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: antaranin

SmartApparel

Description

An easy to use app, that not only collects weather information from the web but also transforms it into useful information. SmartApparel will suggest you what to put on when going out, based on the current weather conditions, so that you don't have to worry about getting wet or cold because you forgot a rain coat.

Problem:
Currently available weather apps, while powerful do not help in interpretation of the information they provide, leaving it up to the user. Thus they force theirs users to interpret the information on their own.
Proposed solution:
Create an app, that not only accesses weather information but processes it as well, providing user with meaningful suggestions, that require no further thought (current aim is to suggest clothing for the day).

Intended User

No specific target group. Technically everyone with a smartphone and interest in saving some time.

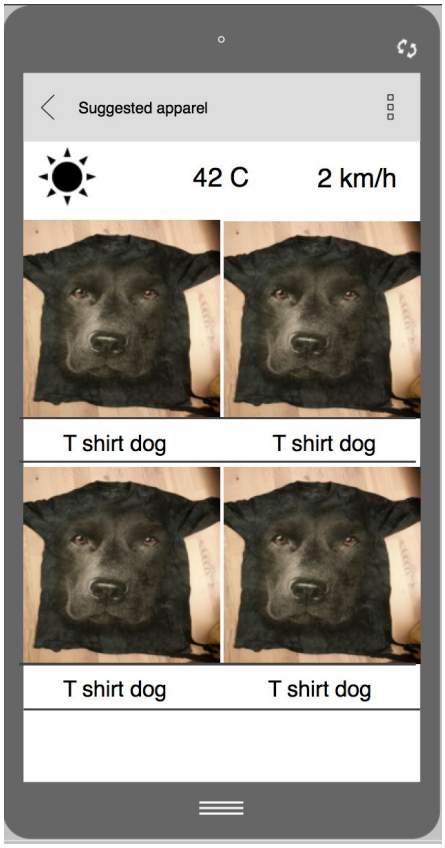
Features

- Extracts weather information from the Internet
- Keeps track of user position
- Takes pictures and crops them if necessary
- Saves data
- Displays widget
- Displays Google Ads

User Interface Mocks

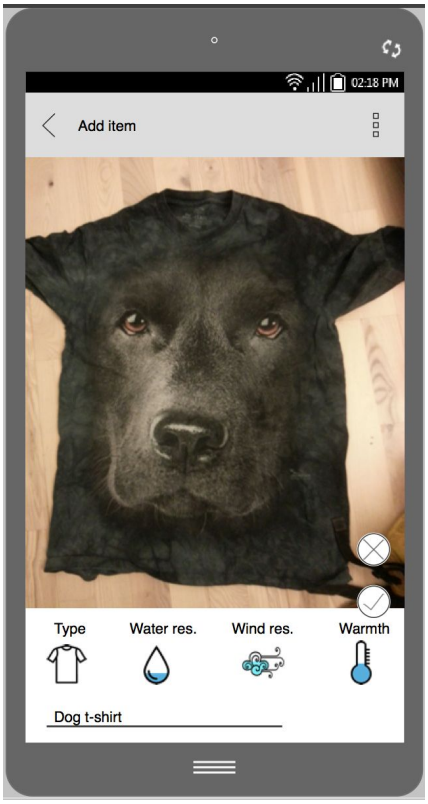
These can be created by hand (take a photo of your drawings and insert them in this flow), or using a program like Photoshop or Balsamiq.

Screen 1



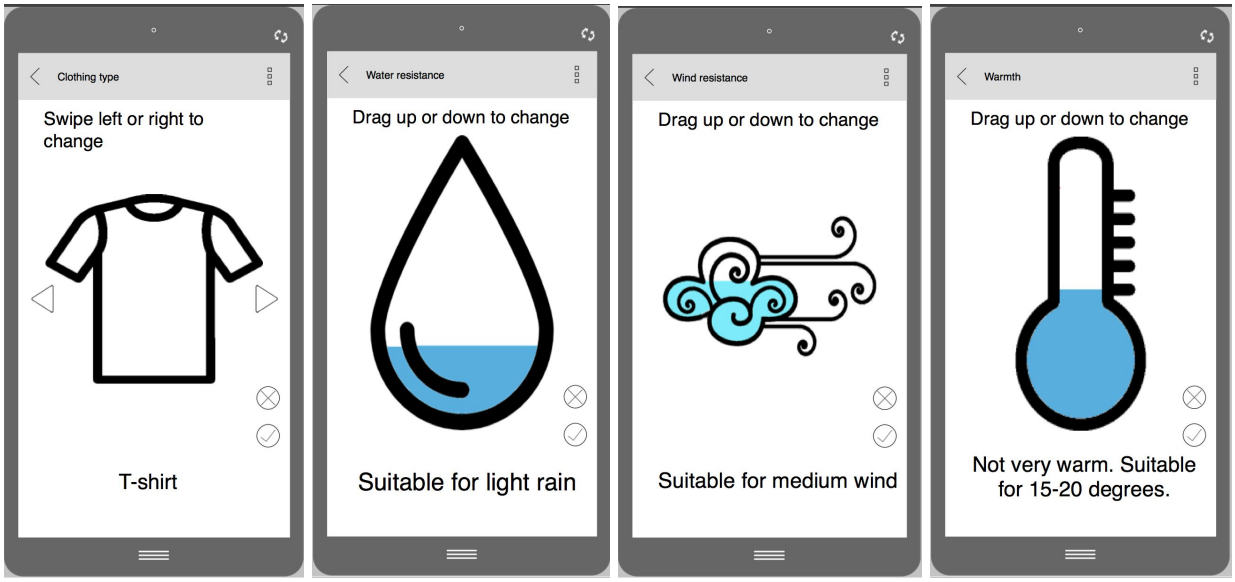
This is the main screen of the application, with suggested clothing for the user. It displays the current weather information, and clothing of the user that would match the current atmospheric conditions. The user can long press on any of the elements to edit them. After the edit is performed the list is recalculated to match the new conditions. The user can also open the options menu (top right corner) and either change app settings or go to the list of items.

Screen 2



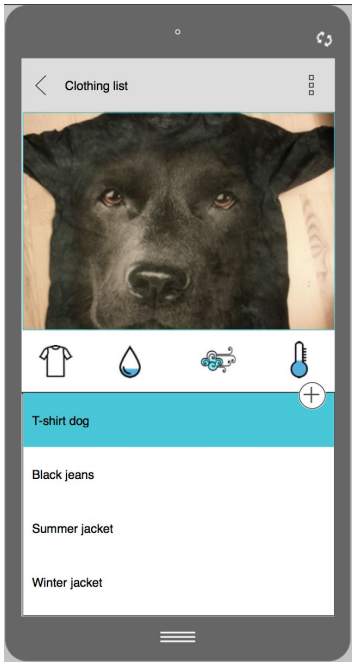
Add/edit item screen. On this screen the user can add a new clothing item or edit existing one. Initially no image is set, with a message “Tap to add image”. If the user taps he is taken to a image chooser/camera activity to take a picture. Assuming that provided image was correct he is then taken to cropping activity (From android crop library - https://github.com/jdamcd/android-crop) so that the image matches required ratio.
The user can also press on any of the four buttons below to set type, water and wind resistance and warmth of clothing article. He can also name the piece of clothing by typing input into field on the bottom. To inform user about it the field is originally empty with hint saying “Enter name”
When the user is done he can press the tick FAB to confirm or cross FAB to cancel. To confirm the user has to at least provide item’s name. Once user confirms the confirm/cancel FABs switch to edit/add FABs allowing user to swiftly add more items (or edit the already created one).
The exception from this is if user adds new item, in such case instead of switching FABs all of the information will be cleared and he will be back to the beginning of adding the item.

Screen 3



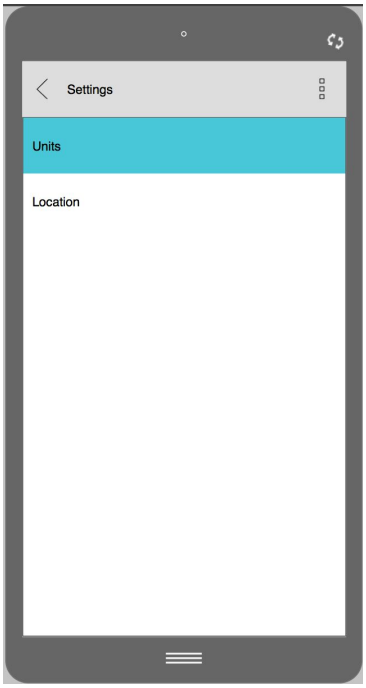
These are parameter modification screens for clothing items. The user reaches them by clicking on any of the four parameter buttons in add/edit item screen. At this point the user can change that parameter by either swiping left right (first screen) or dragging the fill color of the image up or down (other screens). Whenever user changes the value a description below also changes to describe current state of the item. The user can confirm by clicking on the tick FAB or cancel by clicking on cross FAB. Both of these will take user back to the previous screen.

Screen 4



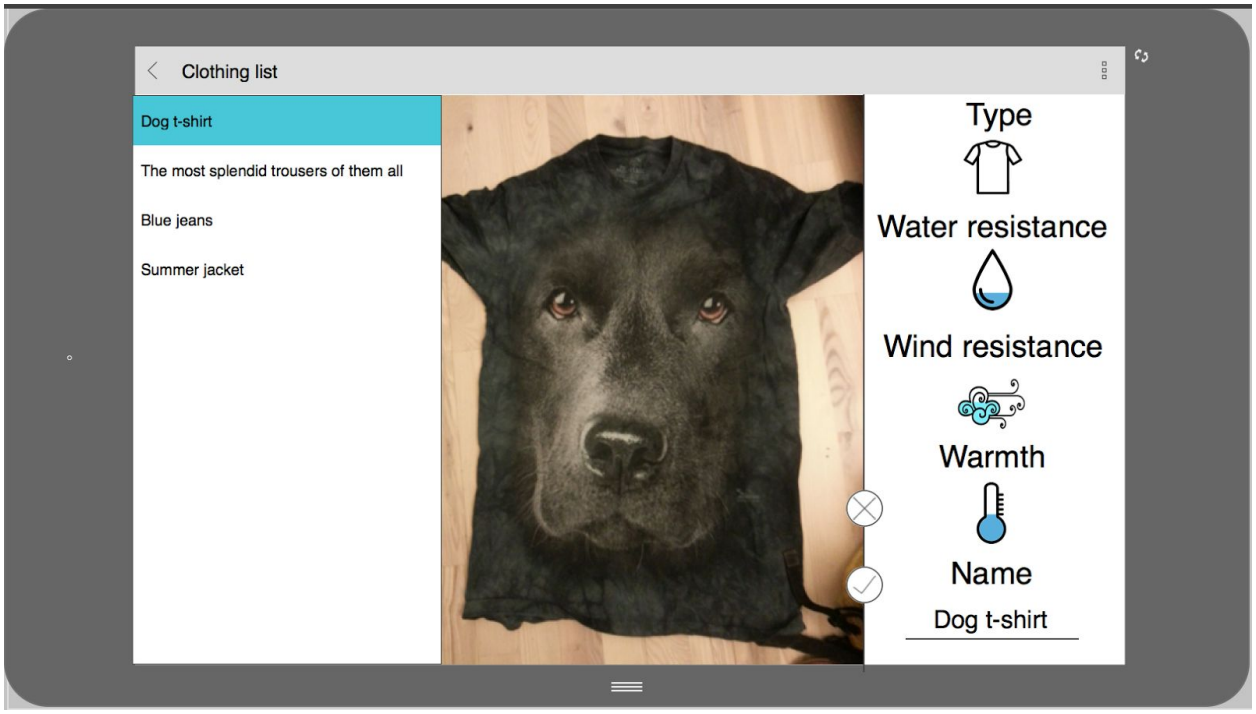
Clothing list screen allows for viewing all of the added items. Whenever an item is selected it's data show up at the top of the screen, including it's icon and parameters (presented as just icons). If the user long presses on any of the list items (or on the selected on on the top for that matter) he will be able to choose from a popup whether he wishes to modify or delete the item. The user can also add a new item by clicking on the FAB.

Screen 5



A small settings screen available from any screen by clicking on the top right menu and selecting preferences. It allows for changing units to metric or imperial as well as manual changing of location.

Screen 6



This is tablet version of the item list screen. Instead of using the space on top of the list to display the item data, the right side of the screen is used for that purpose. Additionally user can modify the item directly from this screen by long pressing on the image, tapping on any of the parameter buttons or changing the name. After any change has been made the user can either discard it by tapping on cross FAB or confirm it by tapping on a tick FAB. User can delete items by long pressing on any of the items in the list and selecting delete from the popup.

Key Considerations

How will your app handle data persistence?

I will build a ContentProvider backed by an SQLite database

Describe any corner cases in the UX.

-

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

Picasso to handle loading of images.
Butterknife to make easy and readable view injections.
Icepick to make it easier to save and restore Activities and Fragments.
Hugo to add easy logging of methods.
Retrolambda to add lambda support to older android versions.
Totally lazy to add collections functional operations (instead of using long and tedious structures like java's for(Object o : objects))
Spring for android for easy network operations.
Gson for (de)serializing json.
Lombok to generate most of data model code, as well as to help preventing null pointers.
Tray for using process safe preferences.
Android support libraries for RecyclerViews, Fabs, AppCompat theme and so on.
Google play services for Location and Admob. If I have enough time also for Maps.
Android-crop for cropping photos so they have a correct ratio.

Possibly (depending on whether I will have enough time)
Spock for testing.

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

- Connect the project to VCS
- Configure gradle so that all the libraries are correctly added.
- Add required plugins and configure them (retrolambda, hugo)

Task 2: Implement UI for Each Activity and Fragment

List the subtasks. For example:

- Build UI for SuggestedApparelActivity
- Build UI for ClothingActivity
- Build UI for AddEditClothingFragment

- Build UI for ItemParamFragment
- Build UI for ClothingListFragment

Task 3: Build data layer

- Create SQLite database to hold items
- Create ContentProvider to access the database

Task 4: Build the item add/edit screen

- Allow user to select image from disk or with camera.
- Crop the image so that it matches the ratio expected.
- Allow user to re-select the image if he long presses on the image.
- Allow user to enter the name of the item and un-gray the confirmation FAB button if he does that.
- Allow user to modify other 4 parameters of the item by opening their corresponding screen.
- Allow user to discard the changes by clicking on corresponding FAB.
- Save the changes in the database once the user clicks on confirmation FAB.

Task 5: Build the item parameters screens

- Allow user to view the currently set parameter of the currently modified item.
- Allow user to change the parameter - inform the user by changing the fill of the main icon (or changing of the icon itself for clothing type) as well as the description of the current parameter.
- Return the result to the previous screen (either the same as original parameter if user cancels, or the new one if he confirms)

Task 6: Build the clothing list screen

Describe the next task. List the subtasks. For example:

- Display the clothing list in a RecyclerView with a usage of CursorLoader.
- Display the preview of the currently selected item
- Allow the user to add/modify the items by going to the add/edit item screen (or if it is on tablet, to do it in the same screen)

- Allow user to delete items.

Task 7: Build weather and location api

- Use Google play services to acquire location of the user.
- Create a sync adapter that will pool the data from the open weather map api based on the current location (or location provided by the user in the settings screen).
- Keep track of the current weather data (but no need to keep track of previous days weather) - using preferences.

Task 8: Build the suggest apparel screen

- Display current weather at the top of the screen.
- Calculate best clothes for the weather based on the weather.
- Display best clothes in a grid, by showing their thumbnails and names (using Picasso).
- Allow user to view and modify items by long pressing on items in the grid.
- Recalculate the items if the user modifies any of the elements or new weather information arrives.

Task 9: Build the suggest apparel widget

- Display best clothes in a grid, by showing their thumbnails and names in the widget.
- When the user taps on the widget he is taken to the app.

Task 10: Add accessibility to the app

- Ensure that app can be used with D-pad.
- Ensure app uses content descriptions and doesn’t rely on sound cues.

Task 11: Add the full tablet support

- Modify the clothing list so it matches it’s tablet design.
- Ensure that screen is used efficiently by the tablet (without wasted space)

Task 12: Add Google ads

- Implement Google ads with the usage of Google Play Services.

Task 13: Build settings screen

- Create settings screen
- Add option to change the units for weather data from metric to imperial
- Add option for the user to manually set his location (by providing his international zip code)

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**"