Name: Lakshmana Kumar Mettu

Class ID: 11

Introduction:

- This ICP would help you to get insights on how JSON data extracting using API's in android.
- And It will help understand how async class is used in android to handle the errors along with using list view and adapter class.

Programming elements:

- Android Studio
- Android Emulator

Objective:

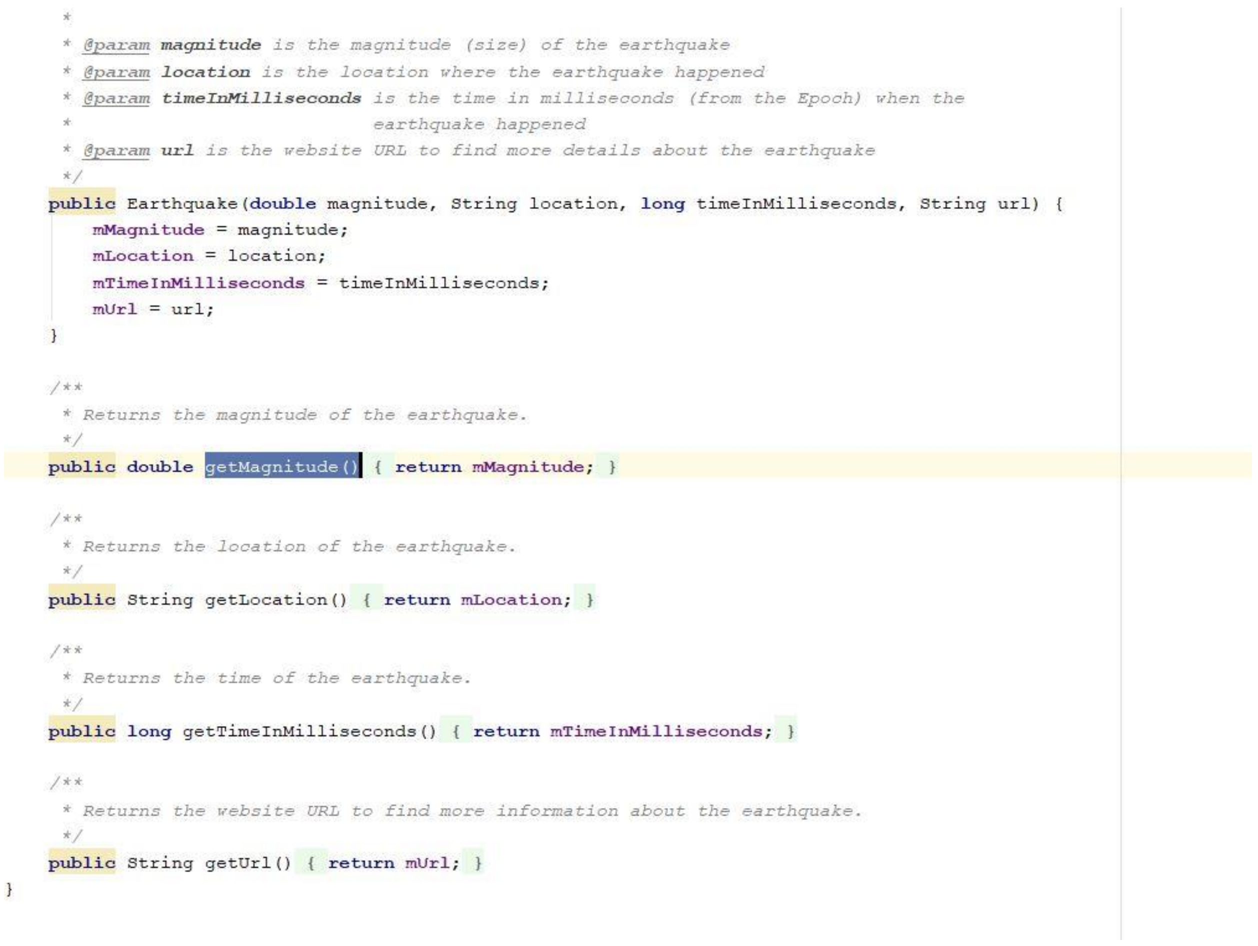
- The main objective of this task are given below.

1. The main activity must show list of earthquakes using USGS Earthquake API.
2. And when user click on any earthquake info it will redirect to corresponding ur in USGS website.

- It has four Activities.

1. Earthquake.java

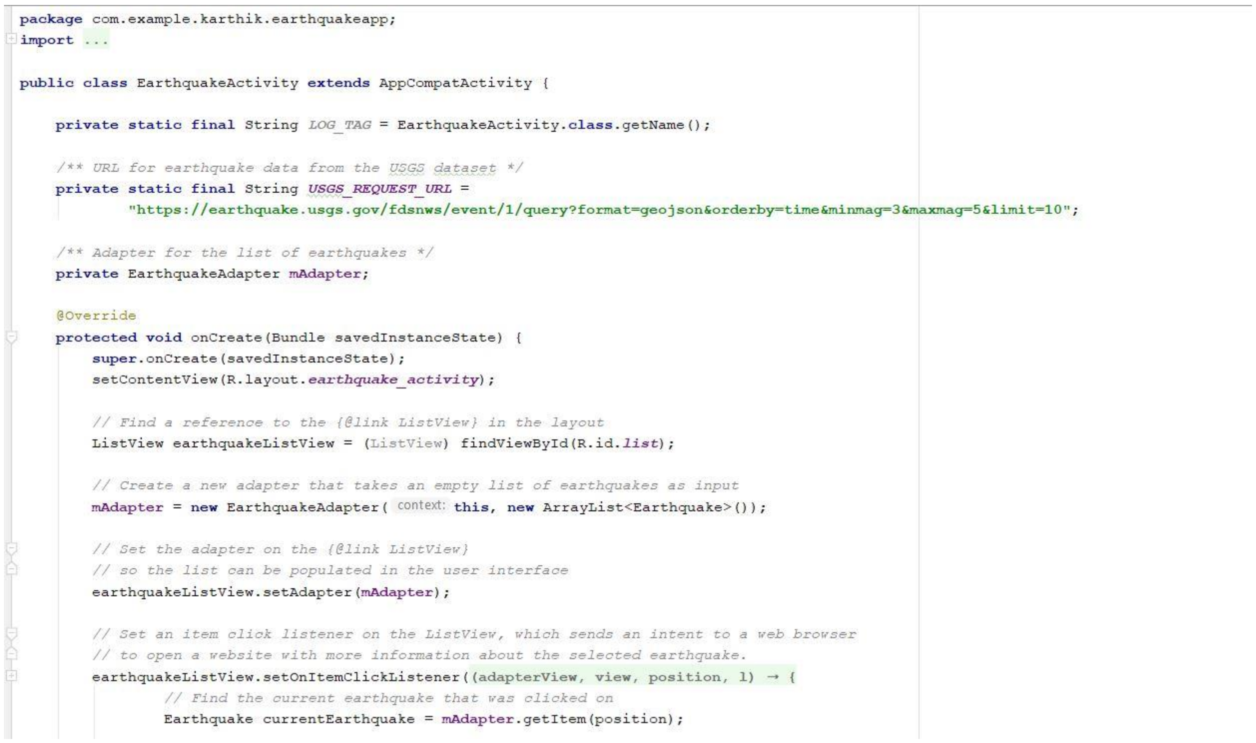
- Defines the required information using get request for location, time, magnitude and get request to obtain the URL.
- Code snippets for Earthquake.java is given below.



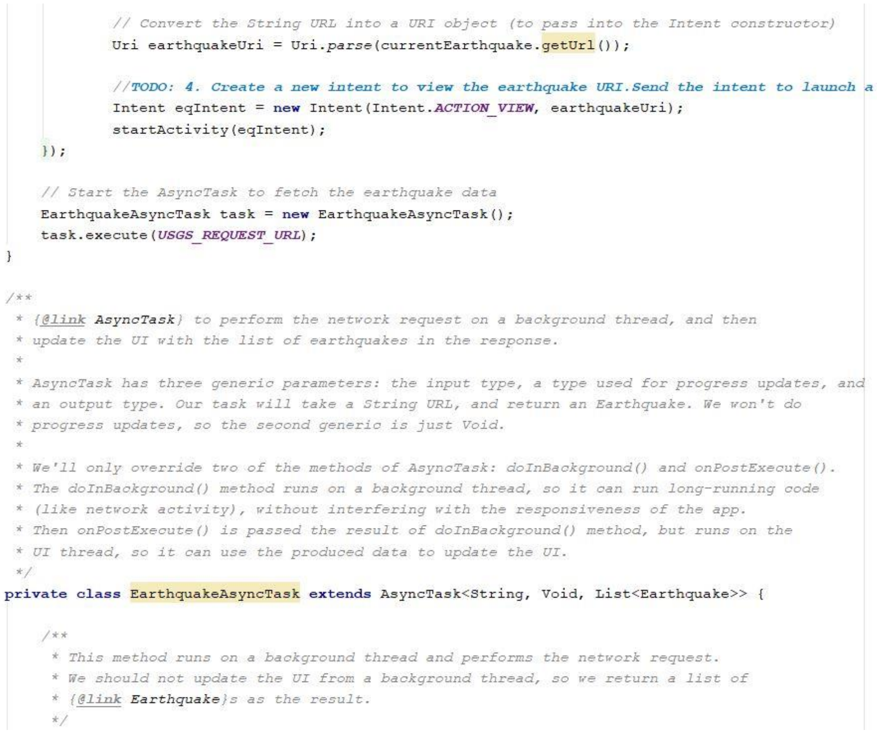
```
     *
     * @param magnitude is the magnitude (size) of the earthquake
     * @param location is the location where the earthquake happened
     * @param timeInMilliseconds is the time in milliseconds (from the Epoch) when the
     *                           earthquake happened
     * @param url is the website URL to find more details about the earthquake
     */
    public Earthquake(double magnitude, String location, long timeInMilliseconds, String url) {
        mMagnitude = magnitude;
        mLocation = location;
        mTimeInMilliseconds = timeInMilliseconds;
        mUrl = url;
    }

    /**
     * Returns the magnitude of the earthquake.
     */
    public double getMagnitude() { return mMagnitude; }

    /**
     * Returns the location of the earthquake.
     */
    public String getLocation() { return mLocation; }

    /**
     * Returns the time of the earthquake.
     */
    public long getTimeInMilliseconds() { return mTimeInMilliseconds; }

    /**
     * Returns the website URL to find more information about the earthquake.
     */
    public String getUrl() { return mUrl; }
}
```

2. EarthquakeActivity.java

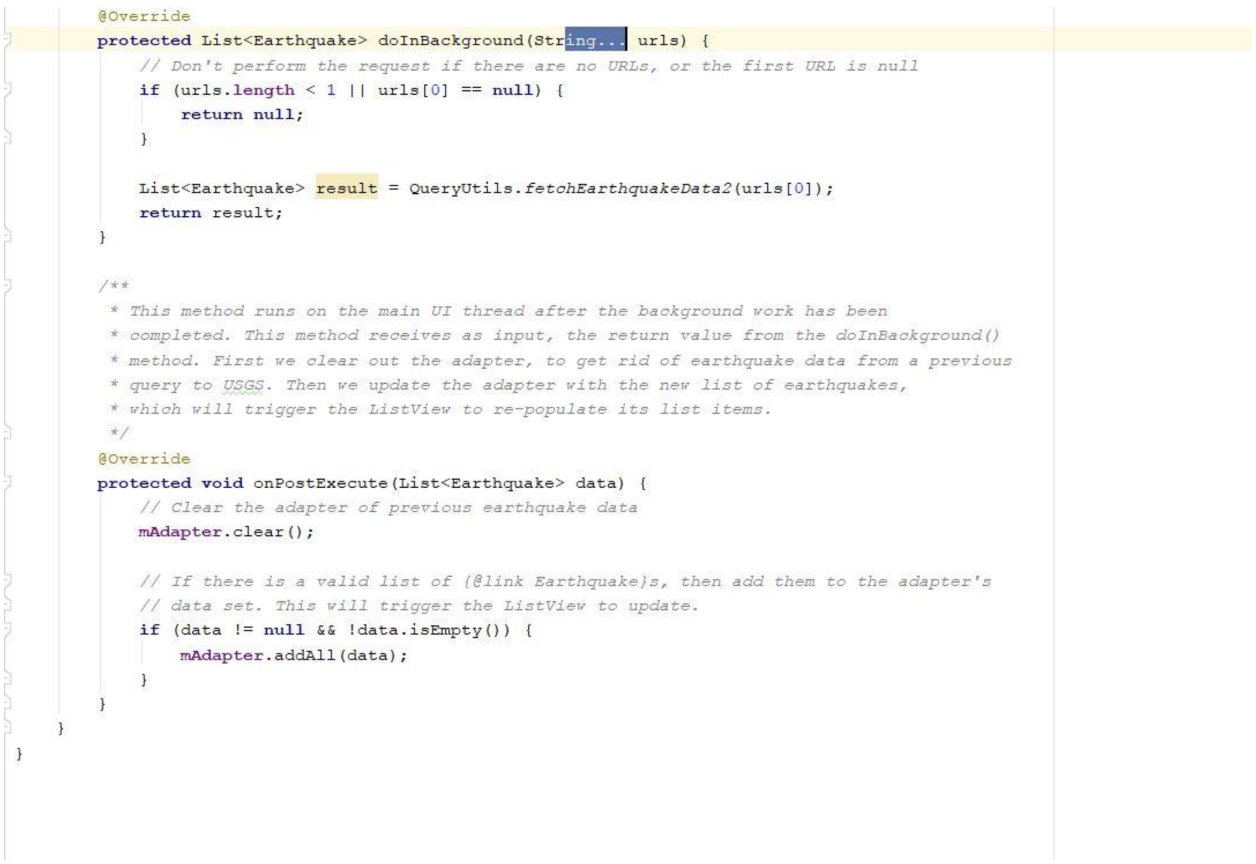
- It is used to define the API URL to obtain earthquake list.
- It contains the list view in order to view the earthquake data into list format.
- It has async task to handle the errors while obtaining the json response from URL String.
- It redirects to Queryutils.java to get the required data.
- It uses the adapter class also. The description for this is given below.
- Corresponding code is shown below.



```java
package com.example.karthik.earthquakeapp;
import ...

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=3&maxmag=5&limit=10";

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

    @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);
```



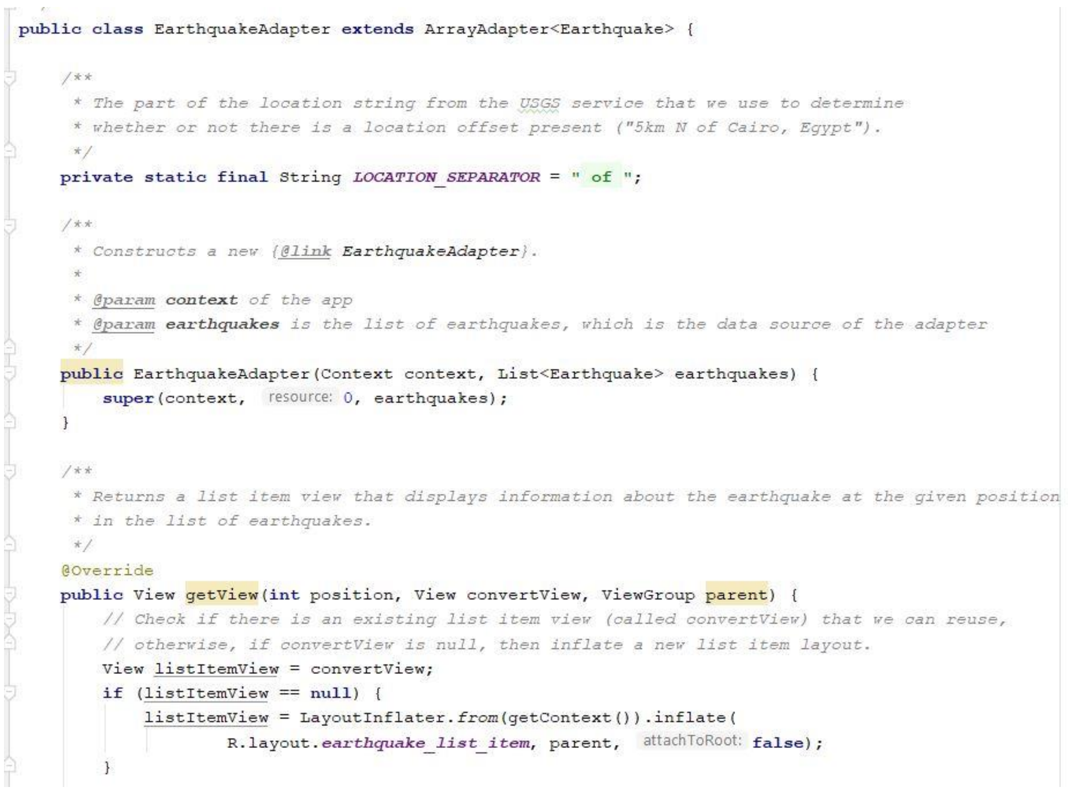
```java
                // Convert the String URL into a URI object (to pass into the Intent constructor)
                Uri earthquakeUri = Uri.parse(currentEarthquake.getUrl());

                //TODO: 4. Create a new intent to view the earthquake URI.Send the intent to launch a new activity
                Intent eqIntent = new Intent(Intent.ACTION_VIEW, earthquakeUri);
                startActivity(eqIntent);
        });

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

    /**
     * {@link AsyncTask} to perform the network request on a background thread, and then
     * update the UI with the list of earthquakes in the response.
     *
     * AsyncTask has three generic parameters: the input type, a type used for progress updates, and
     * an output type. Our task will take a String URL, and return an Earthquake. We won't do
     * progress updates, so the second generic is just Void.
     *
     * We'll only override two of the methods of AsyncTask: doInBackground() and onPostExecute().
     * The doInBackground() method runs on a background thread, so it can run long-running code
     * (like network activity), without interfering with the responsiveness of the app.
     * Then onPostExecute() is passed the result of doInBackground() method, but runs on the
     * UI thread, so it can use the produced data to update the UI.
     */
    private class EarthquakeAsyncTask extends AsyncTask<String, Void, List<Earthquake>> {

        /**
         * This method runs on a background thread and performs the network request.
         * We should not update the UI from a background thread, so we return a list of
         * {@link Earthquake}s as the result.
         */
```



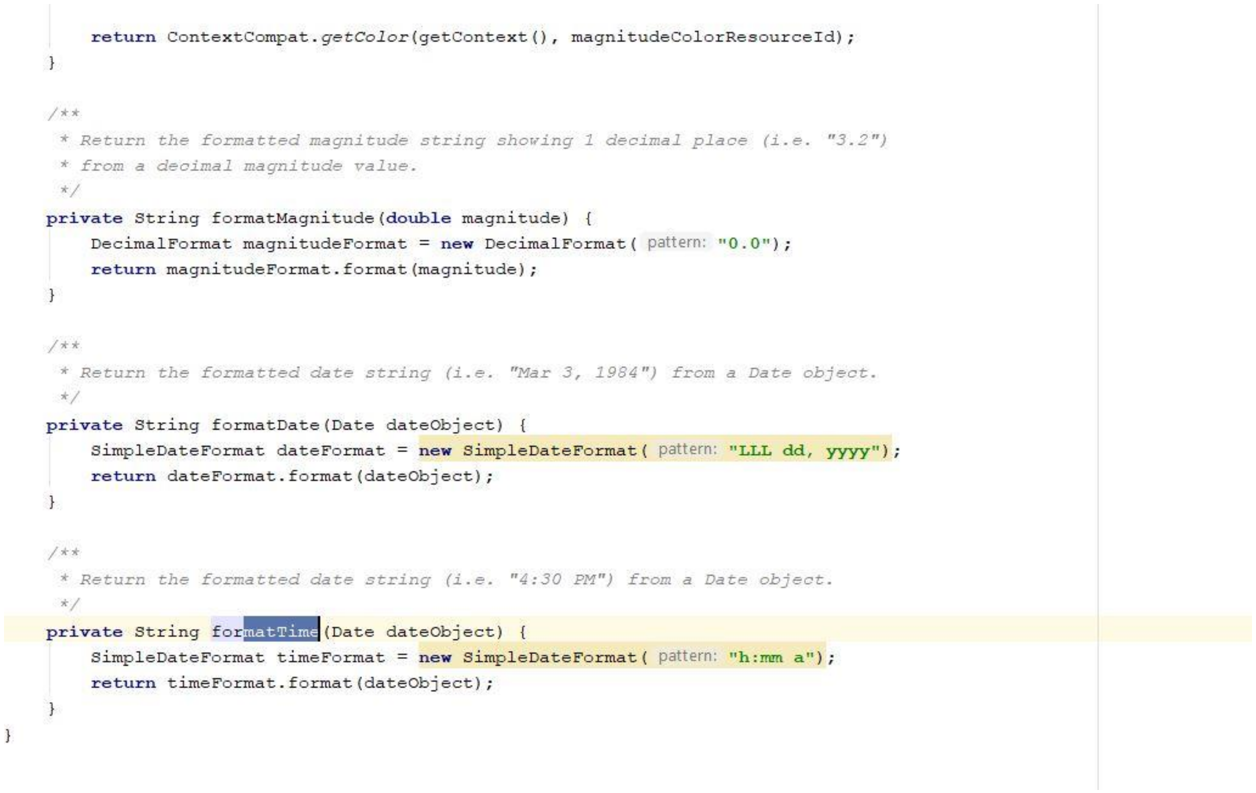
```
        @Override
        protected List<Earthquake> doInBackground(String... urls) {
            // Don't perform the request if there are no URLs, or the first URL is null
            if (urls.length < 1 || urls[0] == null) {
                return null;
            }

            List<Earthquake> result = QueryUtils.fetchEarthquakeData2(urls[0]);
            return result;
        }

        /**
         * This method runs on the main UI thread after the background work has been
         * completed. This method receives as input, the return value from the doInBackground()
         * method. First we clear out the adapter, to get rid of earthquake data from a previous
         * query to USGS. Then we update the adapter with the new list of earthquakes,
         * which will trigger the ListView to re-populate its list 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);
            }
        }
    }
}
```

3. EarthquakeAdapter.java

- The main function of Adapter is used to convert the data to list view.
- Here It directs earthquake data to earthquake list.
- Code snippet is shown here.



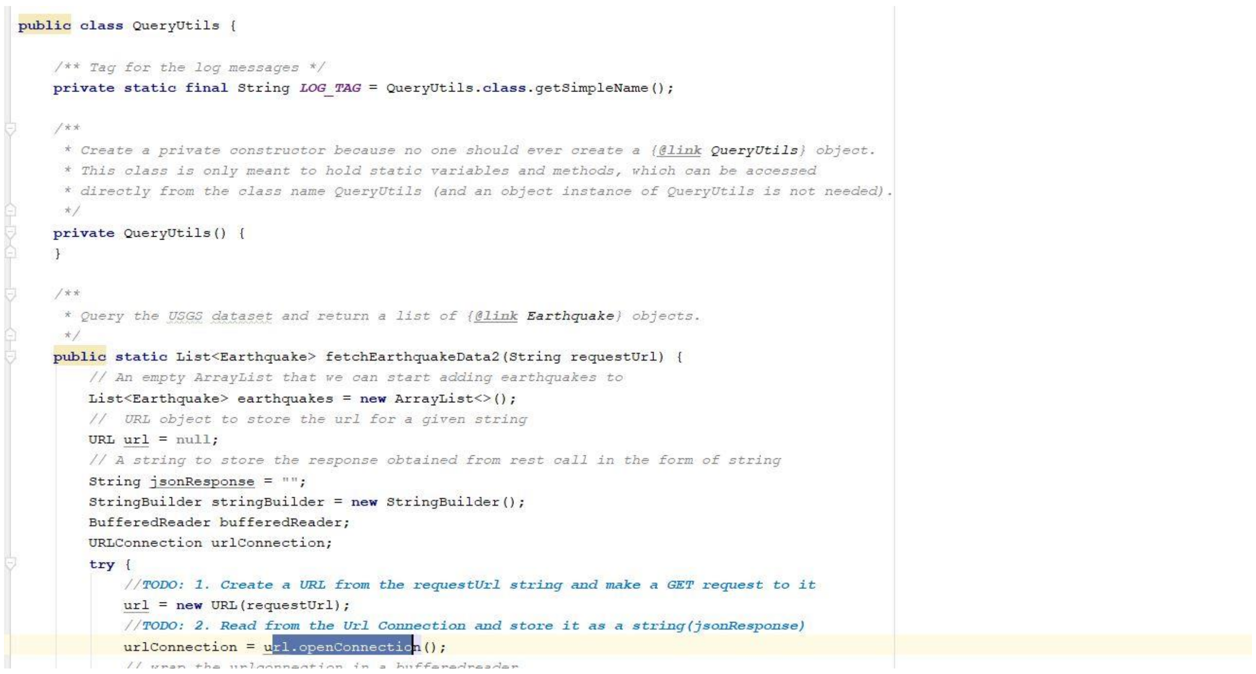
```java
public class EarthquakeAdapter extends ArrayAdapter<Earthquake> {

    /**
     * The part of the location string from the USGS service that we use to determine
     * whether or not there is a location offset present ("5km N of Cairo, Egypt").
     */
    private static final String LOCATION_SEPARATOR = " of ";

    /**
     * Constructs a new {@link EarthquakeAdapter}.
     *
     * @param context of the app
     * @param earthquakes is the list of earthquakes, which is the data source of the adapter
     */
    public EarthquakeAdapter(Context context, List<Earthquake> earthquakes) {
        super(context,  resource: 0, earthquakes);
    }

    /**
     * Returns a list item view that displays information about the earthquake at the given position
     * in the list of earthquakes.
     */
    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        // Check if there is an existing list item view (called convertView) that we can reuse,
        // otherwise, if convertView is null, then inflate a new list item layout.
        View listItemView = convertView;
        if (listItemView == null) {
            listItemView = LayoutInflater.from(getContext()).inflate(
                    R.layout.earthquake_list_item, parent,  attachToRoot: false);
        }
```



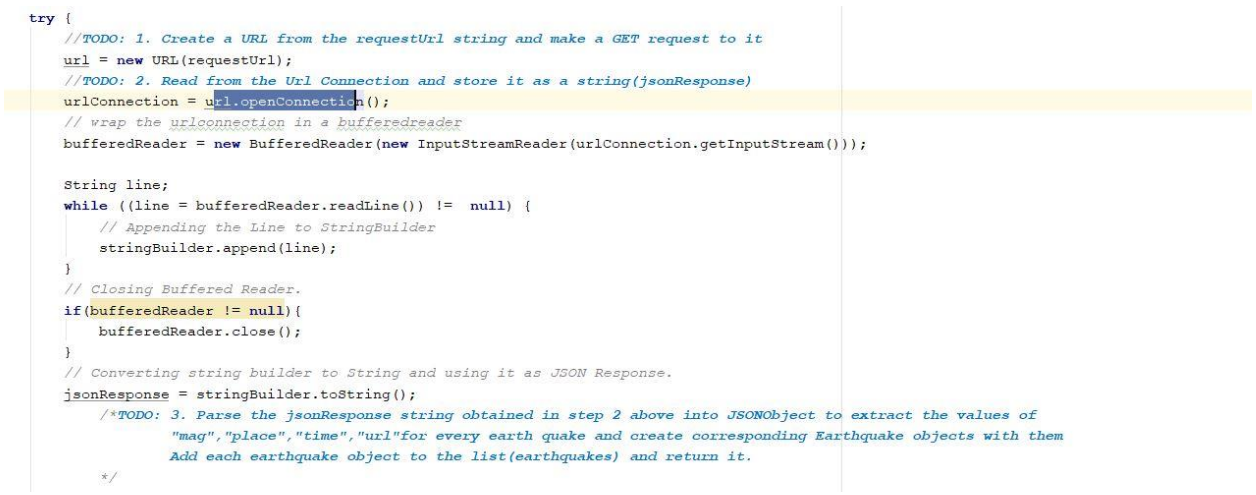
```java
        return ContextCompat.getColor(getContext(), magnitudeColorResourceId);
    }

    /**
     * Return the formatted magnitude string showing 1 decimal place (i.e. "3.2")
     * from a decimal magnitude value.
     */
    private String formatMagnitude(double magnitude) {
        DecimalFormat magnitudeFormat = new DecimalFormat( pattern: "0.0");
        return magnitudeFormat.format(magnitude);
    }

    /**
     * Return the formatted date string (i.e. "Mar 3, 1984") from a Date object.
     */
    private String formatDate(Date dateObject) {
        SimpleDateFormat dateFormat = new SimpleDateFormat( pattern: "LLL dd, yyyy");
        return dateFormat.format(dateObject);
    }

    /**
     * Return the formatted date string (i.e. "4:30 PM") from a Date object.
     */
    private String formatTime(Date dateObject) {
        SimpleDateFormat timeFormat = new SimpleDateFormat( pattern: "h:mm a");
        return timeFormat.format(dateObject);
    }
}
```

4.Queryutils.java

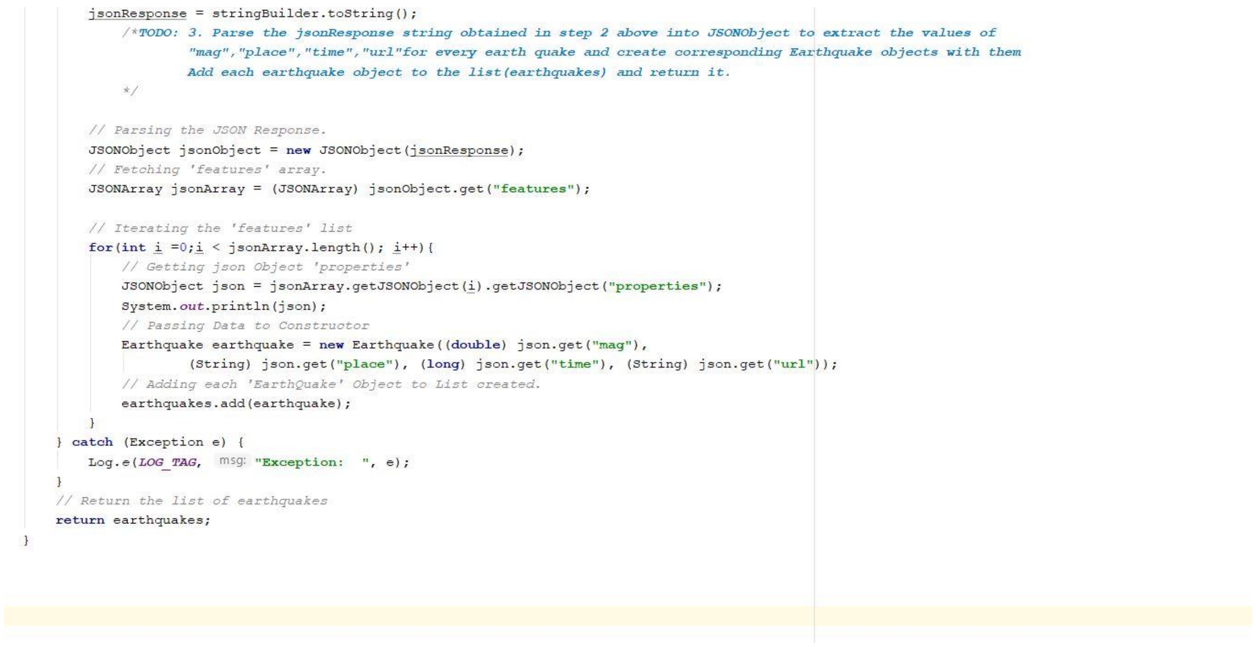
- It plays a major role in this task.
- Usually It converts the URL String to JSON data using get url request.
- Similarly It will obtain time,location,magnitude for particular earthquake using get json request.
- Code snippets are given below.



```
public class QueryUtils {

    /** Tag for the log messages */
    private static final String LOG_TAG = QueryUtils.class.getSimpleName();

    /**
     * Create a private constructor because no one should ever create a {@link QueryUtils} object.
     * This class is only meant to hold static variables and methods, which can be accessed
     * directly from the class name QueryUtils (and an object instance of QueryUtils is not needed).
     */
    private QueryUtils() {
    }

    /**
     * Query the USGS dataset and return a list of {@link Earthquake} objects.
     */
    public static List<Earthquake> fetchEarthquakeData2(String requestUrl) {
        // An empty ArrayList that we can start adding earthquakes to
        List<Earthquake> earthquakes = new ArrayList<>();
        //  URL object to store the url for a given string
        URL url = null;
        // A string to store the response obtained from rest call in the form of string
        String jsonResponse = "";
        StringBuilder stringBuilder = new StringBuilder();
        BufferedReader bufferedReader;
        URLConnection urlConnection;
        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)
            urlConnection = url.openConnection();
```

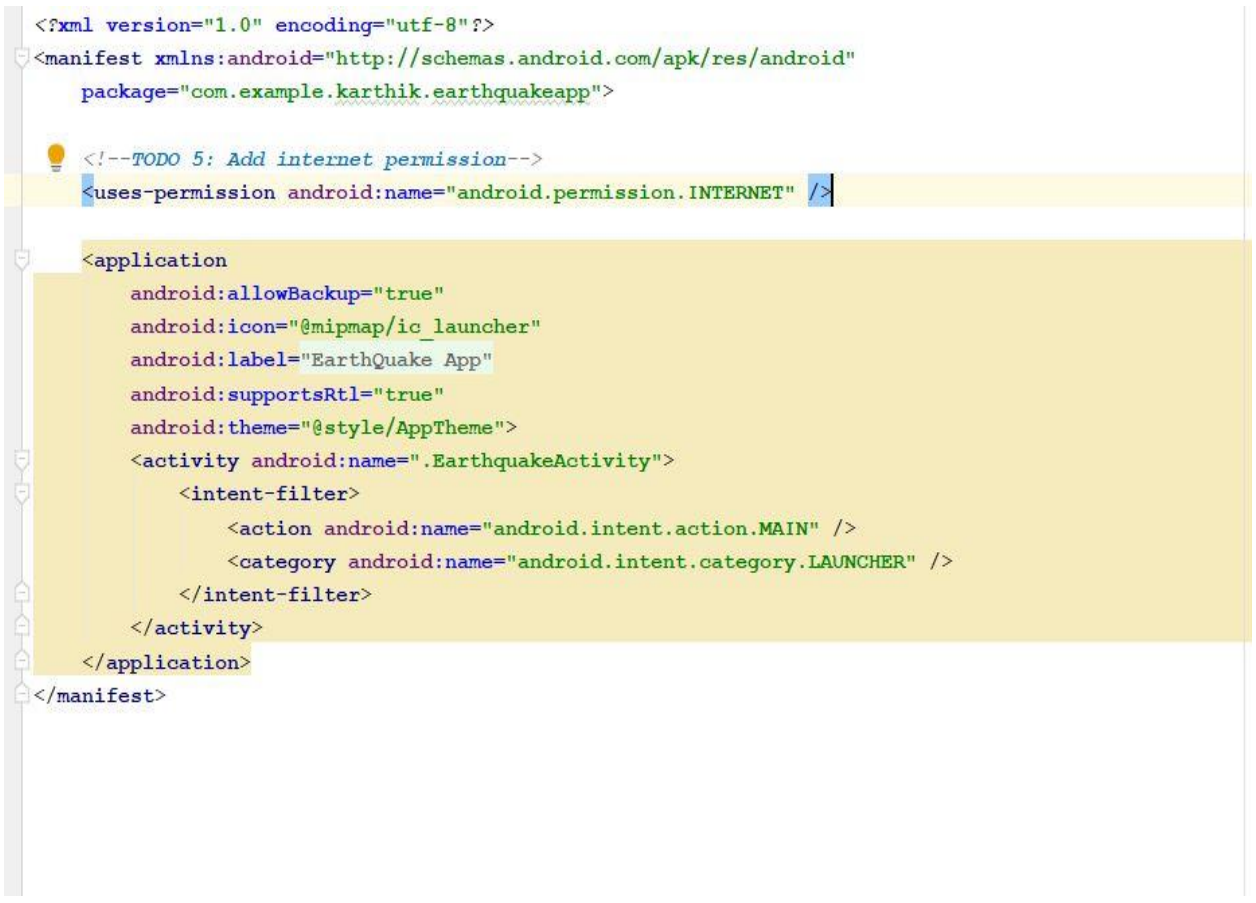


```
try {
    //TODO: 1. Create a URL from the requestUrl string and make a GET request to it
    url = new URL(requestUrl);
    //TODO: 2. Read from the Url Connection and store it as a string(jsonResponse)
    urlConnection = url.openConnection();
    // wrap the urlconnection in a bufferedreader
    bufferedReader = new BufferedReader(new InputStreamReader(urlConnection.getInputStream()));

    String line;
    while ((line = bufferedReader.readLine()) !=  null) {
        // Appending the Line to StringBuilder
        stringBuilder.append(line);
    }
    // Closing Buffered Reader.
    if(bufferedReader != null){
        bufferedReader.close();
    }
    // Converting string builder to String and using it as JSON Response.
    jsonResponse = stringBuilder.toString();
        /*TODO: 3. Parse the jsonResponse string obtained in step 2 above into JSONObject to extract the values of
                "mag","place","time","url"for every earth quake and create corresponding Earthquake objects with them
                Add each earthquake object to the list(earthquakes) and return it.
        */
```



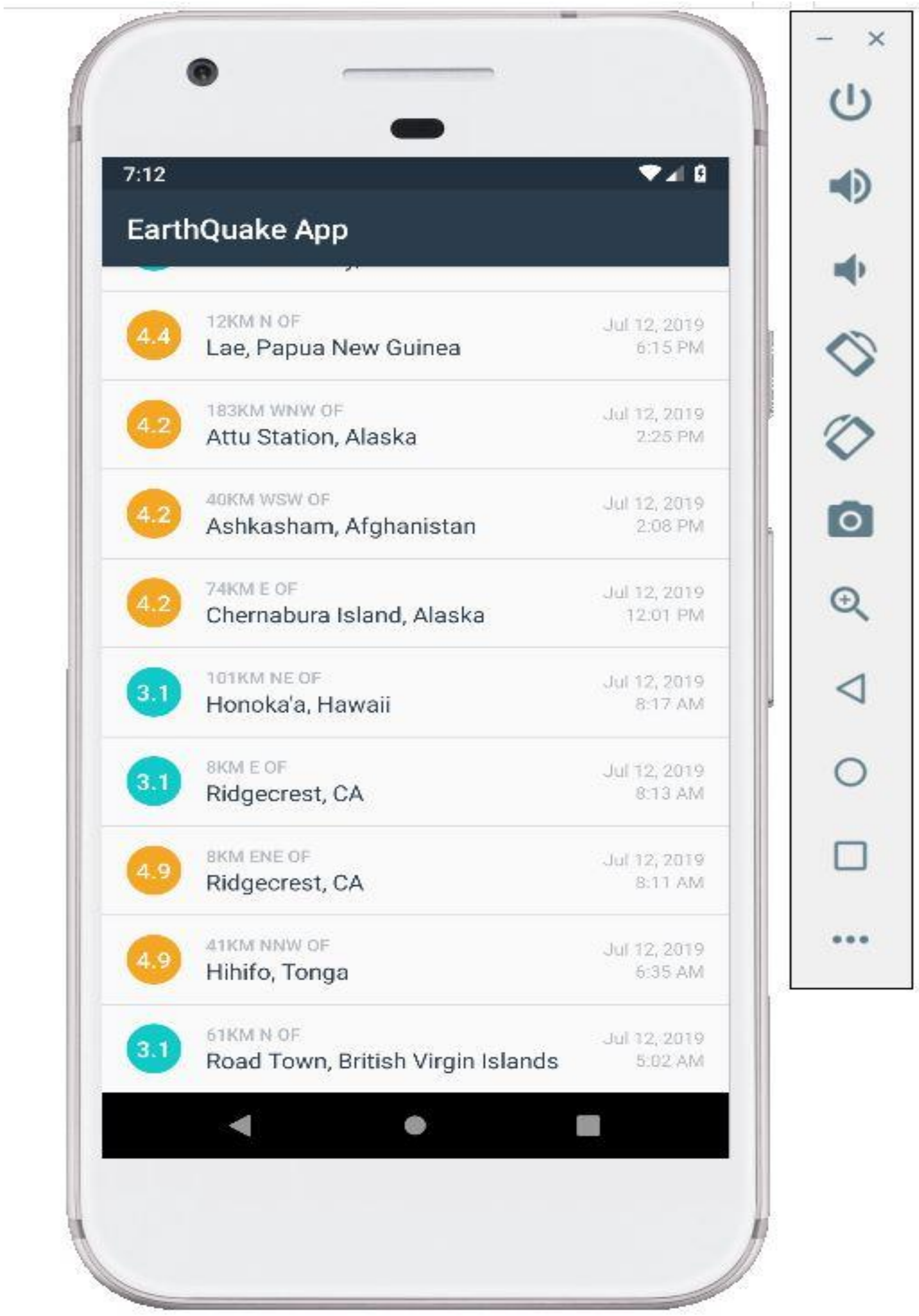
```
        jsonResponse = stringBuilder.toString();
            /*TODO: 3. Parse the jsonResponse string obtained in step 2 above into JSONObject to extract the values of
                    "mag","place","time","url"for every earth quake and create corresponding Earthquake objects with them
                    Add each earthquake object to the list(earthquakes) and return it.
             */

        // Parsing the JSON Response.
        JSONObject jsonObject = new JSONObject(jsonResponse);
        // Fetching 'features' array.
        JSONArray jsonArray = (JSONArray) jsonObject.get("features");

        // Iterating the 'features' list
        for(int i =0;i < jsonArray.length(); i++){
            // Getting json Object 'properties'
            JSONObject json = jsonArray.getJSONObject(i).getJSONObject("properties");
            System.out.println(json);
            // Passing Data to Constructor
            Earthquake earthquake = new Earthquake((double) json.get("mag"),
                    (String) json.get("place"), (long) json.get("time"), (String) json.get("url"));
            // Adding each 'EarthQuake' Object to List created.
            earthquakes.add(earthquake);
        }
    } catch (Exception e) {
        Log.e(LOG_TAG, msg: "Exception:  ", e);
    }
    // Return the list of earthquakes
    return earthquakes;
}
```

- And Accessing internet using manifest.xml file shown below.



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

    <!--TODO 5: Add internet permission-->
    <uses-permission android:name="android.permission.INTERNET" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="EarthQuake App"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".EarthquakeActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

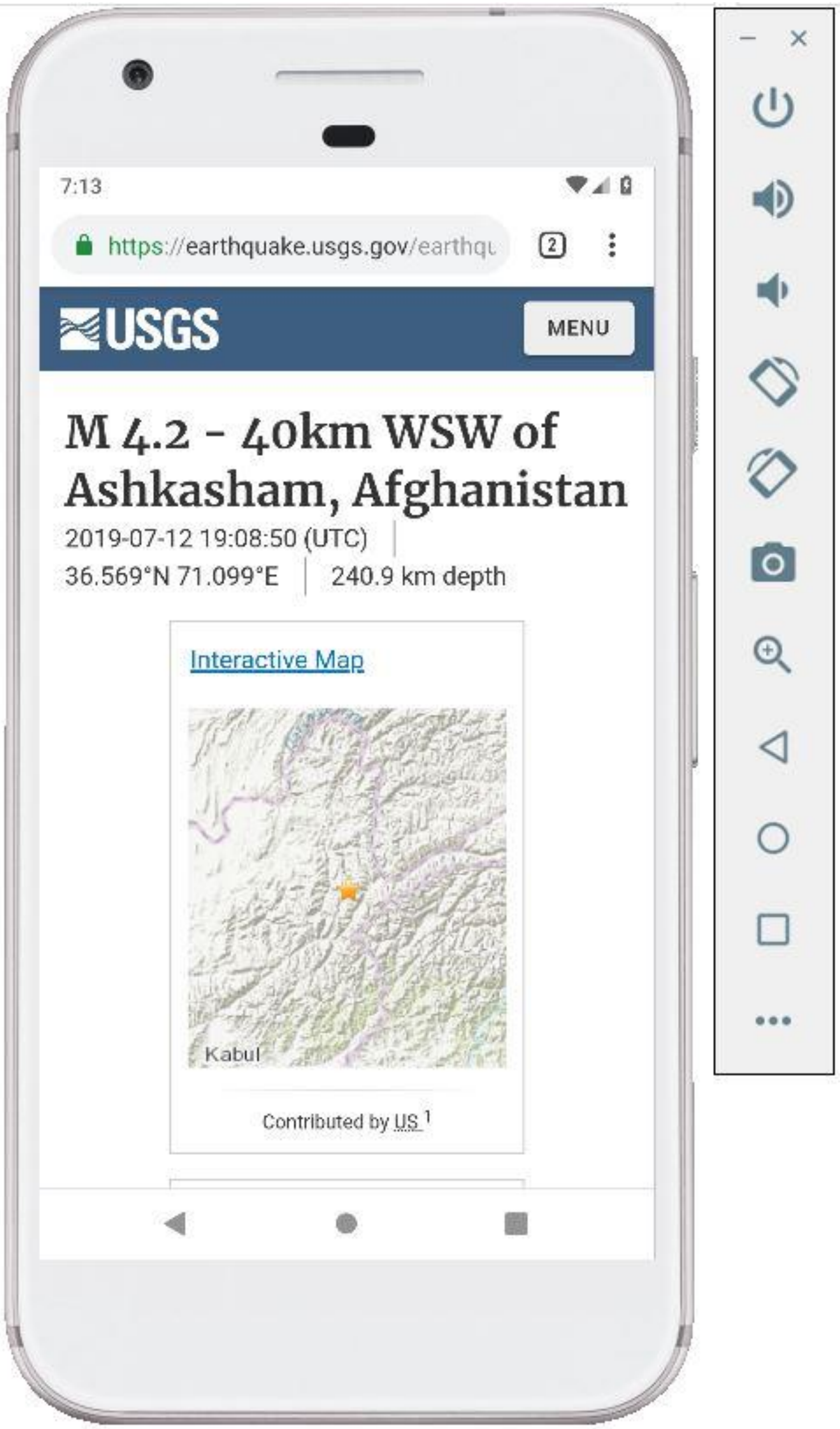
- Corresponding outputs to show the list of earthquake information and when user click on particular earthquake, fetching the respective URL are given below.



7:12

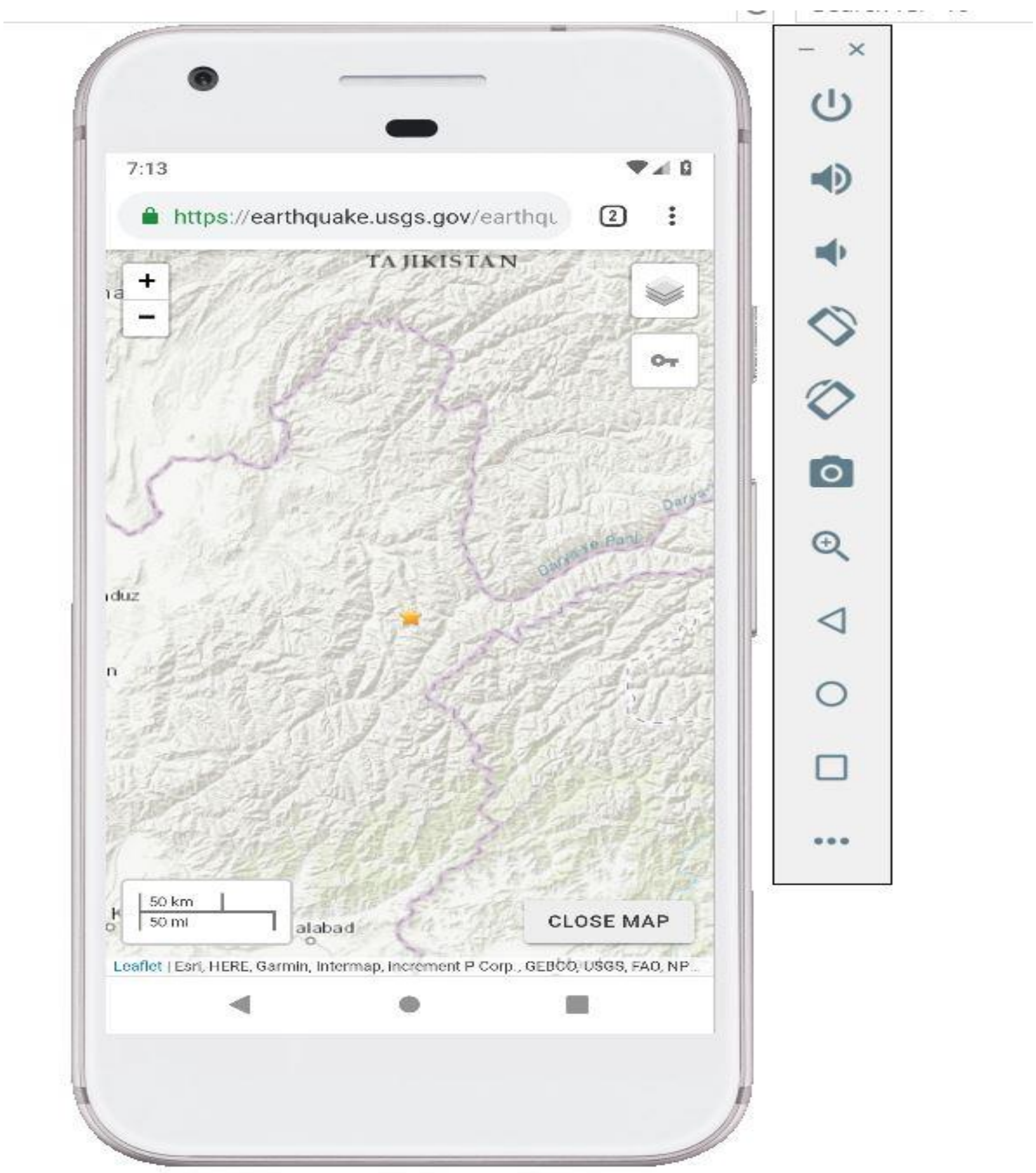
EarthQuake App

4.4	12KM N OF Lae, Papua New Guinea	Jul 12, 2019 6:15 PM
4.2	183KM WNW OF Attu Station, Alaska	Jul 12, 2019 2:25 PM
4.2	40KM WSW OF Ashkasham, Afghanistan	Jul 12, 2019 2:08 PM
4.2	74KM E OF Chernabura Island, Alaska	Jul 12, 2019 12:01 PM
3.1	101KM NE OF Honoka'a, Hawaii	Jul 12, 2019 8:17 AM
3.1	8KM E OF Ridgecrest, CA	Jul 12, 2019 8:13 AM
4.9	8KM ENE OF Ridgecrest, CA	Jul 12, 2019 8:11 AM
4.9	41KM NNW OF Hihifo, Tonga	Jul 12, 2019 6:35 AM
3.1	61KM N OF Road Town, British Virgin Islands	Jul 12, 2019 5:02 AM



M 4.2 - 40km WSW of Ashkasham, Afghanistan

2019-07-12 19:08:50 (UTC)
36.569°N 71.099°E | 240.9 km depth


7:13
https://earthquake.usgs.gov/earthqu
TAJIKISTAN
Daryaye Panj
50 km
50 mi
alabad
CLOSE MAP
Leaflet | Esri, HERE, Garmin, Intermap, Increment P Corp., GEBCO, USGS, FAO, NP...

Conclusion:

- Hence Using this application learnt how to fetch the json data using API's and handling errors using async task along with parsing json data.