Keefer Bibby, TJ Markham, Allan Tuquero, Shelley Suhling

Professor Mongan

CS 275-003 Web and Mobile App Development

27 February 2015

Interim Project Status Report

Our group is working on building an app called TempoTuner. TempoTuner is an app where a user inputs parameters such as tempo, genre, and "hipster-ness" of music. TempoTuner then takes these parameters and builds a Spotify playlist and outputs a link to the playlist for the user to follow.

The group has met three times so far. At the first meeting, we laid out our plans for the project. At the second meeting, we started making a basic UI, testing Spotify OAuth procedures, and working with EchoNest calls. For the UI, TJ originally started worked on making a java applet so that we could use the same code for the Android application. However, since applets are so obsolete, a decision was made to develop a more appropriate JavaScript/HTML/CSS based app.

Shelley started making a HTML/CSS page and TJ later took over to refine it (see Figures 2 - 4). Allan worked on EchoNest calls (see Figures 6 and 7), and also helped Shelley and Keefer work on the OAuth (see Figure 8). Keefer worked on the Android app (see Figure 5), however once the switch was made to JavaScript/HTML/CSS, the Android app was no longer necessary. After each meeting, each member has worked on their individual parts of the project on their off-time before meeting again to touch-base and stay on track for the remainder of the project.

With two weeks remaining for our project each member has been given a different set of tasks to complete. We plan to meet each Thursday and Saturday to bring all the elements together and polish the functionality and appearance of the application. Keefer intends to do more research into Cloudmine and how to integrate that with the project. TJ plans to improve the UI of the Web Application by incorporating Bootstrap and implementing other general UI improvements. Allan is going to get Spotify calls working using JavaScript. Shelley is going work on porting the Spotify OAuth procedures from java to JavaScript. Once everybody's task is complete, everyone will work to get the Application together and working.

The following are lists of what we have done, and what we still need to do, along with the names of the people that will be working on each item. A timeline can also be seen in Figure 1.

What we have done:

- UI (TJ, Shelley)
- Working echonest calls (Allan)
- Oauth - Java (Allan, Shelley, Keefer)
- Android (Keefer)
- Applets (TJ)

What we need to do:

- Improve UI (TJ)
- CloudMine (Keefer)
- Spotify calls (Allan)
- Working OAuth - JavaScript (Shelley)
- Connect all the pieces (everybody)

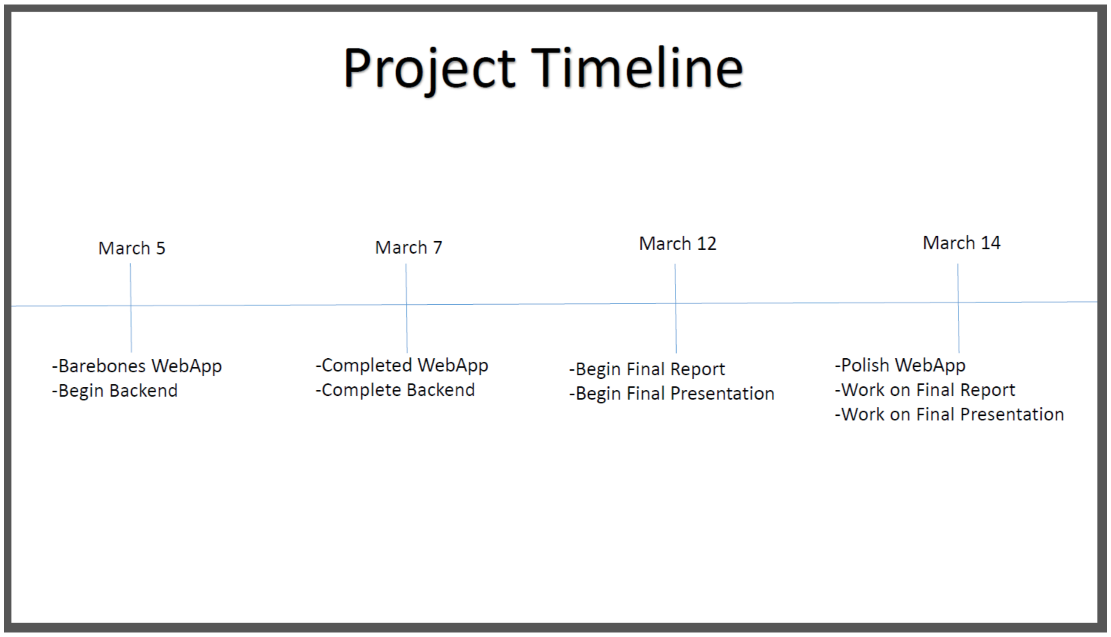


Fig. 1 – Project Timeline

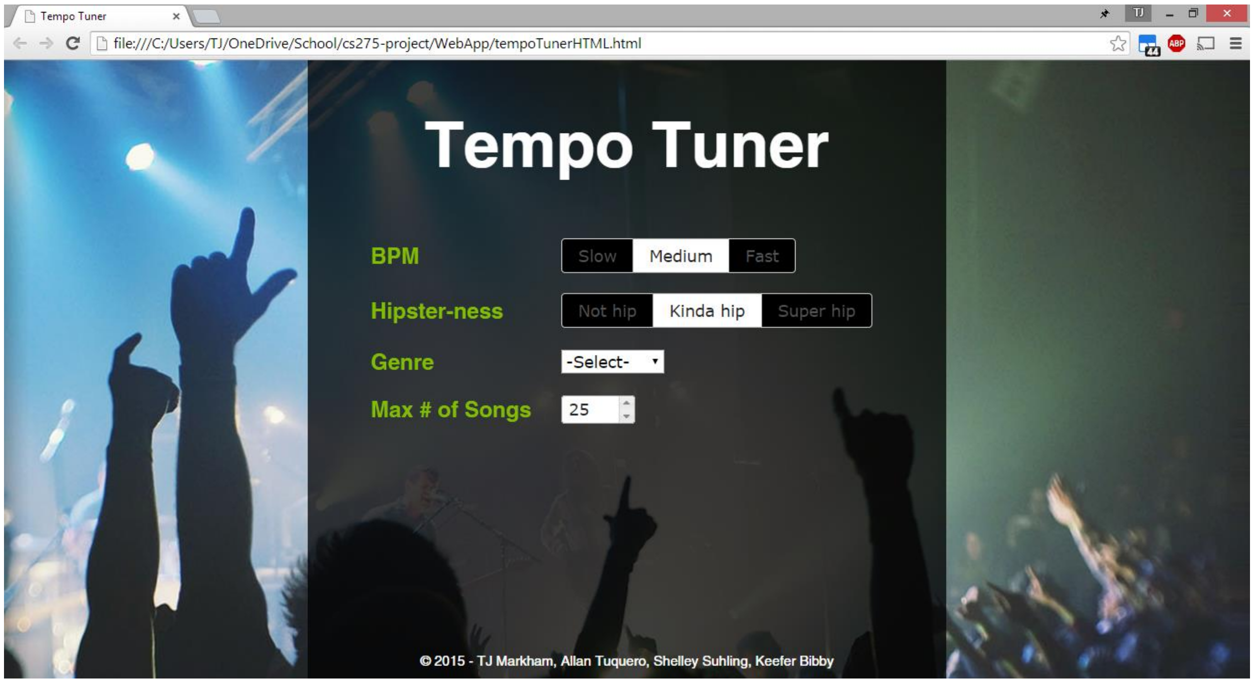

Tempo Tuner
file:///C:/Users/TJ/OneDrive/School/cs275-project/WebApp/tempoTunerHTML.html
Tempo Tuner
BPM
Slow
Medium
Fast
Hipster-ness
Not hip
Kinda hip
Super hip
Genre
-Select-
Max # of Songs
25
© 2015 - TJ Markham, Allan Tuquero, Shelley Suhling, Keefer Bibby

Fig. 2 – UI on Standard Screen

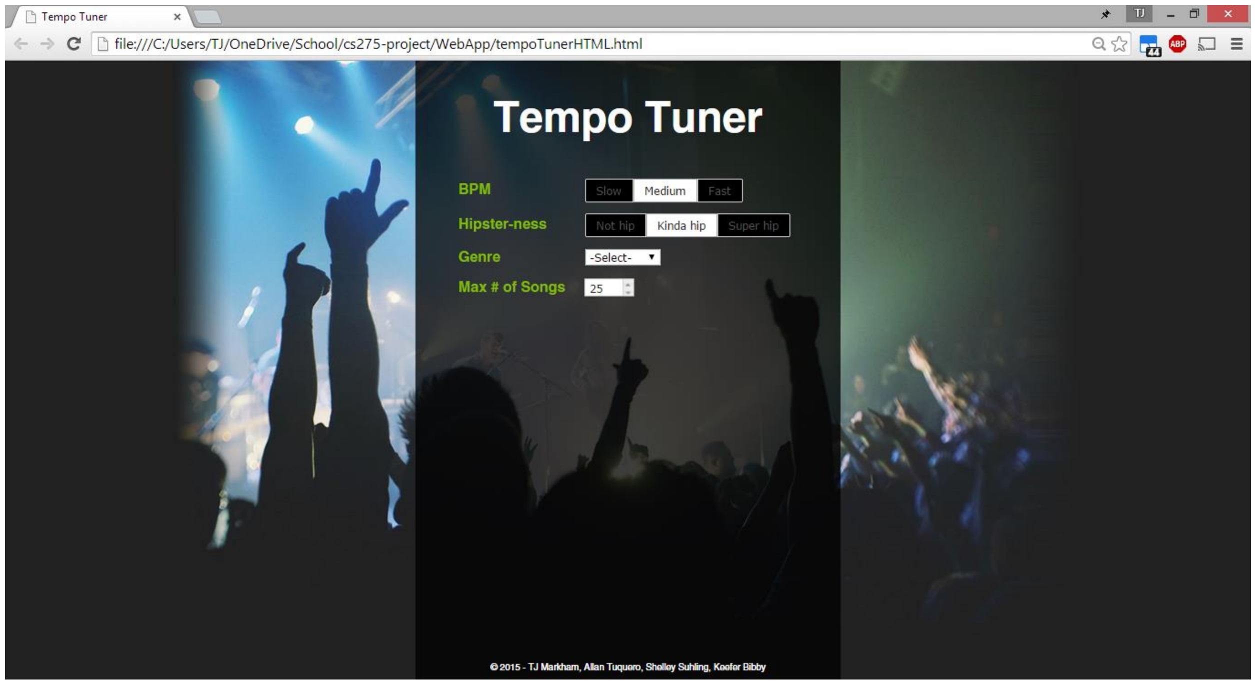

Tempo Tuner
file:///C:/Users/TJ/OneDrive/School/cs275-project/WebApp/tempoTunerHTML.html
Tempo Tuner
BPM
Slow
Medium
Fast
Hipster-ness
Not hip
Kinda hip
Super hip
Genre
-Select-
Max # of Songs
25
© 2015 - TJ Markham, Allan Tuquero, Shelley Suhling, Keefer Bibby

Fig. 3 – UI on Larger Screen

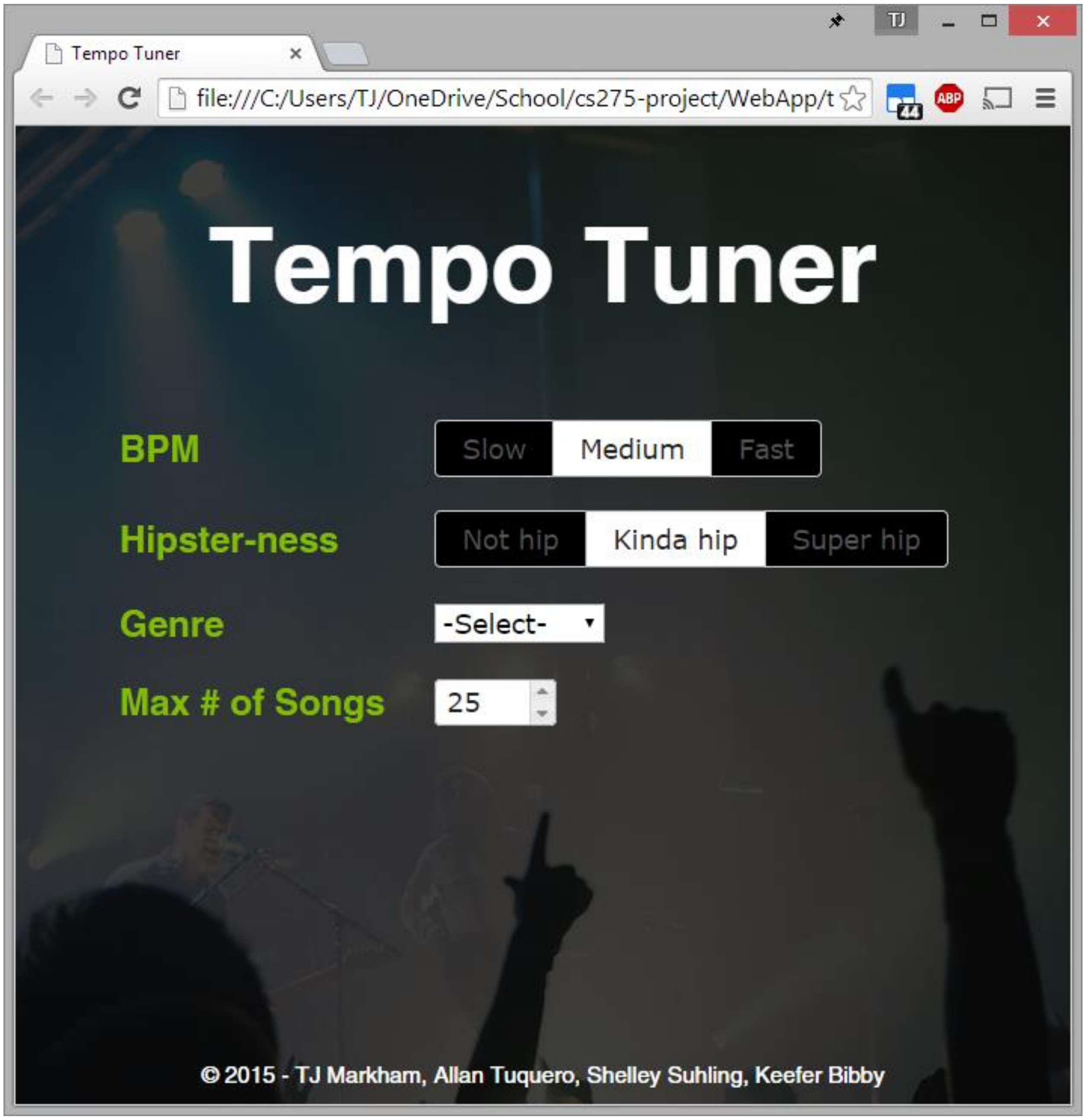

Tempo Tuner
file:///C:/Users/TJ/OneDrive/School/cs275-project/WebApp/t
Tempo Tuner
BPM
Slow
Medium
Fast
Hipster-ness
Not hip
Kinda hip
Super hip
Genre
-Select-
Max # of Songs
25
© 2015 - TJ Markham, Allan Tuquero, Shelley Suhling, Keefer Bibby

Fig. 4 – Example of what an app-view may look like

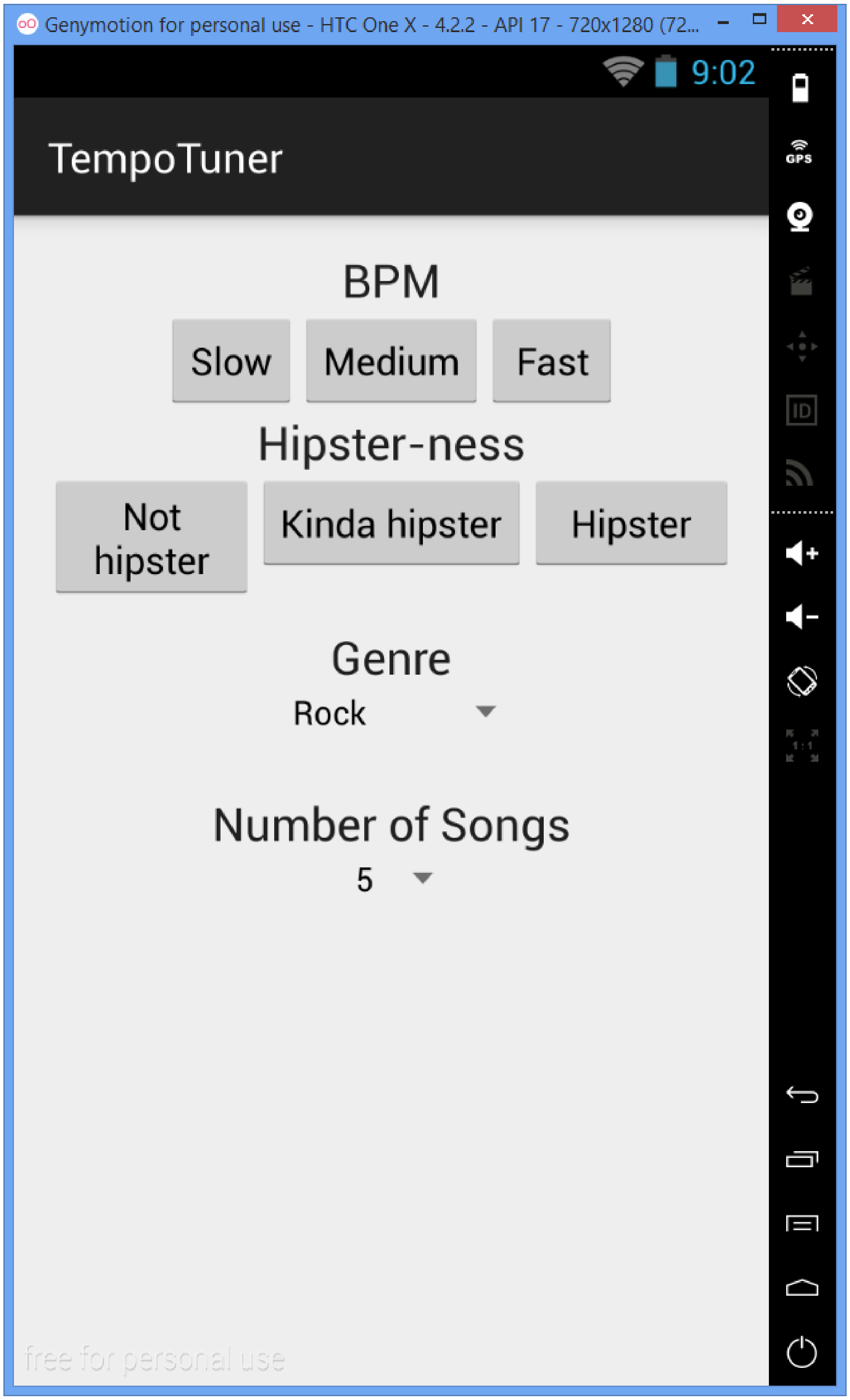
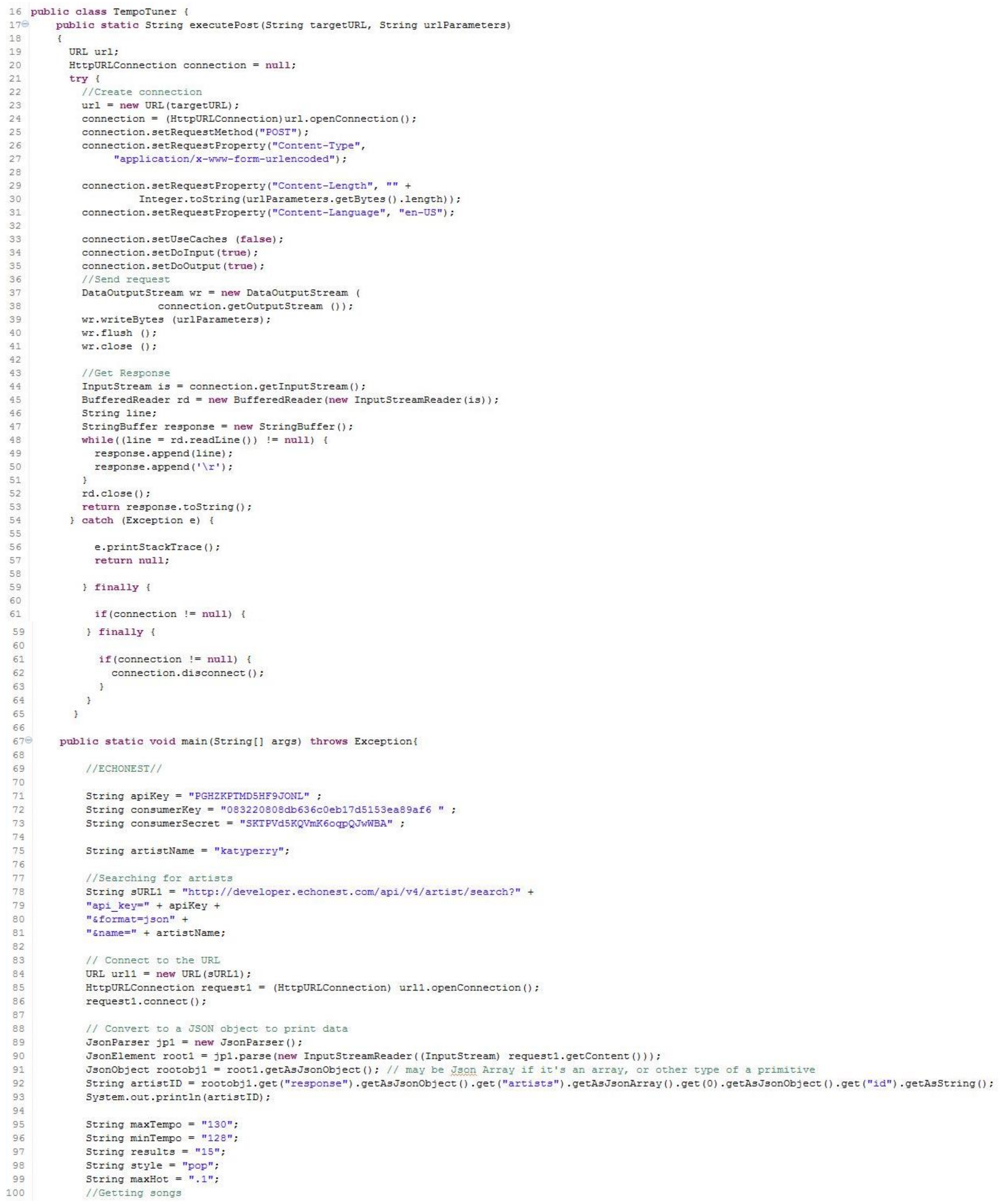


Fig. 5 – Original Android App

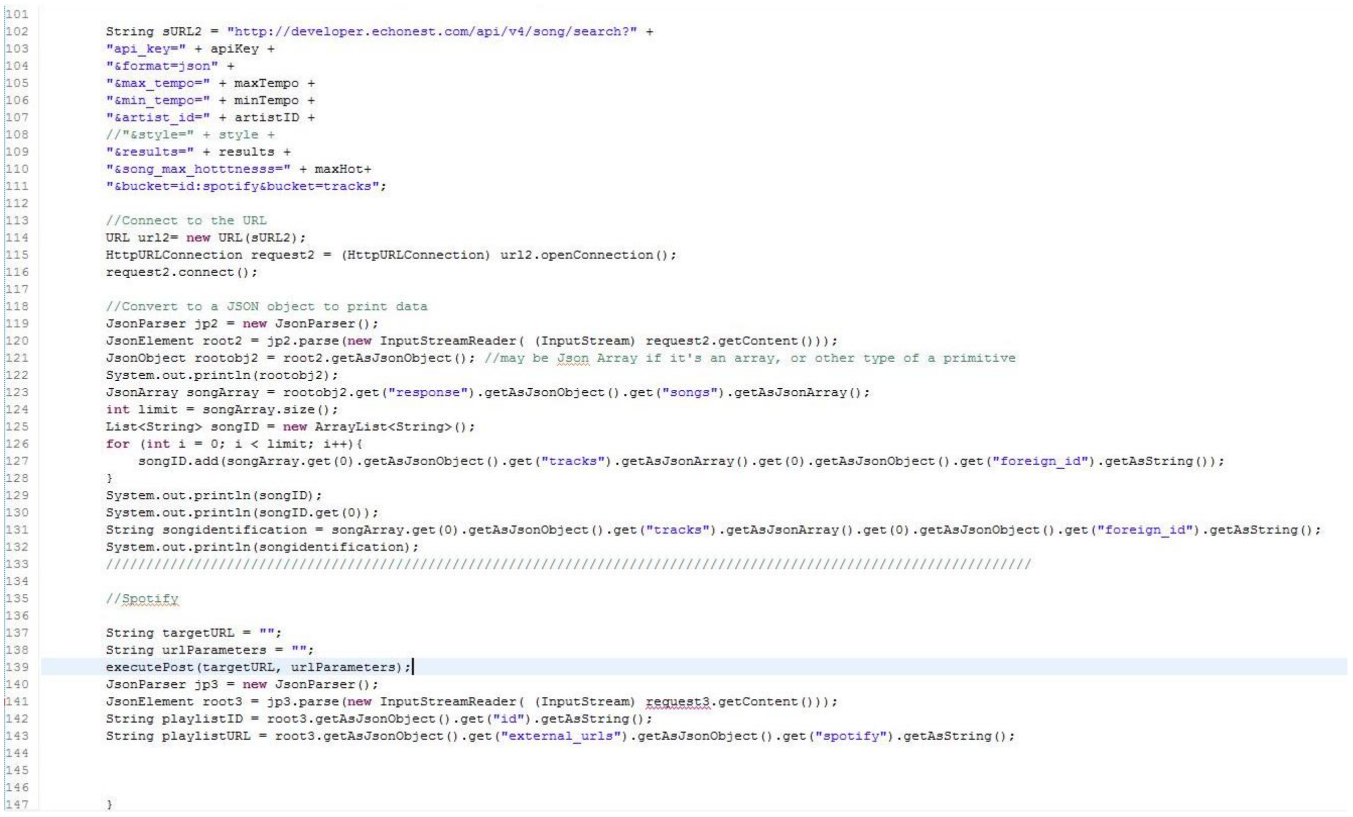


```
public class TempoTuner {
    public static String executePost(String targetURL, String urlParameters)
    {
      URL url;
      HttpURLConnection connection = null;
      try {
        //Create connection
        url = new URL(targetURL);
        connection = (HttpURLConnection)url.openConnection();
        connection.setRequestMethod("POST");
        connection.setRequestProperty("Content-Type",
             "application/x-www-form-urlencoded");

        connection.setRequestProperty("Content-Length", "" +
                 Integer.toString(urlParameters.getBytes().length));
        connection.setRequestProperty("Content-Language", "en-US");

        connection.setUseCaches (false);
        connection.setDoInput(true);
        connection.setDoOutput(true);
        //Send request
        DataOutputStream wr = new DataOutputStream (
                    connection.getOutputStream ());
        wr.writeBytes (urlParameters);
        wr.flush ();
        wr.close ();

        //Get Response
        InputStream is = connection.getInputStream();
        BufferedReader rd = new BufferedReader(new InputStreamReader(is));
        String line;
        StringBuffer response = new StringBuffer();
        while((line = rd.readLine()) != null) {
          response.append(line);
          response.append('\r');
        }
        rd.close();
        return response.toString();
      } catch (Exception e) {

          e.printStackTrace();
          return null;

        } finally {

          if(connection != null) {
```

```
        } finally {

          if(connection != null) {
```

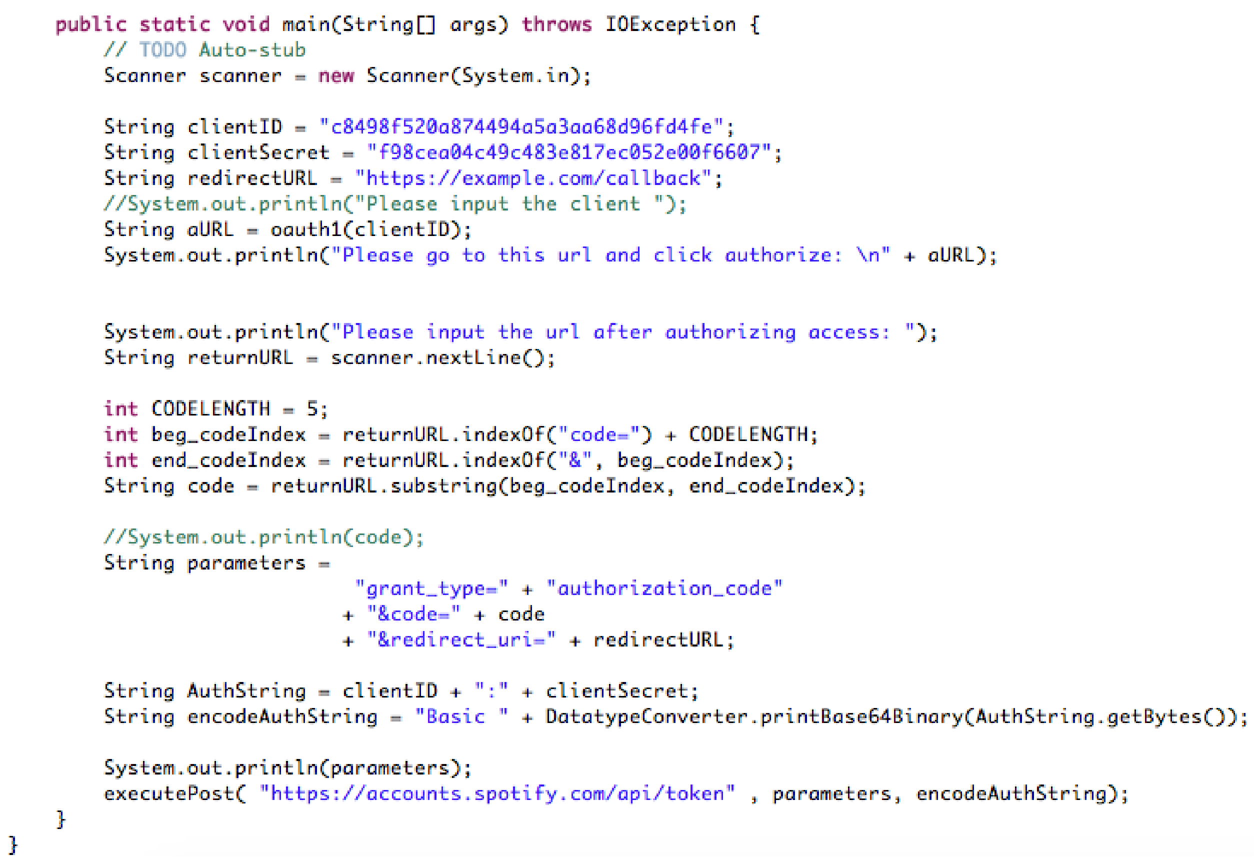
```
            connection.disconnect();
          }
        }
      }

    public static void main(String[] args) throws Exception{

        //ECHONEST//

        String apiKey = "PGHZKPTMD5HF9JONL" ;
        String consumerKey = "083220808db636c0eb17d5153ea89af6 " ;
        String consumerSecret = "SKTPVd5KQVmK6oqpQJwWBA" ;

        String artistName = "katyperry";

        //Searching for artists
        String sURL1 = "http://developer.echonest.com/api/v4/artist/search?" +
        "api_key=" + apiKey +
        "&format=json" +
        "&name=" + artistName;

        // Connect to the URL
        URL url1 = new URL(sURL1);
        HttpURLConnection request1 = (HttpURLConnection) url1.openConnection();
        request1.connect();

        // Convert to a JSON object to print data
        JsonParser jp1 = new JsonParser();
        JsonElement root1 = jp1.parse(new InputStreamReader((InputStream) request1.getContent()));
        JsonObject rootobj1 = root1.getAsJsonObject(); // may be Json Array if it's an array, or other type of a primitive
        String artistID = rootobj1.get("response").getAsJsonObject().get("artists").getAsJsonArray().get(0).getAsJsonObject().get("id").getAsString();
        System.out.println(artistID);

        String maxTempo = "130";
        String minTempo = "128";
        String results = "15";
        String style = "pop";
        String maxHot = ".1";
        //Getting songs
```

Fig. 6 – EchoNext Code pt. 1 (see Figure XX for pt. 2)



```java

        String sURL2 = "http://developer.echonest.com/api/v4/song/search?" +
        "api_key=" + apiKey +
        "&format=json" +
        "&max_tempo=" + maxTempo +
        "&min_tempo=" + minTempo +
        "&artist_id=" + artistID +
        //"&style=" + style +
        "&results=" + results +
        "&song_max_hotttnesss=" + maxHot+
        "&bucket=id:spotify&bucket=tracks";

        //Connect to the URL
        URL url2= new URL(sURL2);
        HttpURLConnection request2 = (HttpURLConnection) url2.openConnection();
        request2.connect();

        //Convert to a JSON object to print data
        JsonParser jp2 = new JsonParser();
        JsonElement root2 = jp2.parse(new InputStreamReader( (InputStream) request2.getContent()));
        JsonObject rootobj2 = root2.getAsJsonObject(); //may be Json Array if it's an array, or other type of a primitive
        System.out.println(rootobj2);
        JsonArray songArray = rootobj2.get("response").getAsJsonObject().get("songs").getAsJsonArray();
        int limit = songArray.size();
        List<String> songID = new ArrayList<String>();
        for (int i = 0; i < limit; i++){
            songID.add(songArray.get(0).getAsJsonObject().get("tracks").getAsJsonArray().get(0).getAsJsonObject().get("foreign_id").getAsString());
        }
        System.out.println(songID);
        System.out.println(songID.get(0));
        String songidentification = songArray.get(0).getAsJsonObject().get("tracks").getAsJsonArray().get(0).getAsJsonObject().get("foreign_id").getAsString();
        System.out.println(songidentification);
        ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

        //Spotify

        String targetURL = "";
        String urlParameters = "";
        executePost(targetURL, urlParameters);
        JsonParser jp3 = new JsonParser();
        JsonElement root3 = jp3.parse(new InputStreamReader( (InputStream) request3.getContent()));
        String playlistID = root3.getAsJsonObject().get("id").getAsString();
        String playlistURL = root3.getAsJsonObject().get("external_urls").getAsJsonObject().get("spotify").getAsString();



        }
```

Fig. 7 – EchoNext Code pt. 2 (see Figure XX for pt. 1)



```
    public static void main(String[] args) throws IOException {
        // TODO Auto-stub
        Scanner scanner = new Scanner(System.in);

        String clientID = "c8498f520a874494a5a3aa68d96fd4fe";
        String clientSecret = "f98cea04c49c483e817ec052e00f6607";
        String redirectURL = "https://example.com/callback";
        //System.out.println("Please input the client ");
        String aURL = oauth1(clientID);
        System.out.println("Please go to this url and click authorize: \n" + aURL);


        System.out.println("Please input the url after authorizing access: ");
        String returnURL = scanner.nextLine();

        int CODELENGTH = 5;
        int beg_codeIndex = returnURL.indexOf("code=") + CODELENGTH;
        int end_codeIndex = returnURL.indexOf("&", beg_codeIndex);
        String code = returnURL.substring(beg_codeIndex, end_codeIndex);

        //System.out.println(code);
        String parameters =
                                 "grant_type=" + "authorization_code"
                                + "&code=" + code
                                + "&redirect_uri=" + redirectURL;

        String AuthString = clientID + ":" + clientSecret;
        String encodeAuthString = "Basic " + DatatypeConverter.printBase64Binary(AuthString.getBytes());

        System.out.println(parameters);
        executePost( "https://accounts.spotify.com/api/token" , parameters, encodeAuthString);
    }
}
```

Fig. 8 – OAuth Code