GitHub Username: mkash32

LyricFinder

Description

Having a trouble time understanding what that pop artist is saying in his new hit single? No worries, LyricFinder lets you find the lyrics of all your favorite songs with a simple search!

Intended User

LyricFinder is intended for all music listeners whether it's for searching song lyrics or discovering hot songs in the area.
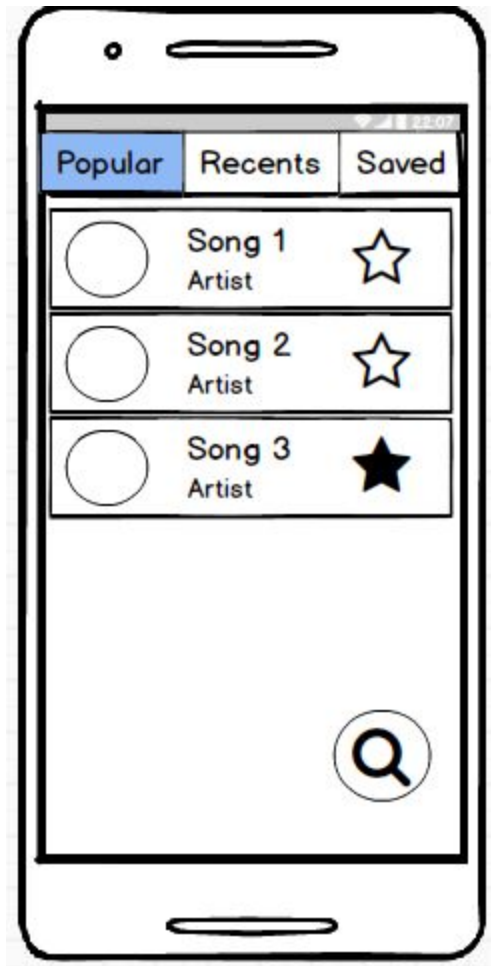
Features

List the main features of your app. For example:
- Search for lyrics
- Lists recent searches, popular tracks in your geographical area
- Save lyrics to starred songs

User Interface Mocks
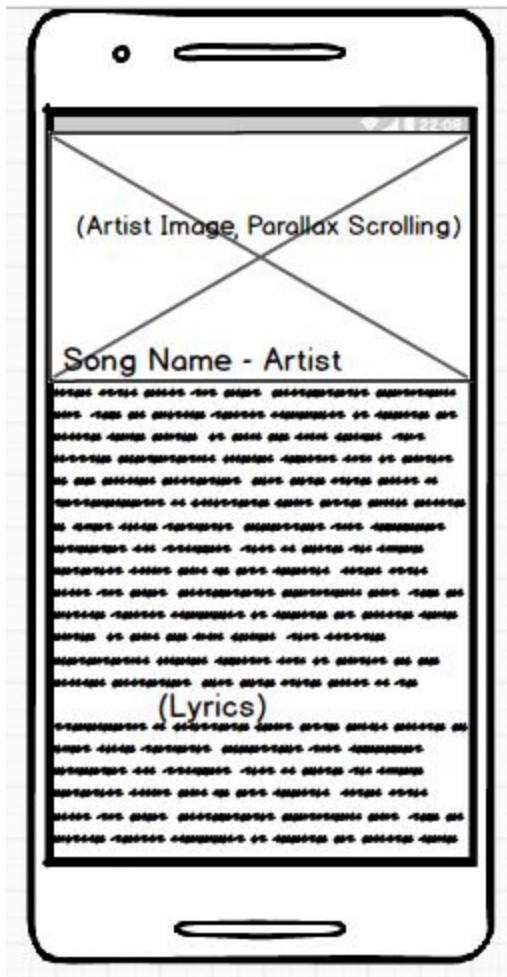
(Created Using Balsamiq Mockups 3)

Screen 1

Main Activity UI:
- Contains a tabbed layout with three tabs: Popular, Recents, and Saved
- Each tab will have the relevant Song items
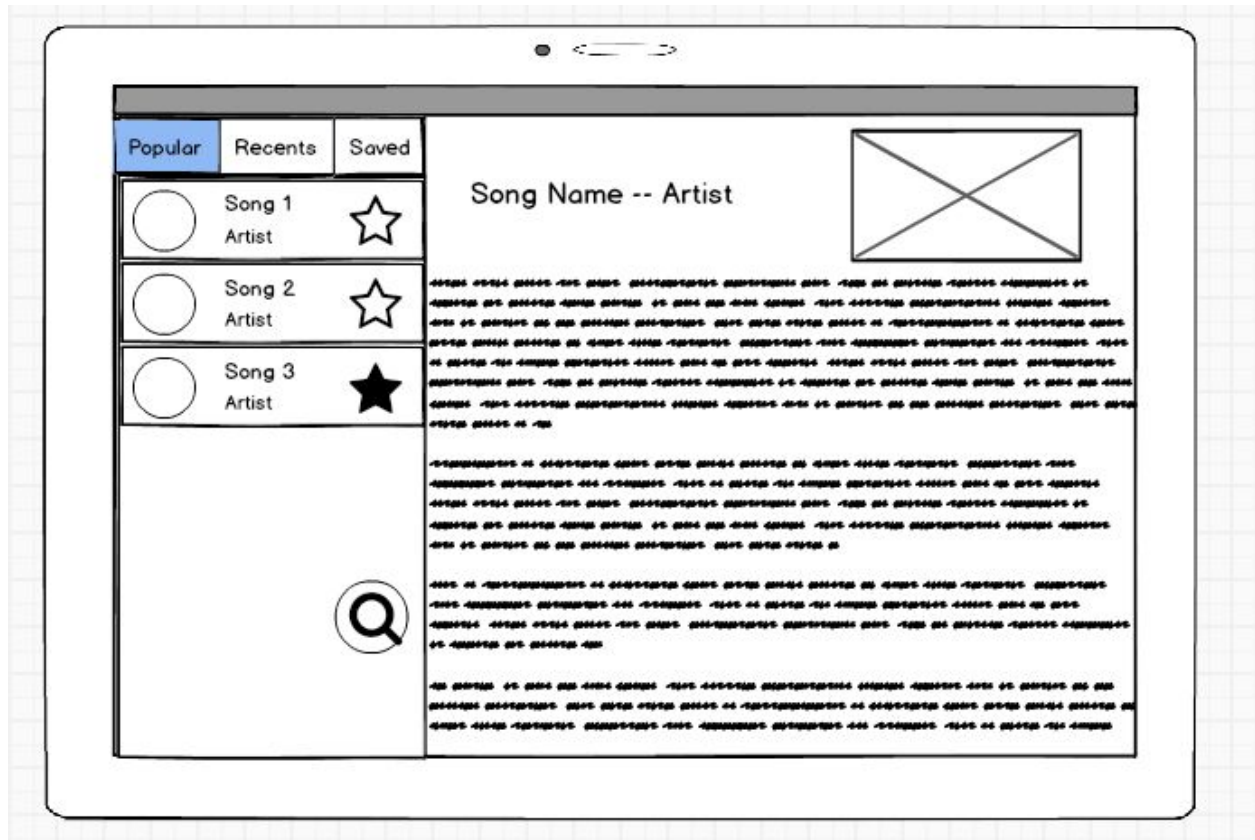- A Floating action button at the bottom which allows to search for music

Screen 2

Lyrics Activity:
- Action Bar has an image of the artist and Parallax Scrolling feature
- Lyrics Scrolling Display below

Screen 3- Tablet

The tablet layout is a combination of the previous two mobile layouts, with the tabbed display on the left, with the selected lyrics display on the right.

Key Considerations

How will your app handle data persistence?

Content Provider will be built.

Describe any corner cases in the UX.

Corner cases:
- Blank screen while loading lyrics, loading bar should be displayed
- Location may not be enabled. Check for location enabled, if not enabled then a dialogue box should be used to allow user to enable location

Describe any libraries you'll be using and share your reasoning for including them.
- Picasso to handle the loading and caching of images of artists or albums
- MusixMatch api is to be used to find the lyrics of the desired song, an open source java library was built for this api. https://github.com/sachin-handiekar/jMusixMatch . Using maven it can be packaged into a jar file and used as a module in android.
- OkHttp will be used to access the REST endpoints of the lastfm api to discover songs, and popular artists/songs in a location. It will also be used to access the REST endpoint of geonames api to attain the country from latitude and longitude.

Describe how you will implement Google Play Services.

2 Google Play Services will be used in this application
1. Location: Locations will be used to recommend tracks which are popular in the user's geographic region
2. AdMob: Display an ad banner within the app

Next Steps: Required Tasks

Task 1: Project Setup

- Configure libraries
    - Using maven jMusixMatch needs to be packaged into a jar file and imported into the project
    - OkHttp
- Create a Static details class which will have all the API endpoints needed for this project

If it helps, imagine you are describing these tasks to a friend who wants to follow along and build this app with you.

Task 2: Implement UI for Each Activity and Fragment

List the subtasks. For example:
- Build UI for MainActivity (Popular music feed, recent searches, saved)
    - List item needs to be created
- Build UI for LyricsActivity

Task 3: Google Play Services

Setup Google Play Services
- Location
    - Account for corner cases when Location is disabled
    - Integrate with Geonames api to retrieve the country name using the coordinates
- Admob

Task 4: API Endpoint Connectivity

Implement the functionality of the app with the use of the api endpoints and jMusixMatch java library. (Using IntentService)
- Create appropriate POJO classes for the music details
- Create utility functions for retrieving the data and parsing the json

Task 5: Persistence

Implement the favorites/saved and Recently Searched feature using a Content Provider.
- Create SQLite database to store the basic music details objects
- Hook up the SQLite database to a Content Provider
- Implement a loader to update the UI

Task 6: Create Signed APK

Signed .apk file is to be generated
- Create Keystore
- Generate Signed .APK