Moody YouTube

CODE DOCUMENTATION
MULTIMEDIA AND NETWORKING COURSE

Ameera Akhtar (ameera.akhtar@student.unitn.it)
Zawar Hussain (zawar.hussain@studenti.unitn.it)

Table of Contents

1. Configuration file:

To work with YouTube server it needs to register application with Google so that it can submit API requests. To register application it must have authorization credentials to be able to use the YouTube Data API. To fulfil the entire above mention requirement I register my application on Google Developer console to get API key.

API Key: `AIzaSyCTwCtQkRDjsD7A1kN7geWyRFy2y21QGEQ`

To run client on Android smartphone, we need some android configuration which is given blow. There is an AndriodManifest.xml file in the project.

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
```

Code Segment:

```
    package="com.zawar.moodyyoutube"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
    android:minSdkVersion="14"
    android:targetSdkVersion="21" />

    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name="android.permission.GET_ACCOUNTS"/>
    <uses-permission android:name="android.permission.USE_CREDENTIALS"/>

    <application
    android:allowBackup="true"
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/AppTheme" >

    <meta-data android:name="com.google.android.gms.version"
    android:value="@integer/google_play_services_version" />

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

   <category android:name="android.intent.category.LAUNCHER" />
  </intent-filter>
  </activity>
  <activity android:name=".PlayerActivity" />
 <activity android:name=".PlaylistActivity" />
      </application>
 </manifest>
```

1. Main Components:

There are two major components in our project. They are explained below in detail,

1. Server
2. Client

1. Server:

In our project, the server is YouTube. Our client application uses the YouTube client Library to communicate with the server. YouTube Client Library is where one can add a variety of server features to the mobile application. Make use of the API to upload videos, manage playlists and much more. Code of YouTube client library is used by client in our project.

To work with YouTube server it needs to register application with Google so that it can submit API requests. To register application must have authorization credentials to be able to use the YouTube Data API. I am also using Google service library to authenticate my account and to use YouTube

2. Client:

In the client section, first of all my application asks for mood from user. User selects the moods from the dropdown list in the first activity (SearchActivity.java). On button click, the application finds configured Google account on the device and asks from user to select.

Code Segment:

```
private void pickUserAccount() {

String[] accountTypes = new String[] { "com.google" };

Intent intent = AccountManager.newChooseAccountIntent(null, null,accountTypes,
false, null, null, null, null);

startActivityForResult(intent, ACCOUNT_CODE);   }
```

User selects one account if there are many on the device, then application communicate with the Google server via Google Play Services Library and ask permission to access the YouTube from user.

private static final String ***SCOPE*** = "oauth2:https://www.googleapis.com/auth/userinfo.profile https://www.googleapis.com/auth/youtube";

The above string asks for the Youtube permission. If user grants the permission, the application moves to next step, else it closes. After granting the permission, Google server generates the token and returns to application to use for the latter API calls.

Code Segment:

```
private void requestToken() {
```

```
             Account userAccount = null;
             String user = authPreferences.getUser();
             for (Account account :
accountManager.getAccountsByType("com.google")) {
                    if (account.name.equals(user)) {
                           userAccount = account;

                           break;
                    }
             }

             accountManager.getAuthToken(userAccount, SCOPE, null, this,
                           new OnTokenAcquired(), null);
      }
```

Application then goes to next step and send search request with the token and key (created from Google Developer Console) to the YouTube server to search videos according to the user's mood. The following code is used to connect with YouTube server. It basically creates the API request to search videos from YouTube.

Code Segment:

```
    public YoutubeConnector(Context context) {

    youtube = new YouTube.Builder(new NetHttpTransport(),
    new JacksonFactory(), new HttpRequestInitializer() {
                                @Override
                                public void initialize(HttpRequest hr) throws
IOException {
                                }

      }).setApplicationName(context.getString(R.string.app_name))
                           .build();

             try {
                    query = youtube.search().list("id,snippet");
                    query.setKey(KEY);
                    query.setMaxResults((long) 50);

      query.setFields("items(id/videoId,snippet/title,snippet/description,snippet
/thumbnails/default/url)");
             } catch (IOException e) {
                    Log.d("YC", "Could not initialize: " + e);
             }
      }
```

Here “keyword” is the selected mood; I append “music” keyword to search only the music from YouTube. Other parameters assure that the search result will give only the music or those videos that are under the music video category in YouTube.

Code Segment:

```
query.setQ(keywords + " music");
query.setType("video");
query.setVideoCategoryId("music");
```

After executing the above request, application got the response from the YouTube as a list of videos search according to the user selected mood.

Code Segment:

```
SearchListResponse response = query.execute();
List<SearchResult> results = response.getItems();

List<VideoItem> items = new ArrayList<VideoItem>();
for (SearchResult result : results) {
VideoItem item = new VideoItem();
item.setTitle(result.getSnippet().getTitle());
item.setDescription(result.getSnippet().getDescription());
item.setThumbnailURL(result.getSnippet().getThumbnails()
                                       .getDefault().getUrl());
if (result.getId() != null) {
item.setId(result.getId().getVideoId());
item.setChannelID(result.getId().getChannelId());
}
items.add(item);
}
return items;
```

SearchListResponse and SearchResult are the objects of YouTube Client Library. These are then converted into VideoItem object (mentioned in the above code) to work with according to my application requirement.

Now, the next step in to show the list on the mobile screen (PlaylistActivity.java). After mapping the Youtube response into VideoItem objects, I display them according to my application design. I display these videos as a list on my mobile screen.

Code Segment:

```
ArrayAdapter<VideoItem> adapter = new ArrayAdapter<VideoItem>(
getApplicationContext(), R.layout.video_item, searchResults) {
                   @Override
public View getView(int position, View convertView, ViewGroup parent) {
if (convertView == null) {
convertView = getLayoutInflater().inflate(
R.layout.video_item, parent, false);
                          }
ImageView thumbnail = (ImageView) convertView.findViewById(R.id.video_thumbnail);
TextView title = (TextView) convertView.findViewById(R.id.video_title);

VideoItem searchResult = searchResults.get(position);
```

```
        Picasso.with(getApplicationContext())

        .load(searchResult.getThumbnailURL()).into(thumbnail);
        title.setText(searchResult.getTitle());
        return convertView;
                     }
              };

        videosFound.setAdapter(adapter);
```

The main objective of this application was to search videos from YouTube according to the user's mood and create a playlist using those videos. Here, application sends a request using the YouTube Client Library to create a playlist on the YouTube channel of the user's YouTube account. For this, application again uses the token generated in the very first step and the API key. The following code creates a public playlist with the desired mood as a title on the YouTube.

Code Segment:

```
playlistSnippet.setDescription("A    public    playlist    created    by    MoodyYoutube
application.");
PlaylistStatus playlistStatus = new PlaylistStatus();
playlistStatus.setPrivacyStatus("public");

Playlist youTubePlaylist = new Playlist();
youTubePlaylist.setSnippet(playlistSnippet);
youTubePlaylist.setStatus(playlistStatus);

YouTube.Playlists.Insert playlistInsertCommand = youtube
.playlists().insert("snippet,status", youTubePlaylist);
playlistInserted = playlistInsertCommand.execute();
```

When playlist creation is done, the application starts adding the videos searched before one by one.

Code Segment:

```
YouTube.PlaylistItems.Insert playlistItemsInsertCommand = null;

for (int i = 0; i < playListItems.size(); i++) {


ResourceId resourceId = new ResourceId();
resourceId.setKind("youtube#video");
resourceId.setVideoId(playListItems.get(i).getId());


PlaylistItemSnippet playlistItemSnippet = new PlaylistItemSnippet();
playlistItemSnippet.setTitle(playListItems.get(i)
                                              .getTitle());
playlistItemSnippet.setPlaylistId(playlistId);
playlistItemSnippet.setResourceId(resourceId);


PlaylistItem playlistItem = new PlaylistItem();
playlistItem.setSnippet(playlistItemSnippet);

playlistItemsInsertCommand = youtube.playlistItems()
.insert("snippet,contentDetails", playlistItem);

PlaylistItem returnedPlaylistItem = playlistItemsInsertCommand
.execute();
if (i == 49) {
playlistcreated = true;
detail.setText("Your playlist has been created. Please click on any song to check
it in youtube application.");
                                  }
                           }
```

The above code runs till the end of search video item list and add these one by one in the playlist. Here, a complete object is created as a requirement of YouTube Client Library by using the title of the videos searched previously and the id of playlist created in the last step.

Now, the final thing is to play this playlist in the YouTube application. Following code open the YouTube application on the Android phone and navigates to the user's created public playlist. Now you can play the videos as a playlist in the YouTube application

Code Segment:

```
Intent intent = new Intent(Intent.ACTION_VIEW);
intent.setData(Uri.parse("http://www.youtube.com/playlist?list="+pc.getChannelID()
));
startActivity(intent);
```

3. Output

Output of our project is to create a public playlist on server (YouTube). Playlist is played by one click on mobile application.