GitHub Username: DomainFlag

Jiggles

Description

Interactive Digital Light-Weight media service where you can discover new music, albums and playlists you will love. Listen to music, share your music and discuss the latest news and releases through forum with people like you that do enjoy listening to music…

Intended User

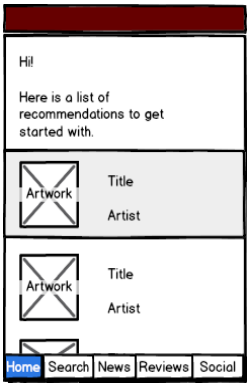
The target group are people that are casual listeners that do like listening to music and are willing to go a step further by interacting with others on forum, share their music and more...

Features

- Discover songs / albums / artists
- Recommendations (songs / albums / artists)
- New releases (with critic's review) / news
- Song playback (notification, widget)
- Create own collection
- CRUD threads / comments on forum

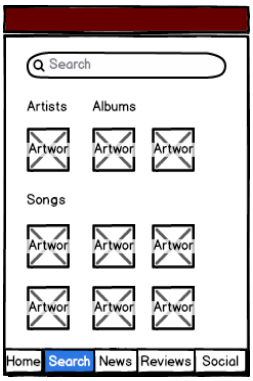
User Interface Mocks

Screen 1



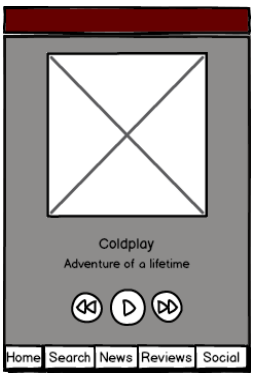
Landing Home page when app is first started, user is exposed with a list of recommendations that the user can chose which he likes the most and start build his collection.

Screen 2



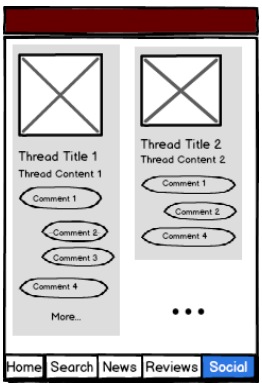
Search fragment, where the user can search for anything he wants (songs/albums/artists) which the result will be shown bellow as set of adaptable grids where he chooses whatever he wants to play.

Screen 3



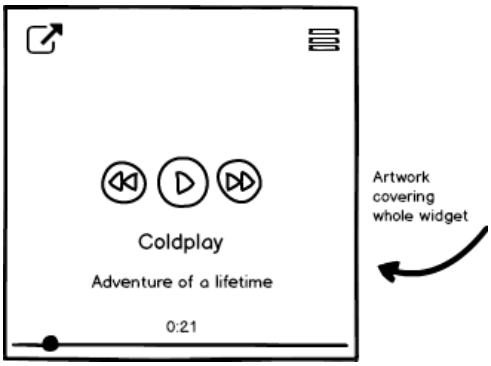
Playback activity which will be used to playback whatever content found out by the user while searching, clicking the recommendation links, the tags under news section that is referring to a song, review tags that is referring to a song being reviewed…

Screen 4



Social forum, where people can discuss different themes like the release of an album / song, how good is the content of an artist, event, or a concert which will be held soon.

Screen 5



The widget will be simple yet powerful utility, from interacting with the content being played out to visualizing the latest review or news. The widget will extend the RemoteViewsService to populate the remote collection view (ListView) with content from user's collection or news/reviews.

Key Considerations

How will your app handle data persistence?

As you can see in the repository there are 2 jiggles folders, where there is a jiggles_end folder which will be used as a remote back-end server for the native app. Thus, the forum(threads) / releases(critics) / news / collections will be preserved on remote MongoDB server, and a Content Provider will be used to cache the latest data to allow the application to have easy access to its SQLite Database and for better abstraction over the correspondence between the application and database. The reason why caching, data exchange over the HTTP is costly and it does take time, thus by caching let's say a list of collection (songs / artists /albums) we will be able to provide an initial list of content by using an AsyncTaskLoader and maintain the data at rotation changes with the SupportLoaderManager. It will be used in conjunction with HTTP requests to sync the cache data with the updated data and update the UI when necessary.

Preferences will be used too, as an example, persisting the type of content which the user wants to see first when opening the app (songs / albums / artist collection or default one the latest releases with critic's reviews / news).

Describe any edge or corner cases in the UX.

- Rotations changes, saving the current state of activity, because the user can have bad UX if all the activity done won't be saved.
- Properly managing the ExoPlayer audio playback state during the activity lifecycle and audio focus for cases like navigation or calling.
- Keeping the user logged in the next time he opens the app by storing the token on local storage for future exchanges with the server.

Describe any libraries you'll be using and share your reasoning for including them.

As the app will be build using Java language in conjunction with Android SDK provided by Android Studio, the latest features will be used, where the project will be built, and all the tasks automated with latest version of grade 3.1.4**v**. To provide standard features the latest stable version of support library will be used: 27.1.1**v**

- Picasso will be used to handle the loading and caching of images. - 2.71828**v**
- Use of Spotify Android Playback SDK for audio playback - 24-noconnect-2.20b**v**
- Use of Spotify Android Auth SDK for logging to Spotify (obtain the credentials) - 24-noconnect-2.20b**v**
- Use of okhttp for managing HTTP responses/requests - 3.11.0**v**
- ExoPlayer will be used for loading local audio files and its playback - r2.8.4**v**

Describe how you will implement Google Play Services or other external services.

Application will be using ads AdMob - 15.0.1**v** which depends on google play services. Also, the Analytics library - 10.2.4**v** provided by google play services we'll be used to track the user's activity and improve UI/UX, and better know the target user group of your app and further improve the app and make the necessary SEO adjustments.

Next Steps: Required Tasks

Task 1: Project Setup

First, is to configure the libraries used for requesting the web server(jiggles_end) to authenticate and get the access token for future interactions. From here, with the token provided, I will be able to set up the necessary URIs for fetching data (especially the Spotify content URI) which from there I will be able to use with the Spotify API for audio playback and start building the app.

Task 2: Implement UI for Each Activity and Fragment

The app will be consisted of MainActivity which will contain a ViewPager which will contain respectively 5 different fragments (Home, Search…) where which will be used in conjunction with 5 different layouts to implement the main functionalities of the app. There will be also a menu button that will expose the Auth functionality (Log out), Offline mode (local music), Preferences and so on...

Task 3: Home fragment

Create 2 different layouts, one that will be used when the app is first opened that will be showed off after you already authenticated. It will contain a List (Recycler View) of recommendations where you will pick which one you like the most & finally building your first collection.

If it was opened before, then a header with the latest song being played out and a grid with most played content will be showed to the user.

Task 4: Search fragment

Creating the fragment, there will be needed a single EditText placed above the layout, and an adaptable layout for different screen sizes will be used to show up the search result. For screens that are below 600dp, a list with 3 dividers (Songs / Albums / Artists) and their respective content will be showed up. For screens above 600dp, a 3x horizontal grid layout will be used in conjunction with 3 dividers where the artists take the most space, and the others take equal space. Each grid layout can be expanded to show even more content then 4 elements max when not expanded, thus the others two will be set to Visibility.GONE.

Task 5: News / Reviews & Social (Forum) fragment

Creating the fragments that each have different layouts (single vertical list for news) and (grid layout for reviews). Each element will contain a Tag that can redirect the user the artist / song / album the news / review is concerned on.

There will be grid layout where all the threads will be showed off, an upper bar will be present for filtering types, a floating action button for thread creation where a modal window will be showed up.

Task 6: Audio Playback

Playback activity, that does contain a single layout to be rendered (artwork, with media buttons, seek bar ...), that does the media playback using the Spotify API. Media playback both for Spotify and Offline mode will be implemented to the respect of the activity lifecycle… Working on notification bar to control the media playback and on audio focus. Create the widget which will be used to control both the Spotify audio playback and ExoPlayer instance, depending on if the user wants to play offline (local music) or online. The communication between the widget and the main app will be based on broadcast intents that will inter-exchange data and current playback state by communicating between the two ends (AppWidgetProvider that does extend the BroadcastReceiver and the main Application)

Task 7: Miscellaneous

Working on orientation changes and preserving the state for better UX. Make the app more accessible, by providing content descriptions to images, using hint text for user interface elements like EditText, better layout design and more intuitive and easy-to-follow navigation. Working on preferences on how the user likes the most the layout and other features more up to come… App will keep all strings in a strings.xml file to internationalize and will enable RTL layout switching on all layouts.