Links and deadlines to submit the projects:

App 1: https://review.udacity.com/#!/projects/55
App 1 Deadline: May 30, 2015

App 2 - Stage 1: https://review.udacity.com/#!/projects/56
App 2 - Stage 1 deadline: June 15, 2015

App 2 - Stage 2: https://review.udacity.com/#!/projects/57
App 2 - Stage 2 deadline: June 30, 2015

App 1 - Cat in the Hat:

Steps to complete this app:

1. Download Android studio
2. Create a new project in Android Studio using the 'application with blank activity' template.
3. Add a button for each app:

    - The Cat
    - Thing 1
    - Thing 2
    - Thingamajigger
    - Sally
    - Nick
    - Dr. Seuss

4. add an onClickListener for each button or write an onClick method
5. Launch a toast from the methods you created in the last step

Design the UI however you wish!

Colors and Styles

When designing your layout for the project, you may enjoy customizing the style and color of your buttons and changing the layout. If you are interested in learning about colors you can use, take a look at the material design color page, from Google.

If you'd like your app to look udacious, feel free to use #F08C35 to match the orange buttons on our site.

Overview

This rubric is here to help you understand the expectations for how your project will be evaluated. It is the same rubric that the person evaluating your project will use. You should look at the rubric before you begin working on this project and before you submit it.

How Grading Works

1. Your project evaluator will be able to see all of your code submissions. They will use this rubric to evaluate your code as well as your written responses.
2. Your grade will simply be "meets specifications" or "does not meet specifications,"

1. You earn “meets specifications” if all criteria “meet specifications”.
2. Your project “does not meet specifications” if any criteria are graded as “does not meet specifications.” In this case, you will have the opportunity to revise and resubmit your work as many times as you like.

Criteria	Does not Meet Specifications	Meets Specifications
Does the interface have buttons representing the characters?		
Does each button launch a toast?		

App 2 - Stage 1:

Task 3: Optimize the entire end to end experience for a tablet

Spotify Streamer, Stage 1: Implementation Guide

Stage 1: Overview

This document should be used as a guide and development plan for Stage 1 of the Spotify Streamer app. It breaks down the build process into smaller, concrete milestones, and provides technical guidance on specific tasks.

Evaluation

Stage 1 is evaluated against the following rubric.

User Experience Design

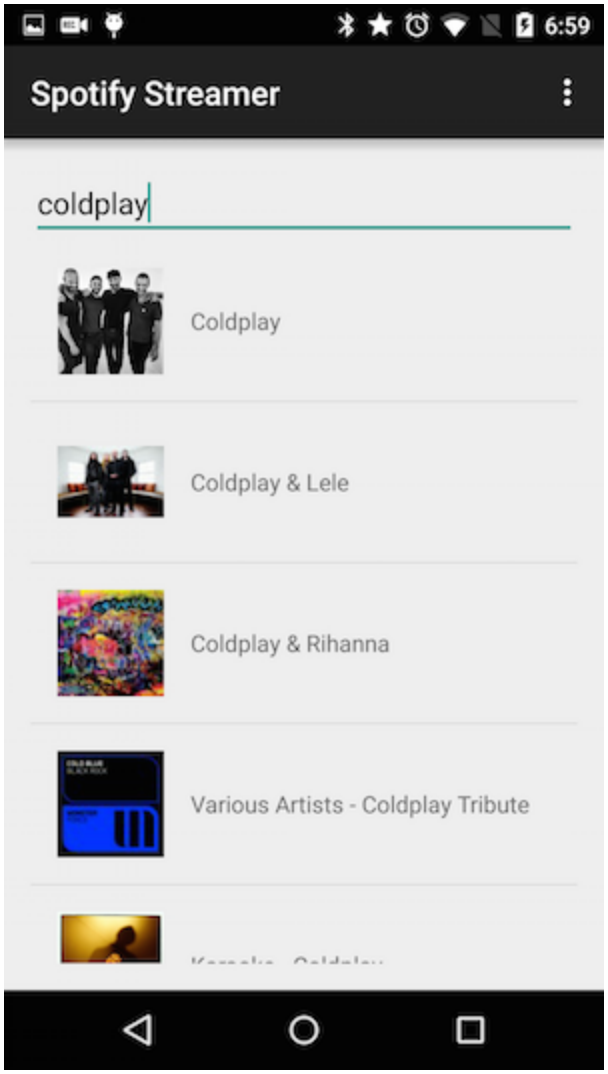
This guide will provide UI mock-ups (or mocks) to establish a baseline for Spotify Streamer's core user experience. Your job is to implement these same experiences, although you're free to customize the visual design to make it your own.

Note: In a real world situation, such mocks are generally provided by a designer or design team. But, in some cases, an individual developer creates their own mocks.

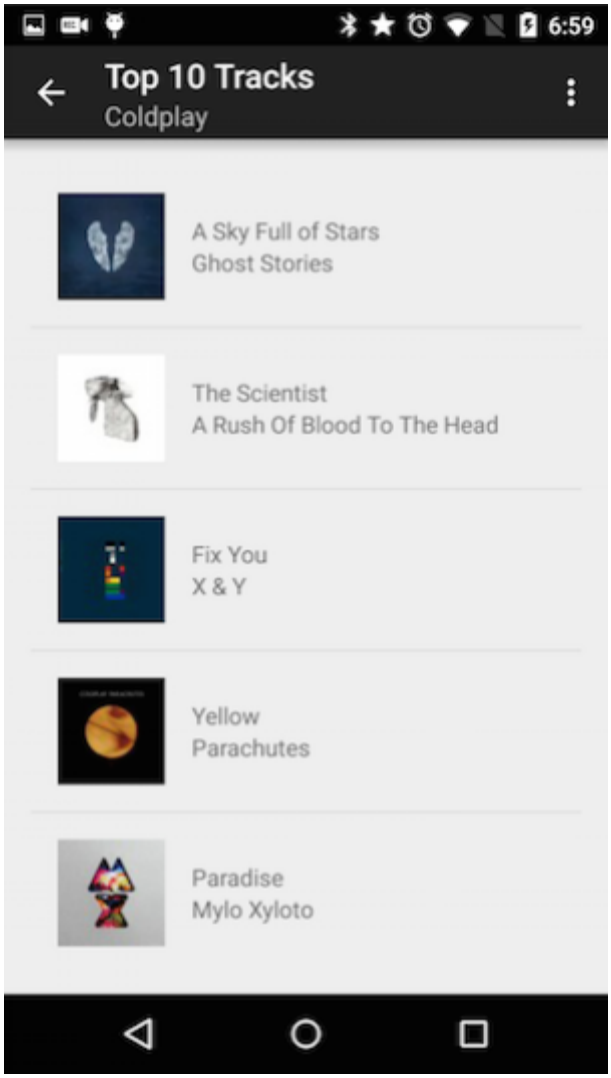
The main purpose of these mocks is to communicate and establish the visual and interactive experience of the app.

Phone Interaction Flow

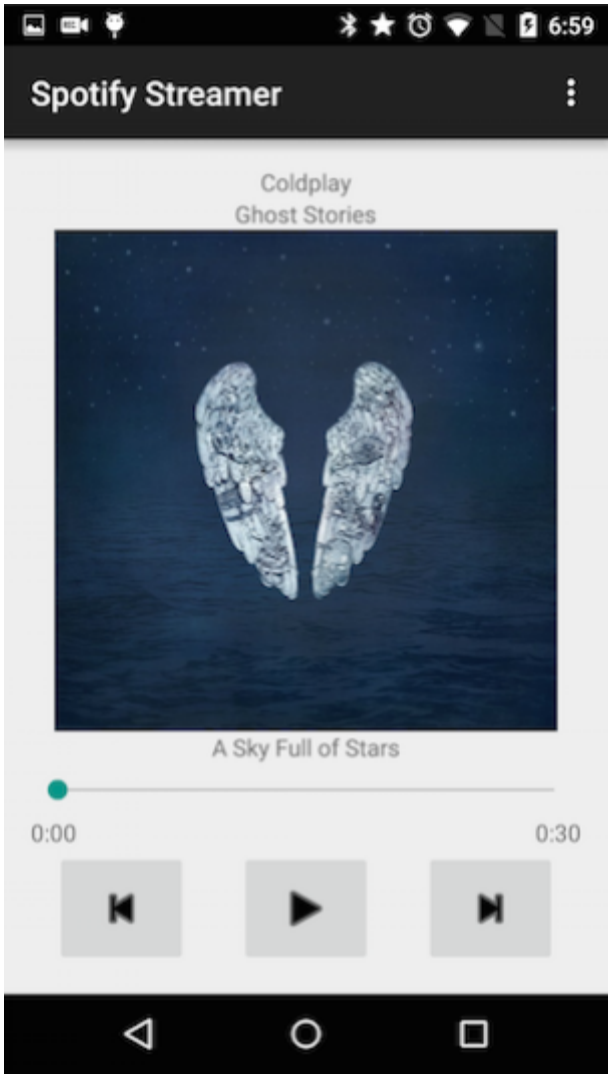
1. Search for an artist and display results



2. Then display the top tracks for a selected artist.



3. When a user selects a track, the track should start playing in a player UI


6:59
Spotify Streamer
Coldplay
Ghost Stories
A Sky Full of Stars
0:00
0:30

Project Setup: Configure Your Project to Use Two Important Libraries

---

This project will utilize two key Android libraries, which we've included to reduce unnecessary extra work and help you focus on applying your app development skills.

1. Picasso - A powerful library that will handle image loading and caching on your behalf
2. Spotify-Web-Api-Wrapper - A clever Java wrapper for the Spotify Web API that provides java methods that map directly to Web API endpoints. In other words, this wrapper handles making the HTTP request to a desired API endpoint, and deserializes the JSON response so you don't need to write any parsing code by hand. When you need to make a particular Web API call, you simply call the corresponding method in the library and act on the return data, already converted to Java objects that you can interact with programmatically.

Library Configuration Instructions

Picasso

In your app/build.gradle file, add:

```
repositories {
    mavenCentral()
}
```

Next, add **compile 'com.squareup.picasso:picasso:2.5.2'** to your dependencies block.

Spotify Wrapper

Compile a new jar from the most recent git clone of the Spotify Web API repository on GitHub. See the README for how to compile the source. Snippet from the README:

> This project is built using Gradle:
>
> 1. Clone the repository: git clone **https://github.com/kaaes/spotify-web-api-android.git**
> 2. Build: **./gradlew** jar

3. Grab the jar and put it in your project. It can be found in **build/libs/spotify-web-api-android-0.1.0.jar**

In order to install the jar file place it into your Android Project under: **<PROJECT_ROOT>/apps/libs directory**

and then add the following statements to your build.gradle dependencies:

```
   compile 'com.squareup.retrofit:retrofit:1.9.0'  // Uses this
to create RESTful Requests
   compile 'com.squareup.okhttp:okhttp:2.2.0'  // Uses this to
create HTTP connections
```

Note: These dependencies are specifically used by the spotify wrapper and you do not need to use them directly. Also don't worry if you don't completely understand these lines. You'll learn more gradle and how it works with Android Studio later in other courses.

If you are having trouble, you can also download the **spotify-web-api-android-0.1.0.jar** from the releases page, though be advised that, as of May 2015, the most recent build was from November 2014, and thus may be a stale build.

Build the UI: Search for an Artist, then Return Their Top Tracks

---

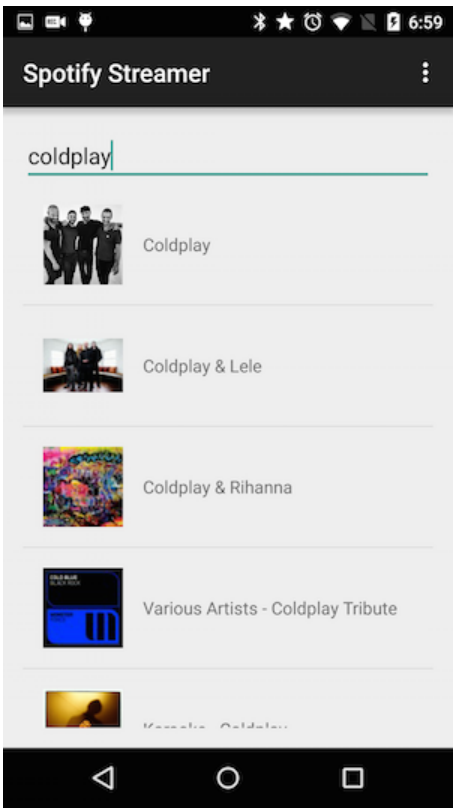
Task 1: UI to Search for an Artist

Design and build the layout for an Activity that allows the user to search for an artist and then return the results in a ListView.

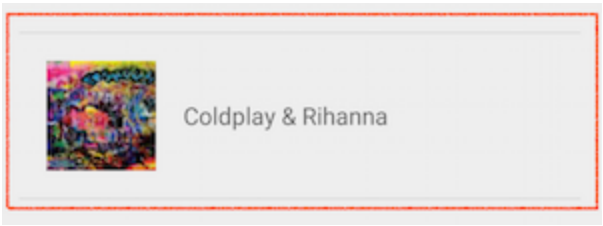
Corner case: If no artists are found, display a toast stating this or write a similar message to the layout.

Note: You should design two layouts:

1. For the artist search activity



2. For an individual artist search result



Guidance:

- See Lesson 1 of Developing Android Apps for support on building UI layouts. For extra polish, check out this section of Lesson 5.

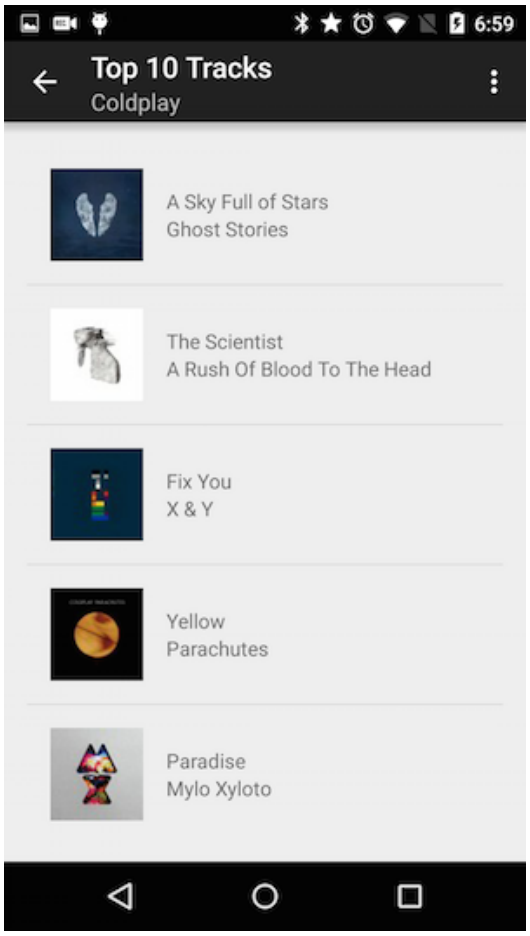
Task 2: UI to Display the top 10 tracks for a selected artist

Design and build the layout for an Activity that displays the top tracks for a select artist.

Note: You should design two layouts:

1. For the Activity that will display the tracks in a ListView
2. For an individual

an individual track (thumbnail, album, track)



Task 3: Query the Spotify Web API

Overview

For the two user interface flows described in tasks 1 & 2, you will need to fetch artist and track data from the Spotify Web API to populate your list views. As mentioned earlier, you will use the Spotify-Web-Api-Wrapper to simplify this task since it handles making the HTTP request and deserializing the JSON response for you. You will be able to extract the artist and track metadata you need directly as java objects.

You will issue the following requests to the Spotify Web API:

For task 1, you will need to request artist data via the **Search for an Item** web endpoint.

- Be sure to restrict the search to artists only by including the item type=”artist”.
- For each artist result you should extract the following data:
    - artist name
    - SpotifyId* - This is **required** by the **Get an Artist’s Top Tracks** query which will use afterwards.
    - artist thumbnail image

For task 2, you will need to request track data via the **Get an Artist's Top Tracks** web endpoint.

- Specify a country code for the search.
- For each track result you should extract the following data:
    - track name
    - album name
    - Album art thumbnail (large (640px for Now Playing screen) and small (200px for list items). If the image size does not exist in the API response, you are free to choose whatever size is available.)
    - preview url* - This is an HTTP url that you use to stream audio

Programming Notes:

- Populate the ListView using a ListView Adapter.
- Fetch data from Spotify in the background using AsyncTask and The Spotify Web API Wrapper

Additional Guidance

Mapping Web API Queries to the Spotify Wrapper Java API

Important Note: Before you write any code using the Web API wrapper, it is important that you familiarize yourself with how the the Spotify Web API works via HTTP.

By doing this exercise, you will not only understand what the wrapper library is doing for you but you'll be able to fully understand the request's parameters as well the JSON response that will be returned.

How do I manually test the Web API via HTTP?

You can do this by submitting test HTTP requests using Spotify's web console or any other REST client of your choice. (@ Udacity, we like clients like PAW or POSTMAN)

Once you're comfortable with the structure of the necessary API requests (and their corresponding responses) in raw form, you can easily implement the same request sequence in Java using the Spotify Wrapper.

Here's a more detailed example:

A Search for "Beyonce" looks like this on the Web Console

https://developer.spotify.com/web-api/console/get-search-item/#complete

Maps to the following Java method in the Web API Wrapper

```
    * Get Spotify catalog information about artists that match a keyword string.
    *
    * @param q The search query's keywords (and optional field filters and
operators), for example "roadhouse+blues"
    * @return A paginated list of results
    * @see <a href="https://developer.spotify.com/web-api/search-item/">Search for an
Item</a>
    */
@GET("/search?type=artist")
public ArtistsPager searchArtists(@Query("q") String q);
```

You can familiarize yourself with the specific Object types such as ArtistsPager using their definitions in code here.

Using Picasso To Fetch Images and Load Them Into Views

You can use Picasso to easily load album art thumbnails into your views using:

```
Picasso.with(context).load("http://i.imgur.com/DvpvklR.png").into(imageView);
```

Picasso will handle properly caching the images.

App 2 - Stage 2:

Spotify Streamer, Stage 2: Implementation Guide

Stage 2: Overview

This document should be used as a guide and development plan for Stage 2 of the Spotify Streamer app. It breaks down the build process into smaller, concrete milestones, and provides technical guidance on specific tasks.

Students are expected to have successfully completed Stage 1 prior to beginning Stage 2.

Supporting course material for this project: Developing Android Apps, Lessons 4-6.

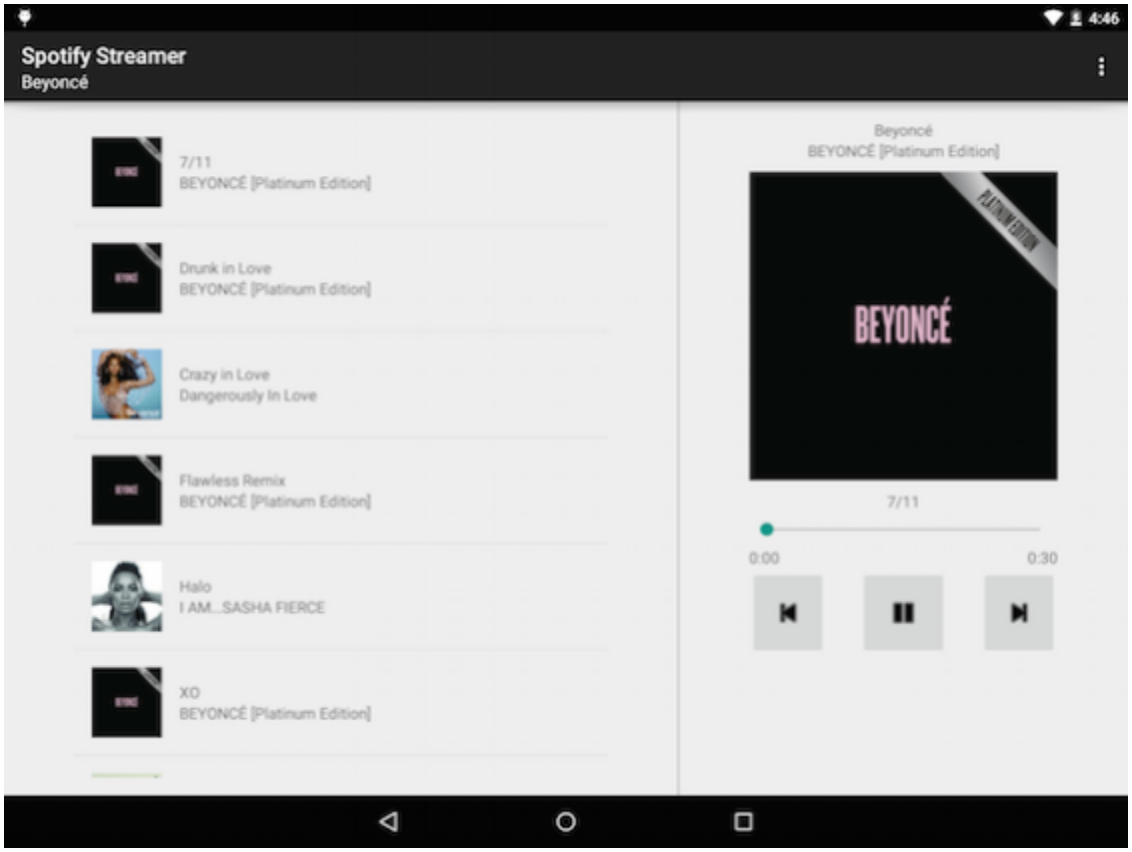
Evaluation

Stage 2 is evaluated against this rubric.

User Experience Design

Tablet Interaction Flow

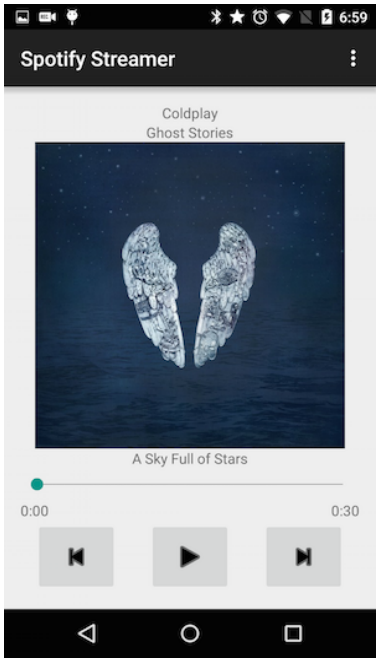
(using a Master Detail Flow)



Build a Track Player and Optimize for Tablet

Task 1: Build a Simple Player UI

Use the layout fundamental skills you learned in Developing Android Apps to build a simple track player. You will launch this view via Intent when a user selects a specific track from the Artist Top Tracks view.



This player UI should display the following information:

- artist name
- album name
- album artwork
- track name
- track duration

This player UI should display the following playback controls:

- Play/Pause Button - (Displays “Pause” when a track is currently playing & displays “Play” when a playback is stopped)
- Next Track - advances forward to the next track in the top track list.
- Previous Track - advances to backward to the previous track in the top track list.
- (Scrub bar) - This shows the current playback position in time and is scrubbable.

Task 2: Implement playback for a selected track

You will use Android Mediaplayer API’s to stream the track preview of a currently selected track.

Please consult the developer.android.com guide on Using MediaPlayer

- Remember that you will be **streaming** tracks versus playing them locally.
- Don’t forget to add the necessary permissions to your app manifest.

Task 3: Optimize the entire end to end experience for a tablet

Migrate the existing UI flow to use a Material-Detail Structure for tablet. You will want to implement two Fragments for your tablet UI; one for search and results and another for Playback.

If you need a review of how to build for tablet, please refer back to Lesson 5 of Developing Android Apps, where the instructors discuss a Master-Detail layout.

Advanced Playback Implementation Guidance:

- Add Links To Media Playback Examples … , developer.android.com documentation
- Google code media playback example