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: Your GitHub username here

The WineOisseur

Description

Design a wine search app that will help users to decide on what type of wine to buy and where to purchase the bottle. The user is able to search by color, region, rating, varietal and price range. They will be able to read reviews by other users, what food pairings go with that particular bottle of wine and details about the winery that produces the wine.

Intended User

Everyone who drinks wine.

Features

List the main features of your app. For example:

- Search wines based on preference
- Saves your favorite wines to a SQLite database
- Will give a list of stores and prices on where to purchase the wine
- Enables the user to read reviews and food pairings that go with the wine

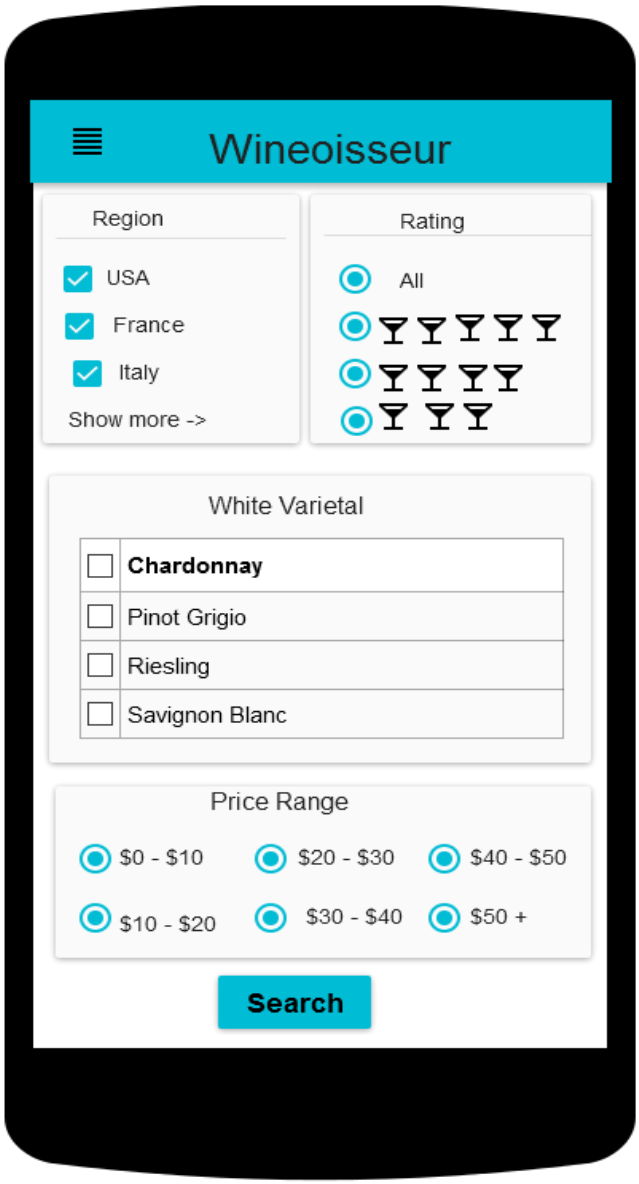
User Interface Mocks

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.

Screen 1 – navigational drawer

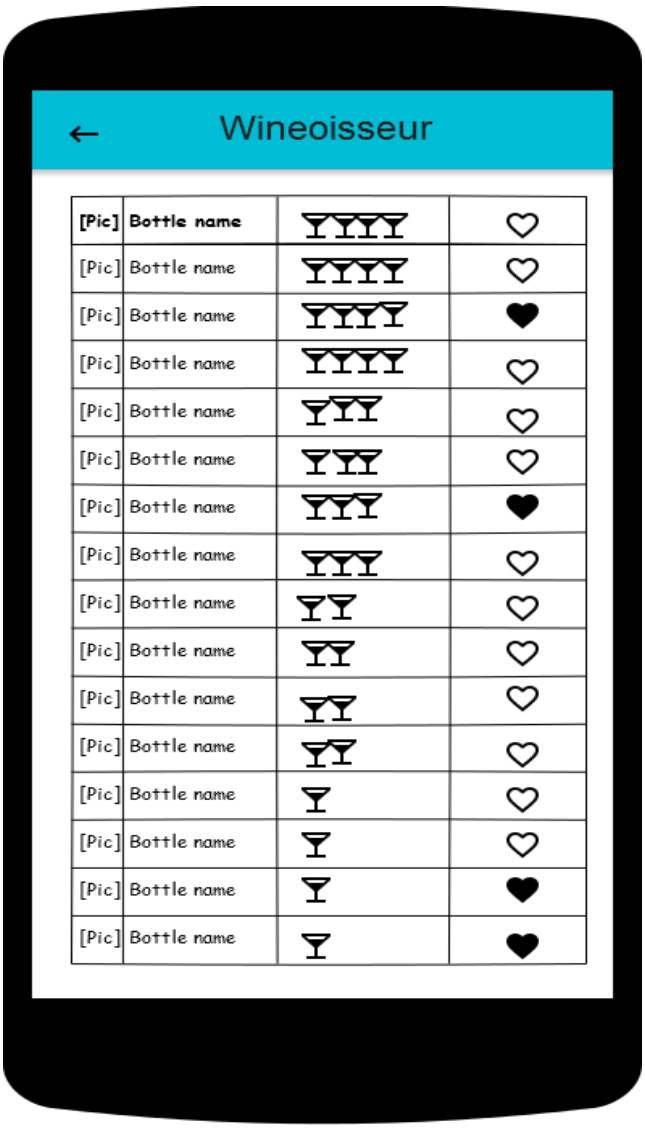
This is the navigational drawer. The user can begin its search by selecting Red, white or dessert. By selecting Favorites you can view wines that were saved in the database if available. The user also has the ability to save its preferences by selecting settings.

Screen 2 - Detailed search screen for wines



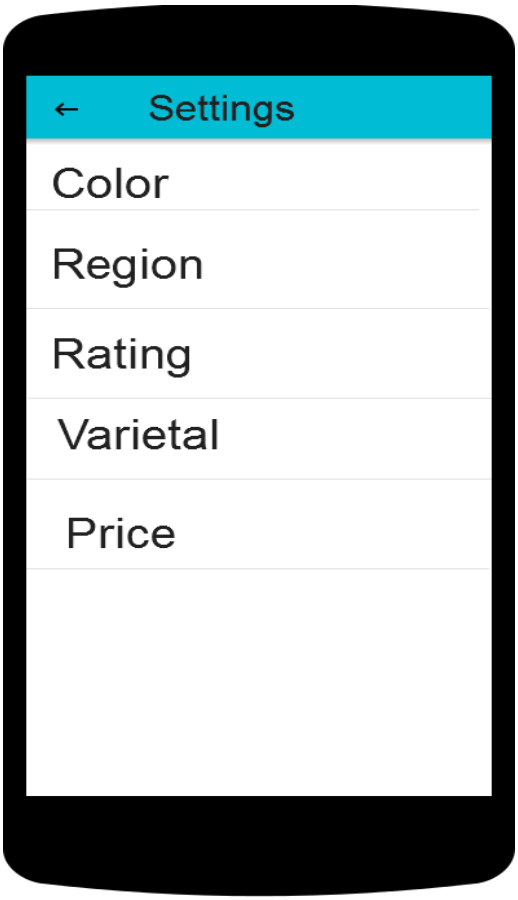
Detailed search screen – In the region if you don't see the country you want to select click on show more and a dialog fragment will pop up with a wider range of countries to choose from.

Screen 3 - Search results



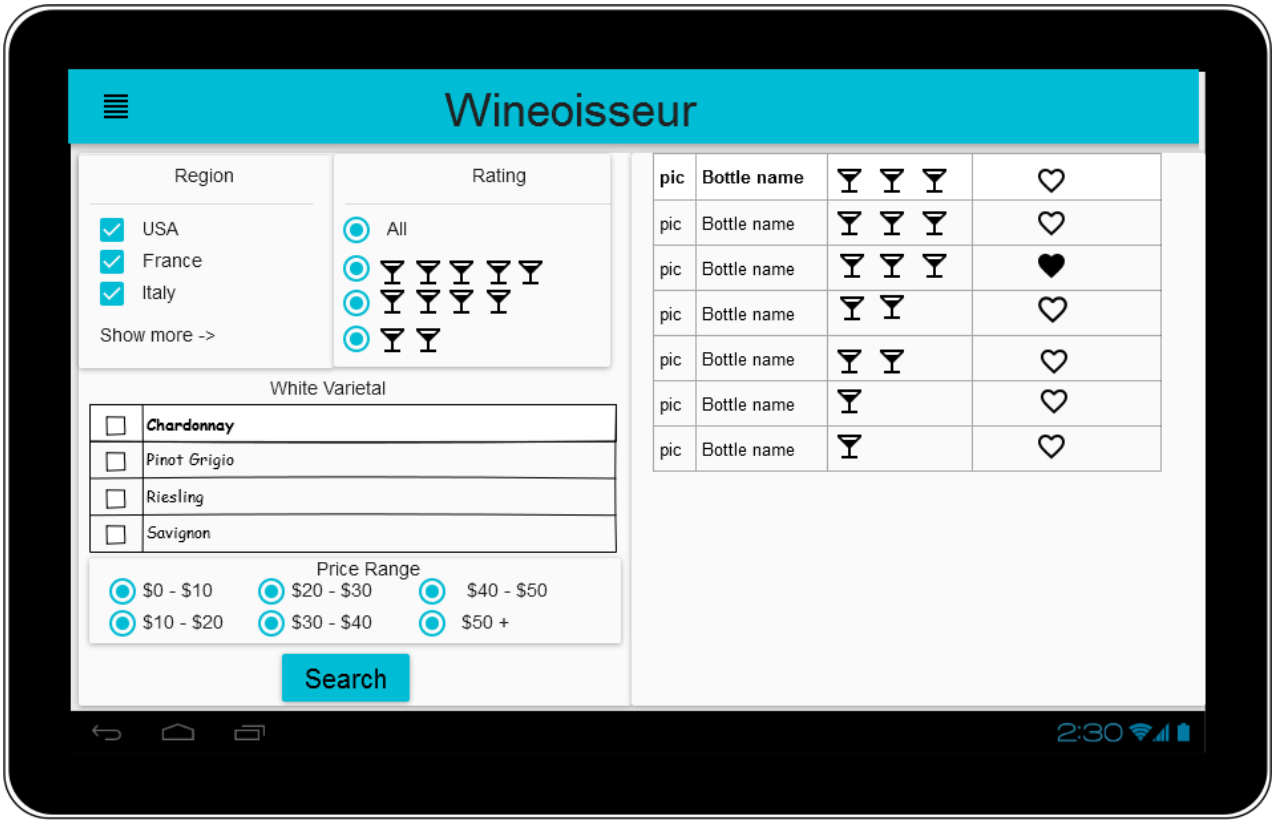
Search results – Recyclerview with a cardview that will display an image of the bottle, a textview with the name of the bottle, user rating and an image of a heart. When you select a row it will direct you to a web site with detailed information about that wine. By Clicking on the outlined heart it will save the bottle to the favorites SQLite database. By clicking on the solid heart it will remove the bottle from the database.

Screen 4 - fragment_settings.xml



Settings screen – will show a list of customizable options for searching for wine.

Screen 5 - Tablet screen



Tablet screen – will show the detailed search screen on the left side and the results on the right side of the screen.

Screen 6

Widget Screen

List of favorite wines

Bottle image

Name / vintage

Bottle image

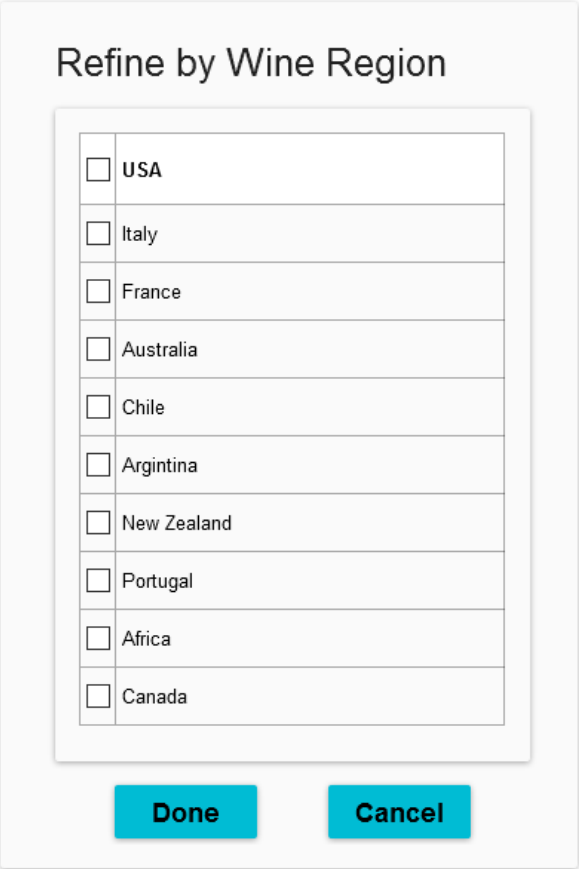
Name / vintage

Bottle Image

Name / vintage

Widget screen – will show a list of favorite wines or wines on sale at your local store.

Screen 6 – Dialog fragment screen for region

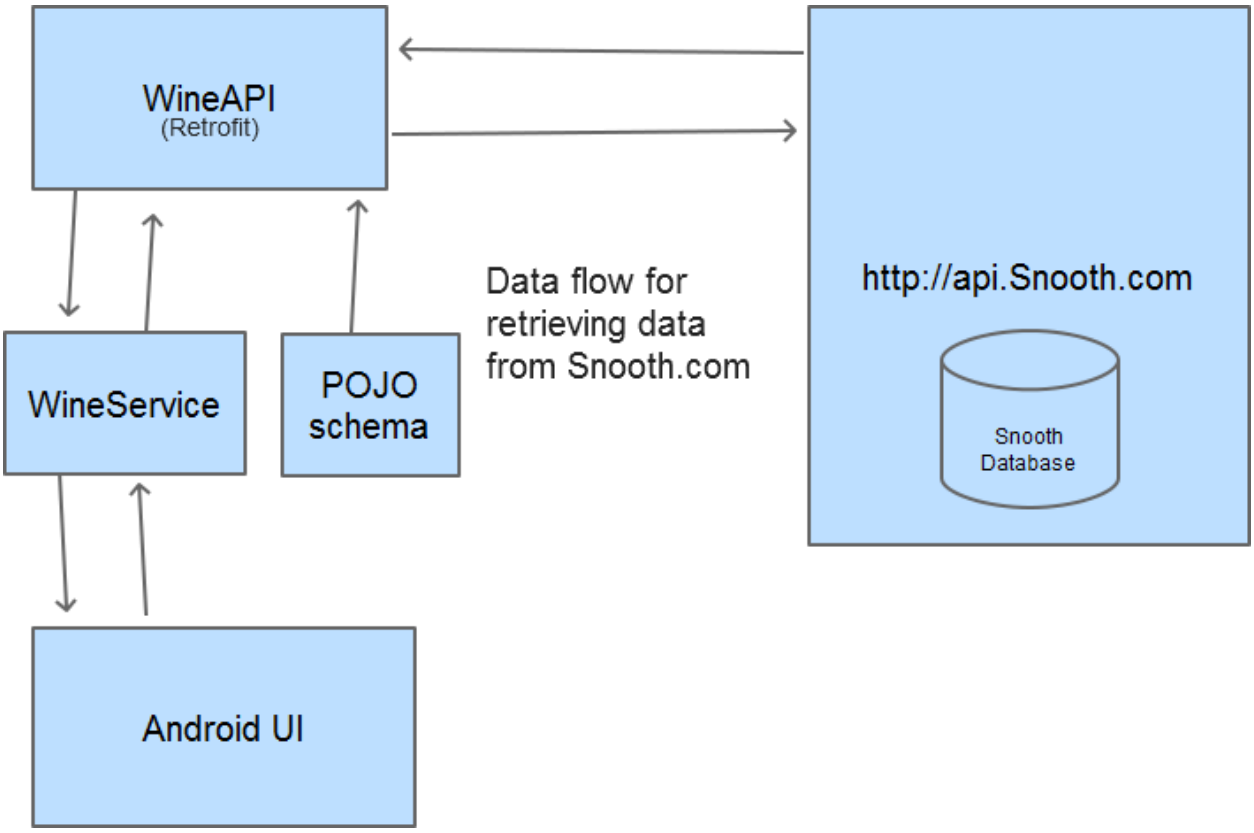


DialogFragment screen – will show a list of all available regions to choose from.

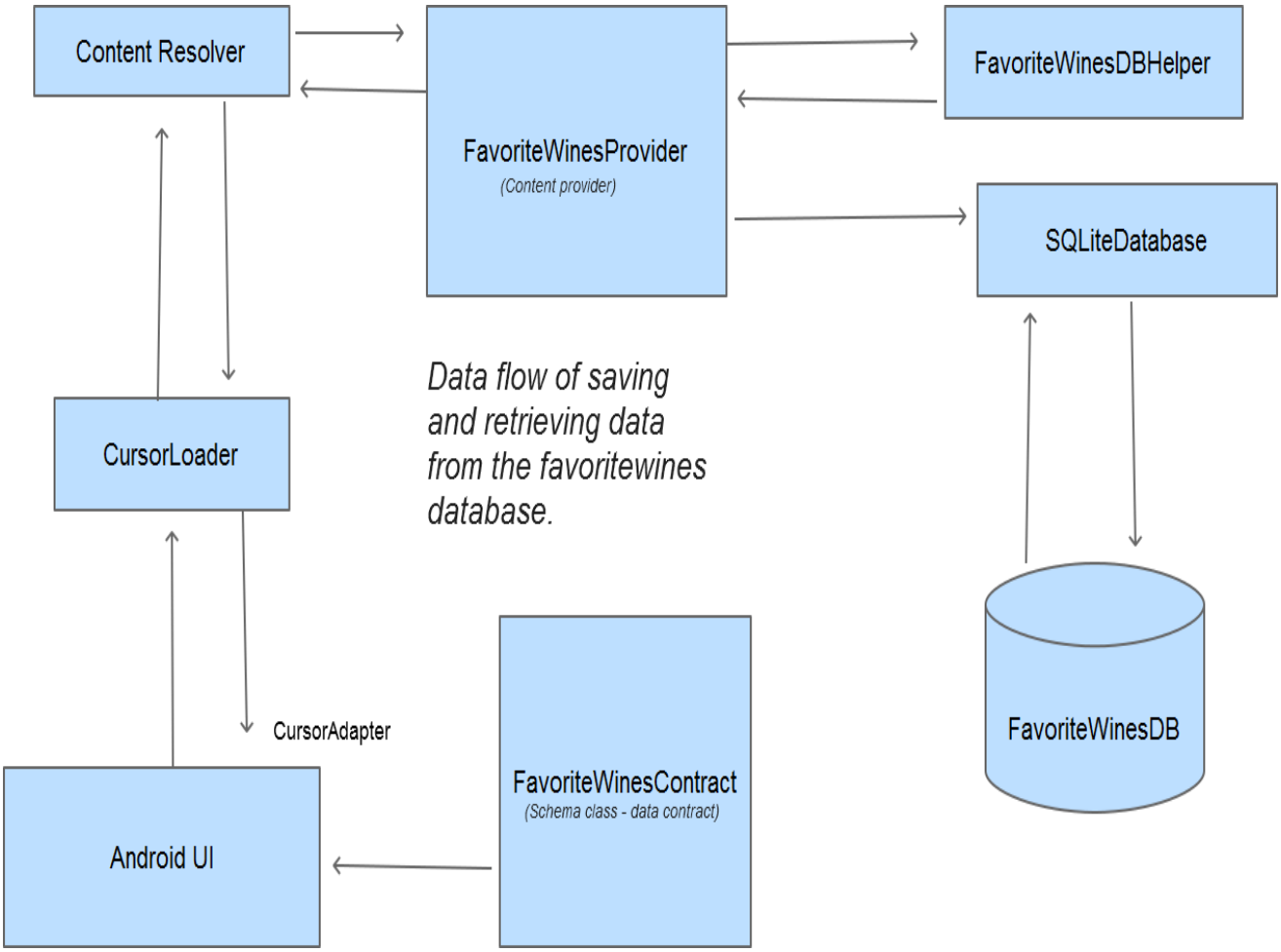
Key Considerations

How will your app handle data persistence?

The wine data will be coming from an external API called snooth, which will require an API key (You can find more information here: https://api.snooth.com/).
I will be using Retrofit with Gson converter to make the API call using a pojo schema.



The app will save some data internally using Content Provider and an SQLite database.


Content Resolver
FavoriteWinesProvider
(Content provider)
FavoriteWinesDBHelper
SQLiteDatabase
Data flow of saving and retrieving data from the favoritewines database.
CursorLoader
CursorAdapter
FavoriteWinesContract
(Schema class - data contract)
FavoriteWinesDB
Android UI

Database Documentation
Database name: favoritewines.db3
Table Name: wine

PK	id	Autoincrement
	Region	String
	Varietal	String
	Name	String
	Code	String
	Winery	String
	Price	string
	Vintage	String
	Link	String
	Image	String
	Snoothrank	String

```
/* DDL information for - wine */
--------------------------------
CREATE TABLE `wine`
-- This table created by SQLite2009 Pro Enterprise Manager
-- Create date:2016-08-29 08:10:03
(
    wineid INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
    region TEXT NOT NULL,
    varietal TEXT NOT NULL,
    name TEXT NOT NULL,
    code TEXT NOT NULL,
    winery TEXT NOT NULL,
    price TEXT NOT NULL,
    vintage TEXT NOT NULL,
    link TEXT NOT NULL,
    image TEXT NOT NULL,
    snoothrank TEXT NOT NULL
)
```

The user settings will be saved using SharedPreferences.

Describe any corner cases in the UX.

- If the wine search returns nothing a toast message will inform the user that the search criteria has returned 0 and will remain on the detailed search screen.
- User will be informed if network is not available either in the toolbar or status bar but will have access to favorites.
- If a failure occurs while saving or removing from the favorites database a toast message will inform the user of the failure.

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

- Retrofit2.x (http://square.github.io/retrofit/): I will be using it as a type-safe HTTP client to retrieve data from an external API
- Converter-gson (https://github.com/square/retrofit/tree/master/retrofit-converters): This is used to convert the file format when using retrofit. It is a converter which uses Gson for serialization to and from JSON.
- Picasso (http://square.github.io/picasso/): Using this library will make it much easier to download images and not having to worry about memory and disk caching.
- Firebase Analytics (https://firebase.google.com/docs/analytics/): Keep track of app usage
- On Orientation changes I will be using onSaveInstanceState(), onViewStateRestored(), I will make my data class parceable in order to write to and restore from a Parcel, this will help me pass data between components.

Describe how you will implement Google Play Services.

In your build.gradle file module:app enter the following:

```
Apply plugin: 'com.androidapplication'
        . . .

Dependencies{
        Compile 'com.google.android.gms:play-services:9.6.1'
        Compile 'com.google.firebase:firebase-core:9.0.2'
}
```

Once that is done save and sync project.

Describe which Google Play Services you will use and how.

Firebase Analytics - It is now recommended. Analytics will enable me to keep track of what devices are using my app and what parts of my application are being used and how often.

Location + context – The user will have the ability to get a list of the closest wine shops according to current location. Example of usage: When we go to a restaurant and didn't realize that wine is not sold at the restaurant but you can bring your own. The app will give a list of the closest wine stores at your current location. Terrorist

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.

Task 1: Project Setup

Write out the steps you will take to setup and/or configure this project. See previous implementation guides for an example.
Project name: Wineoisseur

In order to get the Wineoisseur app to work correctly you will need to register for the snooth api key which is free of charge. You can do this at the following site: https://api.snooth.com/
Once you have your api key you will need to enter it in the gradle.properties file:
WINE_API_KEY = "Enter your Snooth api key here"

You will need to configure the libraries in the gradle.build file located in the app directory they should be targeting the latest libraries:

```
apply plugin: 'com.android.application'

dependencies {
        compile 'com.squareup.retrofit2:retrofit:2.0.1'
        compile 'com.squareup.retrofit2:converter-gson:2.0.1'
        compile 'com.squareup.okhttp3:okhttp:3.2.0'
        compile 'org.glassfish:javax.annotation:10.0-b28'
        compile 'com.squareup.picasso:picasso:2.5.2'

        compile 'com.google.android.gms:play-services:9.6.1'
        compile 'com.google.firebase:firebase-core:9.0.2'
}
```

At this point I know the structure of the database and know what type of information I'm getting back from the API call and or the database. I can now correlate what information is going on what screens.
I also have a good idea on what my package sub directories will look like and will create the following:
Com.dzartek.wineoisseur.

adapterviews
apicall
database
datamodel
contentprovider
fragments
pojomodel
service
widget

Task 2: Implement UI for Each Activity and Fragment

Build the following UI's using material design concepts

- activity_main_wine.xml ActivityMainWine.java
- fragment_wine_search.xml FragmentWineSearch.java
- custom_wine_row.xml WineAdapter.java
- dlgFragment_wine_region.xml DlgFragmentWineRegion.java
- fragment_settions.xml FragmentSettings.java
- widget_wine.xml WidgetWine.java

Task 3: Implement API call & Service

- WineAPI.java
- Create pojo schema using jsonschema2pojo located at http://www.jsonschema2pojo.org/
- WineService.java
- Set up unit testing possibly using espresso

Task 4: Build SQLite Database

- Create database contract class, FavoriteWinesContract.java
- Create database helper class, FavoriteWinesDBHelper.java
- Create content provider, FavoriteWinesProvider.java

- Implement unit testing

Task 5: Create the Wine Widget

- Create WidgetWineProvider
- Create WidgetWineService

Task 6: Implement Google play services

- Implement Google locations

Task 7: Implement Firebase analytics

- Create what events you would like to keep track of.

Task 8: Develop Tablet UI

Develop the app so that different screen sizes are supported.

- Create different screen size folders
- Adapt the code to work with tablet dimensions

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**"