ICP10

ICP Group 4

Name: Sailaja Narra

Email: sntnn@umsystem.edu

ICP
Report: https://drive.google.com/file/d/1AHu01JAtMF9vZYqNC7gQqy2uCUGUqZ5H/view?usp=sharing

ICP Video: https://umsystem.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=b5889463-b855-4020-b886-add8002af13c

GitHub (Sourcecode): https://github.com/UMKC-APL-WebMobileProgramming/ICP10-Mahesh68/tree/main/Source

My Partner

• Partner: Venkata Mahesh Mokkapati

• Email: vmzwn@umsystem.edu

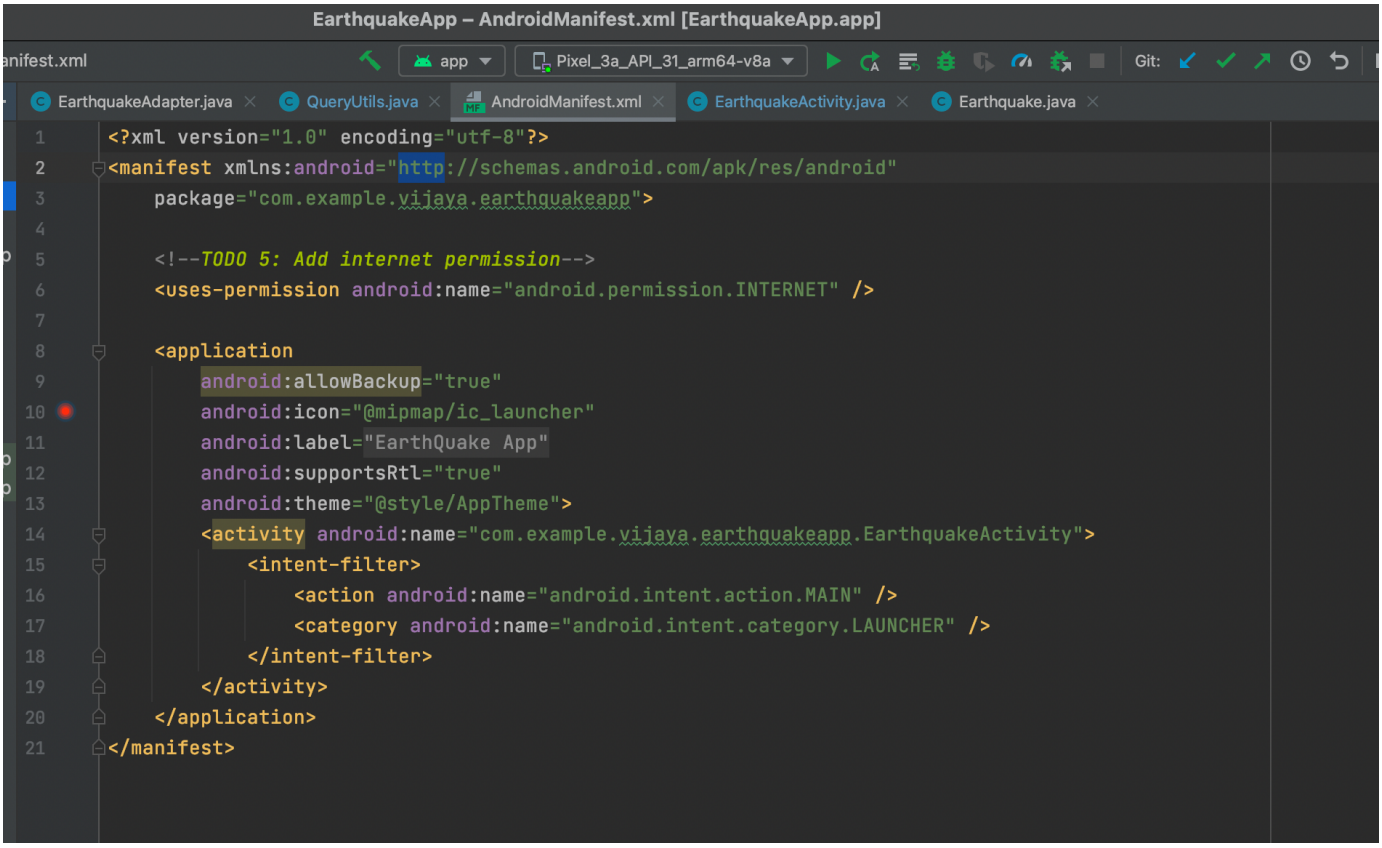
• Repo: https://github.com/UMKC-APL-WebMobileProgramming/ICP10-sailajanarra

In this task we understood about the some of the aspects of android such as fetching JSON data from APIs, parsing the JSON data, handling errors, using the Async task and some JAVA elements.

Description: Rendering and displaying the recent earthquake information.

Here we have the given source code where there are enhancements that has to be done in this ICP.

1. Here internet permissions are added in the manifests file to make sure internet connectivity is possible for the third party apps/plugins.



2. Here Main activity is the earthquake activity where the list of earthquakes are shown as a list.

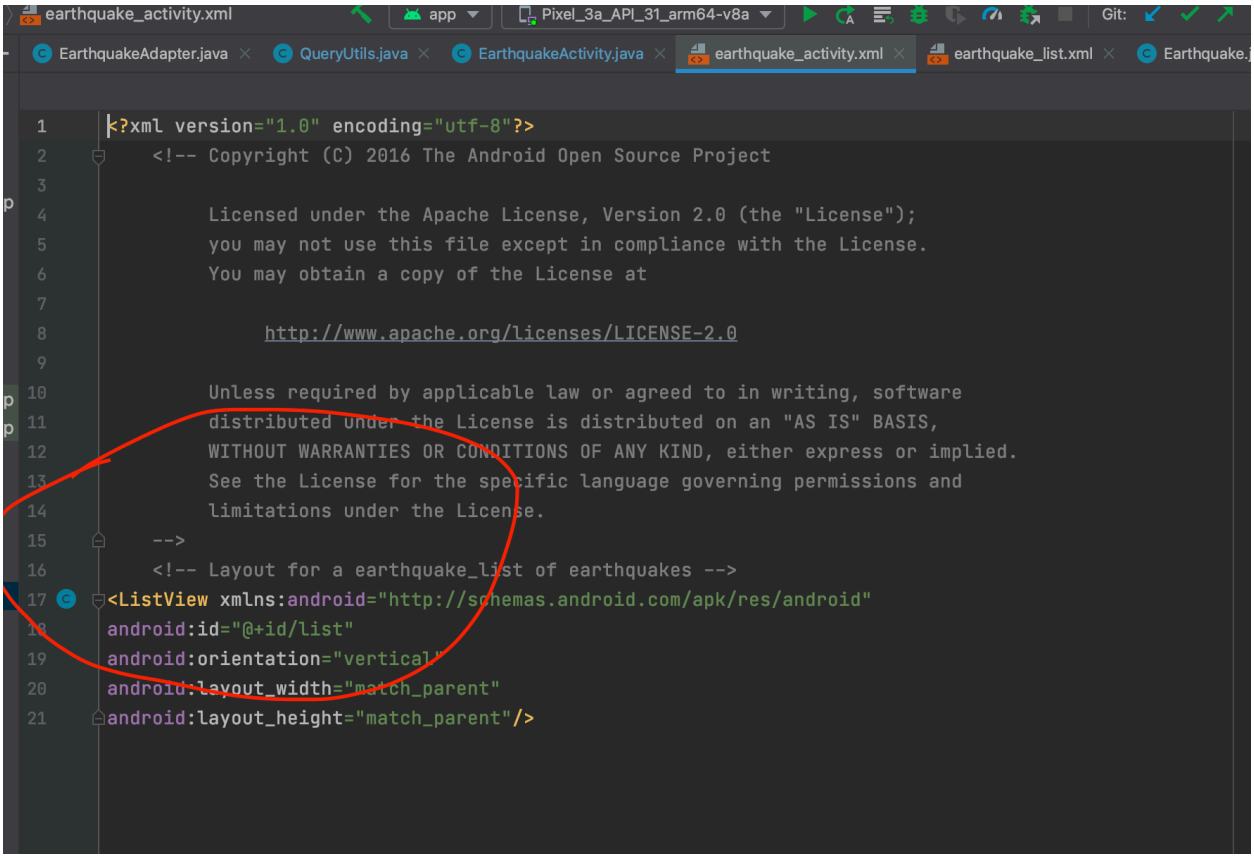
URL for earthquake data

```
public class EarthquakeActivity extends AppCompatActivity {

    private static final String LOG_TAG = EarthquakeActivity.class.getName();

    /**
     * URL for earthquake data from the USGS dataset
     */
    private static final String USGS_REQUEST_URL =
            "https://earthquake.usgs.gov/fdsnws/event/1/query?format=geojson&orderby=time&minmag=4&limit=10";

    /**
     * Adapter for the list of earthquakes
     */
    private EarthquakeAdapter mAdapter;

    public static class HttpsTrustManager implements X509TrustManager {

        private static TrustManager[] trustManagers;
        private static final X509Certificate[] _AcceptedIssuers = new X509Certificate[]{};
```

A new adapter is created for the data items. Here the adapter is used to convert data items into view items.

```java
/**
 * Adapter for the list of earthquakes
 */
private EarthquakeAdapter mAdapter;

public static class HttpsTrustManager implements X509TrustManager {...}

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.earthquake_activity);

    // Find a reference to the {@link ListView} in the layout
    ListView earthquakeListView = (ListView) findViewById(R.id.list);

    // Create a new adapter that takes an empty list of earthquakes as input
    mAdapter = new EarthquakeAdapter( context: this, new ArrayList<Earthquake>());

    // Set the adapter on the {@link ListView}
    // so the list can be populated in the user interface
    earthquakeListView.setAdapter(mAdapter);

    // Set an item click listener on the ListView, which sends an intent to a web browser
    // to open a website with more information about the selected earthquake.
    earthquakeListView.setOnItemClickListener((adapterView, view, position, l) -> {
            // Find the current earthquake that was clicked on
            Earthquake currentEarthquake = mAdapter.getItem(position);
```

List view is created to display the list of scrollable items as shown below in the design page. This list view is passed as a second argument to the adapter in the “EarthquakeActivity.java” file.




```
// Find a reference to the {@link ListView} in the layout
ListView earthquakeListView = (ListView) findViewById(R.id.list);

// Create a new adapter that takes an empty list of earthquakes as input
mAdapter = new EarthquakeAdapter( context: this, new ArrayList<Earthquake>());
```

3. Now the list is populated into the user interface. Now for this list view an on click event listener is added so that it sends an intent to a web browser to open website with more information about the selected earthquake.

Taking the URI of the clicked view item and passed to the created new intent to launch a new activity.

```
    earthquakeListView.setAdapter(mAdapter);

    // Set an item click listener on the ListView, which sends an intent to a web browser
    // to open a website with more information about the selected earthquake.
    earthquakeListView.setOnItemClickListener((adapterView, view, position, l) -> {
            // Find the current earthquake that was clicked on
            Earthquake currentEarthquake = mAdapter.getItem(position);

            // Convert the String URL into a URI object (to pass into the Intent constructor)
            Uri earthquakeUri = Uri.parse(currentEarthquake.getUrl());

            //TODO: 4. Create a new intent to view the earthquake URI.Send the intent to launch a new activity

            Intent eqURIIntent = new Intent(Intent.ACTION_VIEW, earthquakeUri);
            startActivity(eqURIIntent);

    });

    // Start the AsyncTask to fetch the earthquake data
    EarthquakeAsyncTask task = new EarthquakeAsyncTask();
    HttpsTrustManager.allowAllSSL();
    task.execute(USGS_REQUEST_URL);
}
```

Here to perform the API call/network request Async task is used. To perform the operation in the background thread and update the UI with the list of earthquakes in the response.

Async task has three parameters as an input :

1. Input type – String “url”
2. A type used for progress updates
3. Output type – return earthquake

This async task lifecycle has three of the methods:

In this task we will override only two of the methods:

1. doInBakground – This method runs on the background thread and performs the network request. This result of earthquakes is returned to the onpostexecute method. UI will not be updated from the background thread.
2. onPostExecute – This method takes the output of background thread as an input. This method runs on the UI thread so the UI will be updated

from here. Previous data will be cleared from the adapter and then updated with the list of earthquake items.

```
     */
    @Override
    protected void onPostExecute(List<Earthquake> data) {
        // Clear the adapter of previous earthquake data
        mAdapter.clear();

        // If there is a valid list of {@link Earthquake}s, then add them to the adapter's
        // data set. This will trigger the ListView to update.
        if (data != null && !data.isEmpty()) {
            mAdapter.addAll(data);
        }
    }
}
```

In the background thread network request is performed using the “queryutils.java”

Here a new Array list is created to store the response obtained from the REST API call.

4. Creating an URL object:

An empty variable is created to store the JSON response. An obsolute URL is created that contains all the necessary information to reach the resource.

```
// A string to store the response obtained from rest call in the form of string
String jsonResponse = "";
StringBuilder stringBuilder = new StringBuilder();
try {
    //TODO: 1. Create a URL from the requestUrl string and make a GET request to it

    url = new URL(requestUrl);

    //TODO: 2. Read from the Url Connection and store it as a string(jsonResponse)
    //Reading from a URLConnection
    URLConnection urlconnect= url.openConnection();
    BufferedReader in = new BufferedReader(new InputStreamReader(urlconnect.getInputStream()));

    String inputLine;

    while ((inputLine = in.readLine()) !=  null) {
        // Appending the Line to StringBuilder
        stringBuilder.append(inputLine);
    }
    // Closing Buffered Reader.
    if(in != null){
        in.close();
    }
    // Converting string builder to String and using it as JSON Response.
    jsonResponse = stringBuilder.toString();
```

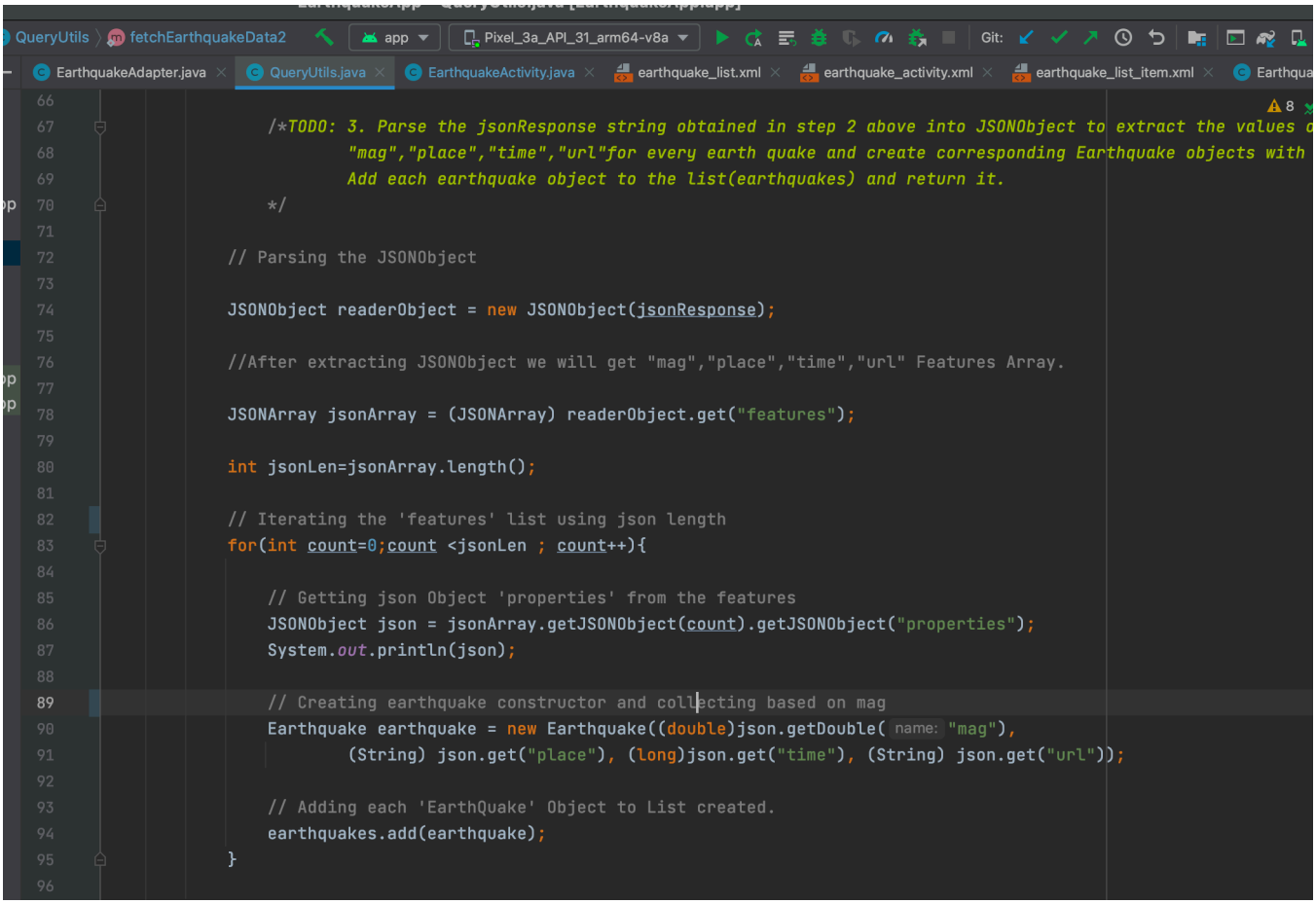
After creating the URL I make use of URL's get input stream method to get a stream from which we can read the contents of the URL.

There is also a openStream() method which will serve the same purpose.

Here I made use of string builder to store the response of the JSON data and then converted back to string and stored in the JSON response variable that's created.

5. The obtained JSON data is extracted to get the properties of "mag", "place", "time", "url" for every earthquake.

Created corresponding earthquake items and added each object to the earthquake list and returned.



Here these properties came as an array of items. So iterated over this array and created an earthquake item for each iteration.

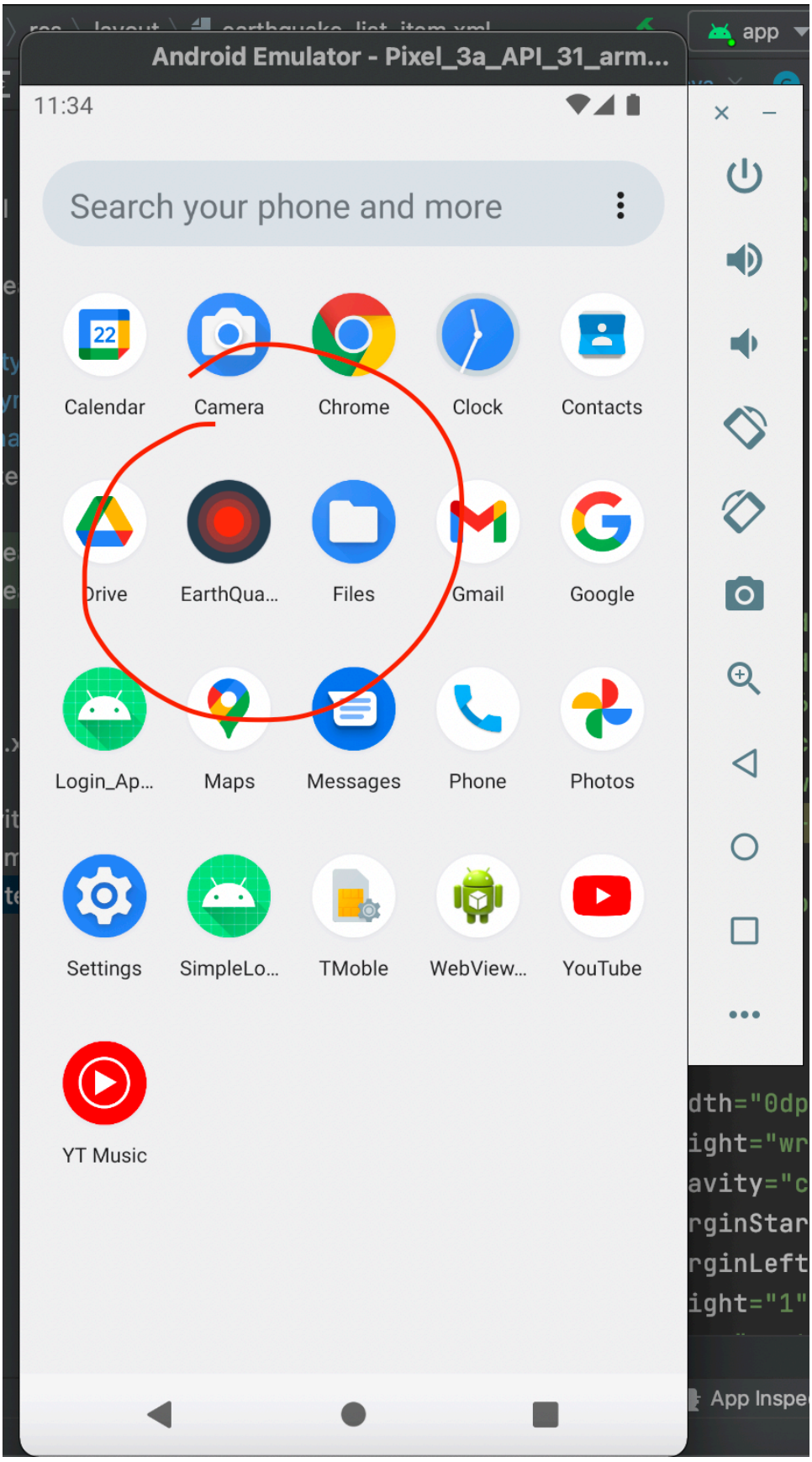
Each list item is displayed as below:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="?android:attr/listPreferredItemHeight"
    android:orientation="horizontal"
    android:paddingStart="16dp"
    android:paddingLeft="16dp"
    android:paddingEnd="16dp"
    android:paddingRight="16dp">

    <TextView
        android:id="@+id/magnitude"
        android:layout_width="36dp"
        android:layout_height="36dp"
        android:layout_gravity="center_vertical"
        android:background="@drawable/magnitude_circle"
        android:fontFamily="sans-serif-medium"
        android:gravity="center"
        android:textColor="@android:color/white"
        android:textSize="16sp"
        tools:text="8.9" />

    <LinearLayout
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_gravity="center_vertical"
        android:layout_marginStart="16dp"
        android:layout_marginLeft="16dp"
        android:layout_weight="1"
```

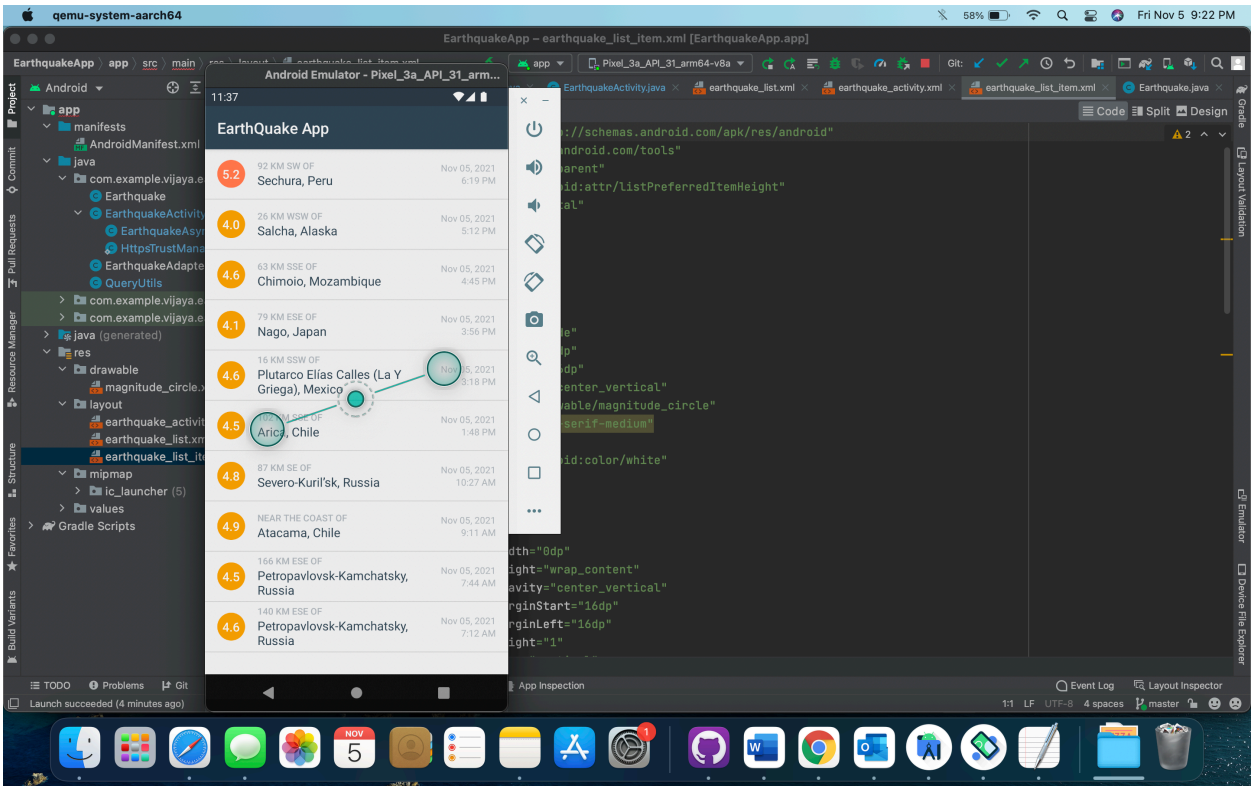
OUTPUT:

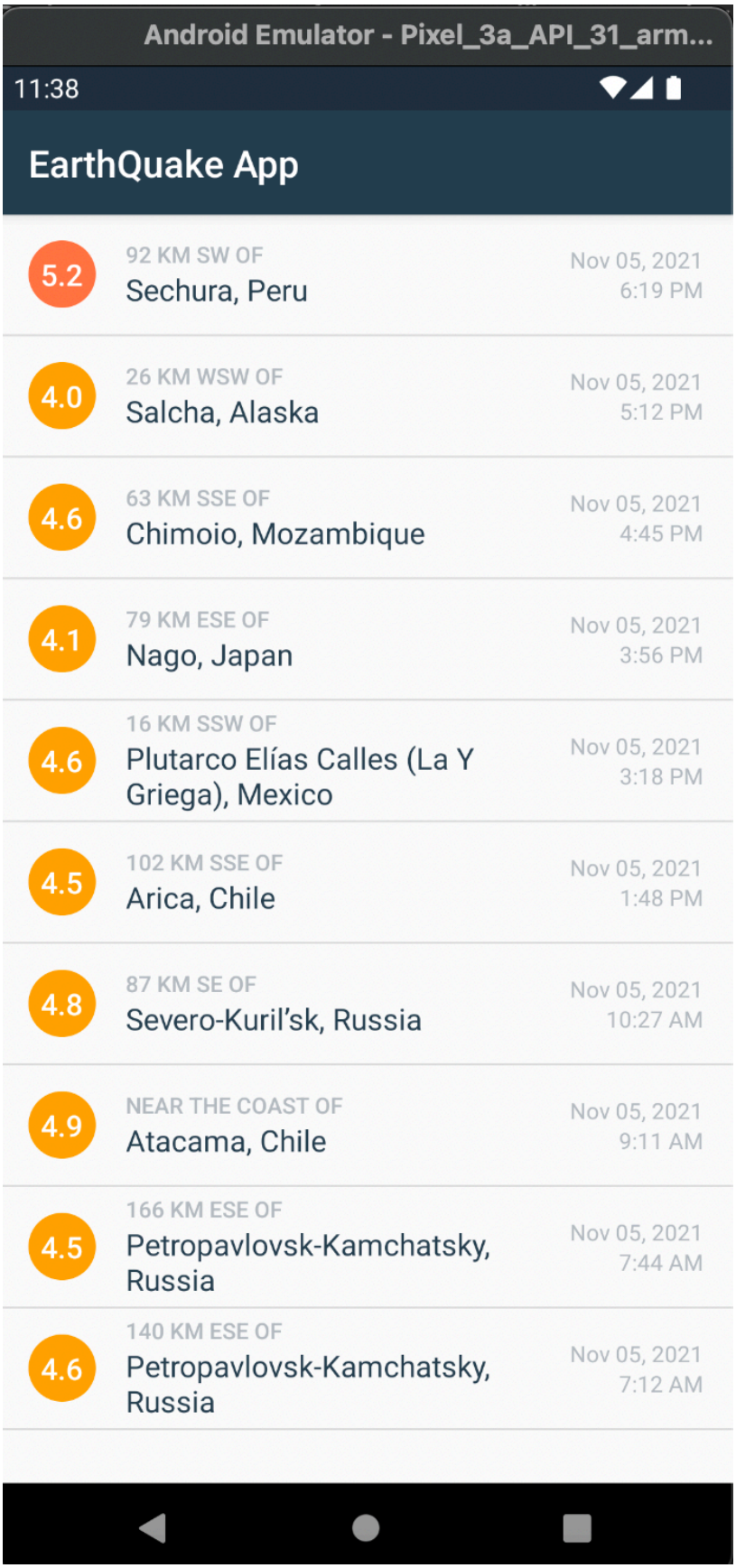
In the home page “Eathquake” app is created as below:

Here the app launcher icon is created with an ic_launcher that’s given with the source code.


app
Android Emulator - Pixel_3a_API_31_arm...
11:34
Search your phone and more
Calendar
Camera
Chrome
Clock
Contacts
Drive
EarthQua...
Files
Gmail
Google
Login_Ap...
Maps
Messages
Phone
Photos
Settings
SimpleLo...
TMoble
WebView...
YouTube
YT Music
dth="0dp
ight="wr
avity="c
rginStar
rginLeft
ight="1"
App Inspe

Home page of application is as below: List of items are as displayed below




Android Emulator - Pixel_3a_API_31_arm...
11:38
EarthQuake App
5.2
92 KM SW OF
Sechura, Peru
Nov 05, 2021
6:19 PM
4.0
26 KM WSW OF
Salcha, Alaska
Nov 05, 2021
5:12 PM
4.6
63 KM SSE OF
Chimoio, Mozambique
Nov 05, 2021
4:45 PM
4.1
79 KM ESE OF
Nago, Japan
Nov 05, 2021
3:56 PM
4.6
16 KM SSW OF
Plutarco Elías Calles (La Y Griega), Mexico
Nov 05, 2021
3:18 PM
4.5
102 KM SSE OF
Arica, Chile
Nov 05, 2021
1:48 PM
4.8
87 KM SE OF
Severo-Kuril'sk, Russia
Nov 05, 2021
10:27 AM
4.9
NEAR THE COAST OF
Atacama, Chile
Nov 05, 2021
9:11 AM
4.5
166 KM ESE OF
Petropavlovsk-Kamchatsky, Russia
Nov 05, 2021
7:44 AM
4.6
140 KM ESE OF
Petropavlovsk-Kamchatsky, Russia
Nov 05, 2021
7:12 AM

On click of any list item corresponding view will be displayed from the earthquake adapter.


Android Emulator - Pixel_3a_API_31_arm...
11:39
earthquake.usgs.gov/earthqu
USGS
MENU
M 4.1 - 79 km ESE of Nago, Japan
2021-11-05 20:56:42 (UTC)
26.246°N 128.667°E
12.5 km depth
Interactive Map
Naha
Contributed by US.[1]
Regional Information

As shown in the above, intent will be displayed in the web browser by loading the activity.

With this ICP is finished with all the necessary enhancements implementations. We learnt about the multi threading and making the network calls using the async tasks.

Thank you.