Slides created by Ilias Chalkidis

Android Application Development Lab 2

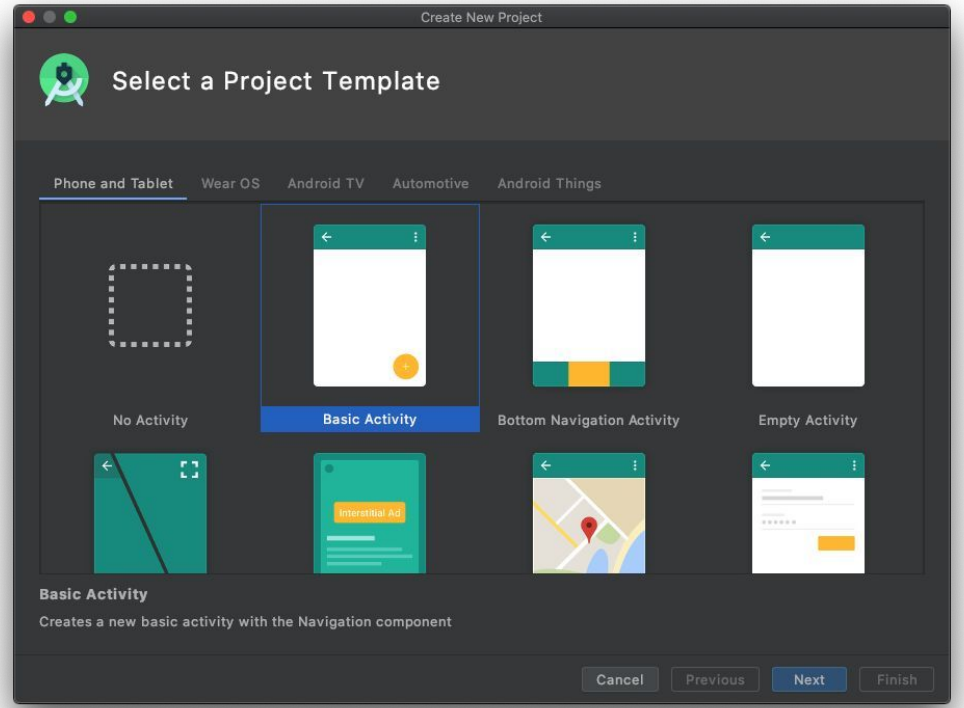
Human-Computer Interaction, AUEB
Εαρινό εξάμηνο 2024-2025

Lab Assistant: Sofia Eleftheriou

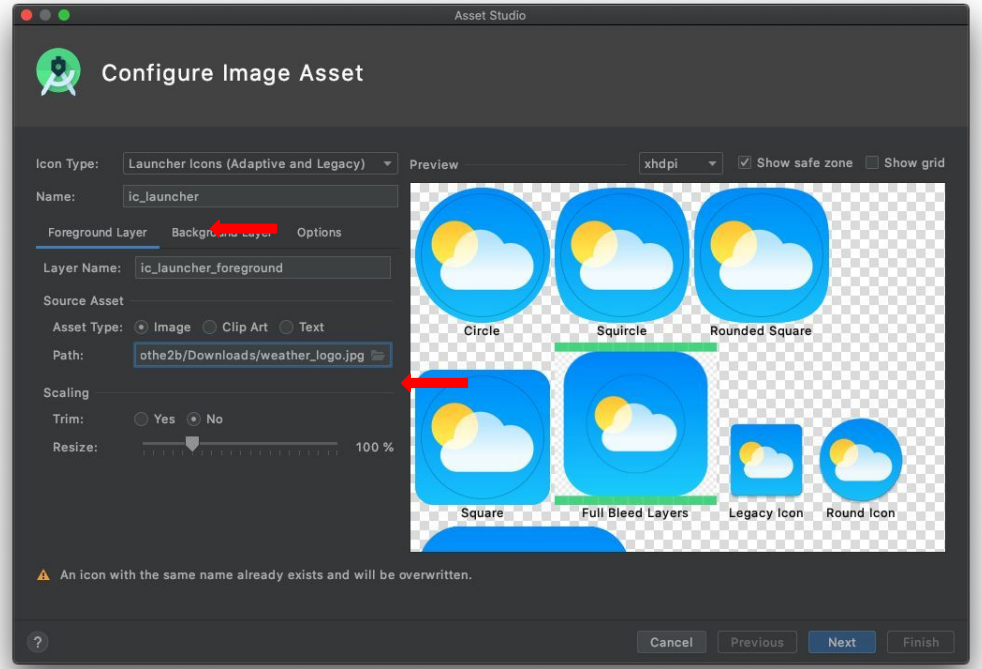
Android Development Advanced Use

- 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

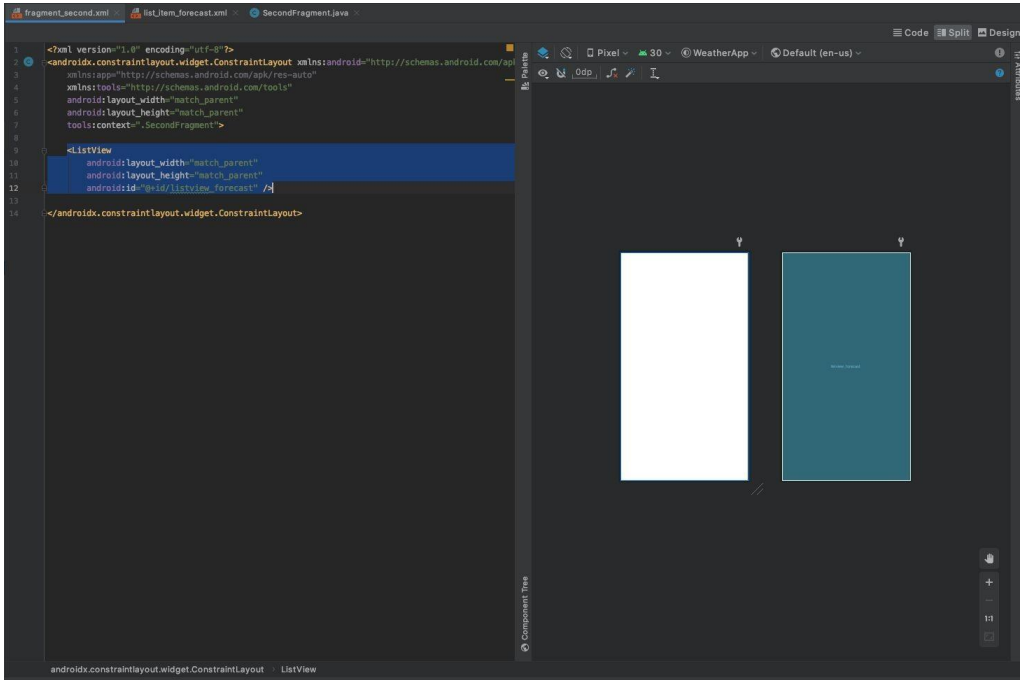
Catch up with Lab 1



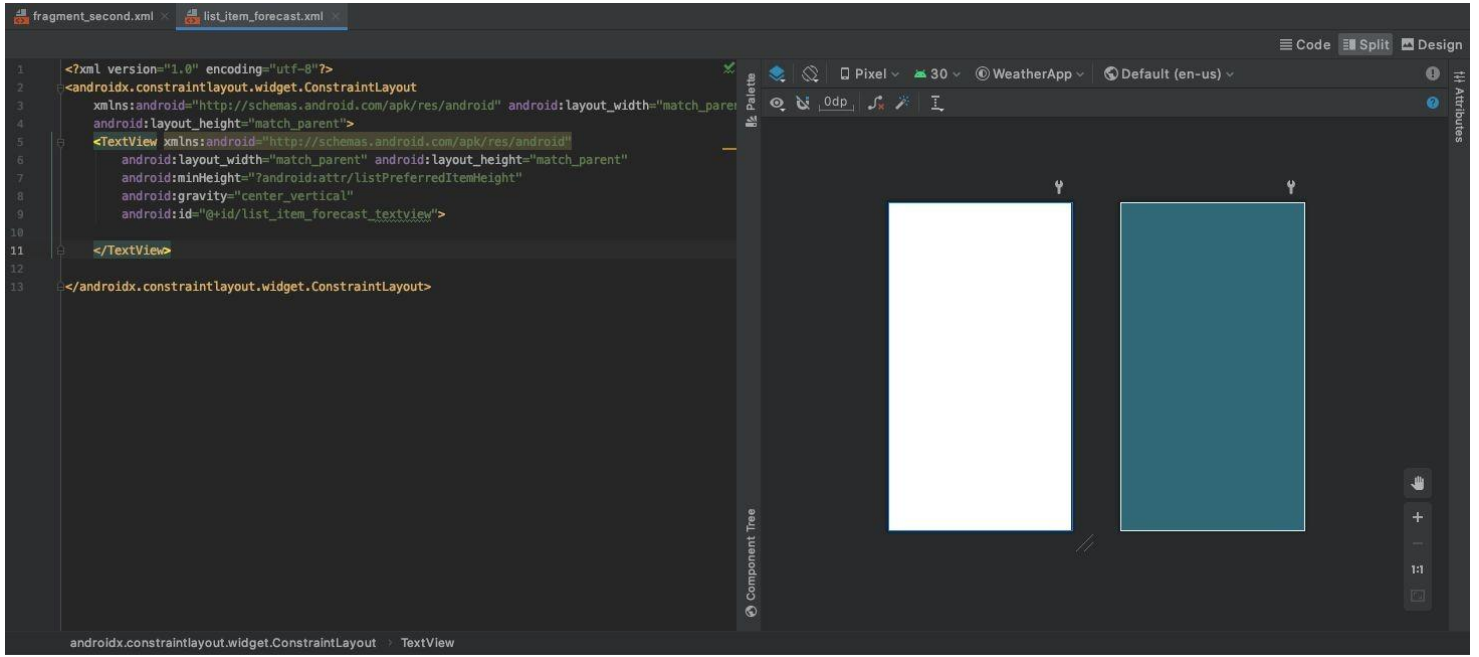
Create new project with Basic Activity



Change Application Icon



Add ListView to layout

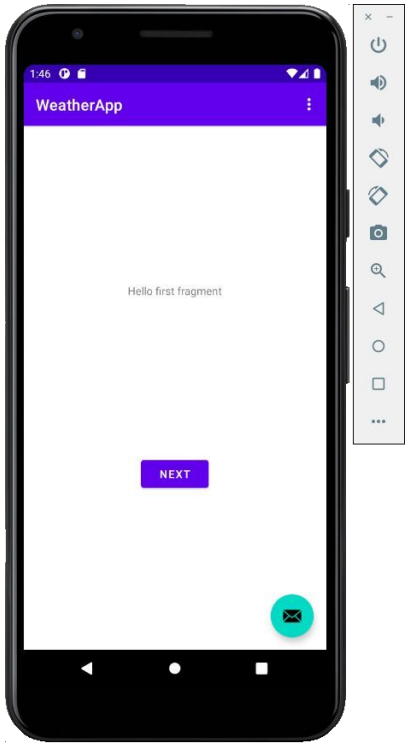


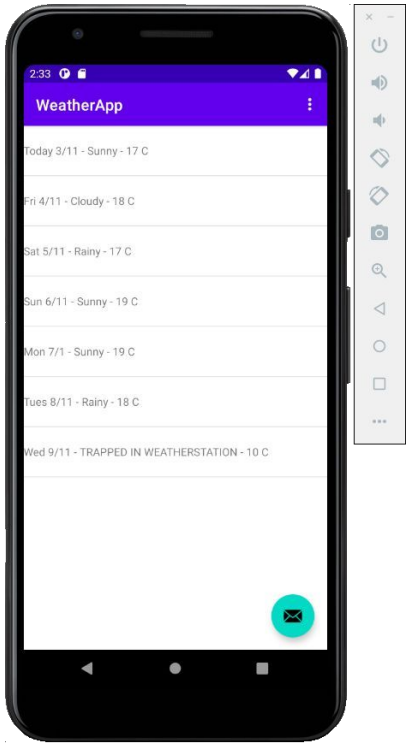
Add new layout with TextView for list items

SecondFragment.java

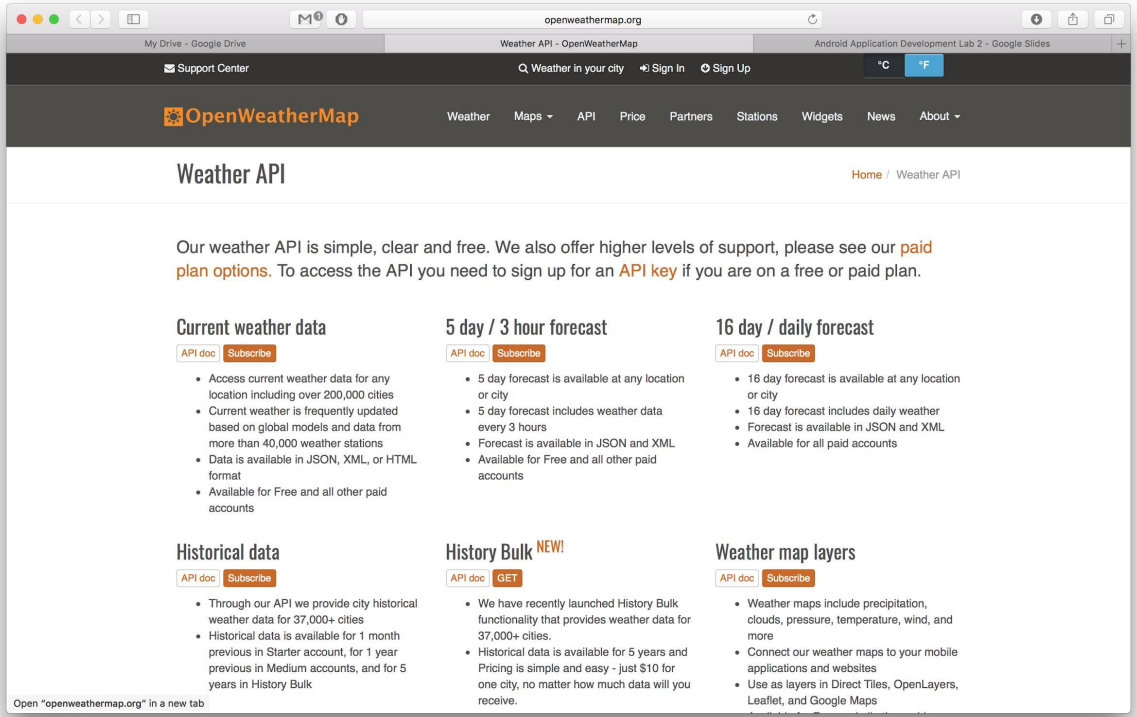
```java
package com.example.weatherapp;

import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import androidx.fragment.app.Fragment;
import java.util.*;
import android.widget.ArrayAdapter;
import android.widget.ListView;

public class SecondFragment extends Fragment {

    @Override
    public View onCreateView(
            LayoutInflater inflater, ViewGroup container,
            Bundle savedInstanceState
    ) {
        ArrayAdapter<String> mForecastAdapter;
        String[] data = {
                "Today 3/11 - Sunny - 17 C",
                "Fri 4/11 - Cloudy - 18 C",
                "Sat 5/11 - Rainy - 17 C",
                "Sun 6/11 - Sunny - 19 C",
                "Mon 7/1 - Sunny - 19 C",
                "Tues 8/11 - Rainy - 18 C",
                "Wed 9/11 - TRAPPED IN WEATHERSTATION - 10 C"
        };

        List<String> weekForecast = new ArrayList<String>(Arrays.asList(data));
        mForecastAdapter = new ArrayAdapter<String>(getActivity(), // The current context (this activity)
                R.layout.list_item_forecast, // The name of the layout ID.
                R.id.list_item_forecast_textview, // The ID of the textview to populate.
                weekForecast);
        View rootView = inflater.inflate(R.layout. fragment_second, container,  attachToRoot: false);
        ListView listView = (ListView) rootView.findViewById(R.id. listview_forecast);
        listView.setAdapter(mForecastAdapter);
        return rootView;

    }

}
```

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

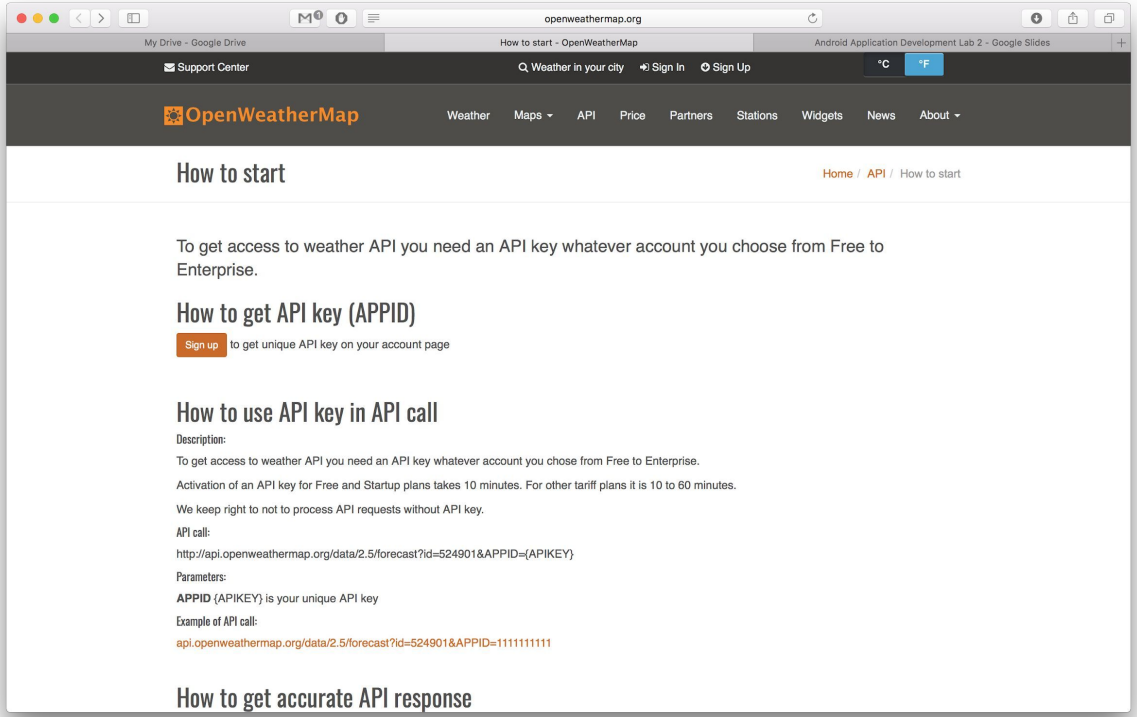

1:46
WeatherApp
Hello first fragment
NEXT


2:33
WeatherApp
Today 3/11 - Sunny - 17 C
Fri 4/11 - Cloudy - 18 C
Sat 5/11 - Rainy - 17 C
Sun 6/11 - Sunny - 19 C
Mon 7/1 - Sunny - 19 C
Tues 8/11 - Rainy - 18 C
Wed 9/11 - TRAPPED IN WEATHERSTATION - 10 C

OpenWeatherMap API



http://openweathermap.org



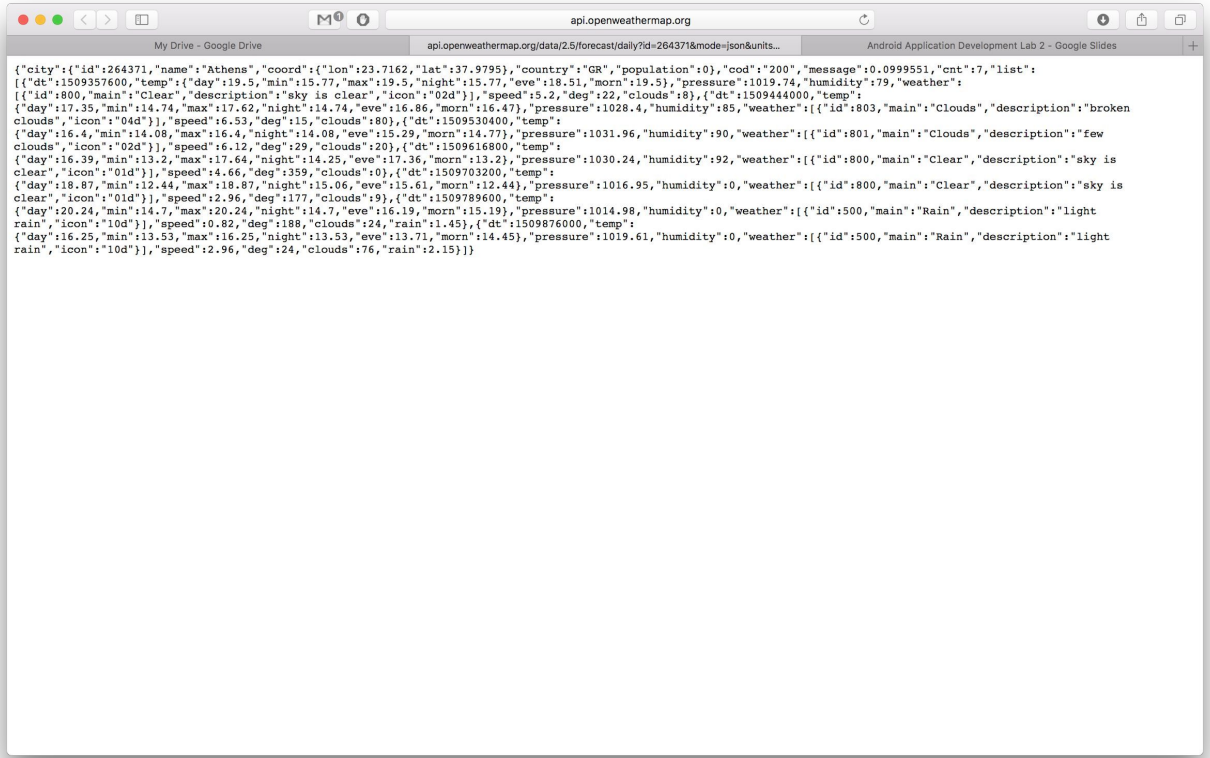
Register & Get API key

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

```java
package com.example.weatherapp;

import ...

public class SecondFragment extends Fragment {

    @Override
    public View onCreateView(
            LayoutInflater inflater, ViewGroup container,
            Bundle savedInstanceState
    ) {
        ArrayAdapter<String> mForecastAdapter;
        String[] data = {
                "Today 3/11 - Sunny - 17 C",
                "Fri 4/11 - Cloudy - 18 C",
                "Sat 5/11 - Rainy - 17 C",
                "Sun 6/11 - Sunny - 19 C",
                "Mon 7/1 - Sunny - 19 C",
                "Tues 8/11 - Rainy - 18 C",
                "Wed 9/11 - TRAPPED IN WEATHERSTATION - 10 C"
        };

        List<String> weekForecast = new ArrayList<>(Arrays.asList(data));
        mForecastAdapter = new ArrayAdapter<String>(getActivity(), // The current context (this activity)
                R.layout.list_item_forecast, // The name of the layout ID.
                R.id.list_item_forecast_textview, // The ID of the textview to populate.
                weekForecast);
        View rootView = inflater.inflate(R.layout. fragment_second, container, attachToRoot: false);
        ListView listView = (ListView) rootView.findViewById(R.id. listview_forecast);
        listView.setAdapter(mForecastAdapter);
        return rootView;

    }

}
```

Array Adapter should be public to be amendable

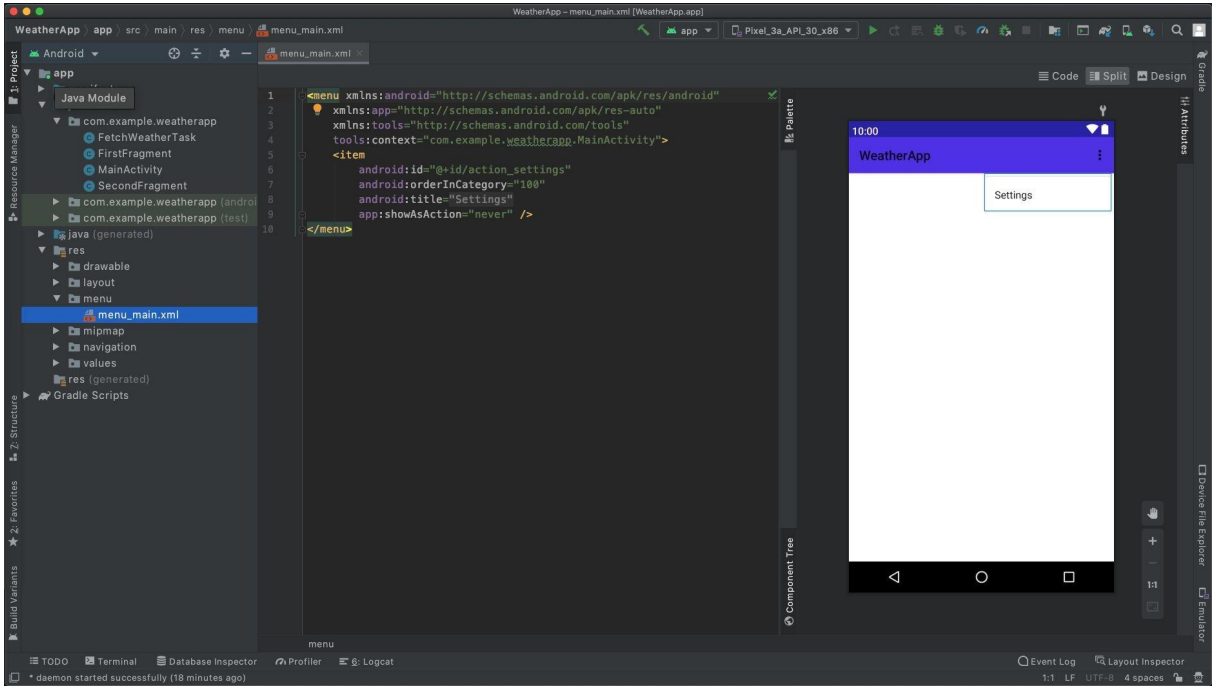
SecondFragment.java | FirstFragment.java

```
import ...

public class SecondFragment extends Fragment {

    public ArrayAdapter<String> mForecastAdapter;

    @Override
    public View onCreateView(
            LayoutInflater inflater, ViewGroup container,
            Bundle savedInstanceState
    ){
        String[] data = {
                "Today 3/11 - Sunny - 17 C",
                "Fri 4/11 - Cloudy - 18 C",
                "Sat 5/11 - Rainy - 17 C",
                "Sun 6/11 - Sunny - 19 C",
                "Mon 7/1 - Sunny - 19 C",
                "Tues 8/11 - Rainy - 18 C",
                "Wed 9/11 - TRAPPED IN WEATHERSTATION - 10 C"
        };

        List<String> weekForecast = new ArrayList<>(Arrays.asList(data));
        mForecastAdapter = new ArrayAdapter<String>(getActivity(), // The current context (this activity)
                R.layout.list_item_forecast, // The name of the layout ID.
                R.id.list_item_forecast_textview, // The ID of the textview to populate.
                weekForecast);
        View rootView = inflater.inflate(R.layout. fragment_second, container, attachToRoot: false);
        ListView listView = (ListView) rootView.findViewById(R.id. listview_forecast);
        listView.setAdapter(mForecastAdapter);
        return rootView;

    }

}
```

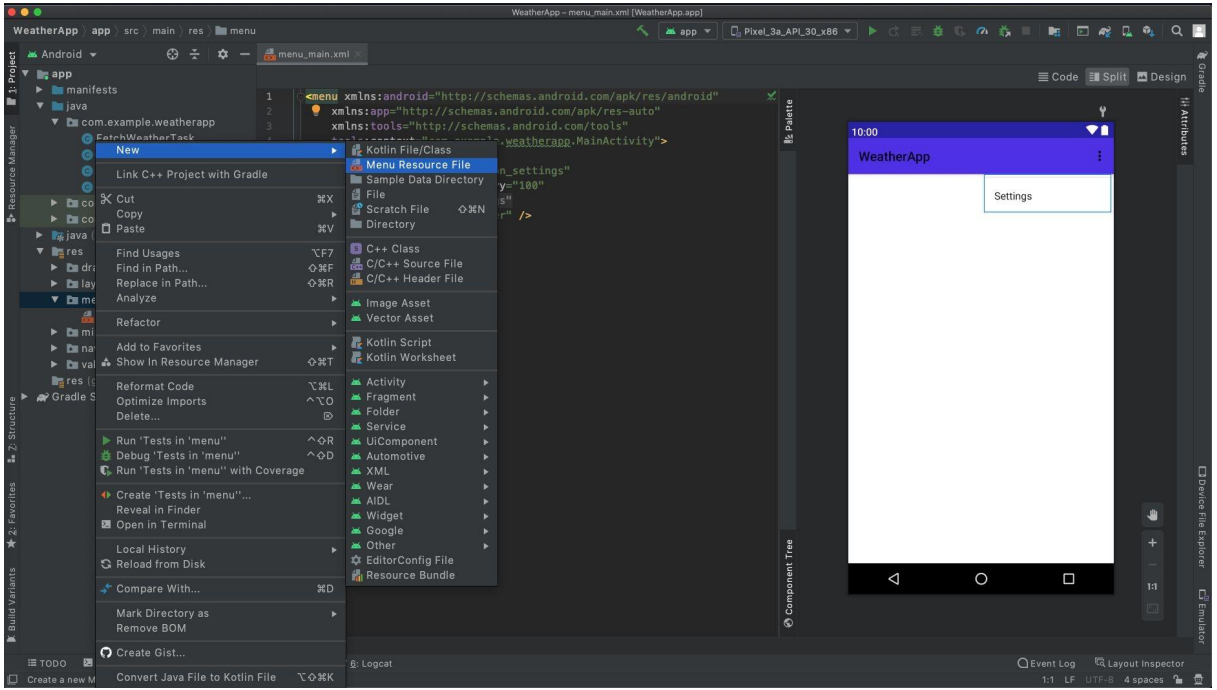
Turn Array Adapter should be public to be amendable

```
public static ArrayAdapter<String> mForecastAdapter;
```

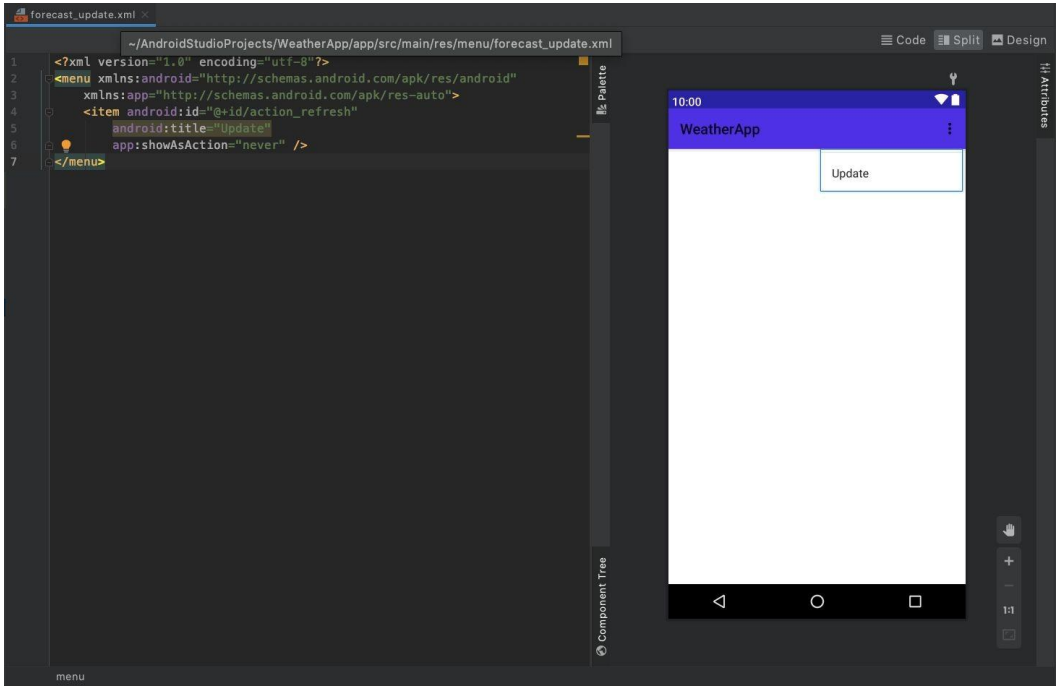
Create new Menu option to update Forecast



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

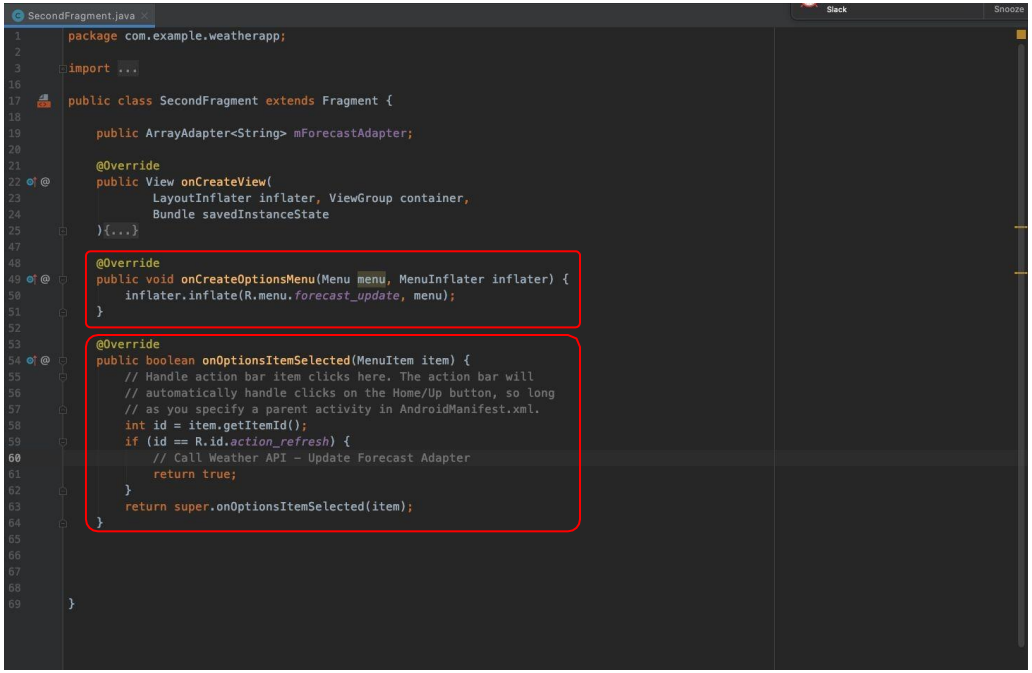


Create new menu for second (Forecast) fragment



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>
```



Update business logic to support new menu

```
import android.view.MenuInflater;
import android.view.Menu;
import android.view.MenuItem;
import androidx.lifecycle.Lifecycle;
```

```
@Override
public void onViewCreated(@NonNull View view, Bundle savedInstanceState) {
   super.onViewCreated(view, savedInstanceState);

   // Get the MenuHost from the parent Activity
   MenuHost menuHost = requireActivity();

   // Add a new MenuProvider to handle the options menu in this fragment
   menuHost.addMenuProvider(new MenuProvider() {

       @Override
       public void onCreateMenu(@NonNull Menu menu, @NonNull MenuInflater menuInflater) {
           // Inflate the menu resource (R.menu.forecast_update) into the fragment's menu
           menuInflater.inflate(R.menu.forecast_update, menu);
       }

       @Override
       public boolean onMenuItemSelected(@NonNull MenuItem item) {
           // Handle menu item clicks
           if (item.getItemId() == R.id.action_refresh) {
               FetchWeatherForecast weatherTask = new FetchWeatherForecast();
               weatherTask.executeOnExecutor(AsyncTask.THREAD_POOL_EXECUTOR, "0000");
               return true; // Indicate that the event was handled
           }
           return false; // Let the system handle other menu actions
       }
   }, getViewLifecycleOwner(), Lifecycle.State.RESUMED); // Ensure the menu is only active when the fragment is visible
}
```

Add Internet permissions

AndroidManifest.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.weatherapp">
    <uses-permission android:name="android.permission.INTERNET"/>
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="WeatherApp"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.WeatherApp">
        <activity
            android:name=".MainActivity"
            android:label="WeatherApp"
            android:theme="@style/Theme.WeatherApp.NoActionBar">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

manifest › uses-permission

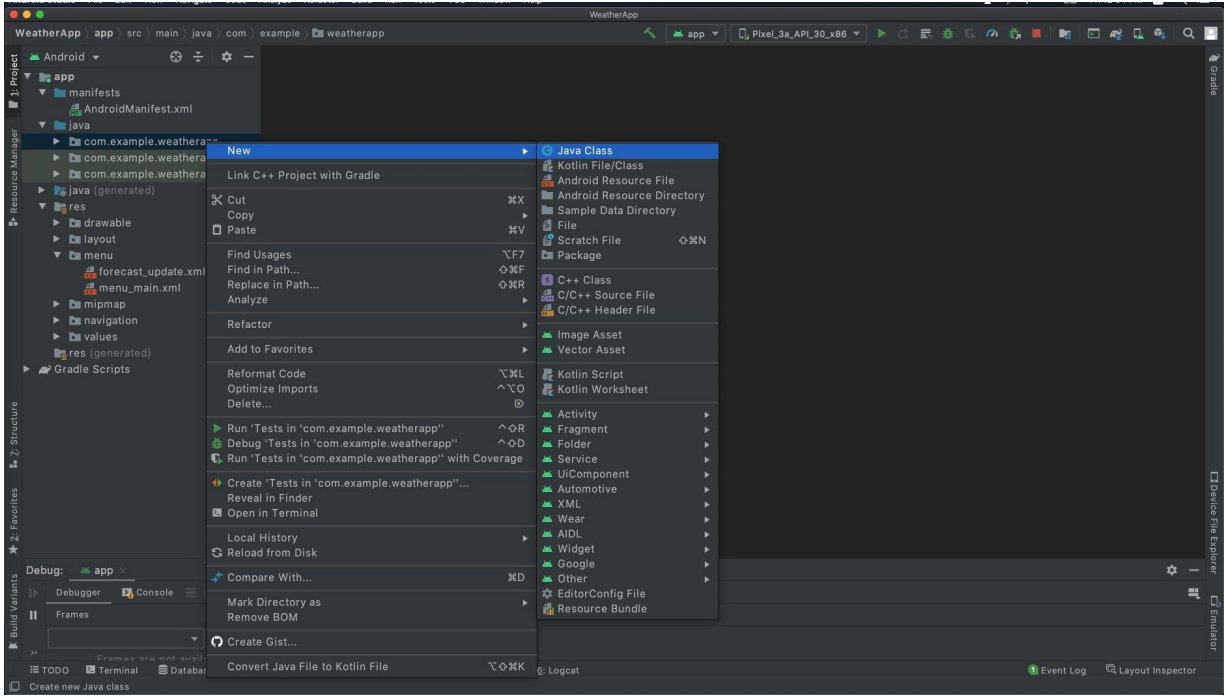
Text | Merged Manifest

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

SecondFragment.java | FetchWeatherTask.java

```java
package com.example.weatherapp;
import ...

public class FetchWeatherTask extends AsyncTask<String, Void, String[]> {

    @Override
    protected String[] doInBackground(String... strings) {
        return new String[0];
    }
}
```

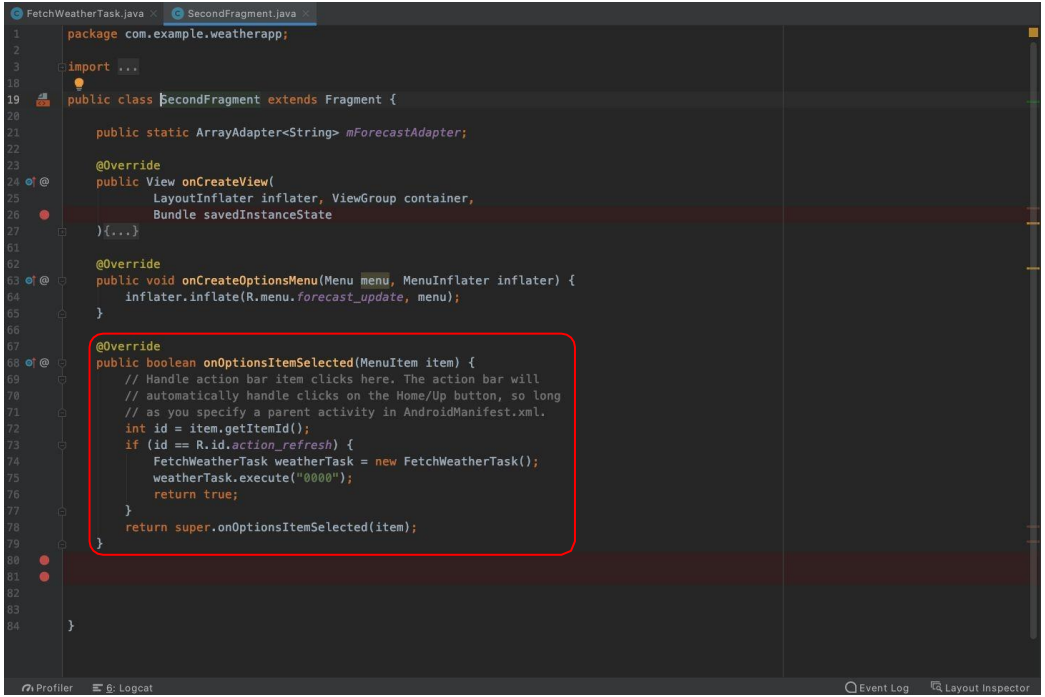
FetchWeatherTask extends AsynchTask

SecondFragment.java | FetchWeatherTask.java

```
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

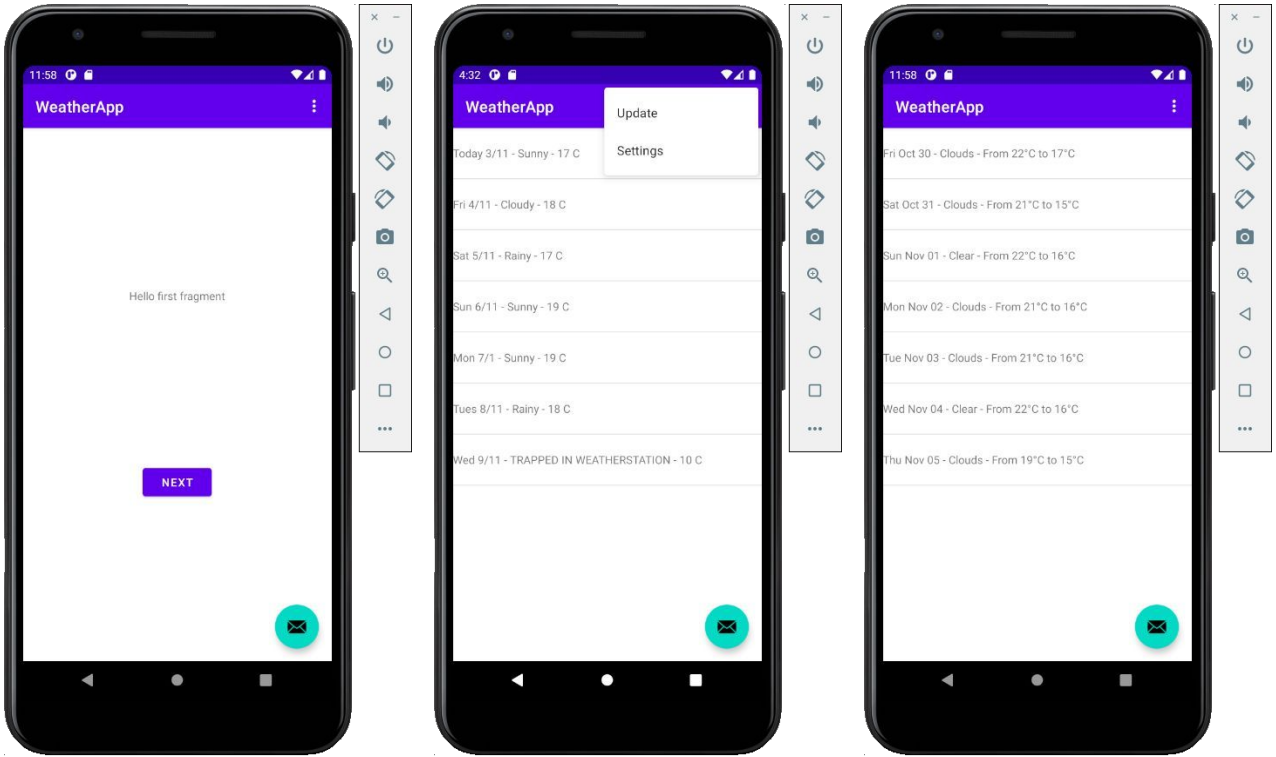
Call Asynchronous Task



Build business logic to fetch and preview forecast

```
import android.os.AsyncTask;


@Override
public boolean onMenuItemSelected(@NonNull MenuItem item) {
    if (item.getItemId() == R.id.action_refresh) {
        FetchWeatherForecast weatherTask = new FetchWeatherForecast();
        weatherTask.executeOnExecutor(AsyncTask.THREAD_POOL_EXECUTOR, "0000");
        return true;
    }
    return false;
}
```

Demonstrate Updates


11:58
WeatherApp
Hello first fragment
NEXT
4:32
WeatherApp
Update
Settings
Today 3/11 - Sunny - 17 C
Fri 4/11 - Cloudy - 18 C
Sat 5/11 - Rainy - 17 C
Sun 6/11 - Sunny - 19 C
Mon 7/1 - Sunny - 19 C
Tues 8/11 - Rainy - 18 C
Wed 9/11 - TRAPPED IN WEATHERSTATION - 10 C
11:58
WeatherApp
Fri Oct 30 - Clouds - From 22°C to 17°C
Sat Oct 31 - Clouds - From 21°C to 15°C
Sun Nov 01 - Clear - From 22°C to 16°C
Mon Nov 02 - Clouds - From 21°C to 16°C
Tue Nov 03 - Clouds - From 21°C to 16°C
Wed Nov 04 - Clear - From 22°C to 16°C
Thu Nov 05 - Clouds - From 19°C to 15°C

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