How to Use this Template

1. Make a copy [File → Make a copy...]
2. Rename this file: “**Capstone_Stage1**”
3. Replace the text in green

Submission Instructions

1. After you’ve completed all the sections, download this document as a PDF [File → Download as PDF]
2. Create a new GitHub repo for the capstone. Name it “**Capstone Project**”
3. Add this document to your repo. Make sure it’s named “**Capstone_Stage1.pdf**”

GitHub Username: nathannak

nathannak/Capstone-Project

YesQoogle!

(i have completed a draft version of this app on my free time while i was working towards my Android nano degree, since it is a big project, i had to plan and start early)

Description:

YesQoogle! Searches for up to three saved queries against three business/venue finding services: Foursquare, Yelp, and Google places manually and automatically. Automatic search kicks in when user moves for a certain distance, this feature can be tuned off. App automatically filters businesses based on ratings which can be manipulated by user, and shows common-between-the-two businesses as well, which are businesses which showed up in two of the three services we used to search against.

Intended User:

Basically everyone can use this app, instead of searching in Google.com, or having using three different services (Yelp, Foursquare, Google Places), they can use one app, and use it to search for their favorite venues, manually and automatically. App filters businesses based on user standards, and enables users to see common businesses which showed up in two out of three search services.

Features:

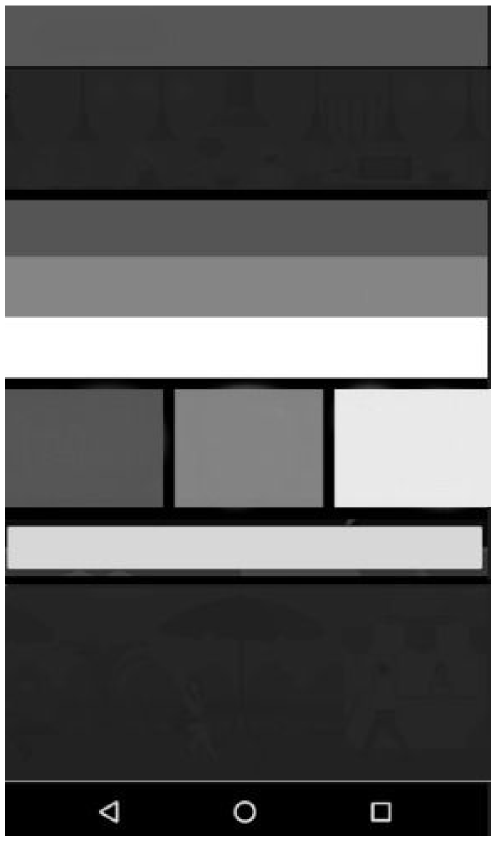
- Keeps track of user location to perform automatic search
- Saves business informations

- Ability to share findings with others
- Ability to automatically filter businesses based on user standards.
- Ability to show common-between-the-two businesses (businesses which satisfied user standard in two of the three services)

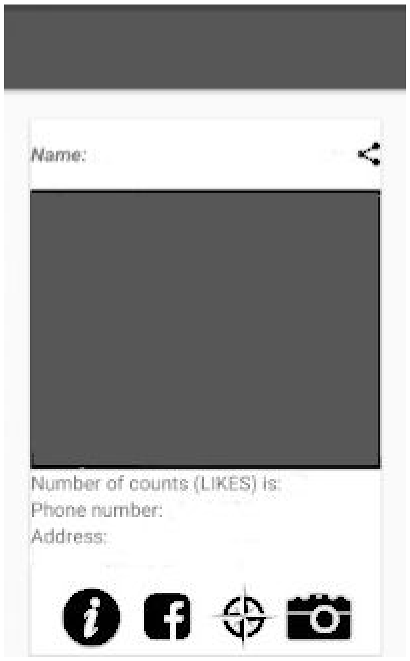
User Interface Mocks:

These can be created by hand (take a photo of your drawings and insert them in this flow), or using a program like Photoshop or Balsamiq.

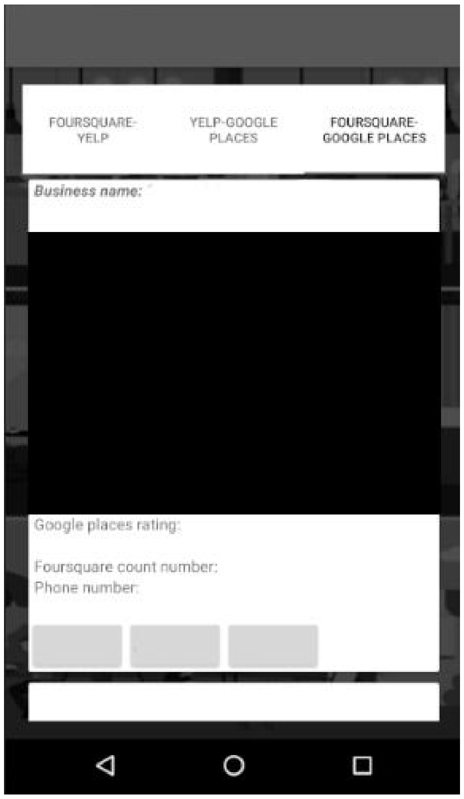
Main screen:



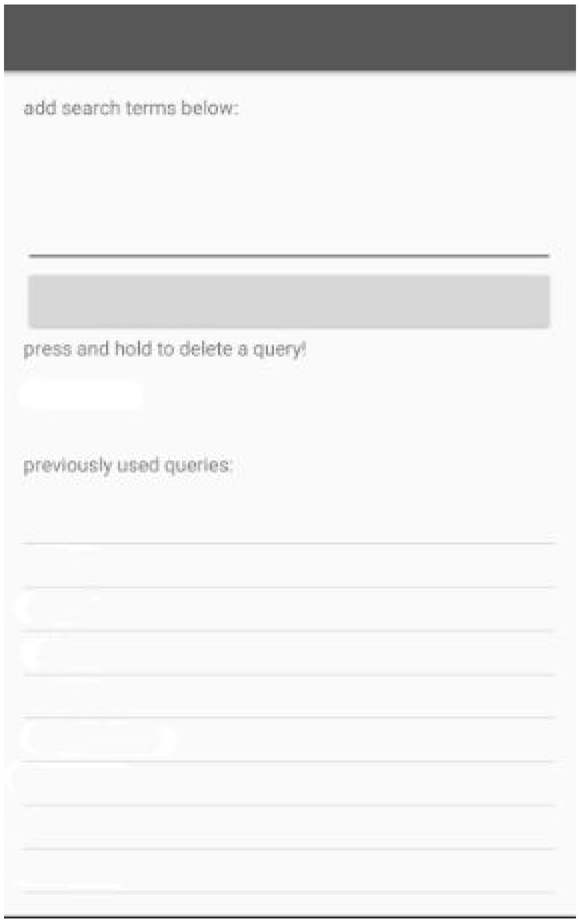
Once any of smaller squares is clicked, detail information about a business will show (using card view and recyclerview):



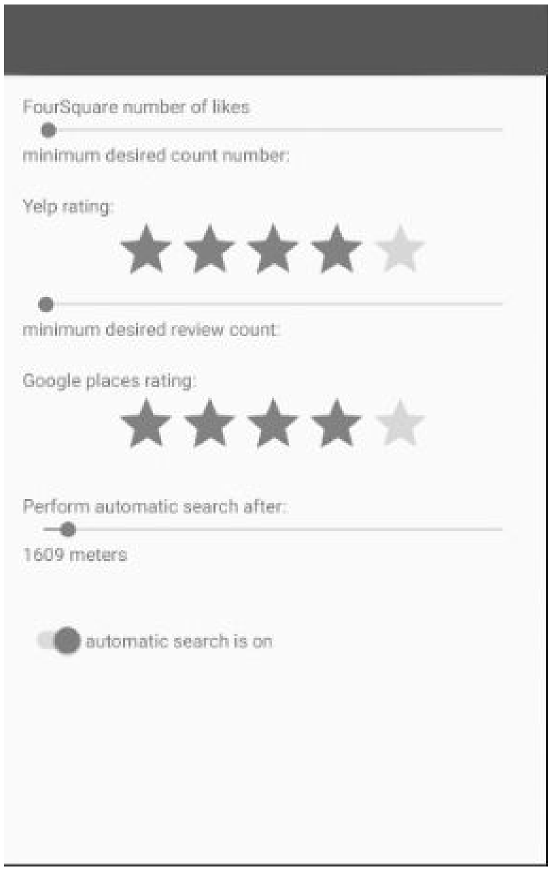
If user clicks on a bigger square common-between-the-two results will show:



Query adding activity:



Activity to modify user standards for an an acceptable business in terms of rating, and comment numbers:


FourSquare number of likes

minimum desired count number:

Yelp rating:

minimum desired review count:

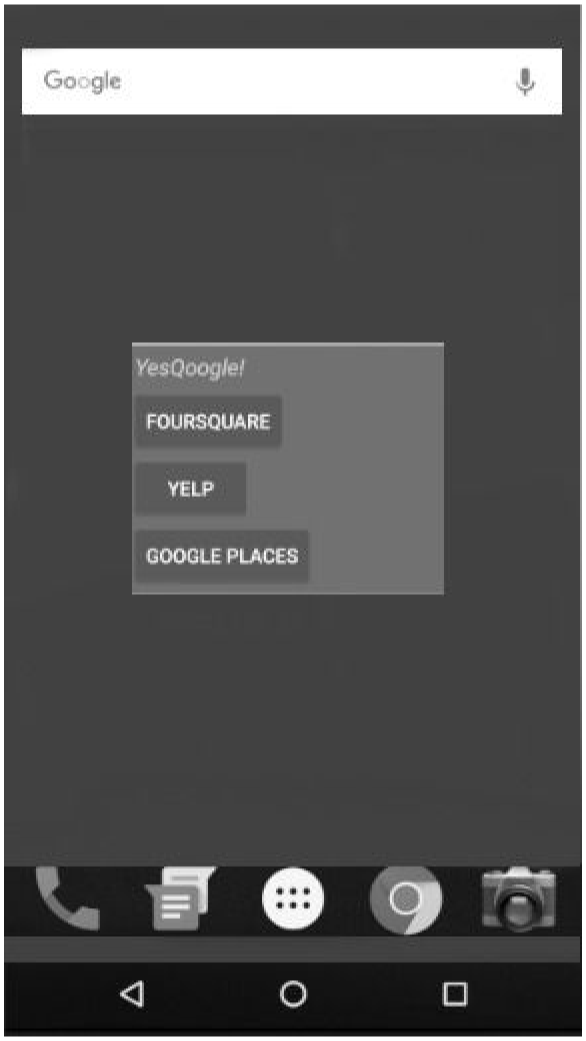
Google places rating:

Perform automatic search after:

1609 meters

automatic search is on

Home Widget:



Key Considerations:

How will your app handle data persistence?

Uses SharedPreferences, and five SQLite databases to save user locations, deleted keywords (this DB also uses content provider/resolver) and databases for three business searching venues.

Describe any corner cases in the UX.

Flow is well controlled, all activity opens with press of a button, which can be navigated back to if necessary by taking the same path. For example if user chooses to see images related to a business s/he can press the ”more images” button, if back button is pressed, same path will take him/her to see more images of that business.

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

'Com.android.volley:volley:1.0.0': To make Async Http connection and download Json.

'com.yelp.clientlib:yelp-android:2.0.0': To connect to yelp restful api.

'com.squareup.retrofit2:retrofit:2.1.0': To make use of yelp restful api

'com.squareup.picasso:picasso:2.5.2': To load images from SQLite databases into image views.

'org.lucasr.twowayview:twowayview:0.1.4': To be able to show images horizontally for common results section.

'com.google.android.gms:play-services:9.6.1': Necessary to use google services such as Google Places api and user geolocation.

‘**Com.google.android.gms:play-services-location:9.6.1': To find user location**

'Com.commonsware.cwac:wakeful:1.0.+': To wake up device to do stuff when phone is sleeping.

Universal Image Loader: Load images in image view similar to Picasso, it is used when image url link redirects to another link.

'Com.squareup.okhttp:okhttp:2.2.0': Facilitates image loading using universal image loader

'Com.squareup.okhttp:okhttp-urlconnection:2.2.0': Facilitates image loading using universal image loader

Describe how you will implement Google Play Services.

Will use Geolocation services (fused location) to keep track of user’s location and also Google places API as one of the business finding services.

Next Steps: Required Tasks

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

Project Steps in order:

1. Design query settings activity to save up to three user keywords in shared preferences, use content provider/resolver combined with a SQLite database to save previously search keywords.

2. Configure JSON parsing and loading from Foursquare using volley, make sure they correctly save to SQLite database, i will need a class for a Foursquare business with members such as address, phone number ...

3. Configure JSON parsing and loading from Yelp using yelp's library and Retrofit, make sure they correctly save to SQLite database, i will need a class for a Yelp business with members such as address, rating ...

4. Configure JSON parsing and loading from Google places API using Volley library, make sure they correctly save to SQLite database, i will need a class for a Google places businesses with members such as phone number, rating …

5. Design an Intentservice for manual search, which searches Foursquare, Yelp, and Google Places in order, and filters then save data in SQLite databases. Service execution should NOT make the phone run out of memory, be careful with memory consumption.

6. Design adapters and activities to show data downloaded from Foursquare, yelp, and google places individually, you can use Universal image loader to load images for Yelp, and Picasso for Google places and Foursquare; all the readings happen from SQLite databases. Add an option to navigate to the business using Google maps, show all the necessary information such as phone number, rating …

7. Add methods to find common-between-the-two search results to find common results between Foursquare-Yelp, Yelp-Google Places, Google Places-Foursquare using businesses phone numbers. Remember same phone numbers are presented in different formatting in different API's, so you have to study them and convert them to the same format, to be able to compare them.

8. Design adapter and activity with Viewpager, tab layout to show common-between-the-two results; use TwoWayView to show images horizontally. Demonstrate ratings and number of comments from both services.

9. Design an activity to modify filtering criteria to be used for filtering search results, for example four star minimum rating for businesses shown in Yelp, etc ... add an option to turn off automatic search, and change how long user should move before automatic search kicks in.

10. Use Memory Analysis Tool to look for memory usage since we are doing heavy lifting, memory analysis is very important, Picasso, and Volley occupy cache memory which should be cleared after use.

Regarding "App design specification demonstrates implementing all features required for Project 8: Capstone, Stage 2 - Build":

I will discuss how app may meet the specs thoroughly below:

MEETS SPECIFICATIONS
App conforms to common standards found in the Android Nanodegree General Project Guidelines : very general, nothing to add.

Core Platform Development

MEETS SPECIFICATIONS
App integrates a third-party library : already discussed in this doc.
App validates all input from servers and users. If data does not exist or is in the wrong format, the app logs this fact and does not crash : i will make sure app is tested completely on different phones with different Android versions.

App includes support for accessibility. That includes content descriptions, navigation using a D-pad, and, if applicable, non-audio versions of audio cues: can you please refresh my memory on these?
App keeps all strings in a `strings.xml` file and enables RTL layout switching on all layouts : app will meet this specification
App provides a widget to provide relevant information to the user on the home screen: added to the design doc, see UI mocks.

Google Play Services

MEETS SPECIFICATIONS
App integrates two or more Google services. Google service integrations can be a part of Google Play Services or Firebase: app uses Google places api and Google Geo location.
Each service imported in the `build.gradle` is used in the app: will keep in mind!
If `Location` is used, the app customizes the user's experience by using the device's location: businesses near user will be searched and processed.

If `Admob` is used, the app displays test ads. If `Admob` was not used, student meets specifications.
If `Analytics` is used, the app creates only one analytics instance. If `Analytics` was not used, student meets specifications.
If `Maps` is used, the map provides relevant information to the user. If `Maps` was not used, student meets specifications.
If `Identity` is used, the user's identity influences some portion of the app. If `Identity` was not used, student meets specifications.

Material Design

MEETS SPECIFICATIONS
App theme extends `AppCompat`: **will keep in mind!**
App uses an app bar and associated toolbars: **will keep in mind!**
App uses standard and simple transitions between activities : **can you please refresh my memory on these?**

Building

MEETS SPECIFICATIONS
App builds from a clean repository checkout with no additional configuration: **will keep in mind!**
App builds and deploys using the `installRelease` `Gradle` task: **will keep in mind!**
App is equipped with a signing configuration, and the keystore and passwords are included in the repository. Keystore is referred to by a relative path: **will keep in mind!**
All app dependencies are managed by `Gradle`: **will keep in mind!**

Data Persistence

MEETS SPECIFICATIONS
App implements a `ContentProvider` to access locally stored data: **only one out of 6 databases use content provider, i only added it to meet specs.**
Must implement at least one of the three 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`.

OR **If it needs to pull or send data to/from a web service or API only once, or on a per request basis (such as a search application), app uses an `IntentService` to do so: App uses multiple Intent Services to do sequential search against 3 API's.** **OR** **It it performs short duration, on-demand requests (such as search), app uses an `AsyncTask`.**
App uses a `Loader` to move its data to its views: only one out of six databases use Content Provider/Resolver. I added it only to meet specs.

Submission Instructions

1. After you've completed all the sections, download this document as a PDF [File → Download as PDF]
2. Create a new GitHub repo for the capstone. Name it "**Capstone Project**"
3. Add this document to your repo. Make sure it's named "**Capstone_Stage1.pdf**"