

[Description](#)

[Intended User](#)

[Features](#)

[User Interface Mocks](#)

[Main Screen](#)

[Search Results Screen](#)

[Details Screen](#)

[Favorites Screen](#)

[Widget](#)

[Key Considerations](#)

[How will your app handle data persistence?](#)

[Describe any corner cases in the UX.](#)

[Describe any libraries you'll be using and share your reasoning for including them.](#)

[Describe how you will implement Google Play Services.](#)

[Next Steps: Required Tasks](#)

[Task 1: Project Setup](#)

[Task 2: Create POJO classes](#)

[Task 3: Create Http Client class](#)

[Task 4: Add Database classes](#)

[Task 5: Implement UI](#)

[Task 6: Implement Search](#)

[Task 7: Save/Restore state](#)

[Task 8: Handle Errors](#)

[Task 9: Implement Sharing Feature](#)

[Task 10: Implement Google Play Services](#)

[Task 11: Create Build Variants](#)

[Task 12: Add Widget](#)

[Task 13: Improve App UI](#)

[Task 14: Add Signing Configuration](#)

[Task 15: Add Support for Accessibility](#)

[Versions](#)

**GitHub Username:** kasiarn87

# Makeup App

## Description

Find the best makeup products available on the market!

## Intended User

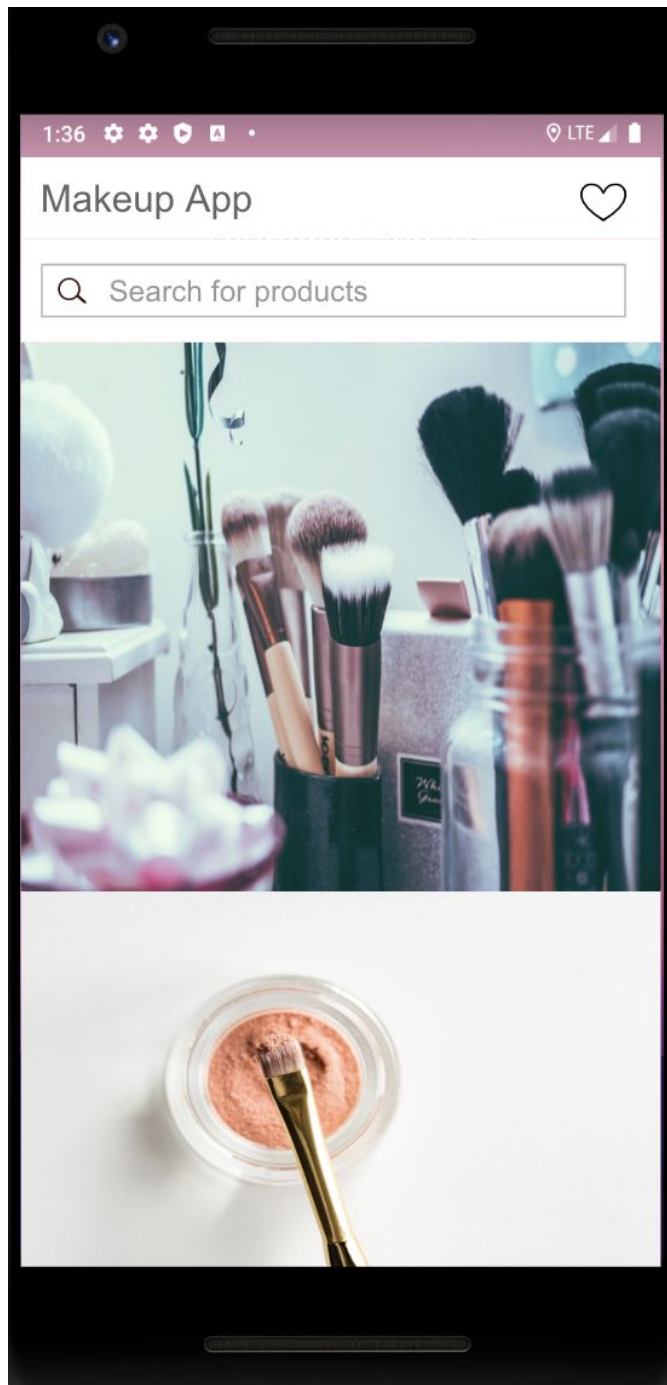
Anyone interested in makeup.

## Features

- Search for makeup products by brand, product type or tag
- Save favorites
- Show favorites
- Share products
- Show product details
- Open product web site

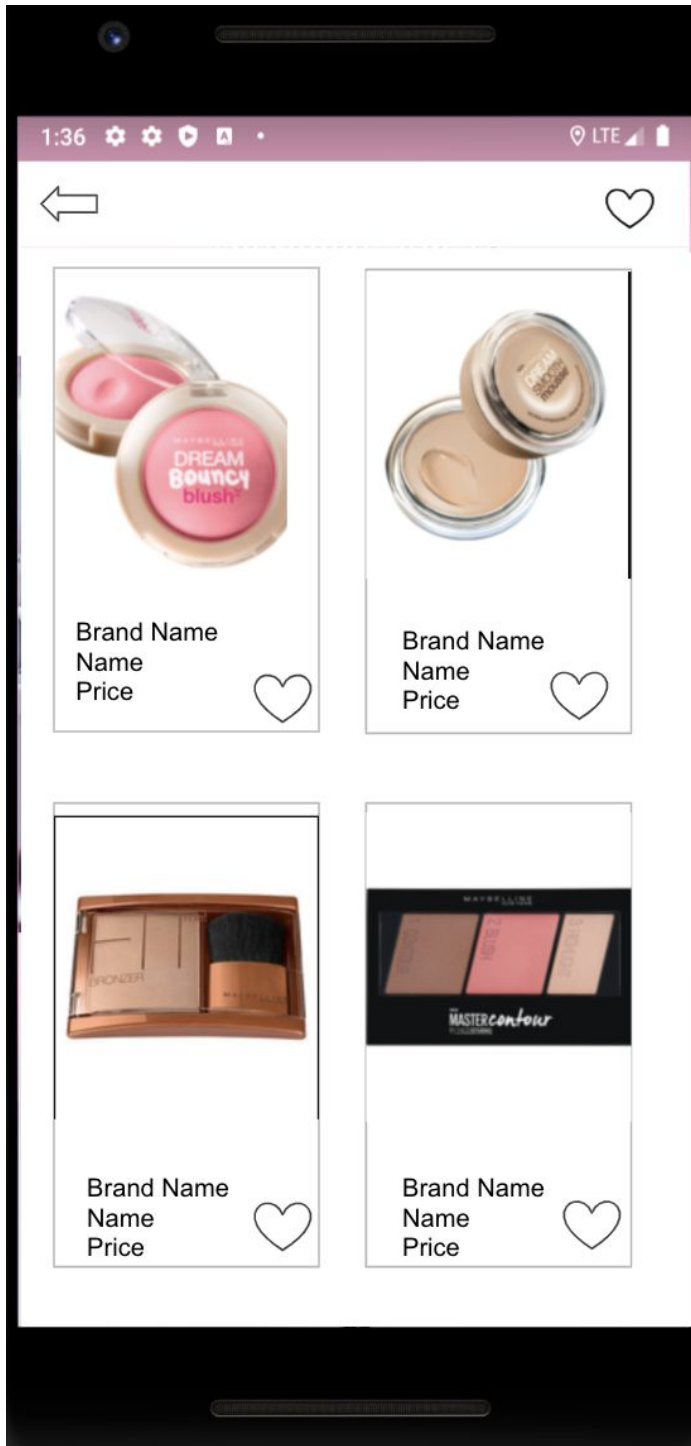
# User Interface Mocks

## Main Screen



Main activity containing a search field and a button to show favorites products.

## Search Results Screen



Screen containing search result. For each result we display an image and some basic info like product brand and price. Each product can be added to favorites.

## Details Screen

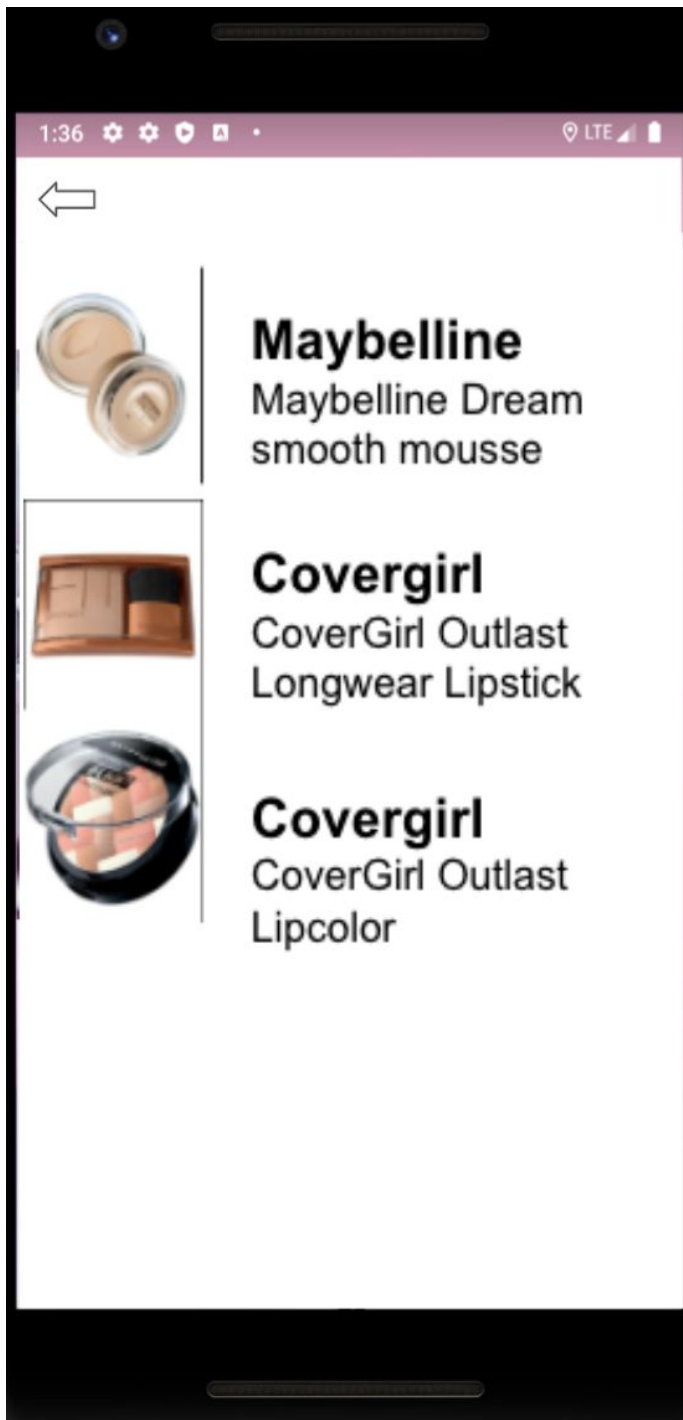


Product details screen.

Product can be shared or added to favorites.

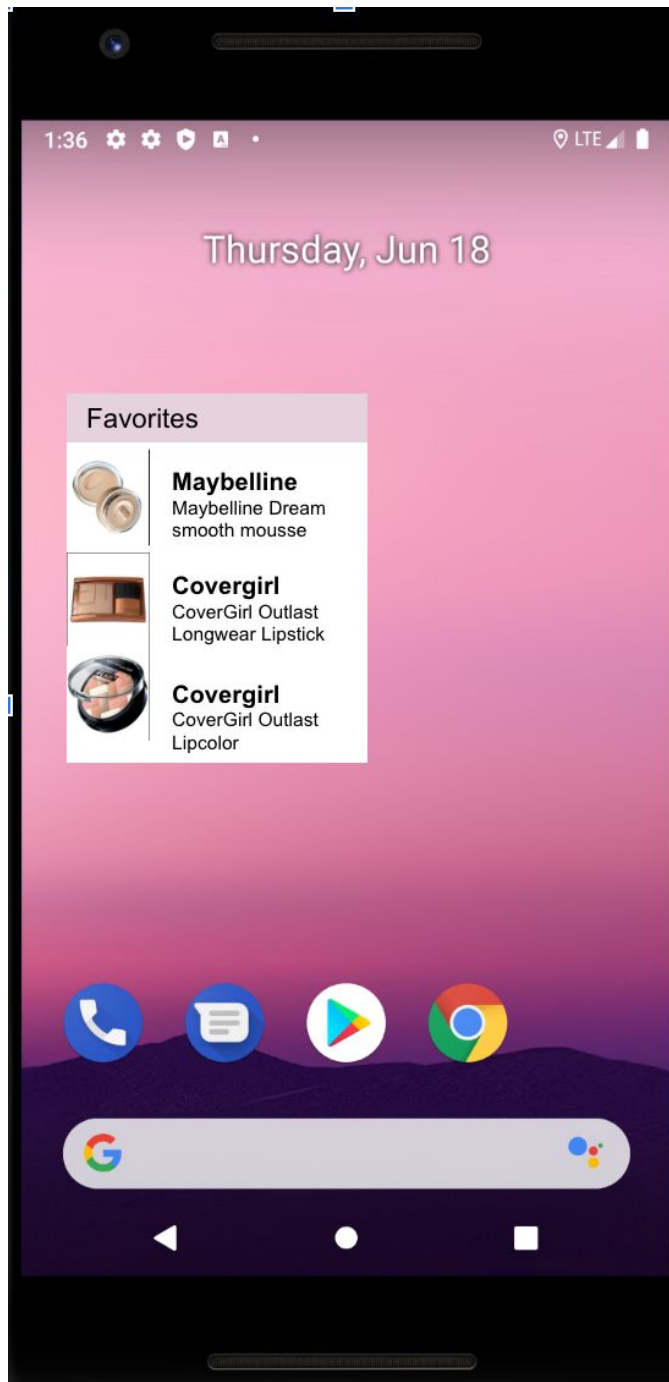
Screen shows info like brand, product name, price, available colors, rating and link to product website.

## Favorites Screen



Displays list of favorite items.

## Widget



Widget displays list of favorite items.

# Key Considerations

## How will your app handle data persistence?

App stores favorite products using Room. Favorites list is updated whenever user adds or removes an item.

## Describe any edge or corner cases in the UX.

- Details screen contains a FAB that serves to add product to favorites.
- Details screen can be scrolled to see more details. When scrolling, the detail image disappears and toolbar, back button and FAB are adjusted to improve the UI.
- If user has no favorite, then Favorite Screen will display appropriate message

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

Picasso to load images (or placeholders, if image is not available).

Retrofit as http client to easily perform network requests and json parsing.

Timber for logging, configured so that logs will appear only in debug app.

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

- App uses Google Play Services AdMob. Ads will be displayed on home screen only in the free variant.
- App uses Google Analytics to log a "share product" event

# Next Steps: Required Tasks

## Task 1: Project Setup

- Create new Android Studio project
- Add lib dependencies to build.gradle
- Define minSdkVersion



## Task 2: Create POJO classes

Define classes that will contain pojos relevant for the app.

Api used is <http://makeup-api.herokuapp.com/>

## Task 3: Create Http Client class

- Use retrofit to create an http client class
- Define http methods that will be used by the app
- Define baseUrl and paths based on api <http://makeup-api.herokuapp.com/>

## Task 4: Add Database classes

- Define Room Database class
- Define Room Dao interface that will contain DB operations
- Define class that extends AndroidViewModel and defines a LiveData object containing favorite products

## Task 5: Implement UI

- Create UI for MainActivity
- Create UI for search results screen, add adapter for results recycler view
- Create Detail Screen activity and implement navigation from search results to details screen
- Create UI for Favorites screen, add adapter for favorites recycler view and implement navigation from home screen and search results screen

## Task 6: Implement Search

- If user searches for a term that is a brand, a product type or a tag, then perform the appropriate query and show results
- If user searches for a term that is not available, show an error message
- Add enums containing list of available brands, product types and tags
- To check if search term is a valid brand, type or tag, compare it with these enum values
- App uses Async Task to perform search

### **Task 7: Save/Restore state**

Save/restore state on each screen, so that screen is not reloaded e.g. when rotating the device

### **Task 8: Handle Errors**

Handle errors such as:

- No internet connection
- Product image not available
- Other product info not available
- No item available

### **Task 9: Implement Sharing feature**

Create intent for sharing product website link from Details Screen

### **Task 10: Implement Google Play Services**

- Add gradle dependency to play-services-ads for the free version
- Add ad to home screen
- Add gradle dependency for firebase analytics
- Register app with firebase and add config file
- Create FirebaseAnalytics instance
- Log an event when user shares a product

### **Task 11: Create Build Variants**

- Create free and paid flavors
- Paid flavor doesn't contain ads
- Remove from paid flavor any unnecessary dependencies

### **Task 12: Add widget**

- Implement widget that will display updated list of favorite items in a recycler view
- Implement intent service to fetch products that will be shown on widget

### Task 13: Improve App UI

- Add elevations where needed
- Make app look good on tablets
- On details screen, make image disappear while scrolling

### Task 14: Add Signing configuration

- Create keystore (in the root of the project) and key
- Create signing configuration in build.gradle file
- Assign signing configuration to build type release
- Make the store file path a relative path

### Task 15: Add Support for Accessibility/Localization

- Move all strings to string.xml file
- Add content-description where needed
- Use margin start/end/left/right attributes
- Add android:supportsRtl="true" to manifest
- Make sure app supports RTL layout on all layouts: activate "Force RTL layout direction" in developers options and check it

## Versions

- App is written in Java
- Project uses following versions of Android Studio, gradle and libraries

Android Studio	4.0
gradle	6.1.1
Timber	4.7.1
retrofit	2.8.1
picasso	2.71828