[Description](#_sm4ra97uwo11)

[Intended User](#_aws88pzfmqca)

[Features](#_zheq5430xrpq)

[User Interface Mocks](#_giquerrw6g46)

[Screen 1](#_a4jdupabry3k)

[Screen 2](#_dpcbbkx5yry)

[Key Considerations](#_gvcvmae8jn8u)

[How will your app handle data persistence?](#_v8my7nhtvz0m)

[Describe any corner cases in the UX.](#_gw69vjn1ico0)

[Describe any libraries you’ll be using and share your reasoning for including them.](#_6yqqubmw5bs)

[Describe how you will implement Google Play Services.](#_qrxg682nywe6)

[Next Steps: Required Tasks](#_v518bncmggeg)

[Task 1: Project Setup](#_8oe8zpk3qsmp)

[Task 2: Implement UI for Each Activity and Fragment](#_rzllsk6uqztx)

[Task 3: Your Next Task](#_fdmohs7hes)

[Task 4: Your Next Task](#_umfwsvmx7tpn)

[Task 5: Your Next Task](#_kjidlkq4xm3u)

**GitHub Username**: neone35

SingSpots

# Description

Find places where you can record your own music. This app shows them in convenient way – on a single map, so you can easily find one nearest to you. Database of places is made up of all the past recording places of all the known artists and bands!

Use this to:

* Find out location of famous music recording places
* Create your own recordings in local places
* See where music is recorded from 1990 until now

# Intended User

Artists, music producers

# Features

* Uses a map
* Shows all known record places
* Provides search

# User Interface Mocks

Continue to next page

# Key Considerations

### How will your app handle data persistence?

### Describe any edge or corner cases in the UX.

Places returned per request should be limited, but all places must be displayed on map.

* For example there 100 places for search term, but limit is 20, so you need 5 requests to get all the places.
* Make this limit easy to tune in code
* Displayed places should be open from 1990

Every pin has a lifespan, meaning after it expires, pin should be removed from the map. Lifespan calculation: open\_year - 1990 = lifespan\_in\_seconds. Example: 2017 - 1990 = 27s

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

**Retrofit 2.4.0** to ease API consumption

**Gson 2.8.5** to parse json data into Java models

**Butterknife 8.8.1** to ease R class binding to code

**AAC (ViewModel & LiveData) 1.1.1** to persist runtime data and observe changes

**Utilcode 1.17.3** to use common functions

**Logger 2.2.0** for simple, pretty and powerful logs

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

# Next Steps: Required Tasks

This is the section where you can take the main features of your app (declared above) and break them down into tangible technical tasks that you can complete one at a time until you have a finished app.

## Task 1: Project Setup

* Use Android Studio 3.1.3.
* Use Gradle 4.9, Android Plugin 3.1.3.
* Use Java 1.8 compiler to support short lambda syntax.
* Implement, sort & sync libraries
* Define & replace constant for library versions
* Update app logo
* Define theme & colors

## Task 2: Implement UI for Each Activity, Fragment and Widget

* Build Uis for activities

## Task 3: Build database & models classes

* Create POJO for PlacesResponse

## Task 4: Build network classes

* Create Retrofit Endpoint Intefrace to consume musicBrainz API
* Create NetworkUtils class with external URLs and helper methods.
* Create AsyncTask to fetch search query

## Task 5: Bonus tasks

* Clean code and best OOP practices
* Kotlin specific features
* Test-covered code