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: fbartnitzek

TasteEmAll

Description

Review and rate every new drink you try - and just try the good ones again.

Too many apps to chose where to note your favorite beers, but none for the other drinks you like?
Lots of ratings in those apps from guys who seem to just like variations of water?

So you tried some and decided to use lists instead?

And in the end you're tired of using multiple lists with hundreds of entries, one for each kind of drink? - Wouldn't it be great to know when and where you tried them without typing and searching and typing? And publish your review to the public if - and only if - you like?

Here is a solution which hopefully will at least work for me :-).

Intended User

Coffee addicts, craft beer fans, amateur sommeliers and whisky lovers - travelers welcome.

Features

Review the following types of drinks:

- beer
- coffee
- wine
- whisky
- generic [user defined would be great - let's see if that works …]

Features:

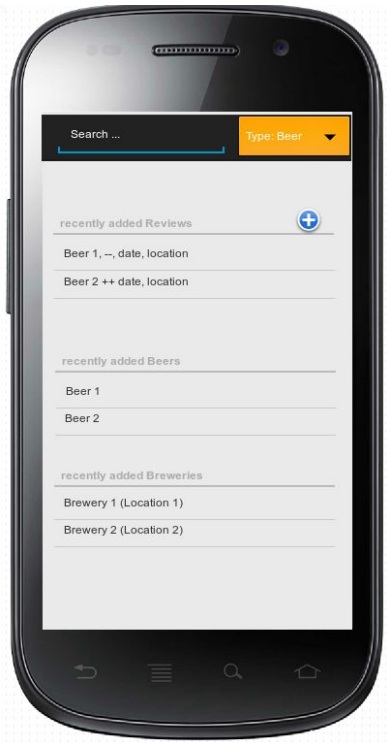
- Add a producer (brewer, roaster, winery, distillery, producer) with name, location, url, pic
- add a drink with a certain type and name, style, ingredients and pic
- add a review with rating, date, location and description of different aspects (smell, taste, look, …) and a pic
- take a picture of the brewery, that amazing bottle or the pub where you tried it
- filter by type
- sort by rating, style and location
- export and import files to json / xml => backup, optionally xslt for blog-entries/…
- Later possible: matching website with multiple users and recommendations of friends ...

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.

In each screen a toolbar is used, which provides a filter for the type of drink (in the MainScreen Beer is selected) and a text or search on the left - depending on the screen.

MainScreen - recently added reviews

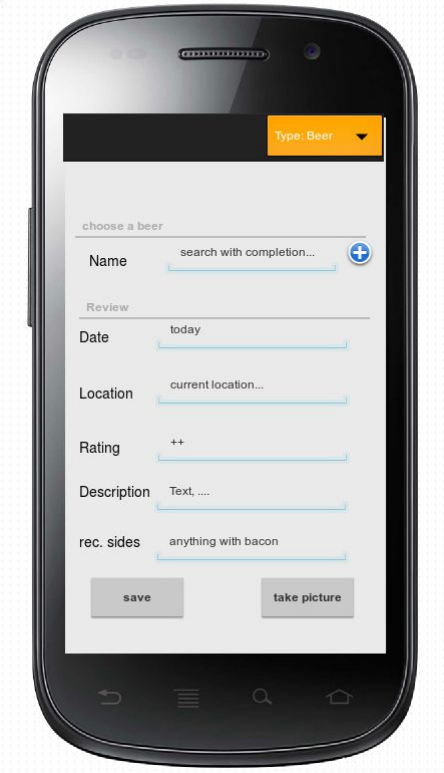


The MainScreen presents the latest additions to the app:

- recently added reviews
- recently added drinks (if type beer is selected: beers)
- recently added producers (if type beer is selected: breweries)

The main action on that screen is adding a review (which is kind of the purpose of the app) - so a floating action button (FAB) is used for that.
Replace the above image with your own mock [click on the above image, then navigate to Insert → Image…]
Provide descriptive text for each screen

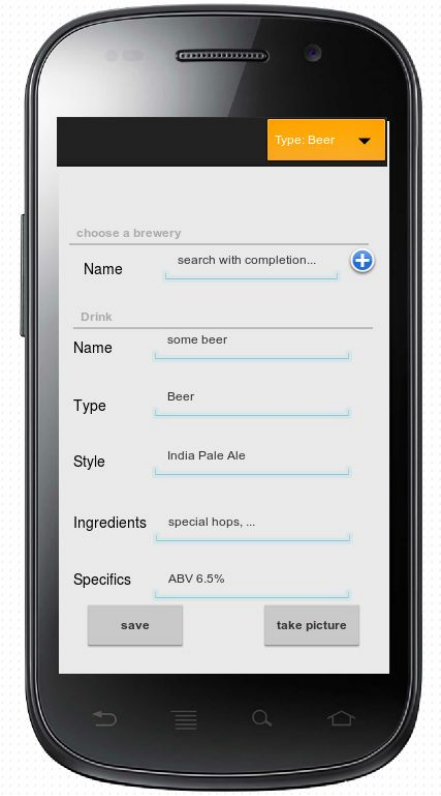
AddReviewScreen



Here a user can add a review for the specified kind of drink.
The most important part of the review is selecting the drink for the review (which will be searched with a kind of autocomplete).
If the drink is not found - the FAB next to the search should be used to add the drink first.
When the drink is found (or filled at returning from AddDrinkScreen) the other views shall be filled.
Date is pre-filled with the current date, location with the current location (GooglePlayServices - getLastLocation).
The rating values are pre-defined (++, +, 0, -, --) and a help/legend should be presented (kind of mouse-over).
A description of the taste and some other fields can optionally be filled.

At the bottom a picture can be added of the review (f.e. the bar you tried it). If an explicit save button or an up-button with a “magical add” by default works better - let’s see in some tests later...

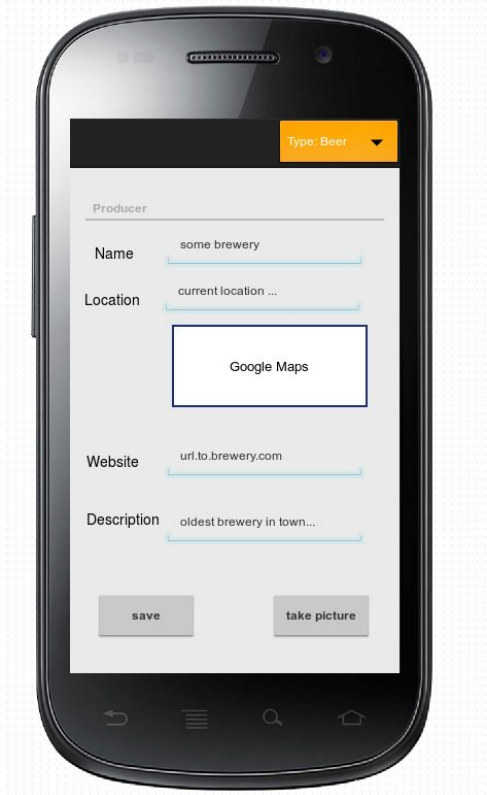
AddDrinkScreen



The drink is the base of every review - in that screen a user can add a drink for the specified type of drink (in that case beer). Every drink is produced by a producer, which must be chosen or added (like the drink in the review).

When the producer is found (or filled at returning from AddProducerScreen) the other views shall be filled.
The name of the drink identifies the drink (primary key in combination with the producer). The type is pre-filled by the filter in the toolbar.
The style should be filled with autocompletion, but may be extended with new values. The remaining fields are optional and need to get different help-texts per drink-type.

AddProducerScreen



The producer is the base of every drink - in that screen a user can add a producer, which may produce any kind of drink (but most likely will just produce one).
It needs to have a name and a location.
The location text might be used with auto-completion and must be validated (if network is available) with google maps.
The result of the maps query should be shown in the image below and the location-entry is changed to the formatted address of the query.

Optionally the url of the producer can be inserted (highly recommended) and a description can be given.
As usual an image of the producer (like a picture of the brewery) can be taken.

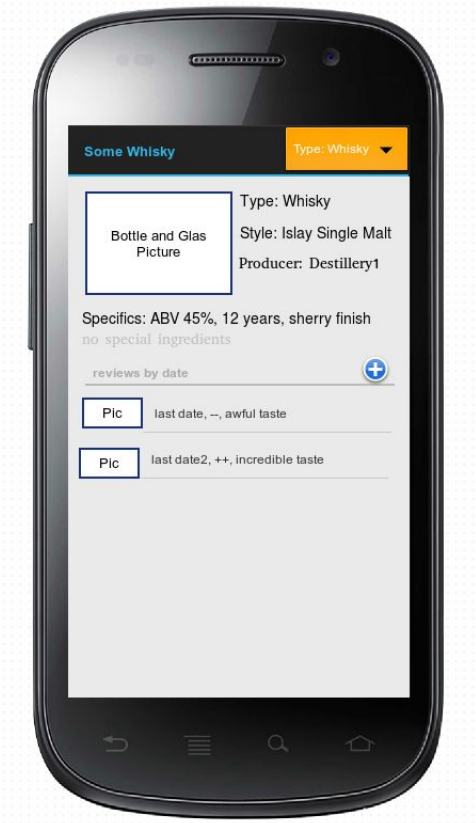
ShowProducerScreen

This Screen will be opened when an producer-entry is clicked in the MainScreen. It presents the name of the producer in the toolbar, a picture of the producer below (if an adapter is present for this type, this picture may be downloaded from an external source, instead of a picture made by the user). The formatted location and picture from google maps is shown next to it (might be better to change the layout depending on the screen width…).
The description and the website of the producer are listed below.

At the bottom the drinks of this producer are presented with their average rating. At this point the user may also add a new drink with the FAB (with pre-filled producer).

It might be useful to allow another search-view (for breweries like brewdog with dozens to hundreds of drinks) and it might also be useful to show “similar producers in that area” - but at first let’s start simple.

ShowDrinkScreen

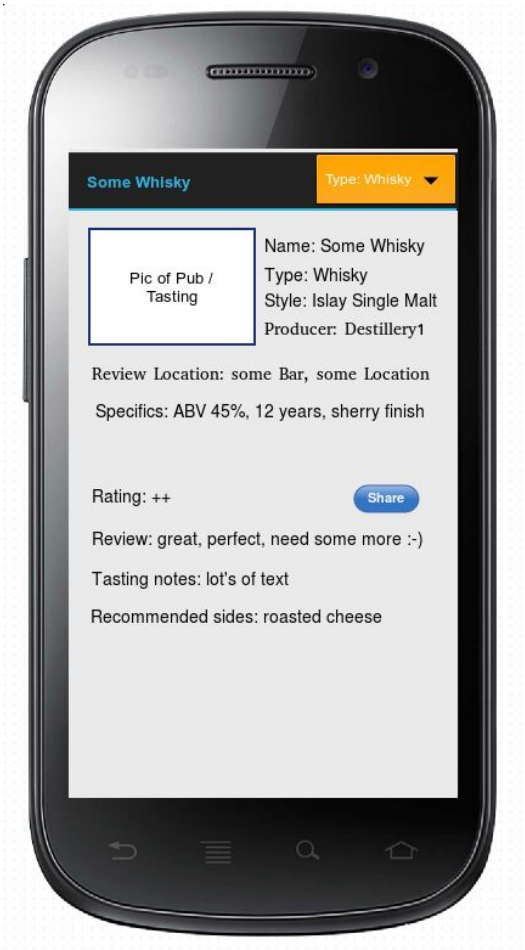


This Screen will be opened when an drink-entry is clicked in the MainScreen or in the previous ShowProducerScreen. It presents the name of this drink in the toolbar. A picture of the drink (if available, optionally gained by an adapter) is presented at the most prominent position. Next to it is the type (f.e. Whisky), the style (just a text) and the producer (on click opens the previous screen) is listed.
Below these identifying characteristics, the optional specifics and ingredients are listed (empty = grey).

At last the reviews for this drink are listed (I guess by date may work best). If the user found this drink in the app and want’s to rate it right away, the FAB-button will open the AddReviewScreen (with pre-filled drink).

More searching seems to be useless in single-mode (maybe a website later), but “similar drinks” seem to be a good idea for later.

ShowReviewScreen



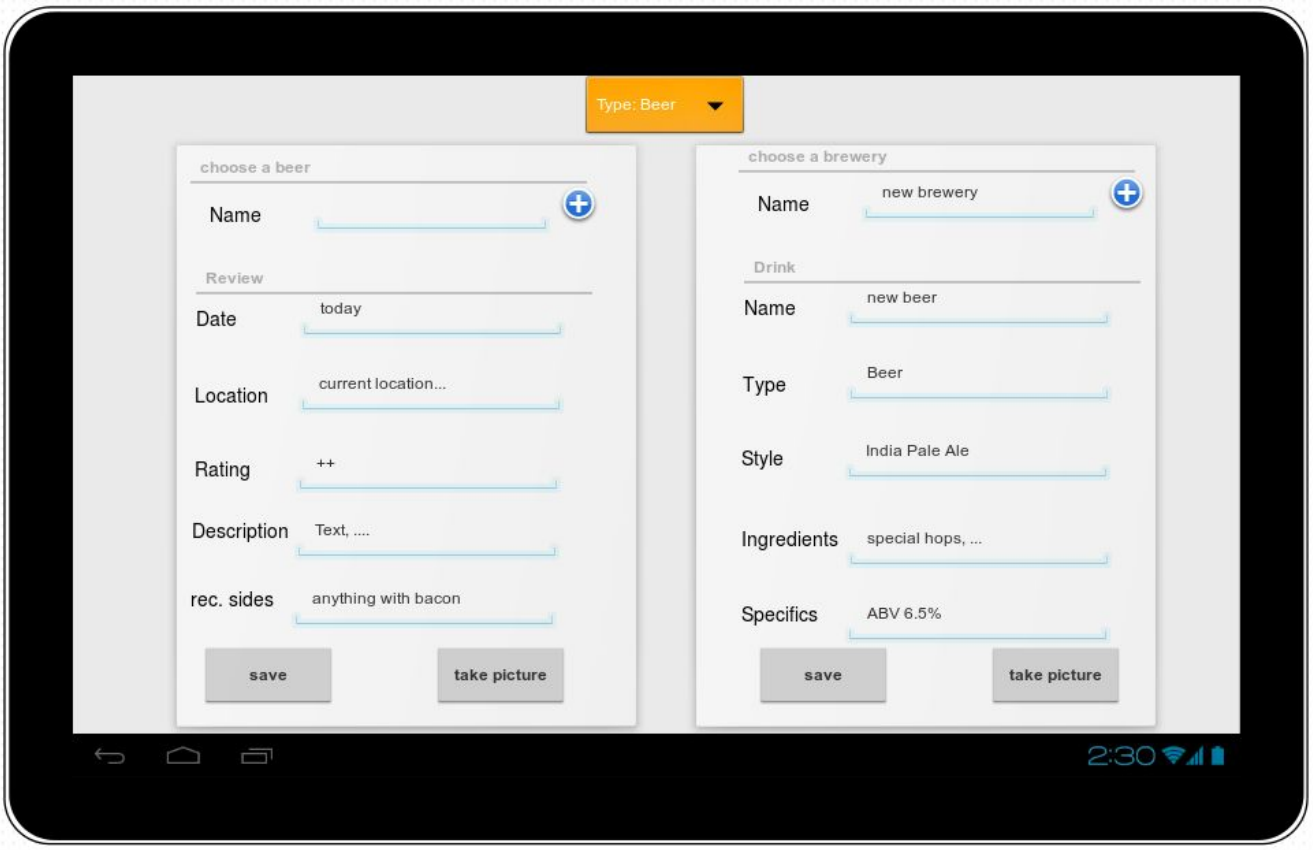
The result and goal of the app - a rating for the drink on hand, so the user can decide if he's trying & buying or avoid it at all cost and better chose another drink.

The upper part of the screen is filled with data to identify the drink / producer (a click on name or producer will start the matching activity to show them) and an optional picture of the tasting.
If no picture of the tasting has been made, the review location may be displayed (f.e. at home, in a bar, part of last holiday or at a friend's home).

Below that identifying data is the rating itself listed (the date should also be displayed...).
A more detailed description and some other optional attributes might seem useful sometimes (like in the picture) - so they should be greyed-out if not needed.

A share-FAB seems to be the only useful action here, maybe some others will follow later (like upload the review to a certain website).

Tablet Screen



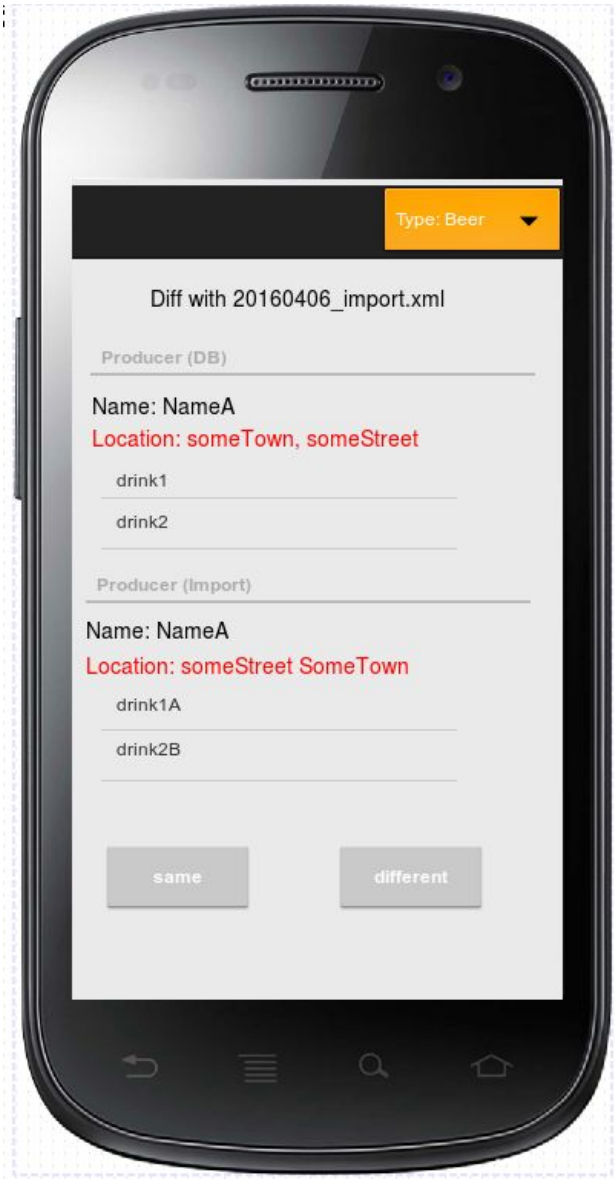
On wider devices it is possible to show the related classes next to the review.
If the user adds a new review and the drink is not yet present, the “add drink screen” opens next to it.
If on the other hand the drink is already present, the “show drink screen” is shown next to it.
On creating the drink with a producer has to be selected - so the views scroll out to the left and at the right appears the “show producer screen” or “add producer screen” (depending on the existence of the producer).

More screens - optional for later versions

Report / Search screen

After inserting some entries with csv / json import a configurable report screen would be interesting.
If that is to complicated to show on a smartphone - maybe it’s “just” a predefined sorting like “best ratings”, “worst ratings”, …
Reports like in the financisto app might also be fascinating - but more complicated with less quantifiable data.
Nonetheless - interesting for later versions of the app.

Diff screen



when importing lists of friends (for producer and drinks)...
might also need another search and "replace/update"...

Could be to complicated to use on a mobile screen.
Then a file-based comparison should be used…

Some tests may follow in later versions...

Add as many screens as you need to portray your app's UI flow.

Key Considerations

How will your app handle data persistence?

Use a local Content Provider with generic tables and type-entries.
Import and Export of existing user-lists based on a specific csv-format.
Export and Import of json / xml files for backups or further usage through the user (later idea: provide a sample xslt-file to html / markdown) / import entries from friends - user name in review would be necessary.

Later: additionally use some predefined API (f.e. http://www.brewerydb.com for the type beer).
Cache the results locally and use an AsyncTask to request them.
Other APIs may follow in later versions (maybe as “addons”).

Describe any corner cases in the UX.

Expected back-button-behaviour:
When adding a producer, while adding a drink, while adding a review, the back-button returns to the previous step:
AddProducer -> AddDrink -> AddReview -> MainScreen

Changing drink-type:
The user can only see drinks and reviews based on the selected drink-type.
A filter-type ‘all’ is possible and maps to the generic drink-type in the AddDrink Screen.
If the attribute type in the AddDrink Screen is changed from the predefined type (based on upper right corner) to another type, the filter in the upper right corner is also changing - so the user can see the resulting drink afterwards.

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

Google Play Services to get LastLocation, reverse geocode location to formatted address and show it in a map.

Libraries to import and export data

- CSV: org.apache.commons:commons-csv
- JSON: com.google.code.gson:gson
- XML: XMLPullParser (seems to be part of android and no external library

Optional photos: seems to be well integrated in android
http://developer.android.com/training/camera/photobasics.html

Picasso or Glide to handle the loading and caching of images.

Optionally Brewerydb.com to request pictures and descriptions of drink-type beer. - later

Maybe some more...

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.

subtasks:

- Configure libraries
- Create submodule for data/persistence (so it may be reused without the android app)

If it helps, imagine you are describing these tasks to a friend who wants to follow along and build this app with you.

Task 2: Implement DatabaseProvider prototypical

subtasks

- Create tables for Producer, Drink and Review with basic test-cases for inserts and selects
- Use submodule data for attribute-definitions (as a dependency)
- Create ContentProvider to insert, update and delete data with basic test-cases

Task 3: Implement UI for Main Activity and Producer Activities

subtasks:

- Create simple main activity layout
- Build UI for MainActivity with ‘add producer’ button which invokes matching activity
- Create simple add producer layout (name, location, save button)
- Build UI for AddProducer with name and location attributes and working save button
- Update MainActivity UI to show Producers with ProducerAdapter for ContentProvider

Task 4: Implement UI for Drink Activities

subtasks:

- Replace MainActivity UI ‘add producer’ button with ‘add drink’ button which invokes matching activity
- Create simple add drink layout (producer, name, type, save button, add producer button)

- Build UI for AddDrink with text-search on producer name (if nothing found - add button revealed), name and type attributes and working save button
- Update MainActivity UI to additionally show Drinks with DrinkAdapter for ContentProvider

Task 5: Implement UI for Review Activities

subtasks:

- Replace MainActivity UI ‘add drink’ button with ‘add review’ button which invokes matching activity
- Create simple add review layout (drink, date, rating, save button, add drink button)
- Build UI for AddReview with text-search on drink name (if nothing found - add button revealed), date and rating attributes and working save button
- Update MainActivity UI to additionally show Reviews with ReviewAdapter for ContentProvider

Task 6: Implement UI for Show Detail Activities

subtasks:

- Create simple show review layout (reuse from add review?, with share button)
- Build UI for ShowReview with attributes producer-name, drink-name, own attributes, reuse ReviewAdapter…?
- Create simple show drink layout (reuse from add review?, with share button)
- Build UI for ShowDrink with attributes producer-name, own attributes, list of reviews for drink, reuse ReviewAdapter and DrinkAdapter…?
- Create simple show producer layout (reuse from add review?, with share button)
- Build UI for ShowProducer with own attributes, list of drinks for producer, reuse DrinkAdapter and ProducerAdapter…?
- Add onClick-invocation in all 3 Show-Activities to other 2 (for ProducerName and DrinkName in ShowReview)

Task 7: Implement import and export for data

subtasks:

- Implement export to csv (atomic = class-based and list-like = all in one list)
- Implement export to json (atomic = class-based)
- Implement import from csv and json
- Implement testcase which imports data from csv and exports to json and the other way around
- Import-on-conflict-behaviour: add with conflict_prefix to resolve later...

Task 7: Add location support

subtasks:

- Add location attribute to Review, and Review UIs
- Add location-permissions to the app
- Use Google Play Services in AddReview to fill the location-attribute with the lastLocation
- Add error-handling if not available...
- Use Google Play Services to “optimize” the Producer-Location while saving and save formattedAddress (with error-handling)

Task 8: Add remaining missing attributes and filter

subtasks:

- Review: description, user-name (pre-filled by preference)
- Drink, Producer - see screenshots
- Working filter in upper right corner for each Show Screen

Task 9: Add Widget

subtasks:

- Create layout file (latest 5 reviews and add button)
- Implement List-Widget
- OnClick opens matching ShowReview-Screen
- Add opens AddReview Screen

Task 10: A11y and tablet-AddReviewScreen

subtasks:

- Add content-description to all textviews
- Check RTL-attributes
- Add AddReviewScreen for tablets, which opens Drink and Producer next to it with existing fragments

Optionally Task 11: Add map screen

subtasks:

- Implement new ShowMap Activity which uses google maps view
- Show Producers on map with one icon and ReviewLocations with another
- Add onClick-behaviour (infos and button for external navigate-intent)
- Error handling if network is not available

Optionally Task 12: ShareActionProvider and beautify UI and transitions

subtasks:

- Add camera-functionality for each AddScreen
- store optional picture for Producer (building), Drink (bottle) and Review (glas)
- show them in ShowScreens
- Use element transitions for name-transitions (f.e. ProducerName.click in ShowReviewScreen)
- Update Share-button to share image (if available) with review description / rating

Optionally Task 13: Implement Beer-API adapter

subtasks:

- Implement Adapter for API http://www.brewerydb.com for the type beer
- In AddReview use API to query drink and producer - if selected add drink and producer to local database
- Load pictures if existing

Add as many tasks as you need to complete your app.

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**"