Slides created by Ilias Chalkidis

Android Application Development Lab 1

Human-Computer Interaction, AUEB
Εαρινό εξάμηνο 2023-2024

Lab Assistant: Sofia Eleftheriou

Android Development Fundamentals

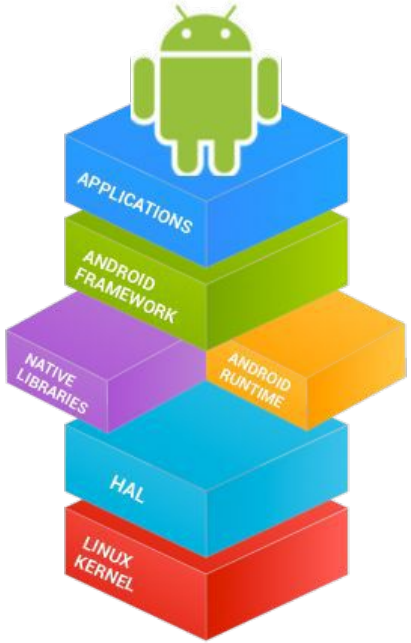
- Android SDK and Development Tools
- Android State Machine
- Android Manifest

Android SDK & Development Tool

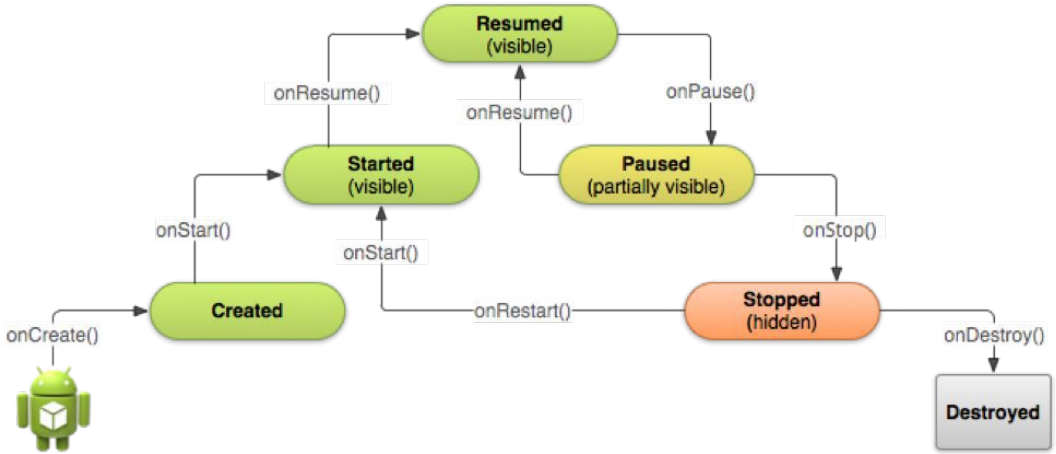
A **software development kit** that enables developers to create applications for the Android platform. The Android SDK includes:

- sample projects with source code,
- development tools,
- an emulator,
- a debugger, and
- required libraries to build Android applications.

Applications are written using the Java programming language and run on Dalvik, a custom virtual machine designed for embedded use which runs on top of a Linux kernel.



Android State Machine



Android State Machine

```
public class Activity extends ApplicationContext {
    protected void onCreate(Bundle savedInstanceState);
    protected void onStart();

    protected void onRestart();

    protected void onResume();

    protected void onPause();

    protected void onStop();

    protected void onDestroy();

}
```

Android Manifest

The Android Application Manifest file includes nodes for each of the following components:

- Activities
- Services
- Content providers
- Broadcast Receivers

that make up your application and uses Intent Filters and Permissions to determine how they interact with each other and other applications. It also offers you attributes that you can use to specify application metadata like icon and theme among other things.

Android Manifest

application – A manifest can contain only one application node. This uses attributes to specify the metadata for your android application(title, icon and theme). Besides that, it acts as a container that includes the Activity, Service, Content Provider and Broadcast Receiver tags for specifying the application components:

activity – For every activity displayed by your android application, an activity tag is required. It must include the main launch Activity and any other screen or dialog that can be displayed. If you try to start an Activity that is not defined, you will get a runtime exception. Each Activity node supports intent-filter child tags which specify which Intents launch the activity.

service – Just like the activity, you must create a new service for each Service class used inside your application. Service tags also support intent-filter child tags to allow late runtime binding.

Android Manifest

provider – provider tags are used for each of your application's Content Providers. Content Providers are used to manage database access and sharing within and between applications.

receiver – You can register a Broadcast Receiver by adding a receiver tag without having to launch your application first. Broadcast Receivers are more like global event listeners – once registered, they will execute whenever a matching Intent is broadcast by an application.

Android Manifest

uses-permission – this is part of the security model. It declares permissions you have determined that your application needs to operate properly. The permissions you include will always be presented to the user to either grant or deny during installation. Many native android services require permissions for example those that have cost or security implications. Examples : location services, SMS, Camera etc.

permission – You need to define a permission in the android application manifest file before restricting access to any application component. Other apps will then need to include a uses-permission tag in their manifest file and have it granted before they can use these protected components.

Android Manifest

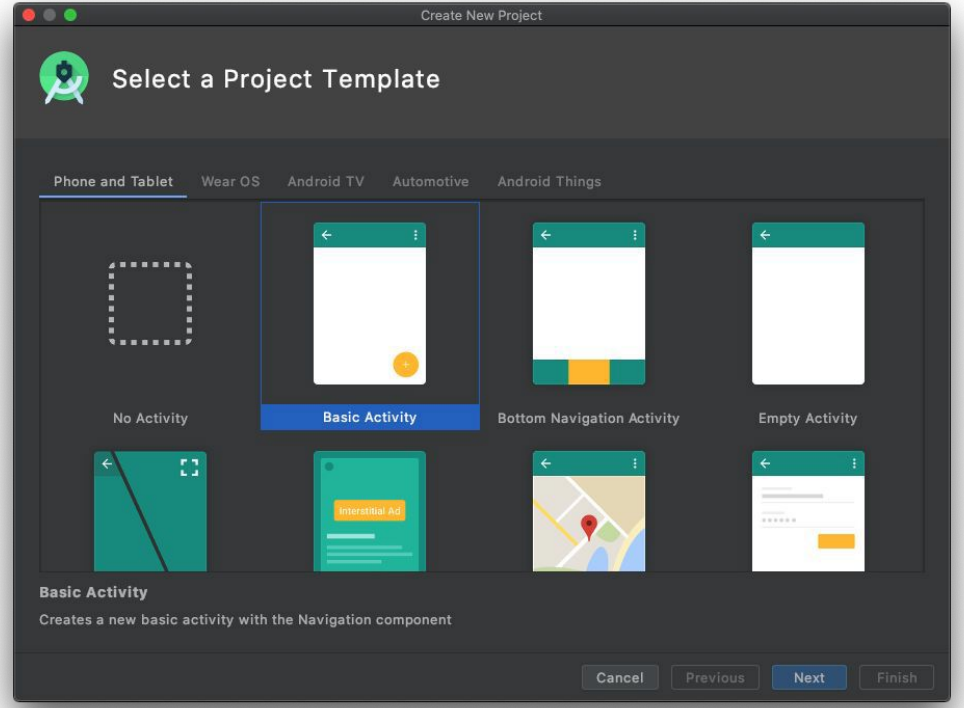
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="com.example.kiddo.myapp">

<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.CAMERA" />

<application
android:icon="@mipmap/my_ic_launcher"
android:label="@string/app_name"
android:theme="@style/AppTheme">
<activity
android:name=".MainActivity"

<intent-filter>
<action android:name="android.intent.action.MAIN" />
</intent-filter>
</activity>

<service android:name=".MainService" />
</service>
</application>

</manifest>
```

Android Development Studio

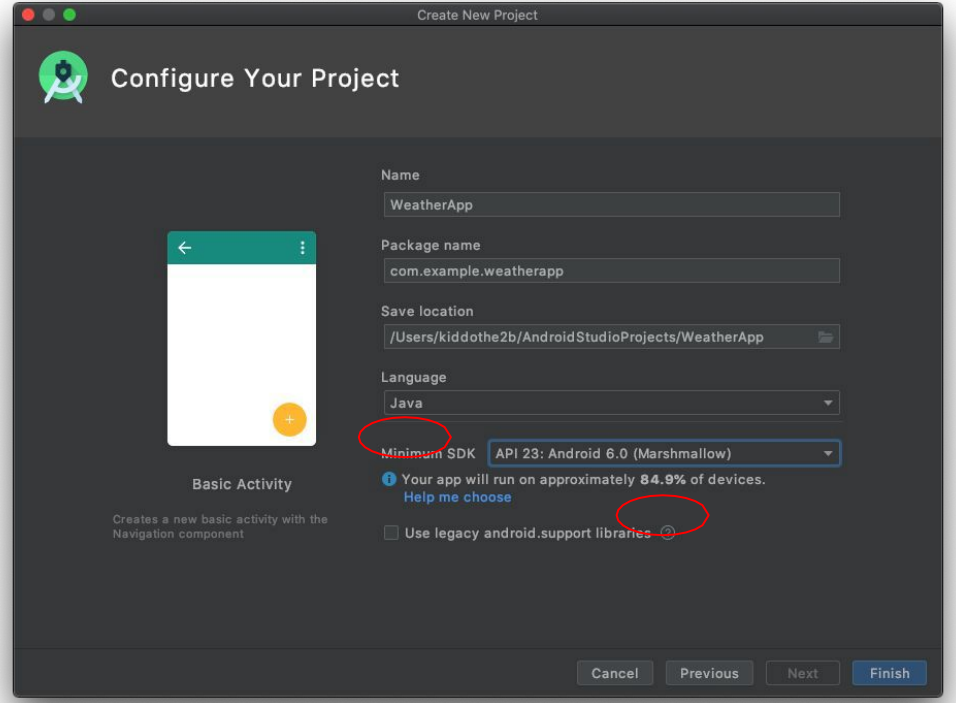
- Create Project - Default Activity
- Change Application Icon
- Build and Run Project
- Create new Activity
- Add mock data - Preview in Activity

Create Project - Default Activity

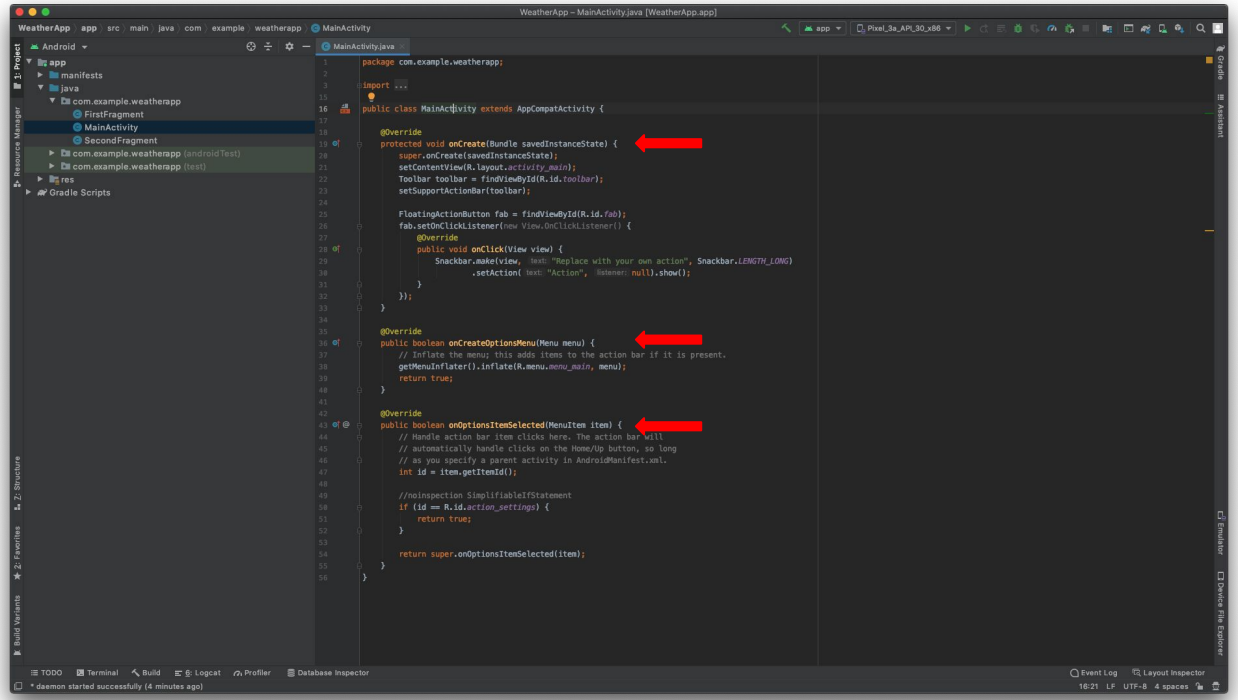
Create new project



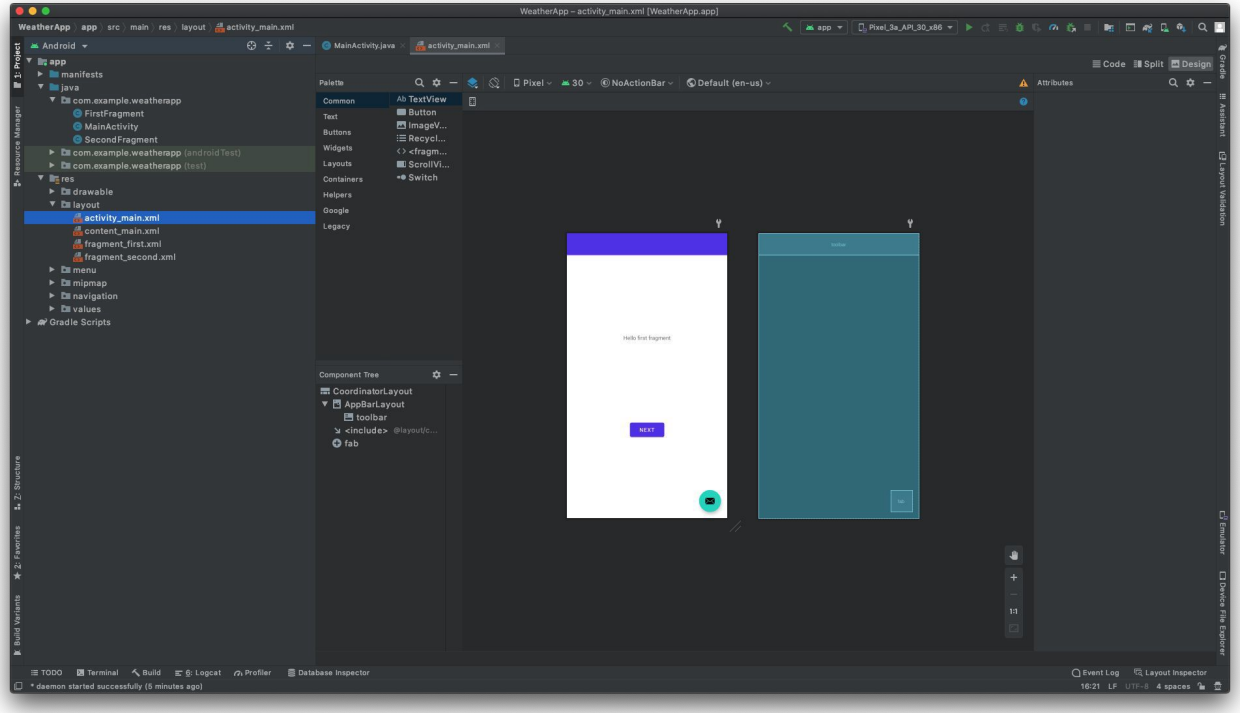
Select Template (Basic Activity)



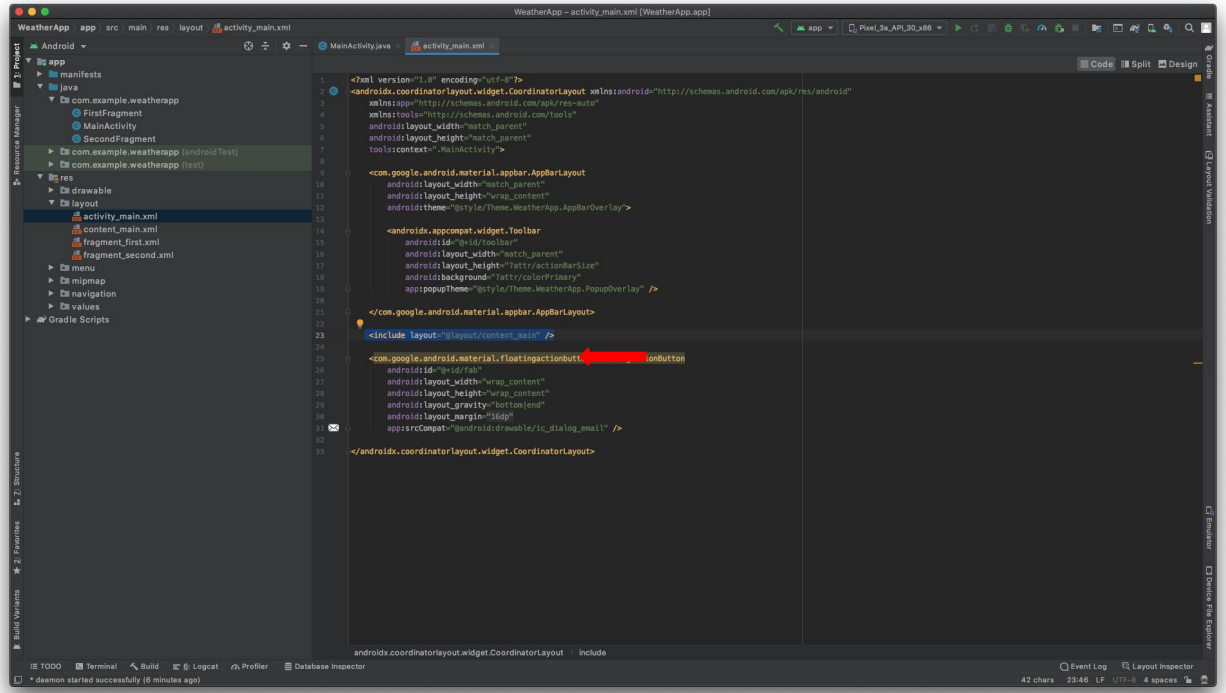
Select Language (Java) and Minimum SDK



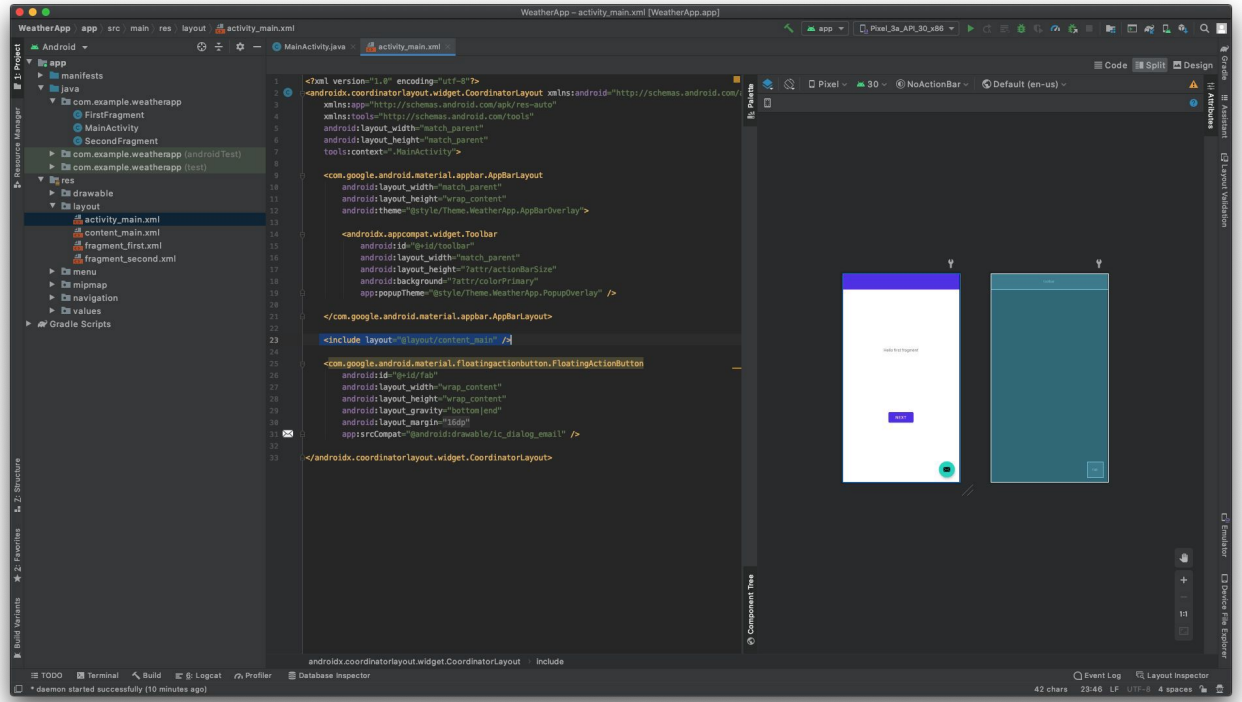
Core functionality in MainActivity



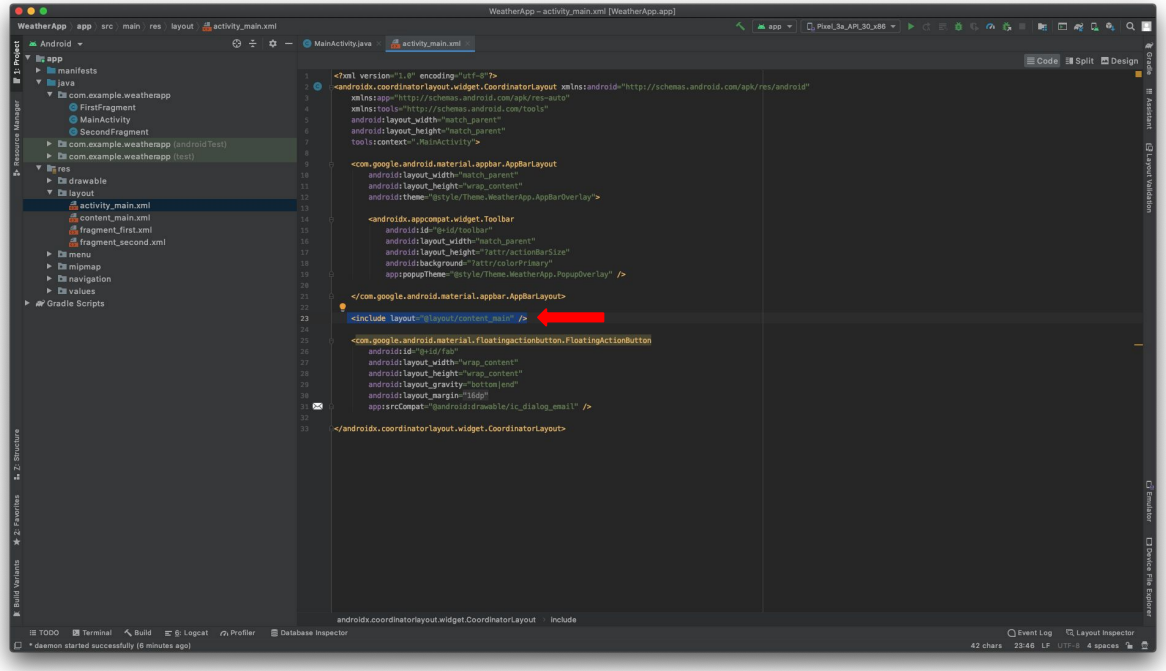
Preview screen layout in Design Mode



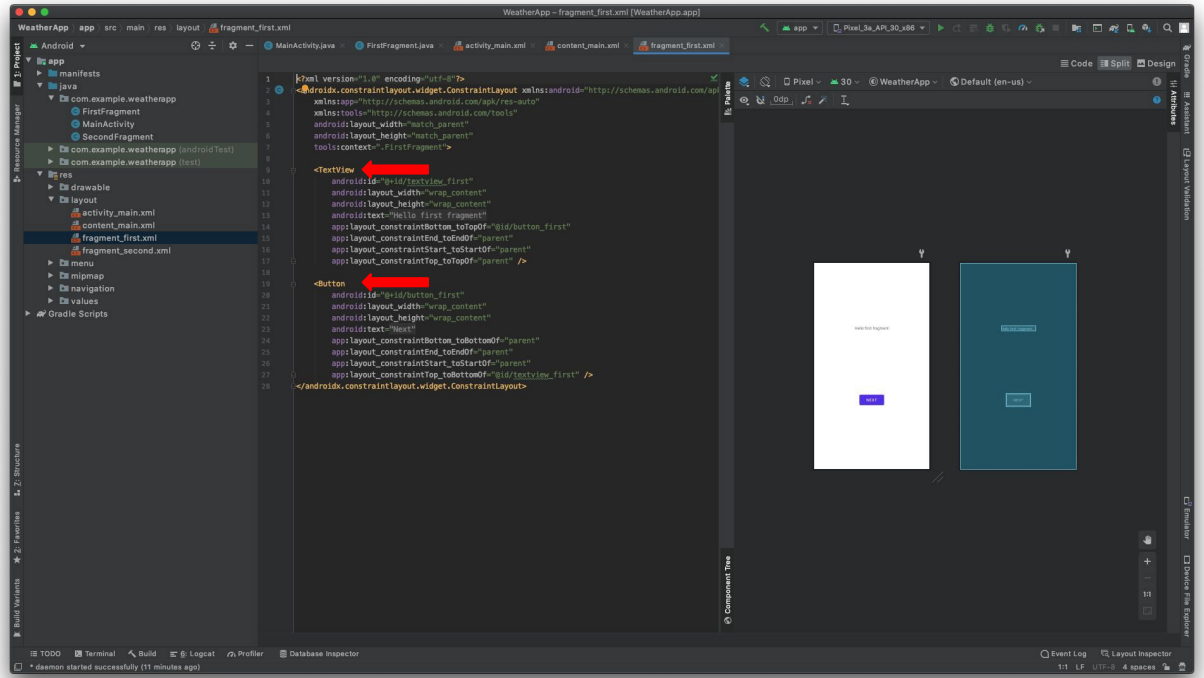
Preview screen layout in XML



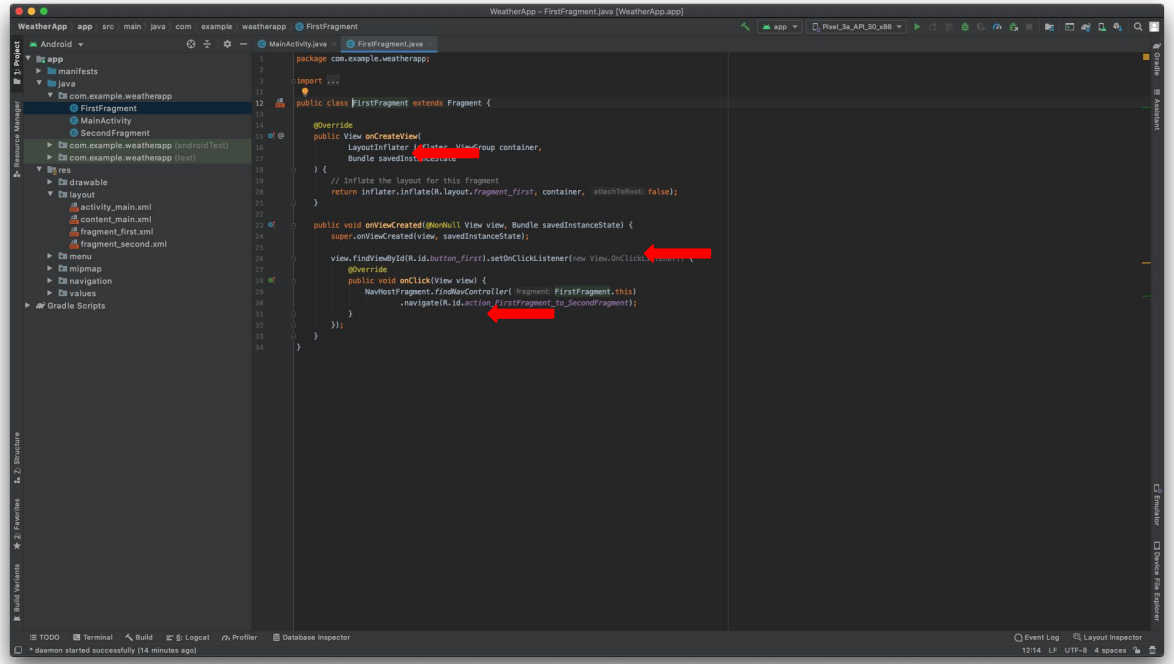
Preview screen layout (Split View → Code/Graphics)



Main Activity Layout includes Main content

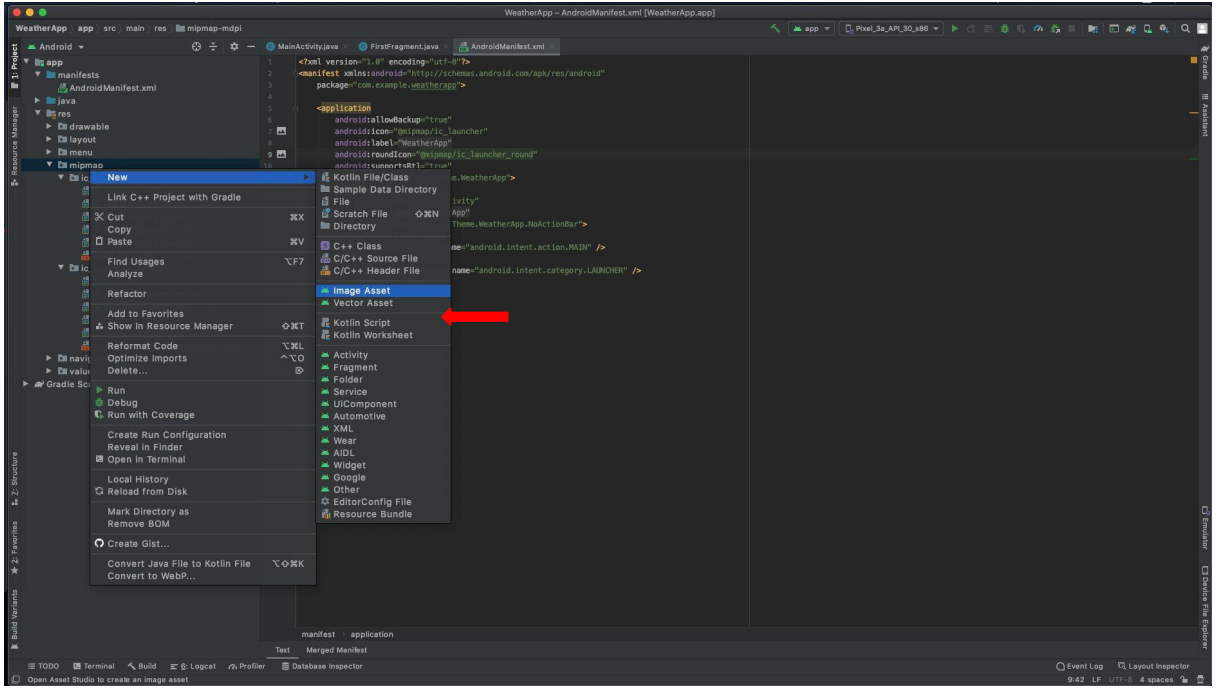


1st Fragment (Text + Button)

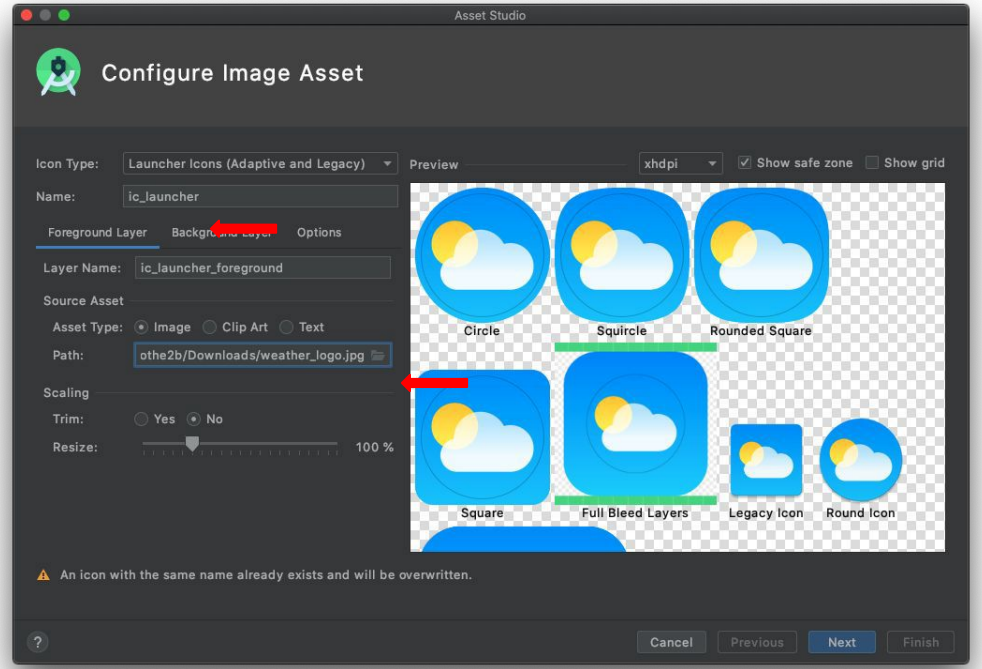


(1) Show fragment and (2) onClick move to next fragment

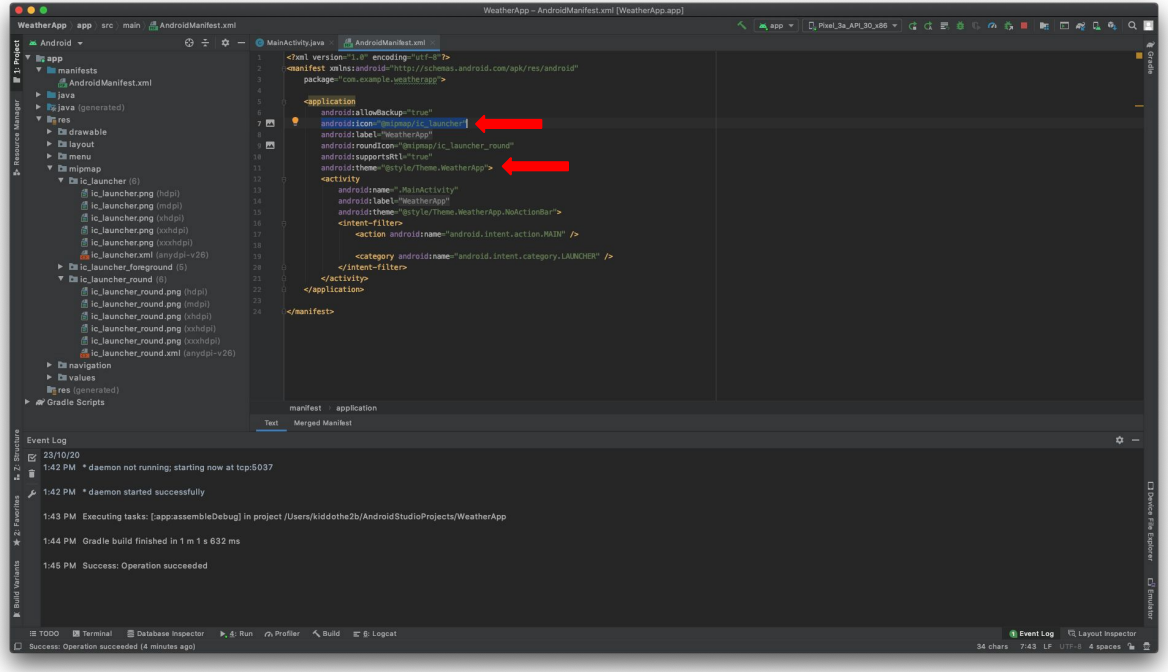
Change Application Icon



Create new Image asset



Select Image

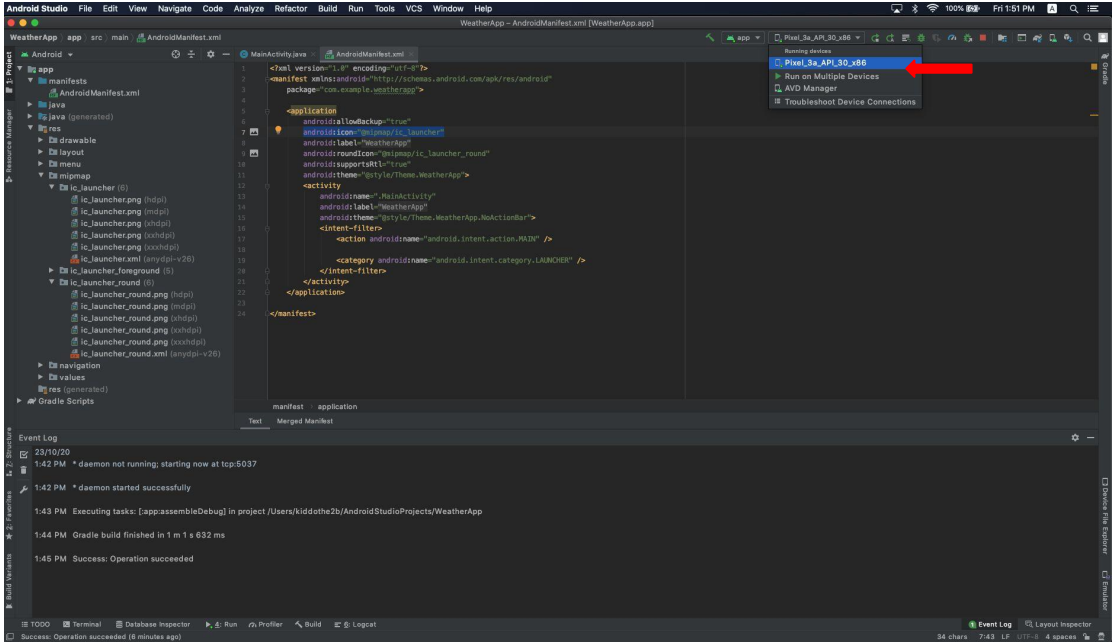


Manifest defines application icon

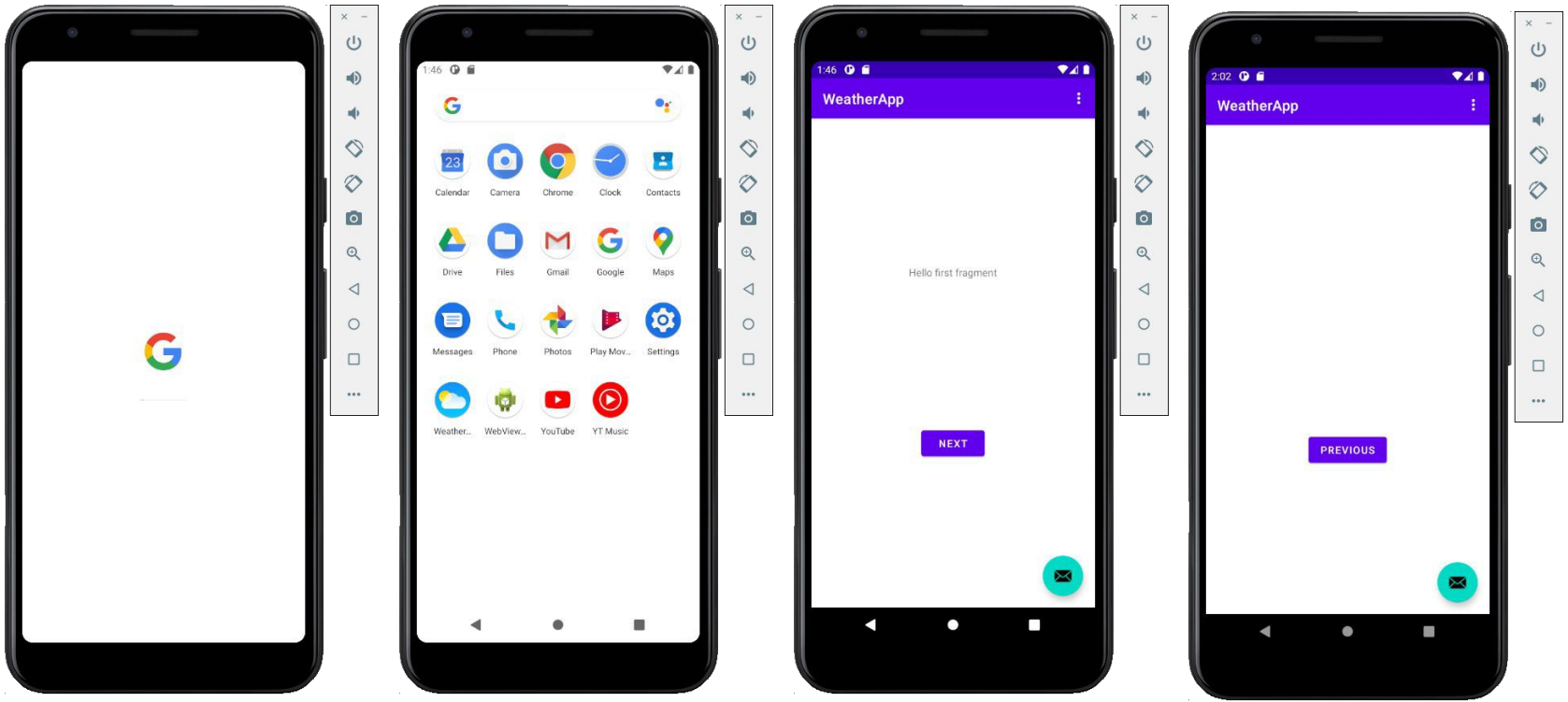
Build and Run Project

Add the following line in app gradle in dependencies

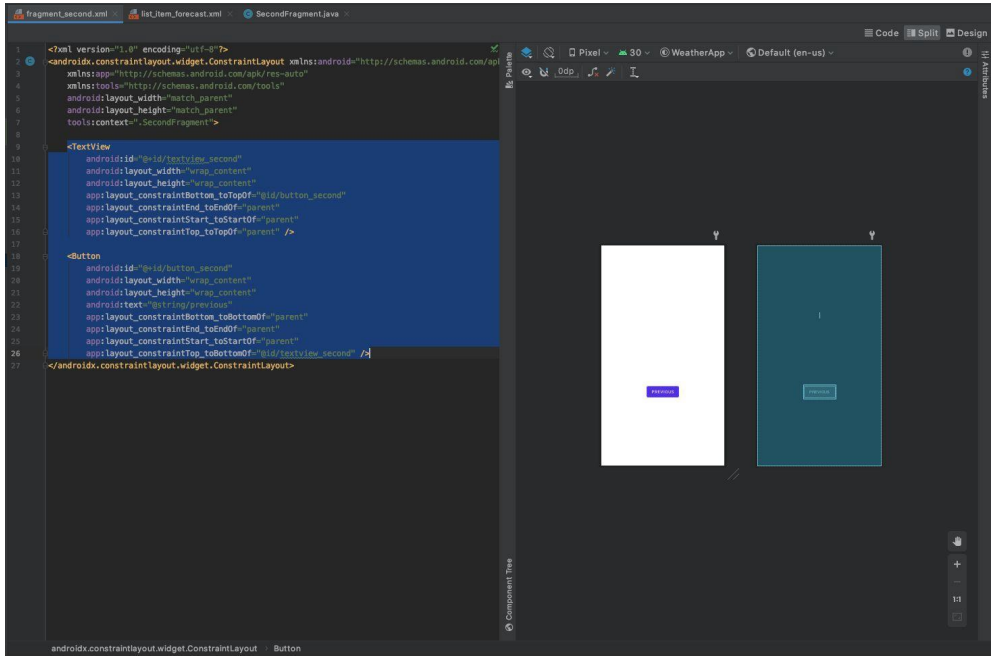
```
implementation(platform("org.jetbrains.kotlin:kotlin-bom:1.8.0"))
```



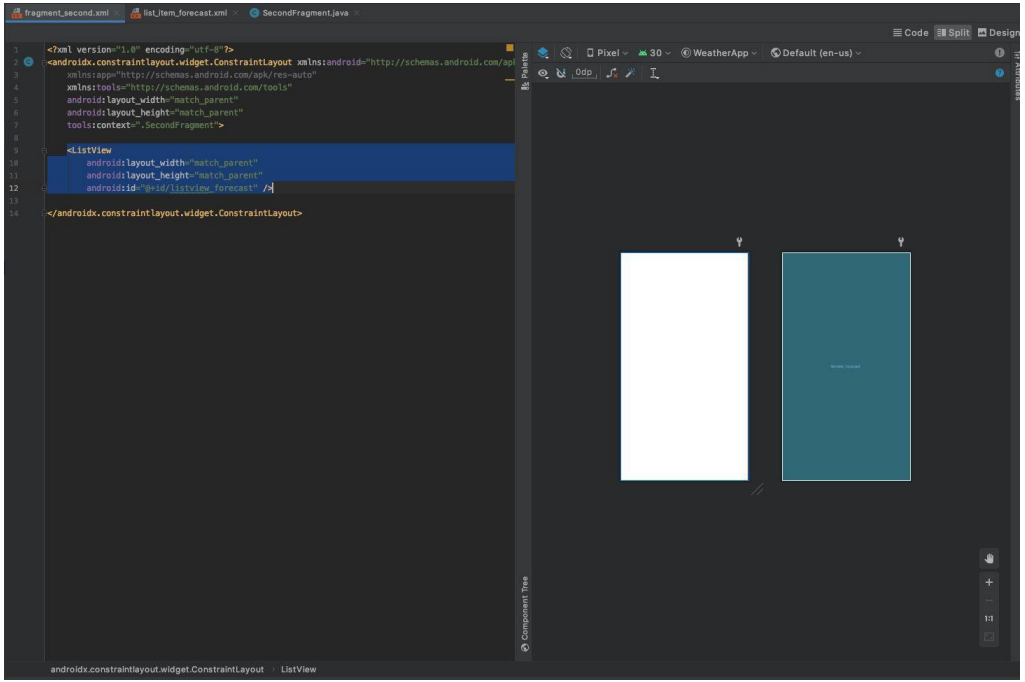
Select virtual device for emulation


1:46
Calendar
Camera
Chrome
Clock
Contacts
Drive
Files
Gmail
Google
Maps
Messages
Phone
Photos
Play Mov...
Settings
Weather...
WebView...
YouTube
YT Music
1:46
WeatherApp
Hello first fragment
NEXT
2:02
WeatherApp
PREVIOUS

Create new Activity

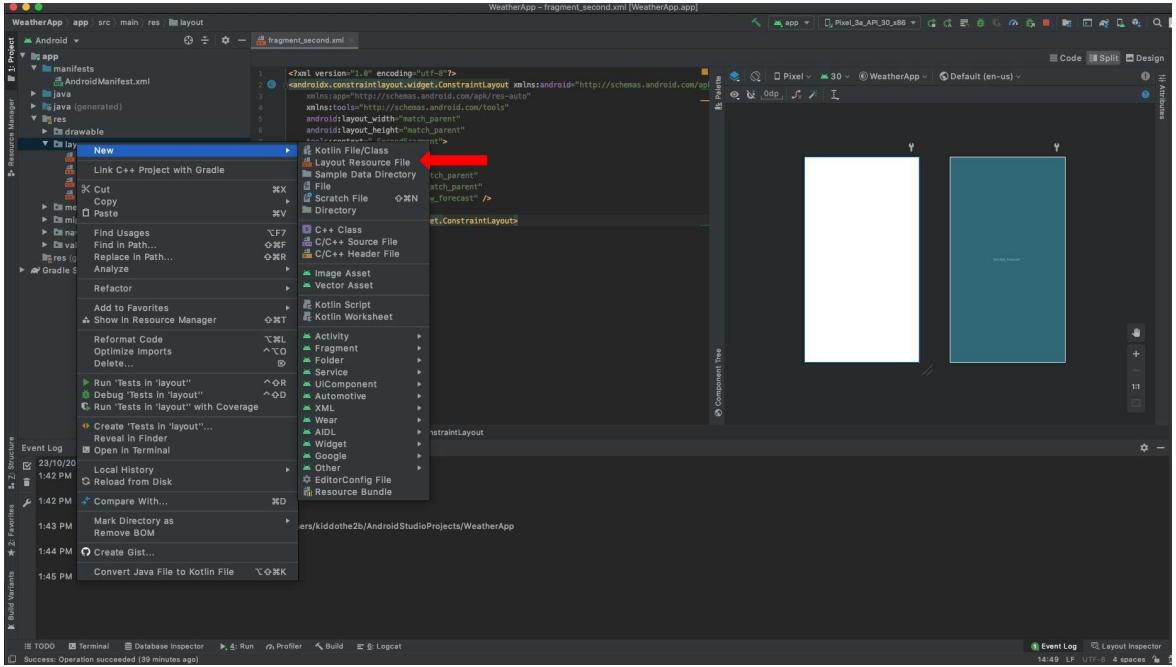


Remove text and button from second fragment

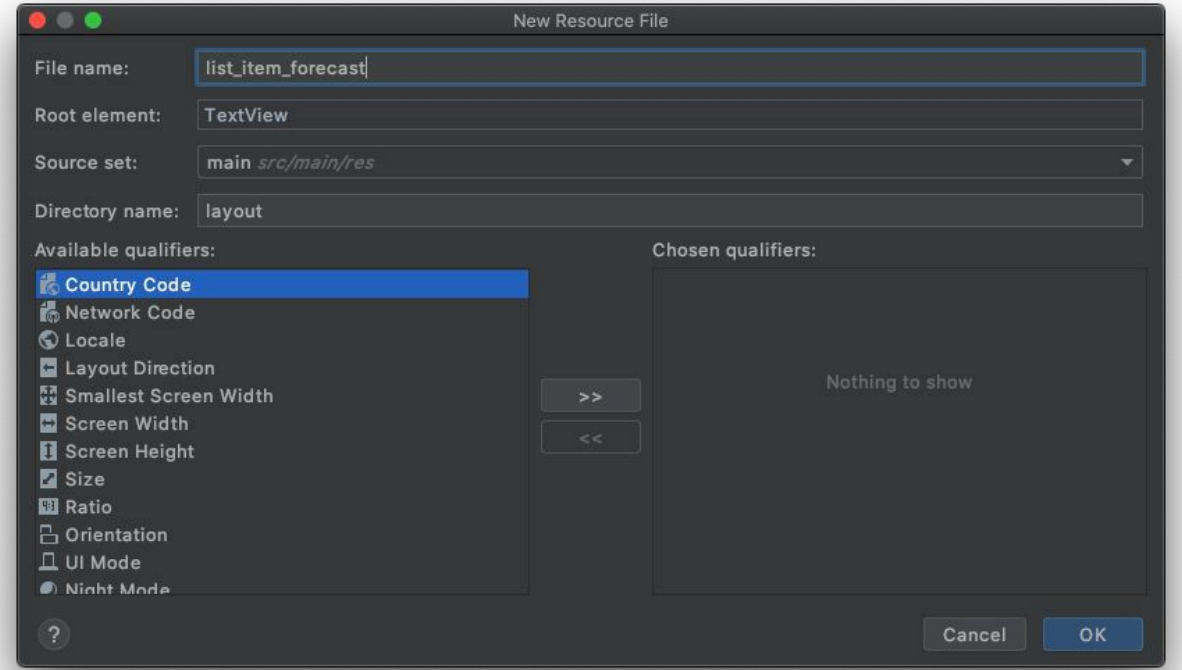


Replace with ListView

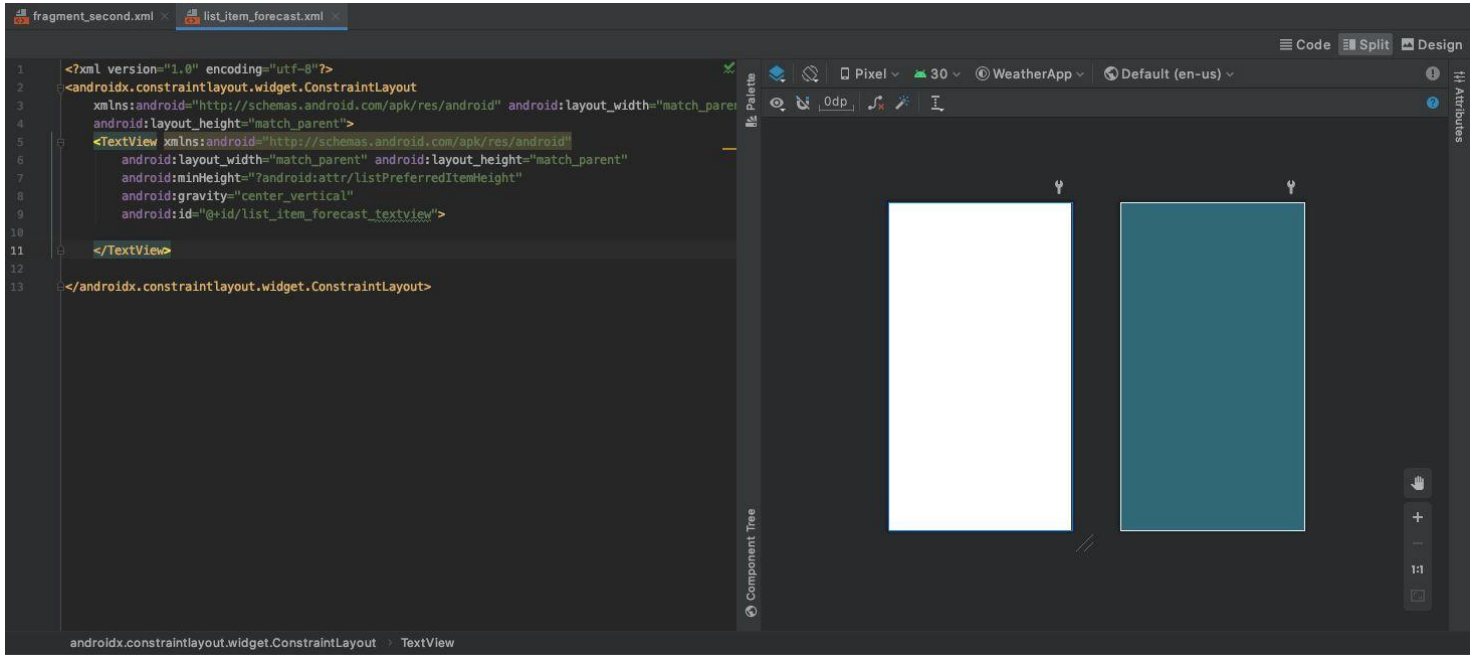
```
<ListView
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:id="@+id/listview_forecast" />
```



Create new layout file



Create a TextView for List items



TextView specifications

```
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent" android:layout_height="match_parent"
    android:minHeight="?android:attr/listPreferredItemHeight"
    android:gravity="center_vertical"
    android:id="@+id/list_item_forecast_textview">

</TextView>
```

Add mock data - Preview in Activity

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.annotation.NonNull;
import androidx.fragment.app.Fragment;
import androidx.navigation.fragment.NavHostFragment;

public class SecondFragment extends Fragment {

    @Override
    public View onCreateView(
            LayoutInflater inflater, ViewGroup container,
            Bundle savedInstanceState
    ) {
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.fragment_second, container,  attachToRoot: false);
    }

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

        view.findViewById(R.id.button_second).setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                NavHostFragment.findNavController( fragment: SecondFragment.this)
                        .navigate(R.id.action_SecondFragment_to_FirstFragment);
            }
        });
    }
}
```

Old business logic / Move back button

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

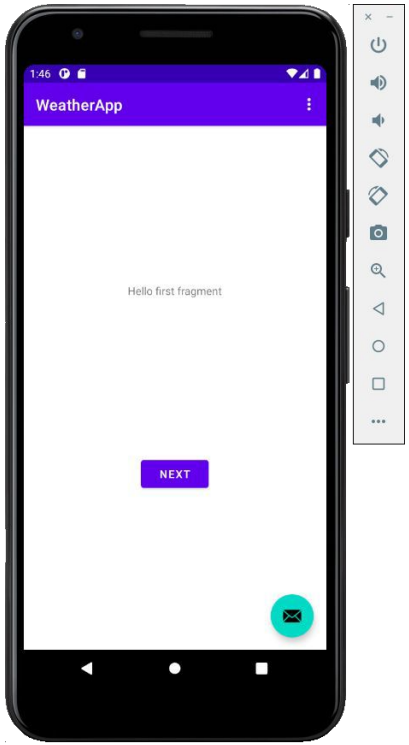
SecondFragment.java

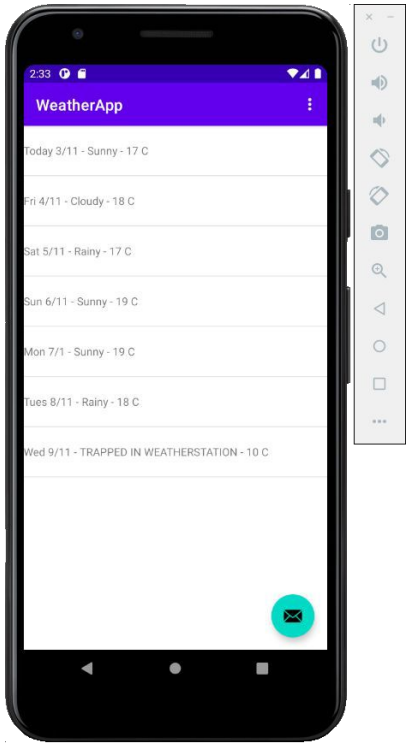
```java
package com.example.weatherapp;

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;

    }

}
```

Fix imports

```
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_main, container, false);
//ListView listView = (ListView) rootView.findViewById(R.id. listview_forecast);
binding = FragmentSecondBinding.inflate(inflater, container, false);
ListView listView = (ListView) binding.listviewForecast;
listView.setAdapter(mForecastAdapter);
return binding.getRoot();
```

```
import java.util.*;
import android.widget.ArrayAdapter;
import android.widget.ListView;
```


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

Review of Lab 1

- Android Development Fundamentals
  - Android SDK and Development Tools
  - Android State Machine
  - Android Manifest
- Android Development Studio
  - Create Project - Default Activity
  - Change Application Icon
  - Build and Run Project
  - Create new Activity
  - Add mock data - Preview in Activity