```
1   import android.graphics.Color;
2   import android.net.Uri;
3   import android.os.AsyncTask;
4   import android.os.Bundle;
5   import android.support.v7.app.AppCompatActivity;
6   import android.text.Spannable;
7   import android.text.SpannableString;
8   import android.text.Spanned;
9   import android.text.TextPaint;
10  import android.text.TextUtils;
11  import android.text.method.LinkMovementMethod;
12  import android.text.style.ClickableSpan;
13  import android.text.style.ForegroundColorSpan;
14  import android.util.Log;
15  import android.view.Menu;
16  import android.view.MenuItem;
17  import android.view.View;
18  import android.widget.Button;
19  import android.widget.TextView;
20  import android.widget.Toast;
21
22  import com.google.android.gms.appindexing.Action;
23  import com.google.android.gms.appindexing.AppIndex;
24  import com.google.android.gms.common.api.GoogleApiClient;
25
26  import org.json.JSONArray;
27  import org.json.JSONException;
28  import org.json.JSONObject;
29
30  import java.io.BufferedReader;
31  import java.io.IOException;
32  import java.io.InputStream;
33  import java.io.InputStreamReader;
34  import java.net.HttpURLConnection;
35  import java.net.MalformedURLException;
36  import java.net.URL;
37  import java.util.LinkedList;
38
39  public class MainActivity extends AppCompatActivity {
40
41      private Button ourButton;
42      private TextView ourMessage;
43      private String result;
44      private int i = 0;
45      private int currentWord =0;
46
47
48      private TextView statusView;
49      /**
50       * ATTENTION: This was auto-generated to implement the App Indexing API.
51       * See https://g.co/AppIndexing/AndroidStudio for more information.
52       */
53      private GoogleApiClient client;
54
55      private JSONArray loadUrl(String location) {
56          LinkedList<String> lines = new LinkedList<String>();
57
58          // Download URL to array of lines
59          try {
```

```
60              // Open URL
61              URL url = new URL(location);
62              HttpURLConnection conn = (HttpURLConnection) url.openConnection();
63
64              // Open buffered reader
65              InputStream is = conn.getInputStream();
66              InputStreamReader isr = new InputStreamReader(is);
67              BufferedReader buf = new BufferedReader(isr);
68
69              // Read data into lines array
70              String line;
71              while ((line = buf.readLine()) != null)
72                  lines.add(line);
73
74              // Close keep-alive connections
75              conn.disconnect();
76          }
77          catch (MalformedURLException e) {
78              Log.d("SfvLug", "Invalid URL: " + location);
79          }
80          catch (IOException e) {
81              Log.d("SfvLug", "Download failed for: " + location);
82          }
83
84          // Load text into JSON Object
85          String text = TextUtils.join("", lines);
86          try {
87              return new JSONArray(text);
88          }
89          catch (JSONException e) {
90              Log.d("SfvLug", "Invalid JSON Object: " + text);
91              return null;
92          }
93      }
94
95
96      /* Display the meeting status to the user */
97      private void setStatus(final JSONArray status) {
98
99          String text = null;
100
101             // Extract info from JSON object and format output messages
102             try {
103                 JSONObject quote = status.getJSONObject((i % 3));
104
105                 String qouteString = quote.getString("quote");
106
107                 Spannable wordtoSpan = new SpannableString(qouteString);
108
109                 int prev = 0, count = 0;
110
111                 //For loop for each word and create a new spannable
112                 for(String highLightWord: qouteString.split(" ")) {
113                     final int LOCALCOUNT = count++;
114
115                     ClickableSpan clickableSpan =
116                         new ClickableSpan() {
117                             @Override
118                             public void onClick(View textView) {
```

```
119                 Toast toastMessage = Toast.makeText(MainActivity.this, "Click t
he HighLight word " + LOCALCOUNT + " .", Toast.LENGTH_SHORT);
120                 toastMessage.show();
121                 currentWord++;
122                 setStatus(status);
123                 }
124
125                 @Override
126                 public void updateDrawState(TextPaint ds) {
127                 super.updateDrawState(ds);
128                 ds.setUnderlineText(true);
129                 }
130             };
131
132             if(LOCALCOUNT == currentWord) {
133                 wordtoSpan.setSpan(clickableSpan, prev, prev + highLightWord.length()
, Spanned.SPAN_EXCLUSIVE_EXCLUSIVE);
134             }
135             else if ( currentWord > LOCALCOUNT){
136                 wordtoSpan.setSpan(new ForegroundColorSpan(Color.BLUE), prev, prev +
highLightWord.length(), Spannable.SPAN_EXCLUSIVE_EXCLUSIVE);
137
138             }
139             prev = prev + highLightWord.length() + 1;
140
141         }
142         ourMessage.setMovementMethod(LinkMovementMethod.getInstance());
143         ourMessage.setText(wordtoSpan);
144
145     }
146     catch (NullPointerException e) {
147         ourMessage.setText("Error downloading meeting status");
148
149     }
150     catch (JSONException e) {
151
152         ourMessage.setText("Error parsing meeting status: ");
153
154     }
155
156 }
157
158
159     /* Trigger a refresh of the meeting status */
160     private void loadStatus() {
161         // Set URI location
162         String location = "http://52.33.120.235/juan.php";
163         //peer-reader.ddns.net
164         //ip: 52.33.120.235
165         ourMessage.setText("Loading status.. ");
166
167
168         // Update status in a background thread
169         //
170         // In Android, we normally cannot access the network from the main
171         // thread; doing so would cause the user interface to freeze during
172         // data transfer.
173         //
174         // Again, android provides several ways around this, here we use an
```

```
175         // AsyncTask which lets us run some code in a background thread and
176         // then update the user interface once the background code has
177         // finished.
178         //
179         // The first Java Generic parameter are:
180         //   1. String     - argument for doInBackground, from .execute()
181         //   2. Void       - not used here, normally used for progress bars
182         //   3. JSONObject - the return type from doInBackground which is
183         //                   passed to onPostExecute function.
184         new AsyncTask<String, Void, JSONArray>() {
185
186             // Called from a background thread, so we don't block the user
187             // interface. Using AsyncTask synchronization is handled for us.
188             protected JSONArray doInBackground(String... args) {
189                 // Java passes this as a variable argument array,
190                 // but we only use the first entry.
191                 return MainActivity.this.loadUrl(args[0]);
192             }
193
194             // Called once in the main thread once doInBackground finishes.
195             // This is executed in the Main thread once again so that we can
196             // update the user interface.
197             protected void onPostExecute(JSONArray status) {
198                 ourMessage.setText("Loading Post Execute.. ");
199                 MainActivity.this.setStatus(status);
200             }
201
202         }.execute(location);
203     }
204
205
206     @Override
207     protected void onCreate(Bundle savedInstanceState) {
208         super.onCreate(savedInstanceState);
209         //This is content on the screen
210         setContentView(R.layout.activity_main);
211         //Link Button to Textview
212         ourButton = (Button) findViewById(R.id.button);
213         ourMessage = (TextView) findViewById(R.id.textView);
214
215
216         //Listen on the button to be click
217         View.OnClickListener ourOnClickListener =
218             new View.OnClickListener() {
219                 @Override
220                 public void onClick(View v) {
221                     currentWord = 0;
222                     loadStatus();
223                     i++;
224                 }
225             };
226     //
227         ourButton.setOnClickListener(ourOnClickListener);
228
229         // ATTENTION: This was auto-generated to implement the App Indexing API.
230         // See https://g.co/AppIndexing/AndroidStudio for more information.
231         client = new GoogleApiClient.Builder(this).addApi(AppIndex.API).build();
232     }
233
```

```
234
235     @Override
236     public boolean onCreateOptionsMenu(Menu menu) {
237       // Inflate the menu; this adds items to the action bar if it is present.
238       //Add menu_main on screen
239       getMenuInflater().inflate(R.menu.menu_main, menu);
240       return true;
241     }
242
243     @Override
244     public boolean onOptionsItemSelected(MenuItem item) {
245       // Handle action bar item clicks here. The action bar will
246       // automatically handle clicks on the Home/Up button, so long
247       // as you specify a parent activity in AndroidManifest.xml.
248       int id = item.getItemId();
249
250       //noinspection SimplifiableIfStatement
251       if (id == R.id.action_settings) {
252           Toast toastMessage = Toast.makeText(this, "Text value is now " + ourMessage
.getText(), Toast.LENGTH_LONG);
253           toastMessage.show();
254
255           return true;
256       }
257
258       return super.onOptionsItemSelected(item);
259     }
260
261     @Override
262     public void onStart() {
263       super.onStart();
264
265       // ATTENTION: This was auto-generated to implement the App Indexing API.
266       // See https://g.co/AppIndexing/AndroidStudio for more information.
267       client.connect();
268       Action viewAction = Action.newAction(
269               Action.TYPE_VIEW, // TODO: choose an action type.
270               "Main Page", // TODO: Define a title for the content shown.
271               // TODO: If you have web page content that matches this app activity's
content,
272               // make sure this auto-generated web page URL is correct.
273               // Otherwise, set the URL to null.
274               Uri.parse("http://host/path"),
275               // TODO: Make sure this auto-generated app deep link URI is correct.
276               Uri.parse("android-app://coolcatmeow.org.helloworld/http/host/path")
277           );
278       AppIndex.AppIndexApi.start(client, viewAction);
279     }
280
281     @Override
282     public void onStop() {
283       super.onStop();
284
285       // ATTENTION: This was auto-generated to implement the App Indexing API.
286       // See https://g.co/AppIndexing/AndroidStudio for more information.
287       Action viewAction = Action.newAction(
288               Action.TYPE_VIEW, // TODO: choose an action type.
289               "Main Page", // TODO: Define a title for the content shown.
290               // TODO: If you have web page content that matches this app activity's
```

```
content,
291             // make sure this auto-generated web page URL is correct.
292             // Otherwise, set the URL to null.
293             Uri.parse("http://host/path"),
294             // TODO: Make sure this auto-generated app deep link URI is correct.
295             Uri.parse("android-app://coolcatmeow.org.helloworld/http/host/path")
296         );
297     AppIndex.AppIndexApi.end(client, viewAction);
298     client.disconnect();
299 }
300 }
```