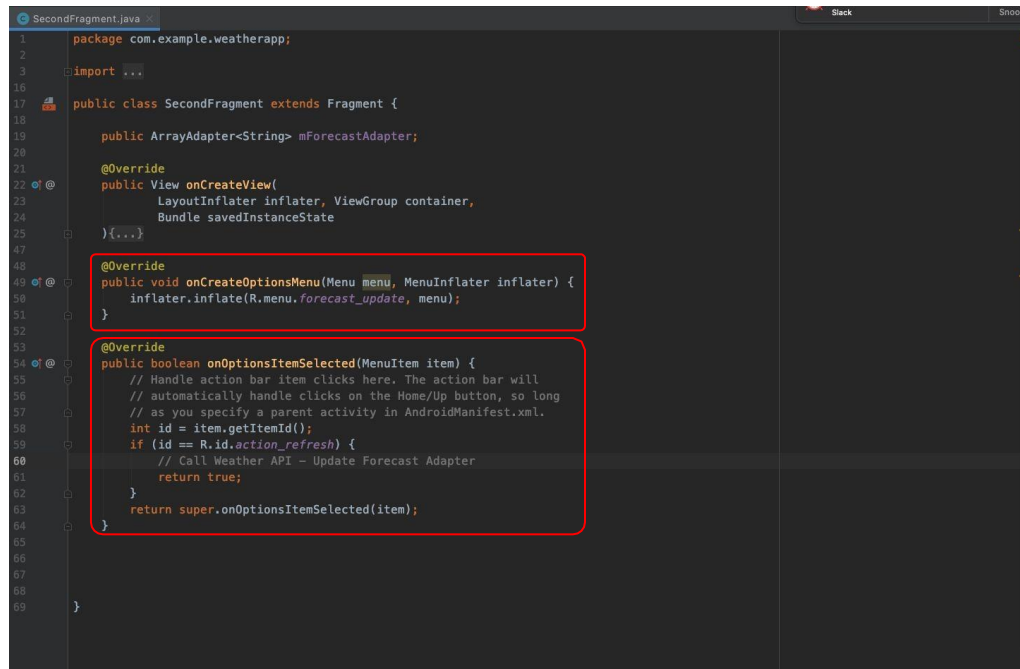
Create new menu for second (Forecast) fragment



Add menu option (Update Forecast)



```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android"
      xmlns:app="http://schemas.android.com/apk/res-auto">
  <item android:id="@+id/action_refresh"
        android:title="Update"
        app:showAsAction="never" />
</menu>
```

```
1 package com.example.weatherapp;
2
3 import ...
4
16
17 public class SecondFragment extends Fragment {
18
19     public ArrayAdapter<String> mForecastAdapter;
20
21     @Override
22     public View onCreateView(
23         LayoutInflater inflater, ViewGroup container,
24         Bundle savedInstanceState
25     ){...}
26
47
48     @Override
49     public void onCreateOptionsMenu(Menu menu, MenuInflater inflater) {
50         inflater.inflate(R.menu.forecast_update, menu);
51     }
52
53
54     @Override
55     public boolean onOptionsItemSelected(MenuItem item) {
56         // Handle action bar item clicks here. The action bar will
57         // automatically handle clicks on the Home/Up button, so long
58         // as you specify a parent activity in AndroidManifest.xml.
59         int id = item.getItemId();
60         if (id == R.id.action_refresh) {
61             // Call Weather API - Update Forecast Adapter
62             return true;
63         }
64         return super.onOptionsItemSelected(item);
65     }
66
67
68
69 }
```

Update business logic to support new menu



```
import android.view.MenuInflater;
```


```
@Override
```

```
public void onCreateOptionsMenu(Menu menu, MenuInflater inflater) {  
    inflater.inflate(R.menu.forecast_update, menu);  
}
```

```
@Override
```

```
public boolean onOptionsItemSelected(MenuItem item) {  
    // Handle action bar item clicks here. The action bar will  
    // automatically handle clicks on the Home/Up button, so long  
    // as you specify a parent activity in AndroidManifest.xml.  
    int id = item.getItemId();  
    if (id == R.id.action_refresh) {  
        // Call Weather API - Update Forecast Adapter  
        return true;  
    }  
    return super.onOptionsItemSelected(item);  
}
```

Add Internet permissions



```
1 <?xml version="1.0" encoding="utf-8"?>
2 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3 package="com.example.weatherapp">
4 <uses-permission android:name="android.permission.INTERNET"/>
5 <application
6     android:allowBackup="true"
7     android:icon="@mipmap/ic_launcher"
8     android:label="WeatherApp"
9     android:roundIcon="@mipmap/ic_launcher_round"
10    android:supportsRtl="true"
11    android:theme="@style/Theme.WeatherApp">
12    <activity
13        android:name=".MainActivity"
14        android:label="WeatherApp"
15        android:theme="@style/Theme.WeatherApp.NoActionBar">
16        <intent-filter>
17            <action android:name="android.intent.action.MAIN" />
18
19            <category android:name="android.intent.category.LAUNCHER" />
20        </intent-filter>
21    </activity>
22 </application>
23
24 </manifest>
```

manifest > uses-permission

Text Merged Manifest

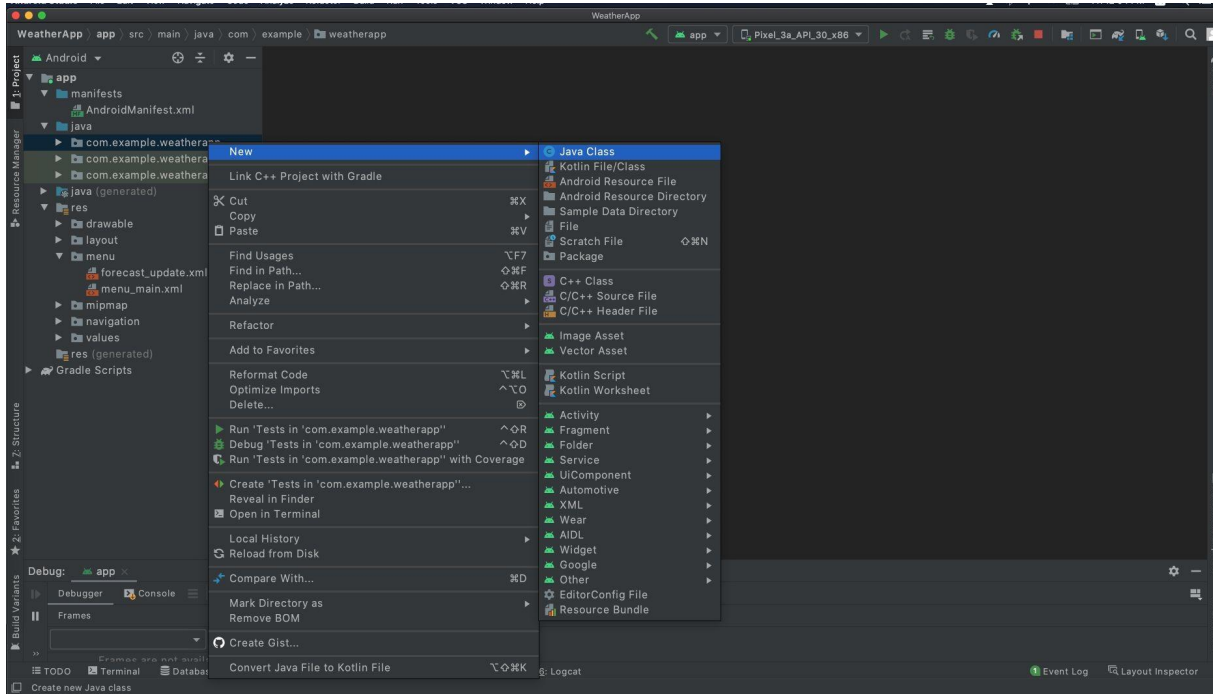
Add permission for internet in
manifest

Create Asynchronous Task to call API

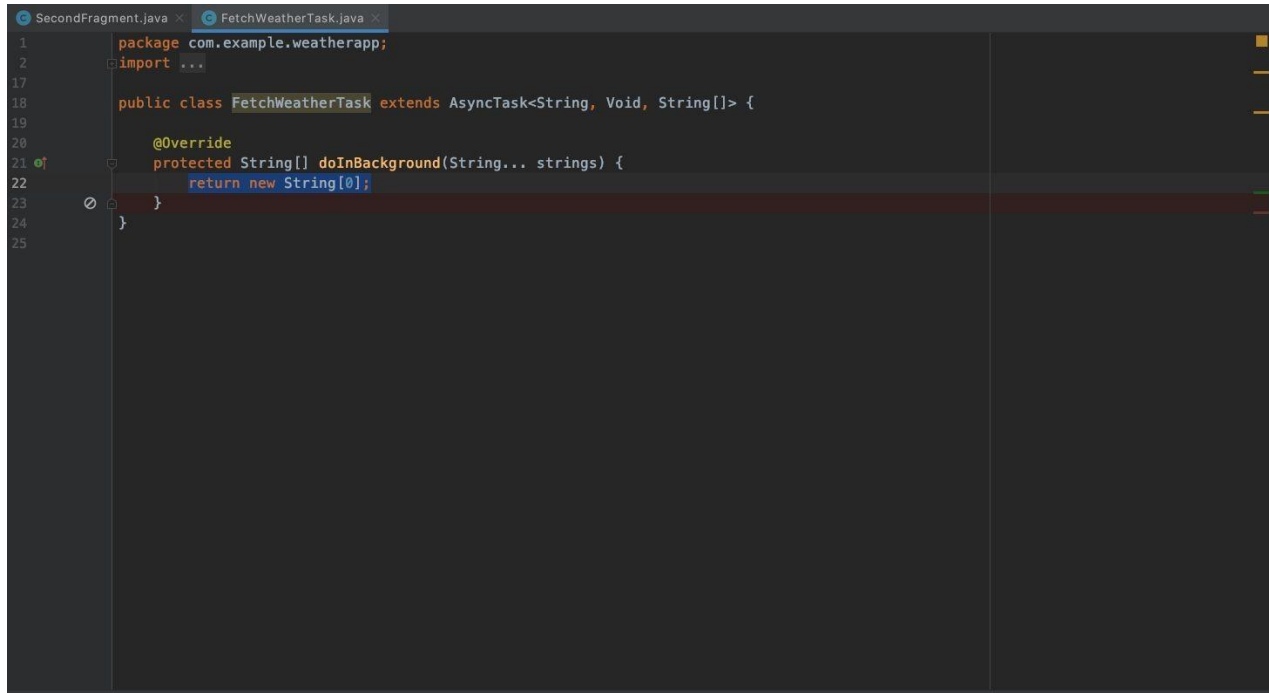


Next we need to apply a HTTP Request for weather forecast:

- Make HTTP Requests using the API URL
- Read HTTP Responses from the input stream in JSON format
- Clean up and Save the results
- Log any errors




Create new Java class for Asynchronous Task



```
1 package com.example.weatherapp;
2 import ...
17
18 public class FetchWeatherTask extends AsyncTask<String, Void, String[]> {
19
20     @Override
21     protected String[] doInBackground(String... strings) {
22         return new String[0];
23     }
24 }
25
```

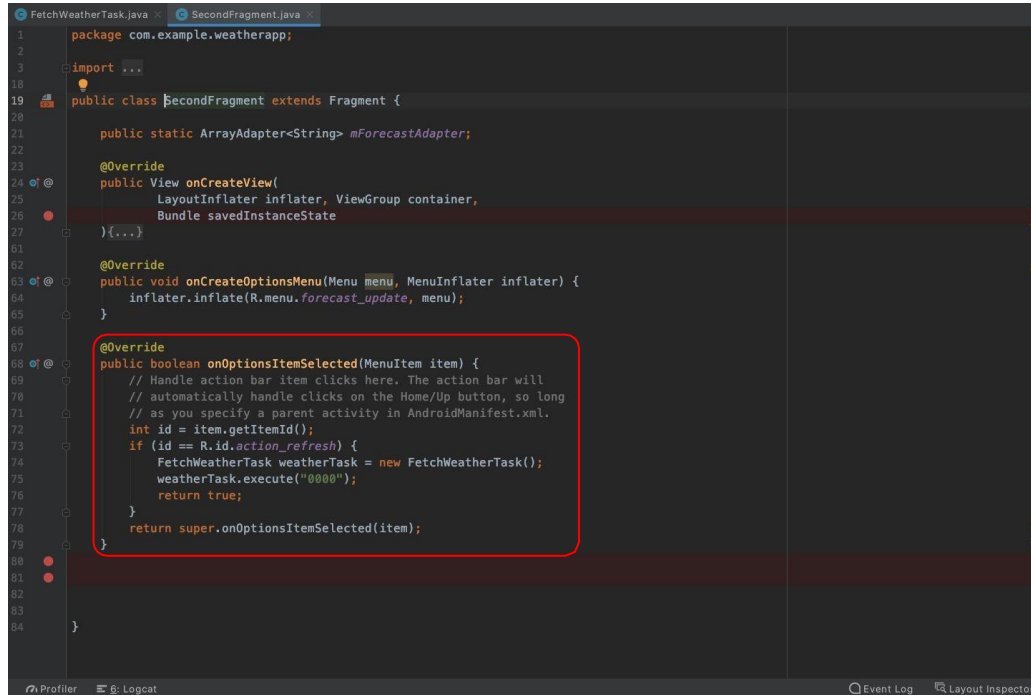
FetchWeatherTask extends AsyncTask



```
1 package com.example.weatherapp;
2 import ...
17
18 public class FetchWeatherTask extends AsyncTask<String, Void, String[]> {
19     private final String LOG_TAG = FetchWeatherTask.class.getSimpleName();
20
21     /** Prepare the weather high/lows for presentation. */
24     @ private String formatHighLows(double high, double low) {...}
32
33     /** Take the String representing the complete forecast in JSON Format and ...*/
40     @ private String[] getWeatherDataFromJson(String forecastJsonStr, int numDays)
41         throws JSONException {...}
96
97     @Override
98     @ protected String[] doInBackground(String... params) {...}
176
177     @Override
178     @ protected void onPostExecute(String[] result) {...}
187 }
188
```

Build business logic to fetch and preview forecast

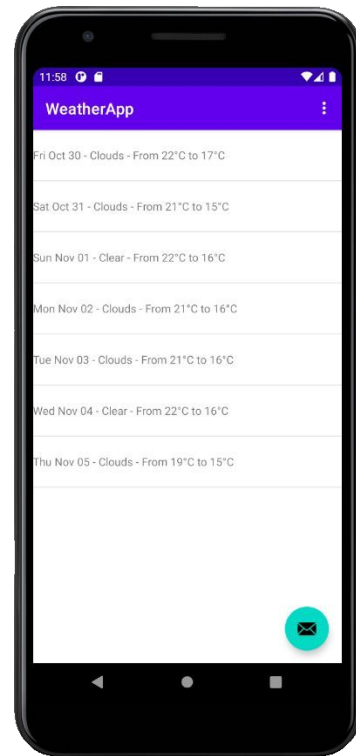
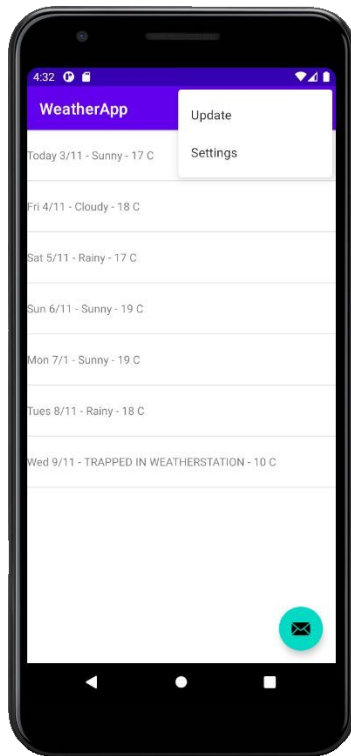
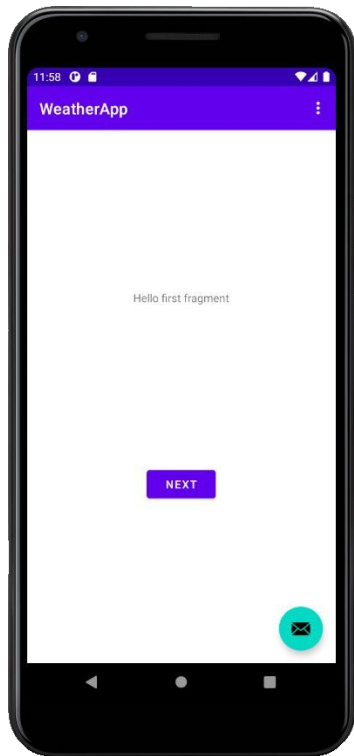
Call Asynchronous Task



```
1 package com.example.weatherapp;
2
3 import ...
4
18
19 public class SecondFragment extends Fragment {
20
21     public static ArrayAdapter<String> mForecastAdapter;
22
23     @Override
24     public View onCreateView(
25         LayoutInflater inflater, ViewGroup container,
26         Bundle savedInstanceState
27     ){...}
28
29
30
31
32     @Override
33     public void onCreateOptionsMenu(Menu menu, MenuInflater inflater) {
34         inflater.inflate(R.menu.forecast_update, menu);
35     }
36
37
38     @Override
39     public boolean onOptionsItemSelected(MenuItem item) {
40         // Handle action bar item clicks here. The action bar will
41         // automatically handle clicks on the Home/Up button, so long
42         // as you specify a parent activity in AndroidManifest.xml.
43         int id = item.getItemId();
44         if (id == R.id.action_refresh) {
45             FetchWeatherTask weatherTask = new FetchWeatherTask();
46             weatherTask.execute("0000");
47             return true;
48         }
49         return super.onOptionsItemSelected(item);
50     }
51
52
53
54 }
```

Build business logic to fetch and preview forecast

Demonstrate Updates





Review of Lab 2

- Catch up with Lab 1
- Introduce OpenWeatherMap API
- Update code - Use Public Forecast API
 - Make property publicly available
 - REST call to API - JSON response
 - Create new menu option to update forecast
 - Add internet permission
 - Create Asynchronous Task to call API
 - Call Asynchronous Task
 - Demonstrate updates