



Music Mania

by

Gaurav Rane
CS5200
Prof Jose Annunziato

TABLE OF CONTENTS

1. INTRODUCTION
2. MOTIVATION BEHIND THE APP
3. IMPORTANT FEATURES
4. TECHNOLOGY USED
5. DATABASE DIAGRAM
6. APPLICATION QUERIES
7. SNAPSHOTS
8. CHALLENGES
9. FUTURE SCOPE
10. PROJECT REFERENCES
11. SOURCE CODE LINK ON BITBUCKET

INTRODUCTION

Music Mania is an android application used to search, save and recommend music. This is the simplest one line summary of the application. Users of the application are able to search for music via three categories – Song Name, Artist Name and Album Name.

Once the music for their choice is found, users can view the YouTube video of the music item, see it's detailed description, add that music item to their recommend list which will then help get music recommendations based on items in the list. Users are able to get recommendations based on all the three categories - Song Name, Artist Name and Album Name. This will help them expand their music search and add more songs to their favorite list.

Users can also add songs to their favorite playlists (which can be viewed later on). Users can also visit the profiles of other users who have listened to the song and thus view their playlists.

2. MOTIVATION BEHIND THE APP

- Need for a complete music app
 - There is not a single complete music app on the Play Store. Users should not have to navigate to different apps for doing different functionalities.
- Every other app on the Google Play Store allows only a selected feature
 - I have tried to think of a mash up of all the major features while making this app.
- Music recommendation is a cool feature
 - Recommending music based on users choice of music is an interesting and trending feature.
- Search via artist, album and song
 - Searching via all three music categories is rare to be found in applications.

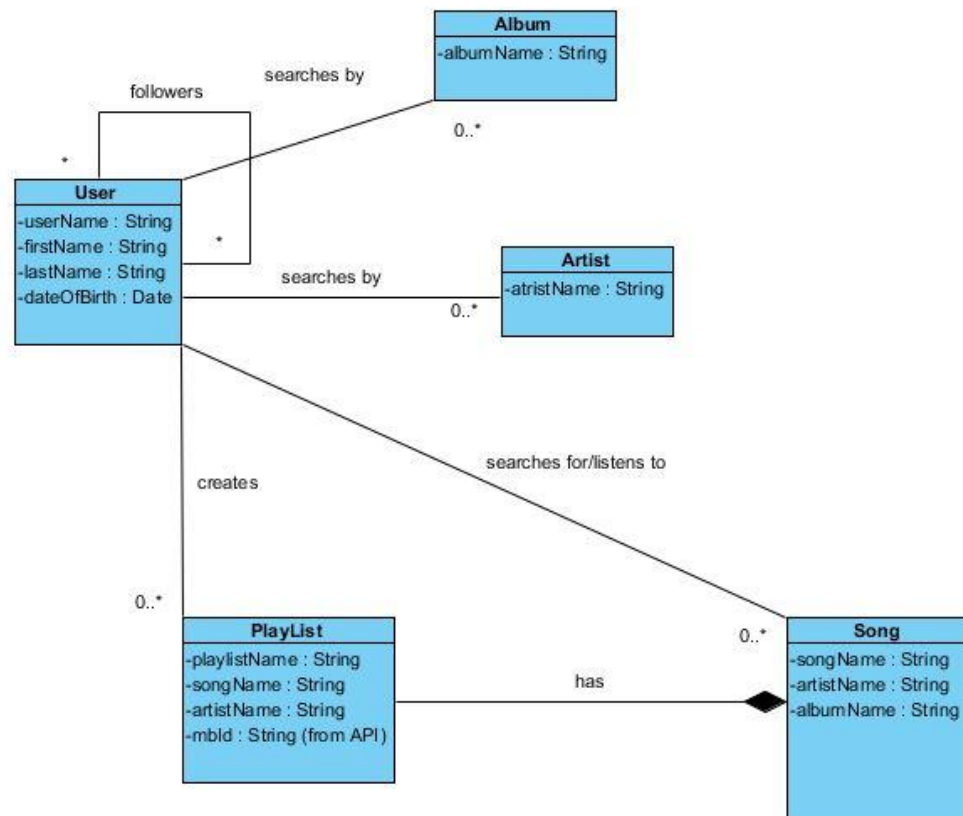
IMPORTANT FEATURES

- Search music (artist, album, song)
 - Searching by all the three categories
- Create new playlists
 - Users can create new custom playlists
- Add songs to playlists
 - Users can add songs to the playlist
- Get music recommendation (artist, album, song)
 - Users can get music recommendations based on their choice of music category.
- User login
 - Users need to have an login username and password to use the above features
- User profile (statistics)
 - Users can see their profile which shows statistics and other cool summary of their music history
- Follow/Unfollow users
 - Users can follow other users to get accesses to their playlist.
- Visit follower's playlists
 - Users can get accesses to their folllower's playlist.

TECHNOLOGY USED

- Android SDK(Java)
 - I have used native Android SDK tool to write the code for this app. I have used Eclipse integrated with the Google Android SDK inbuilt.
- SQLite Database
 - I have used the SQLite Database which is internal to the Android System for storing the data in the database tables.
- Last.Fm API
 - I have used the Last.Fm API to get the information about the Music searches and also for music recommendation. It has tons of methods to access various dimensions of music.
- YouTube Data API (v3)
 - I have used the YouTube Data API(v3) to get the YouTube music video based on the searched music query.
- YouTube Android Player API
 - I needed to use the YouTube Android Player API to integrate the YouTube video in my android application.

DATABASE DIAGRAM



The following is my database schema.

The users can create multiple playlist and store songs of their choice. Also they store the searched artist name, song name and album name to get the desired music recommendations. Plus the users can follow other users (so that is represented as a self-referential relationship)

APPLICATION QUERIES

Following are the application queries:

- User following each other

1. add to follower list

E.g. Nick follows Larry

```
-- insert into follower_table Values('Nick','Larry')
```

2. check if following

E.g. Check if Nick follows Larry

```
-- select count(*) from follower_table where follower = 'Nick'  
AND following = 'Larry'
```

```
-- Check count greater than 1
```

3. get list of Followers (input following name)

E.g. list of users following 'Larry'

```
-- select * from follower_table where following = 'Larry'
```

4. remove following relationship

E.g. remove 'Nick' from following 'Larry'

```
-- delete from follower_table where following = 'Larry' AND  
follower = 'Nick'
```

- User related queries

1. Add new user to list of users

E.g. Add 'Lisa' to list of users.

```
-- insert into User_table values ('Lisa', 'Lisa123')
```

2. Check if user table is empty

```
-- select count(*) from User_table
```

```
-- return true if count > 1 or else return false.
```

3. Check user credentials.

E.g. Check if 'Mike' is in the list of users table.

```
-- select pwd from User_table where user_name = 'Mike'
```

```
-- compare pwd with the user entered pwd.
```

4. Get other list of users other than other user.

E.g. Get list of users than 'Ryan'

```
-- select * from User_table where user_name != 'Ryan'
```

- User playlist table

1. Add new user playlist

E.g. Add 'Happy Songs' for 'Mike'

```
-- insert into User_Playlist values('Mike', 'Happy Songs')
```

2. Get all the playlist names for the user.

E.g. get all playlists for 'Mike'

```
-- select * from User_Playlist where user_name = 'Mike'
```

3. get user playlists count

E.g. get 'Mike' playlists count

```
-- select count(*) from User_Playlist where user_name = 'Mike'
```

4. delete user playlist

E.g. delete 'Rock songs' playlist for 'Mike'

```
-- delete * from User_Playlist where user_name = 'Mike' AND  
playlist_name = 'Rock Songs'
```

```
-- delete * from Playlist where user_name = 'Mike' AND  
playlist_name = 'Rock Songs'
```

- Playlist Table

1. Add a song to the user playlist

E.g. Add 'Happy' by 'Pharrel Williams' into 'Happy Songs' playlist for 'Mike'

```
-- insert into Playlist values('Happy Songs','Mike','Happy','Pharrel  
Williams','AS343FEEFEFET343')
```

2. Get all the songs from the playlist name for the user

E.g. get all songs for 'Happy songs' playlist for the user 'Mike'

```
-- select * from Playlist where user_name = 'Mike' AND  
playlist_name = 'Happy Songs'
```

3. Get all the mbids of songs from the playlist name for the user

E.g. get all mbids of songs for 'Happy songs' playlist for the user 'Mike'

*-- select mbid from Playlist where user_name = 'Mike' AND
playlist_name = 'Happy Songs'*

4. Get the count of songs from the playlist name for the user

E.g. get count of songs for 'Happy songs' playlist for the user
'Mike'

-- select count() from Playlist where user_name = 'Mike' AND
playlist_name = 'Happy Songs'*

5. delete song from user playlist

E.g. delete 'Stairway To Heaven' from 'Rock songs' playlist for
'Mike'

*-- delete * from Playlist where user_name = 'Mike' AND
playlist_name = 'Rock Songs' AND song_name = 'Stairway To Heaven'*

- Song Recommendation Table

1. Add song to user recommendation list

E.g. Add 'Rise Up' by 'Blue' for 'Mike'

-- insert into Recommend_Song values('Mike', 'Rise Up', 'Blue')

2. Get all the recommended song names for the user.

E.g. get all recommended songs names for 'Mike'

*-- select * from Recommend_Song where user_name = 'Mike'*

3. Get all the mbids of the recommended songs for the user

E.g. get all mbids of recommended songs for the user 'Mike'

-- select mbid from Recommend_Song where user_name = 'Mike'

4. Get the count of recommended songs for the user

E.g. get count of recommended songs for the user 'Mike'

```
-- select count(*) from Recommend_Song where user_name =  
'Mike'
```

5. delete song from recommended song list

E.g. delete 'Stairway To Heaven' from recommended songs list for 'Mike'

```
-- delete * from Recommend_Song where user_name = 'Mike'  
AND song_name = 'Stairway To Heaven'
```

- Album Table

1. Add album to user recommendation list

E.g. Add 'Mylo Xyloto' by 'Coldplay' for 'Mike'

```
-- insert into Recommend_Album values('Mike','Mylo Xyloto',  
'Coldplay')
```

2. Get all the recommended album names for the user.

E.g. get all recommended album names for 'Mike'

```
-- select * from Recommend_Album where user_name = 'Mike'
```

3. Get all the mbids of the recommended album for the user

E.g. get all mbids of recommended album for the user 'Mike'

```
-- select mbid from Recommend_Album where user_name = 'Mike'
```

4. Get the count of recommended albums for the user

E.g. get count of recommended albums for the user 'Mike'

```
-- select count(*) from Recommend_Album where user_name =  
'Mike'
```

5. delete album from recommended song list

E.g. delete 'Freedom' by 'Akon' from recommended songs list for 'Mike'

```
-- delete * from Recommend_Album where user_name = 'Mike'  
AND album_name = 'Freedom' AND artist_name = 'Akon'
```

-Artist Table

1. Add artist to user recommendation list

E.g. Add 'Blue' for 'Mike'

```
-- insert into Recommend_Artist values('Mike', 'Blue')
```

2. Get all the recommended artist names for the user.

E.g. get all recommended artist names for 'Mike'

```
-- select * from Recommend_Artist where user_name = 'Mike'
```

3. Get all the mbids of the recommended artist for the user

E.g. get all mbids of recommended artist for the user 'Mike'

```
-- select mbid from Recommend_Artist where user_name = 'Mike'
```

4. Get the count of recommended artists for the user

E.g. get count of recommended artists for the user 'Mike'

-- select count() from Recommend_Artist where user_name = 'Mike'*

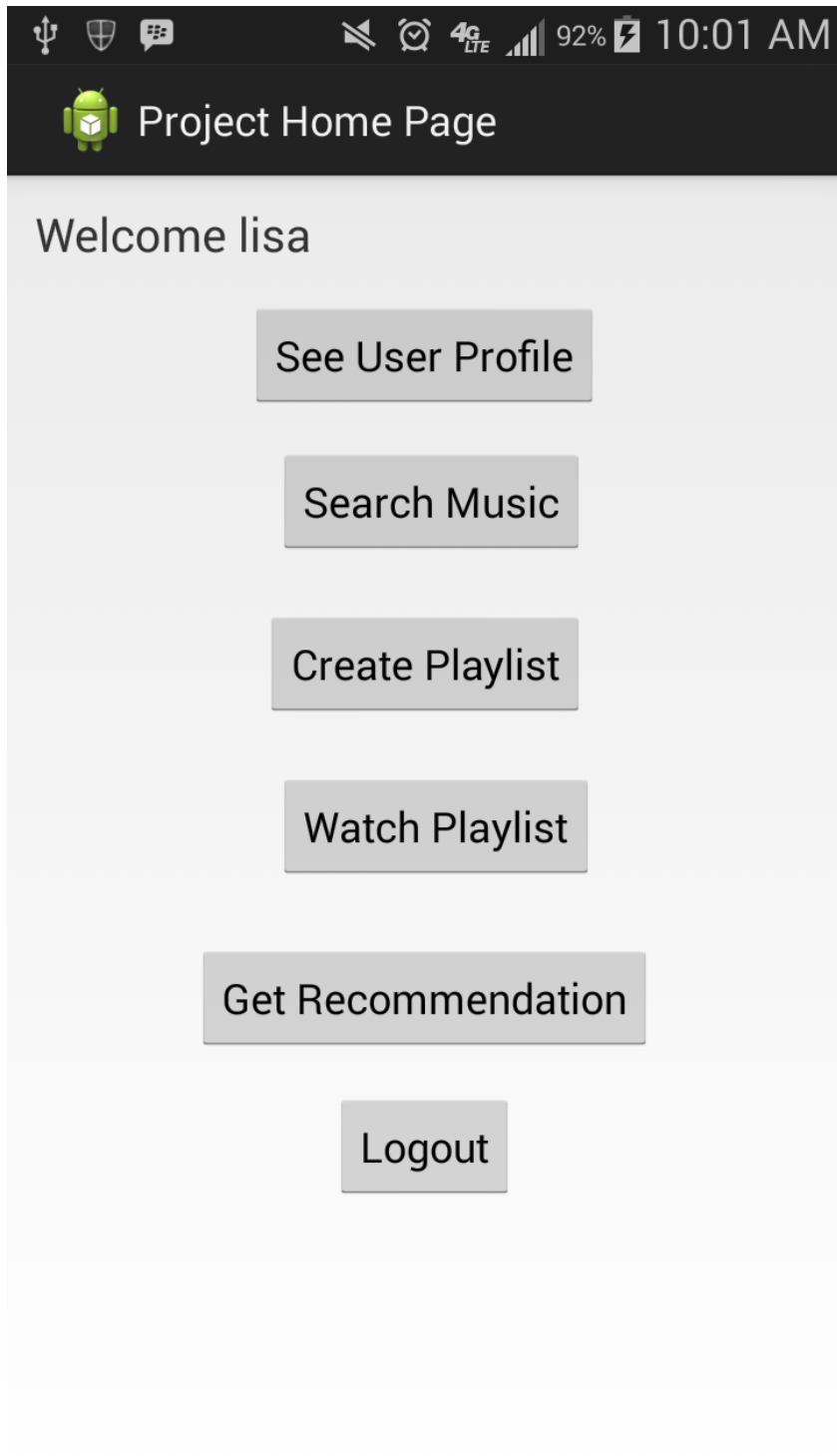
5. delete song from recommended artists list

E.g. delete 'Eagles' from recommended artists list for 'Mike'

*-- delete * from Recommend_Artist where user_name = 'Mike'*
AND artist_name = 'Eagles'

SNAPSHOTS

Project Home Page



User Profile Page

User Profile statistics

UserName - lisa

Number of playlists - 1

Number of playlist songs - 2

Number of recommended songs - 3

Number of recommended albums - 0

Number of recommended artists - 0

Follow User

Go Home

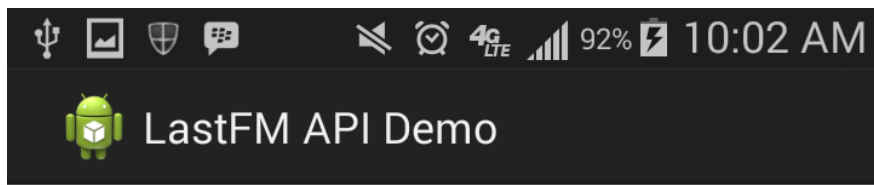
List of other users followed

ryan

Follower playlists

Unfollow User

Music Search Page



LastFM API Demo

- ☐ album
- ☐ artist
- ☒ song

Enter the song name to search

happy

Search Query

List of Music Searches Page

Android status bar: 10:02 AM, 92% battery, 4G LTE, and various system icons.

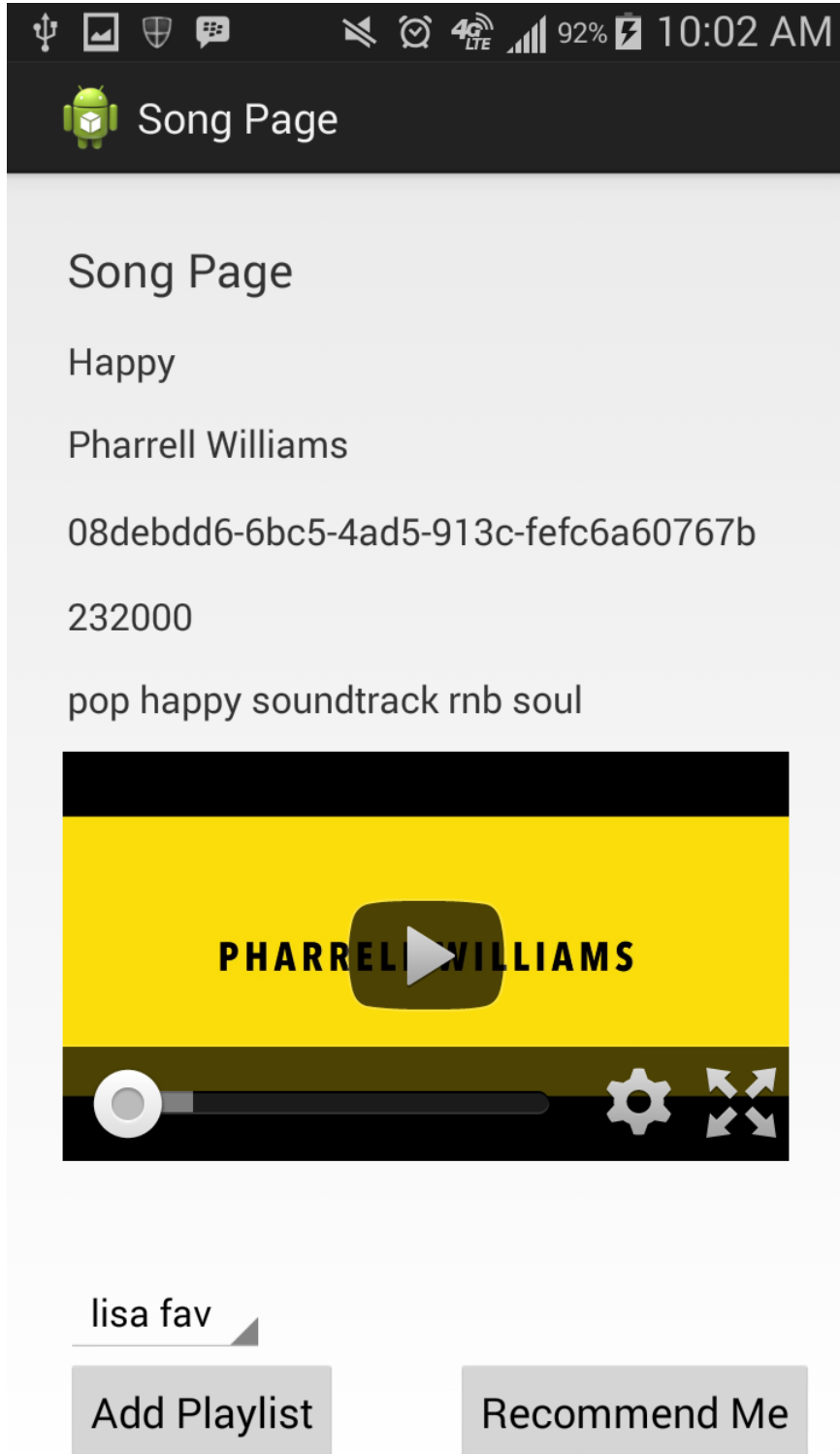
List of Song Searches

Page 1 of 3


- You Could Be Happy-Snow Patrol
- My Happy Ending-Avril Lavigne
- Happy Up Here-Röyksopp
- Shiny Happy People-R.E.M.
- Happy-Pharrell Williams
- So Happy I Could Die-Lady Gaga
- Happy Ending-Mika
- Merry Happy-Kate Nash
- Happy Together-The Turtles
- Only Happy When It Rains-Garbage

Prev Next

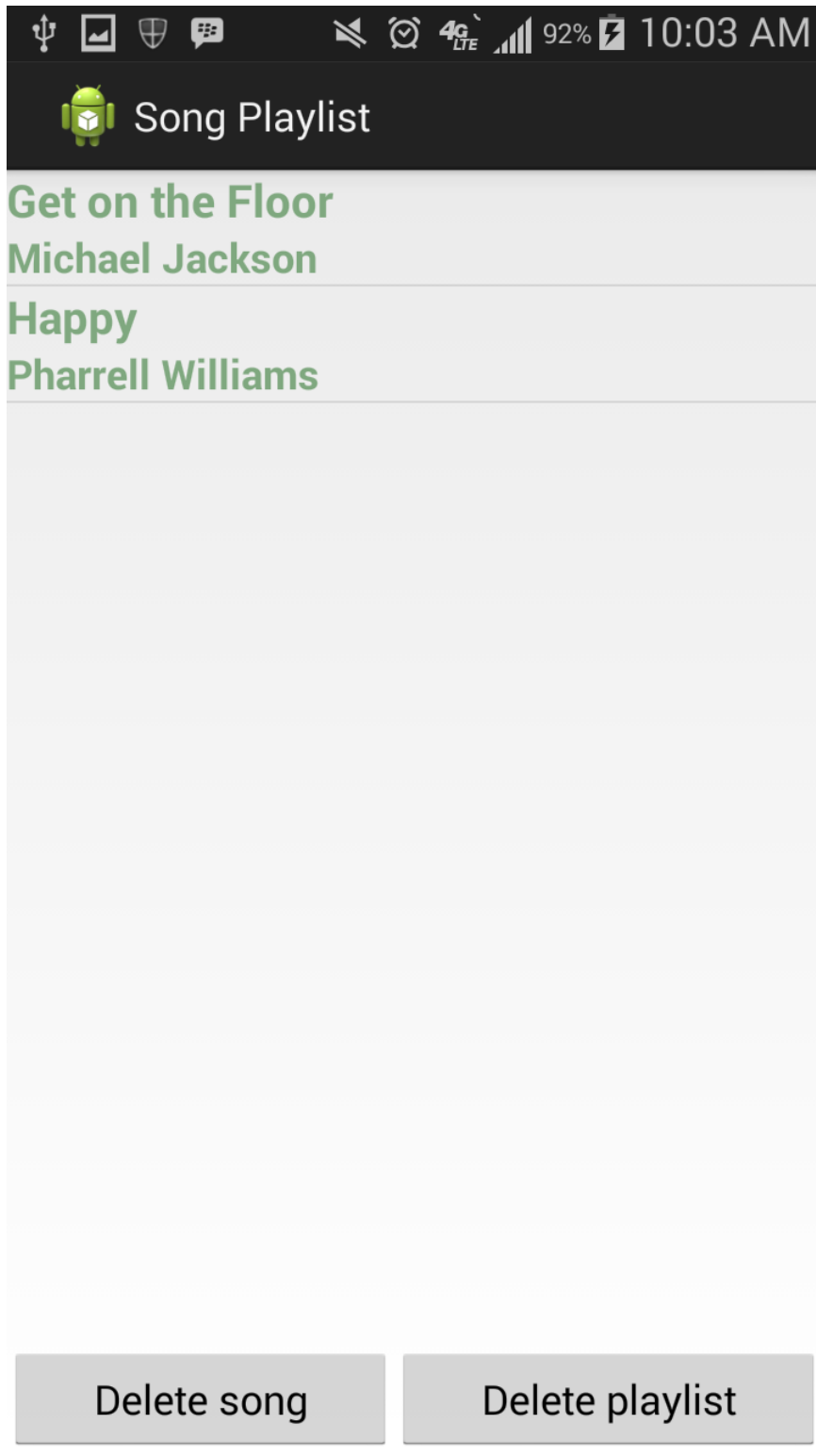
Music Home Page



Recommendation List Page

 Recommendations
Dance Again-Jennifer Lopez feat. Pitbull
Live It Up-Jennifer Lopez feat. Pitbull
'Naive'-The Kooks
Mardy Bum-Arctic Monkeys
When the Sun Goes Down-Arctic Monkeys
Lets Dance To Joy Division-The Wombats
Are You Gonna Be My Girl-Jet
All of Me-John Legend
Money on My Mind-Sam Smith
Not a Bad Thing-Justin Timberlake
Air Balloon-Lily Allen
Take Back the Night-Justin Timberlake

Playlist Page



CHALLENGES FACED

- Integrating YouTube video on the activity page
 - This was quite challenging as the documentation for the YouTube API was deprecated and not up-to-date. I had to do a lot of trial and error stuff to get it running.
- SQLite Database connection and queries
 - I didn't use raw SQL statements but instead I used the prepared statements and inbuilt methods to get the SQL statements. Figuring how that works took some time.
- Logic for recommending music
 - Logic for recommending music was not straight forward. I had to fetch all the music items in the recommendation list extract out music details and tags from the music items and then get the related music. I had to think through a lot to adapt this code for all the three music categories.
- Pagination for music lists

I wanted to use a generalized pagination layout for all the events and that was a little tricky to do. But I figured out after reading a lot of online tutorials. Doing the pagination on the android screens is a little handy as the screen sizes are never the same.

FUTURE SCOPE

- Improve YouTube video search and enable full screen mode
 - I wanted to improve the YouTube video searching feature as not every time do I get videos related to the content. This is because of the discrepancies between YouTube users.
- Sharing music across different social media platforms
 - I wanted to enable music sharing across popular social media platforms.
- Enable hands free search and voice commands
 - I also wanted to use hands free commands in the future if possible to enable better experience for the users.
- Improve UI design
 - UI design is an important factor in this app and it needs to be improved for user experience.

PROJECT REFERENCES

Here are the following project references

- <http://codezone4.wordpress.com/2012/11/19/android-login-system-using-sqlite/>
- <http://www.last.fm/api/account/75ef2e0ade33b40cfc0f765b27d17f5?pending=0&new=1>
- <http://www.techrepublic.com/blog/software-engineer/using-googles-youtube-api-in-your-android-apps/>
- http://www.tutorialspoint.com/android/android_login_screen.htm
- <http://www.androidhive.info/2012/01/android-json-parsing-tutorial/>
- <http://www.vogella.com/tutorials/AndroidSQLite/article.html>
- <http://lyle.smu.edu/~coyle/cse7345/handouts/s12.AndroidCookbook.Steele.ch9.SQLite.pdf>

SOURCE CODE LINK ON BITBUCKET

This is the link for the project source code on BitBucket.

<https://grane@bitbucket.org/grane/musicmania.git>