How to Use this Template

1. Make a copy [Select All → Copy → Paste into new document]
2. Name your document file: “**Capstone_Stage1**”
3. Replace the text in green

GitHub Username: alexpfx

My Beer Collection

Description

This is an app for beers lovers. The purpose of this application is to allow you to remember all the beers you have ever tasted in your life. With this app you also have in your hands information about thousands beers from all over the world.

Intended User

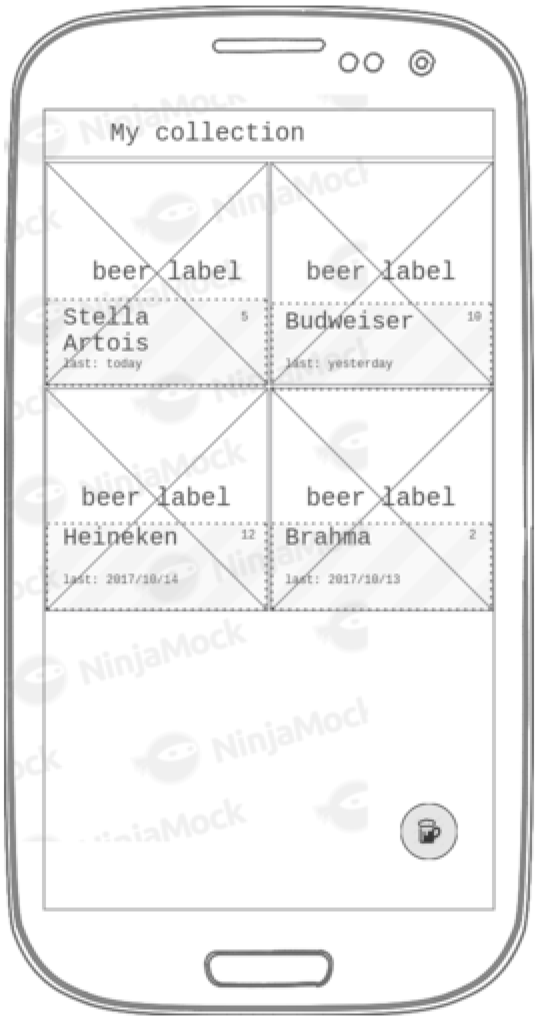
For all beer lovers and collectors, who wants to maintain a historical record about beers they've ever tasted. Or for anyone who is curious to meet different beers from around the world.

Features

- Allow users to maintain a collection of beers, remembering beers they've already tasted
- Allow users to search for beers
- Allow users to get detailed information about beers, like ABV (Alcohol by Volume), IBU (International Bitterness Unit), SRM (Standard Reference Method), Brewery who makes that beer, the beer style, the serving temperature, among others...
- Allow users to have a widget on the homescreen of their cell phone displaying their beer collection

User Interface Mocks

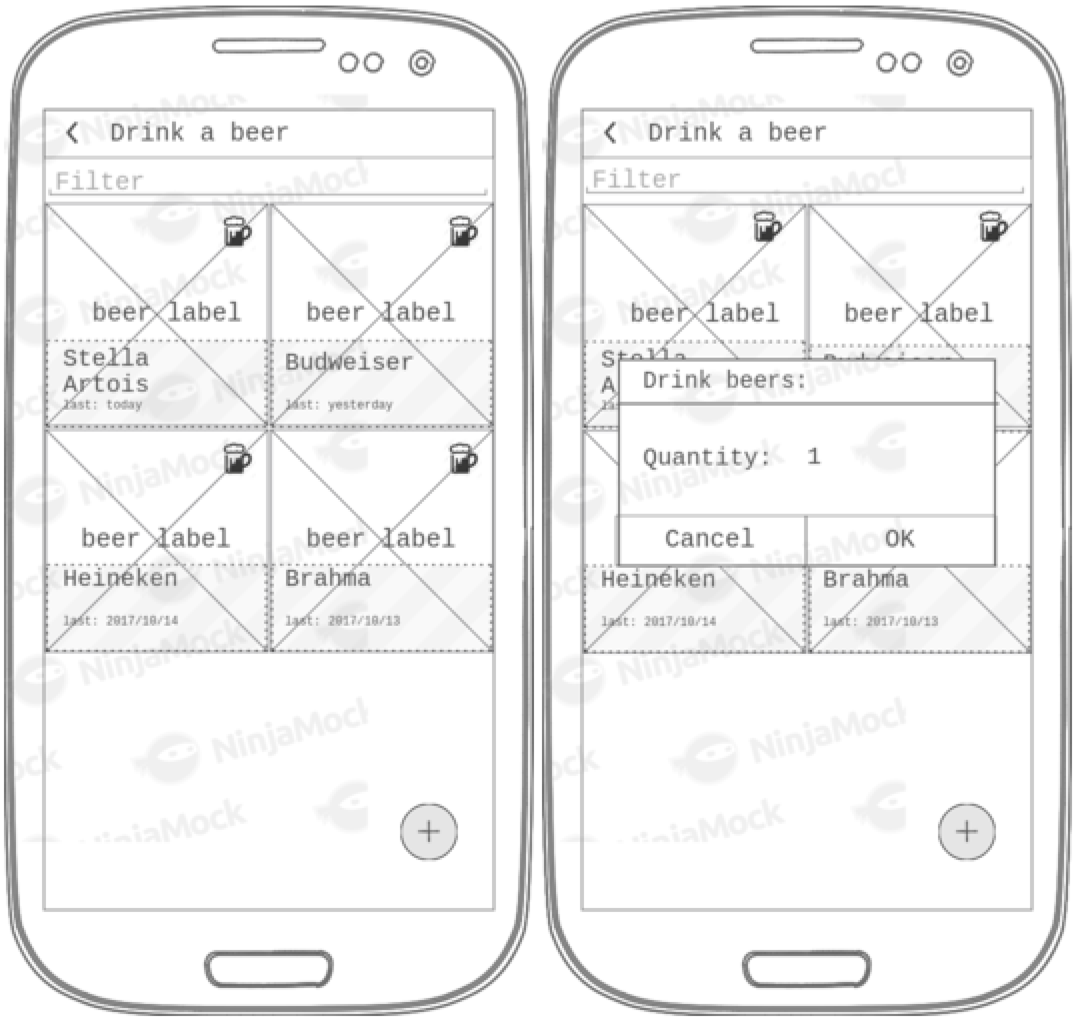
My Collection Screen



When user starts the App this screen is shown. It shows the user's beers collection, which are the beers he register previously. The screen shows a grid where each cell is backgrounded with the beer label. Each cell shows the beer name and the last date one beer of that label was consumed. And the quantity of beers of that label that was consumed by the user.

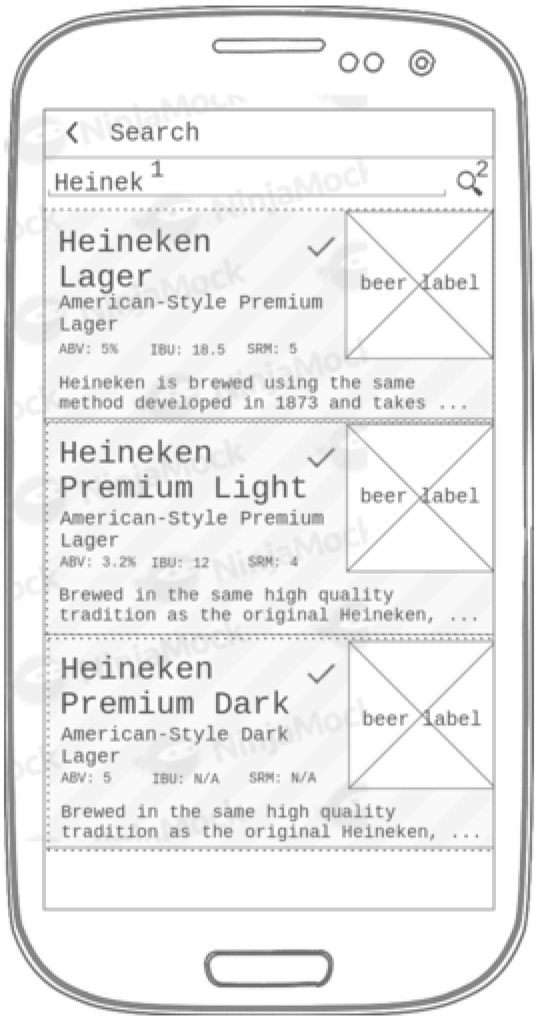
At the bottom of the screen, there is a FAB button that takes the user to the screen "Drink a Beer".

Drink a Beer Screen



This screen allows user to register that he drank a beer that he already has in his collection. A grid is showing and the user can use the filter at the top. When user clicks on the mug a dialog is opened allowing it to enter the quantity of beers that was consumed. If the beer he wants to register is not in his collection yet, he should to add it to collection, by pressing the FAB, that will take the user to “Search Screen”.

Search Screen



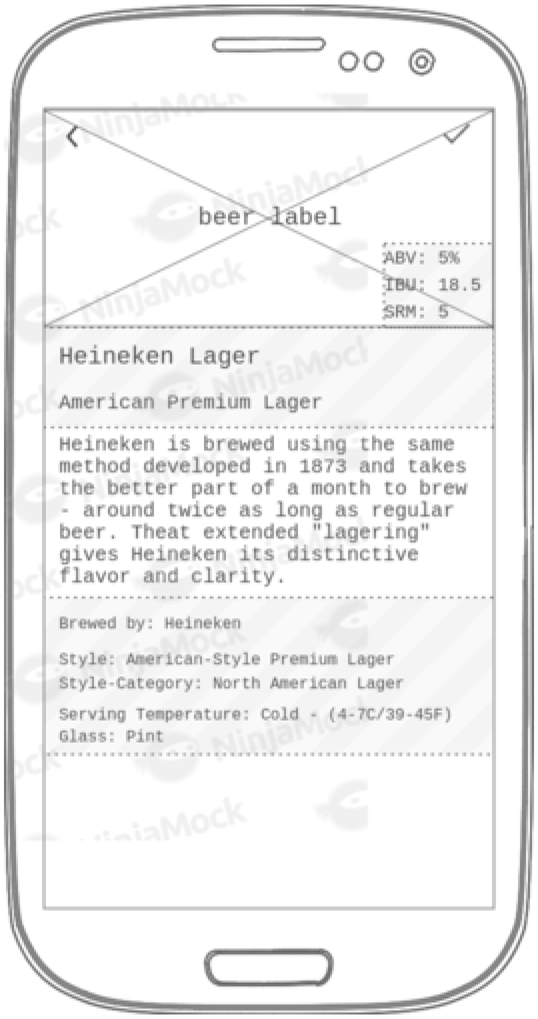
Heineken Lager
American-Style Premium Lager
ABV: 5% IBU: 18.5 SRM: 5
Heineken is brewed using the same method developed in 1873 and takes ...

Heineken Premium Light
American-Style Premium Lager
ABV: 3.2% IBU: 12 SRM: 4
Brewed in the same high quality tradition as the original Heineken, ...

Heineken Premium Dark
American-Style Dark Lager
ABV: 5 IBU: N/A SRM: N/A
Brewed in the same high quality tradition as the original Heineken, ...

This screen allows the user to search for a beer in database. User has to type the search expression (1) and then press the search button (2). When results come to screen it will be added to a list. Each list item shows the information about a beer found by the search. Then when the user found the desired beer, he can click on the accept button and the beer will be added to collection and the user will be taken back to the “Drink a Beer” screen, where he can inform that he have consumed that beer. if the user clicks elsewhere on the list item he will be taken to the Beer Detail Screen.

Beer Detail Screen



Heineken is brewed using the same method developed in 1873 and takes the better part of a month to brew - around twice as long as regular beer. Theat extended "lagering" gives Heineken its distinctive flavor and clarity.

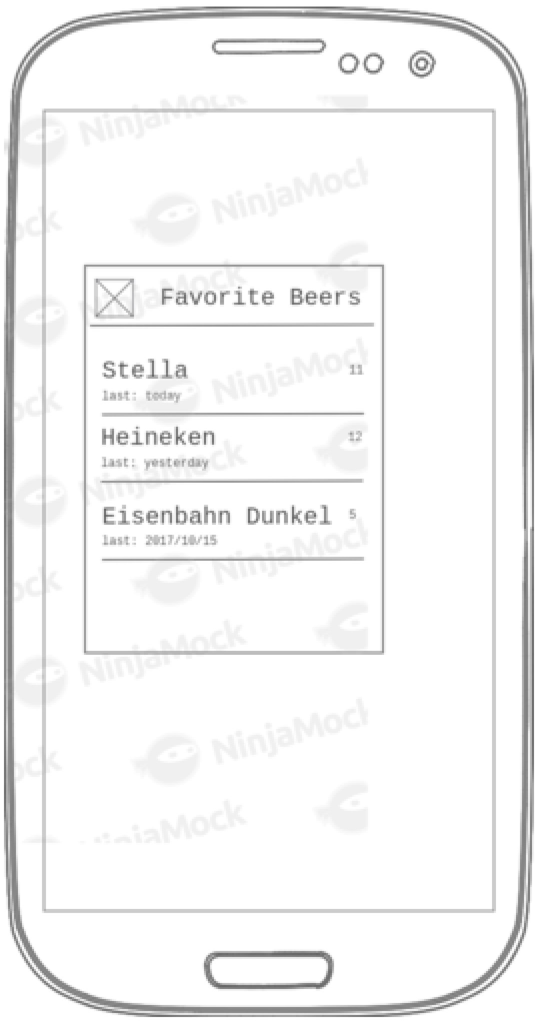
Brewed by: Heineken

Style: American-Style Premium Lager
Style-Category: North American Lager

Serving Temperature: Cold - (4-7C/39-45F)
Glass: Pint

This screen shows detailed information about a beer. As on search screen, there is a accept button, which when pressed will take the user to “Drink a Beer” screen, where he can inform that he’ve consumed that beer.

Widget Screen



This is the screen for the App’s Widget. It shows beers in user’s collection ordered by the date it was consumed.

Key Considerations

How will your app handle data persistence?

App will user Firebase Realtime Database to store data fetched from a REST API for the purpose of cache and to save local app data. App will use shared-preference to store local user settings.

Describe any edge or corner cases in the UX.

- When user enter the “My Collection” screen and the user’s beer collection is empty, a message is shown instead of the grid.
- In the Drink a Beer Screen, when user clicks on the beer mug to register a beer that he’ve consumed, a dialog is showed showing the default quantity: 1. If he clicks that number, a NumberPicker is opened allowing the user to inform the quantity.
- Data providers: At first, the information about beers will be provided by a online Rest Service: BreweryDb (http://www.brewerydb.com). It is possible that new services to that purpose are added or even that this service is replaced as needed.
- The UX components will not access Rest Endpoints directly. At first, the data will be fetched and stored into a Firebase Database, where online capabilities flag will be enabled, allowing offline access. Then, App components will access data from that database instance.
- Internet connectivity: when the user try to access some data that is not cached in firebase database and there is no internet connection, the APP will display a message notifying the situation.

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

- Retrofit
    - used for consuming the beer REST API
- Gson
    - Java serialization/deserialization library to JSON Objects
- Picasso
    - used for loading and caching images.
- ButterKnife
    - allows views inject through annotations
- Dagger2
    - Used for dependency injection
- Event Bus
    - event library
- Android-iconics or Android-iconify:
    - facilitates the management and use of vector icons
- Palette Api:

  - used to extract colors from images and use them in layouts
- Espresso
  - To build user interface testing
- Hamcrest
  - Provide a collection of Matchers for tests

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

- Firebase Realtime Database
  - It will be used for database management.
- Firebase Google Analytics
  - To collect and analyze App usage data.
- BreweryDb Rest api (http://www.brewerydb.com/developers/)
  - It's a REST based API that will provide data about beers.

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

- Create a new empty project in Android Studio, setting the SDK minimum version to API 18 (Kitkat).
- Add required libraries to gradle build file.
- Create a GitHub repository and push an initial commit
- Setup and test a Firebase Database account
- Setup and test a Firebase Google Analytics account
- Create an account and request an API key from brewerydb.com

Task 2: Implement Rest API and Data Layer

- Implement a class that uses Retrofit to communicate with brewerydb web services, providing methods to access each endpoint needed by App.
- Create a IntentService to handle the search for beers

- Create database layer in Firebase

Task 3: Create the architectural model

- Decide if dependency injection using Dagger2 will be used.
- Decide which architectural design model - like MVP (Model View Presenter) MVVM (Model View - View Model) or traditional android Model View architectural pattern - will be used.
- Once this is decided, create the skeleton (if necessary) to support the architecture and dependency injection.

Task 4: Implement UI for Each Activity and Fragment

- Build UI for My Collection Screen
- Build UI for Drink a Beer Screen
- Build UI for Search Screen
- Build UI for Build Detail Screen
- Build UI for Widget Screen

Task 5: Connect Data layer to UI

- Implement the glue between the UI Interfaces and the data layer.

Task 6: Test

- Create Espresso tests to handle the main user interactions with the UI

Task 7: Sign App

- Implement signing configuration
- Generate a signed APK

Submission Instructions

- After you've completed all the sections, download this document as a PDF [File → Download as PDF]
    - Make sure the PDF is named "**Capstone_Stage1.pdf**"
- Submit the PDF as a zip or in a GitHub project repo using the project submission portal

If using GitHub:

- Create a new GitHub repo for the capstone. Name it "**Capstone Project**"
- Add this document to your repo. Make sure it's named "**Capstone_Stage1.pdf**"