UDACITY

Android Fundamentals

Final Project Description:

Simple A/V Jukebox

App Design Summary

"Pick an app to build, and draft a short (less than one page) summary of what the app does, and what problem it solves."

The concept for this app is that when the user inputs the name of a feature music track or performing artist, the app will begin streaming stills and video from a google image search for the specified terms onscreen. Meanwhile the app will begin playing audio segments pulled from soundcloud, spotify, youtube, or some other free music service for the specified search term. If the audio found is a component of a video stream, the two will be displayed and played synchronized; if the audio is a standalone stream, it will be played while images from the google image search are drifted on and off screen.

Mocks of User-facing screens

"Create mocks for all user-facing screens, and at least one alternate mock for tablet screens. Hand-drawn sketches are OK, but they must be in a digital format that you can share with your Coach."

Figure 1: *Initial screen simply prompting user for input*

Figure 2: *The initial screen as input is spoken*

Figure 3: *The initial screen as search results are returned*

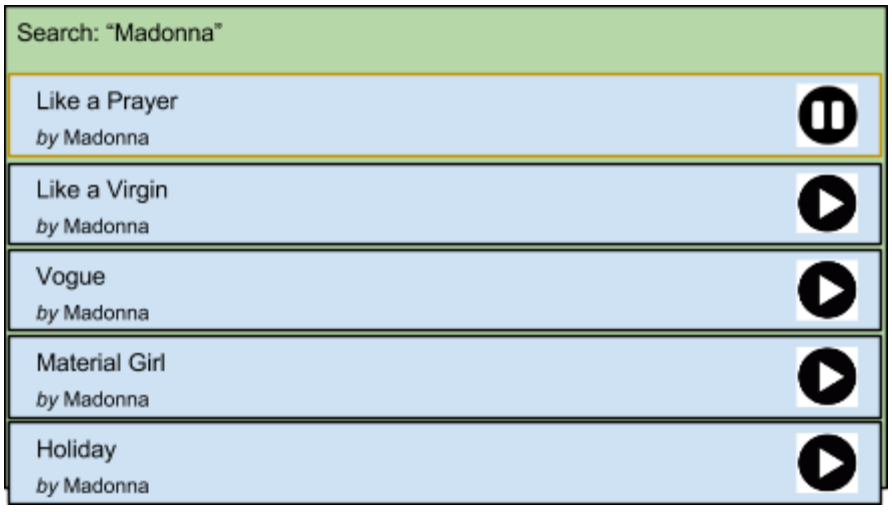

Figure 4: *The list of tracks returned from the search, with the first track automatically playing*

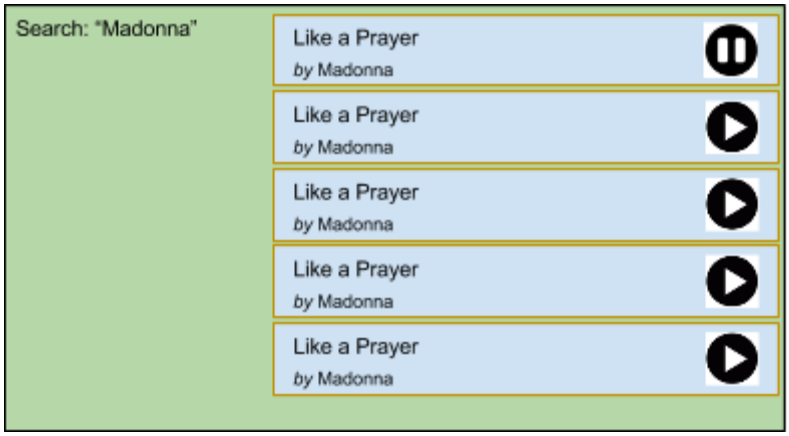

Figure 5: *The list of tracks returned from the search, when displayed on a large-screen device*
This design was not implemented

References

The initial version of the project app was created using the instructions at
https://developer.spotify.com/technologies/spotify-android-sdk/tutorial/

Instructions for building a messagable service bound to an activity were taken from
http://developer.android.com/guide/components/bound-services.html

The implementation of the incoming telephone call broadcast receiver was initially copied from
http://androidcookbook.com/Recipe.seam?recipeId=1109
...and was subsequently modified to register the receiver programmatically instead of via the application manifest.

Appendix 1: Project Completion Steps

1. Pick an app to build, and draft a short (less than one page) summary of what the app does, and what problem it solves.
2. Create mocks for all user-facing screens, and at least one alternate mock for tablet screens. Hand-drawn sketches are OK, but they must be in a digital format that you can share with your Coach.
3. Build your app!
4. Save a copy of the "Android Fundamentals Project Self-Evaluation". Follow the instructions in the doc to explain your thought process to the grader and confirm your app meets specifications.
5. Create a signed .apk of your app. Instruction are available here.
6. Zip your source code, your signed .apk, and all accompanying documents including the "Android Fundamentals Project Self-Evaluation".
7. Send an email to androidapps-project@udacity.com that includes following:
    a. Your zip file from step 5 above
    b. A list of Web sites, books, forums, blog posts, github repositories, etc. that you referred to or used in creating your submission (add N/A if you did not use any such resources).
    c. Please carefully read the following statement and include it in your email: *"I hereby confirm that this submission is my work. I have cited above the origins of **any** parts of the submission that were taken from Websites, books, forums, blog posts, github repositories, etc. By including this in my email, I understand that I will be expected to explain my work in a video call with a Udacity coach before I can receive my verified certificate."*

Appendix 2: Project Requirements

Project must:

- ✓ Be entirely self-contained on Android devices (No external devices/bluetooth peripherals)
- ✓ Include a problem description of the problem your app solves.
- ✓ Include mocks for all user-facing screens.
- ✓ Include a signed .apk of your app
- ✓ Include at least one alternate mock for tablet / large screens
- ✓ Have at least two distinct screens.
- ✓ Work properly with the app lifecycle (rotate screen changes)
- ✓ Use permissions responsibly.

- ✓ Use Intents to move inside your app or to an outside app.
- ✓ Create and use your own ContentProvider
- ✓ Create and use your own custom Loader

For extra Udacious-ness, include at least two of the following (optional):

- ✓ Receive Broadcast events and do something meaningful
- ❏ Create and use a Custom View
- ❏ Implement ShareActionProvider
- ✓ Use Notifications

Appendix 3: Project Self-Evaluation

Android Fundamentals Project Self-Evaluation

Instructions: Once you've completed your Final Project, please evaluate it against the components of the rubric below. For each criteria that you met, put an "X" in either the "Does Not Meet Specifications" or the "Meets Specifications" box. For some criteria, we ask you to provide an explanation of where and how it was implemented in your app. This is a chance for you to briefly explain to the grader your thought-process during development. Once you are done, include this with the source code and accompanying files you are submitting. Then, give yourself a pat on the back for making a great app!

Required Components

To "meet specifications", your app must fulfill all of the criteria listed in this section of the rubric.

Criteria	Does Not Meet Specifications	Meets Specifications
Standard Design		
App does not redefine the expected function of a system icon (such as the Back button).		X

App does not replace a system icon with a completely different icon if it triggers the standard UI behavior.		X
App does not redefine or misuse Android UI patterns, such that icons or behaviors could be misleading or confusing to users.		X
Navigation		
App supports standard system Back button navigation and does not make use of any custom, on-screen "Back button" prompts.		X
All dialogs are dismissible using the Back button.		X
Pressing the Home button at any point navigates to the Home screen of the device.		X
Permissions		
App requests only the absolute minimum permissions that it needs to support core functionality.		X
App does not request permissions to access sensitive data or services that can cost the user money, unless related to a core capability of the app.		X
Please elaborate on why you chose these permissions: **android.permission.INTERNET**: to perform the spotify track search, and to stream the selected audio from spotify **android.permission.RECORD_AUDIO**: to obtain the spoken text that will be used for the spotify track query		
Performance and Stability		
App does not crash, force close, freeze, or otherwise function abnormally on any targeted device.		X
ContentProvider		
App retrieves and caches data from a server using a ContentProvider.		X

If it regularly pulls or sends data to/from a web service or API, app updates data in its cache at regular intervals using a SyncAdapter.		N/A
Please elaborate on how/where you implemented a ContentProvider and SyncAdapter: The app's functionality of retrieving a list of freely available audio tracks from Spotify was accomplished with a simple content provider that only supports one URI context. Because the searches were initially envisioned as always being on demand (and therefore immediate), I elected not to implement a SyncAdapter -- e.g., the content provider always performs a search on the Spotify web API to retrieve the requested content. Given the luxury of more time, I might refactor the implementation to cache the retrieved data locally into a SQLiite database and include a middle-tier SyncAdapter. However, the SyncAdapter functionality was not as necessary as the other features that I included in the initial design, and so were left to subsequent releases for implementation.		
User/App State		
App correctly preserves and restores user or app state.		X
When the app is resumed after the device wakes from sleep (locked) state, the app returns the user to the exact state in which it was last used.		X
When the app is relaunched from Home or All Apps, the app restores the app state as closely as possible to the previous state.		X
Please elaborate on how/where your app correctly preserves and restores user or app state: The initial "MainActivity" (which admittedly is very simple) stores its state (one of WAITING_FOR_INPUT, PARTIAL_INPUT_RECIEVED, INPUT_COMPLETE__SEARCHING_FOR_TRACKS, INPUT_COMPLETE__SEARCH_COMPLETE, or INACTIVE)		

in onSaveInstanceState(), and when onRestoreInstanceState() is called the state is restored. When the onscreen activity contains a search result track list (a TrackListFragment), the scrolled list position is maintained.		

Optional Components

To receive “exceeds specifications”, your app must fully implement all of the criteria listed under at least two of the four categories below (e.g. Notifications, ShareActionProvider, Broadcast Events, and Custom Views).

Criteria	Does Not Exceed Specifications	Exceeds Specifications
Notifications		
Notifications do not contain advertising or content unrelated to the core function of the app.		X
Notifications are persistent only if related to ongoing events (such as music playback or a phone call).		X
Multiple notifications are stacked into a single notification object, where possible.		X
App uses notifications only to indicate a context change relating to the user personally (such as an incoming message).		X
App uses notifications only to expose information/controls relating to an ongoing event (such as music playback or a phone call).		X
Please elaborate on how/where you implemented Notifications in your app: When the app plays a selected audio clip (either automatically or as a result of user election), the track title is included in a notification. When the clip is finished or paused, the notification is dismissed.		

If the user taps the notification, a track list containing the playing track is displayed.		
ShareActionProvider		
Uses ShareActionProvider to share content with an outside application.	X	
Makes use of Intent Extras to send rich content.	X	
Please elaborate on how/where you implemented ShareActionProvider: N/A		
Broadcast Events		
App intercepts broadcast events.		X
App responds to Broadcast events in a meaningful way.		X
Please elaborate on how/where you implemented Broadcast Events: When the app is playing audio and an incoming telephone call rings, the media player service pauses any playing clips until the phone state returns to idle. When the app plays a selected audio clip (either automatically or as a result of user election), the track title is included in a broadcast intent with the action "com.josephbanta.avjukebox" and the extra text "NowPlaying" as the track title.		
Custom Views		
App creates and uses a custom View.	X	

App uses a novel View that couldn't sufficiently be satisfied by the core Views in Android.	X	
Please elaborate on how/where you implemented Custom Views: N/A		