How to Use this Template

1. Create a new document, and copy and paste the text from this template into your new document [Select All → Copy → Paste into new document]
2. Name your document file: "**Capstone_Stage1**"
3. Replace the text in green

Capstone Stage 1 project (author: Tomasz Milczarek)

GitHub Username: https://github.com/miltomasz

Pricefy

Description

Write a brief summary of what your app does. What problem does your app solve?

The app allows to take photos of objects that the user wants to buy. Next it recognizes what is in the picture and finds the same or similar items in the Amazon store. It shows a list of these items along with the price and gives user the opportunity to purchase the selected item in the Amazon Shopping app or on the Amazon website.

Intended User

Who is your intended user?

The app is for customers of stores, shopping malls, local shops etc. (generally for all people who do shopping) who want to check the market price of a product. Users also can check if there is other similar (competitive) product in Amazon store, check its price and possibly make a purchase via Amazon Shopping mobile app or via mobile browser.

Features

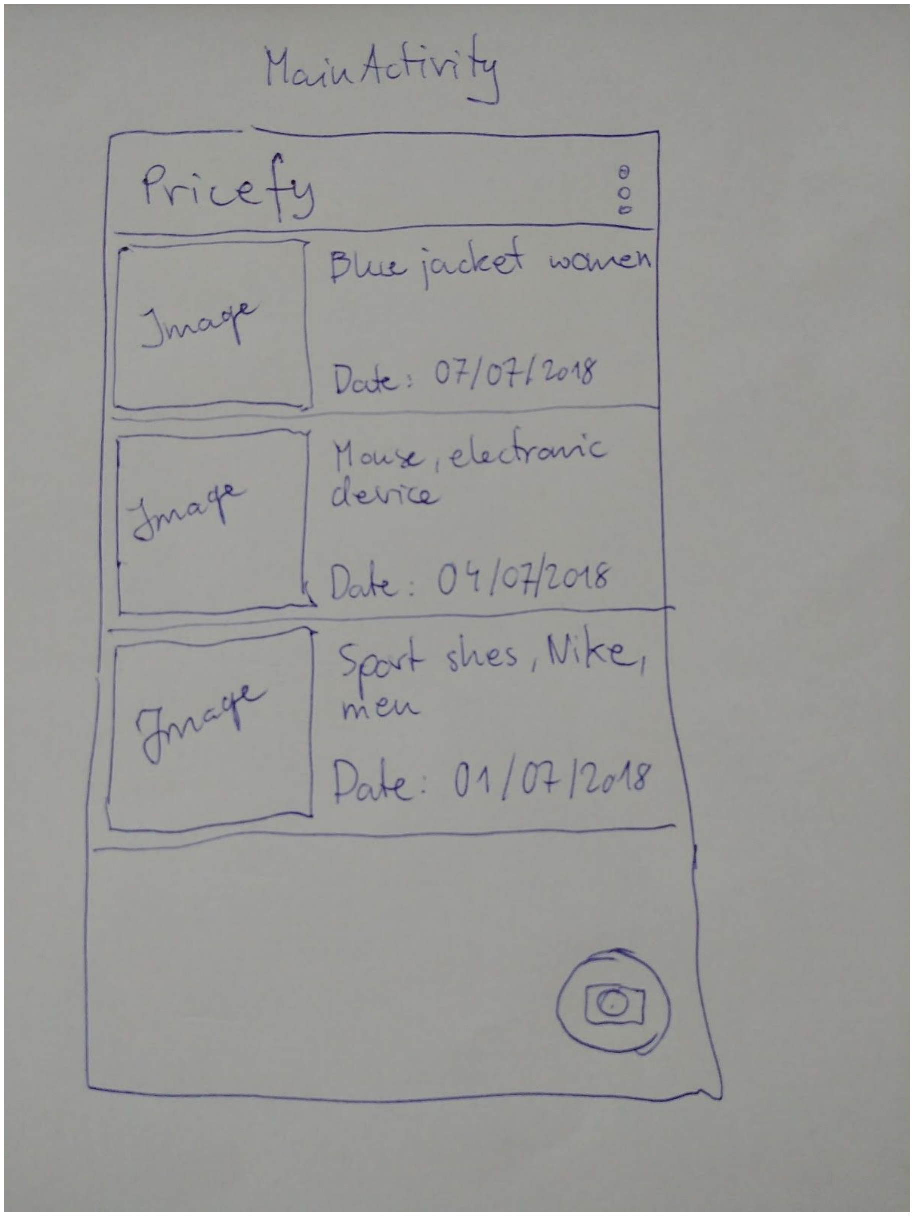
List the main features of your app:

- Takes pictures of product
- Sends it to Google Vision API to make image recognition
- Displays list of taken photos along with recognized tags
- Displays list of items found in Amazon store basing on tags returned by Google Vision API. The presented items have prices which may be informative for users.
- Saves those lists in local database
- Allows user to buy selected product via Amazon Shopping app or Amazon website
- Amazon store is an example here. The app architecture will allow to add easily a new internet shopping platform (store). For example: Allegro.pl in Poland

User Interface Mocks

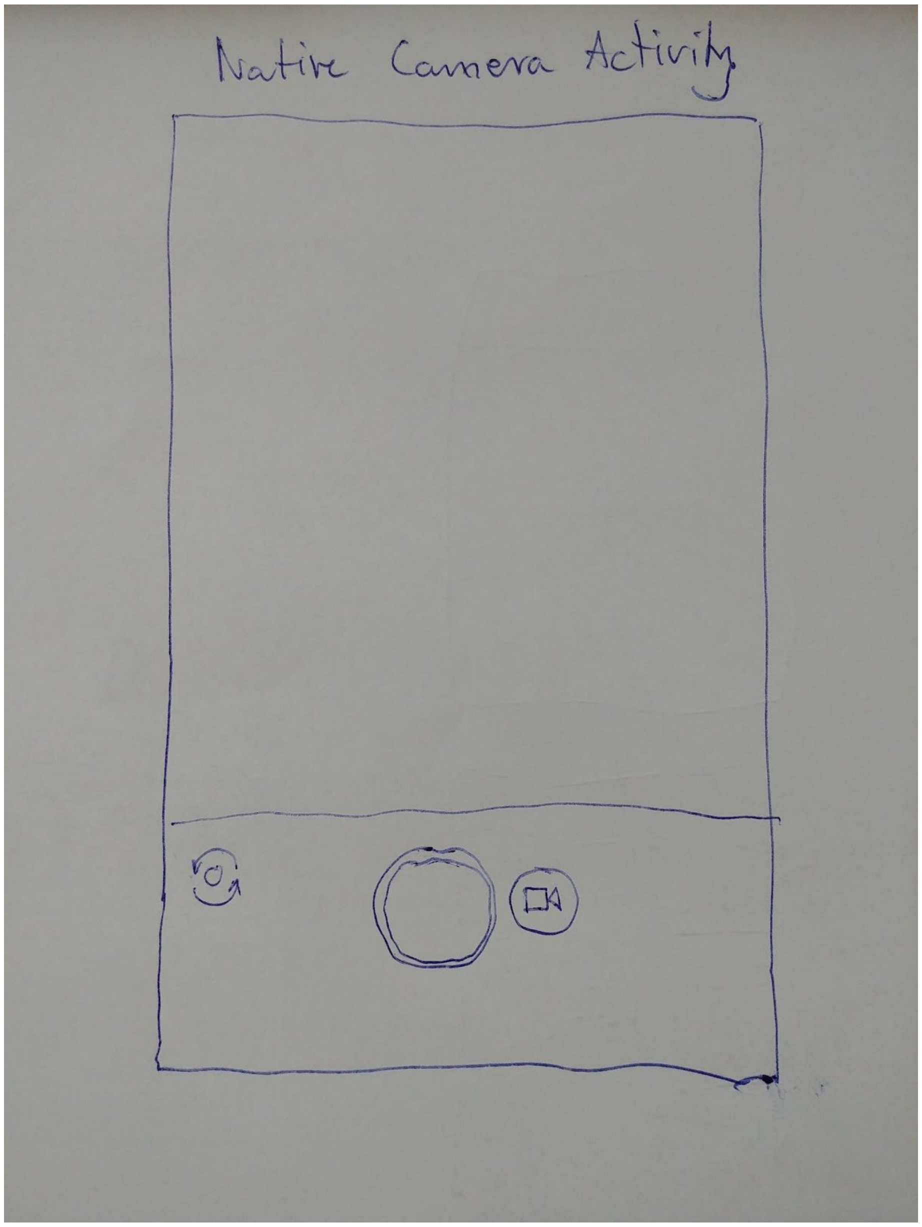
These can be created by hand (take a photo of your drawings and insert them in this flow), or using a program like Google Drawings, www.ninjamock.com, Paper by 53, Photoshop or Balsamiq.

Screen 1



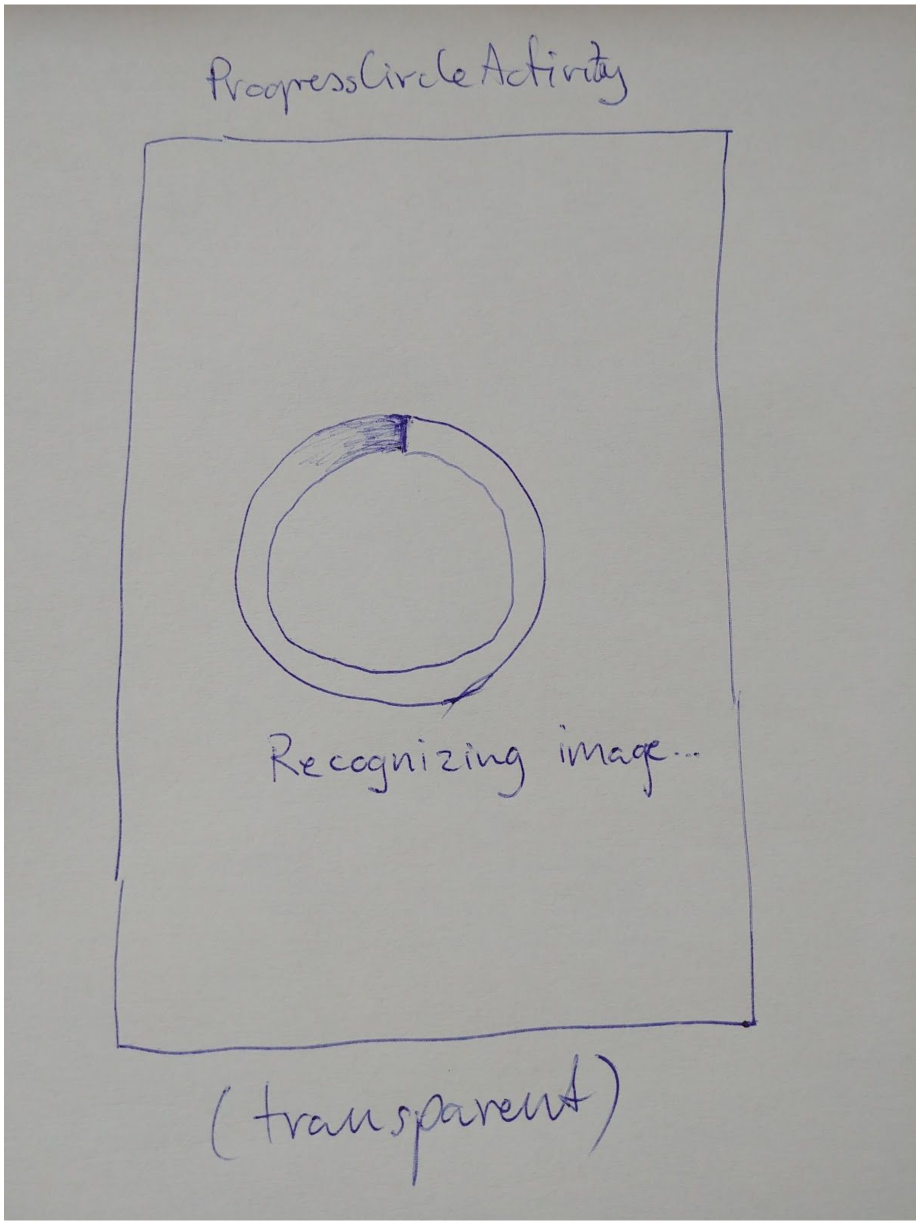
MainActivity screen - it holds list of images taken by user. The list is implemented using RecyclerView. Each item view contains list of recognized tags as a title. It also contains date/time when image was taken.
Tapping one of the item views opens **ResultActivity** (Screen 4).
At the bottom of the screen there is a floating action bar which opens native Android camera activity.

Screen 2



Native **Android camera** activity **-** user can take picture.

Screen 3

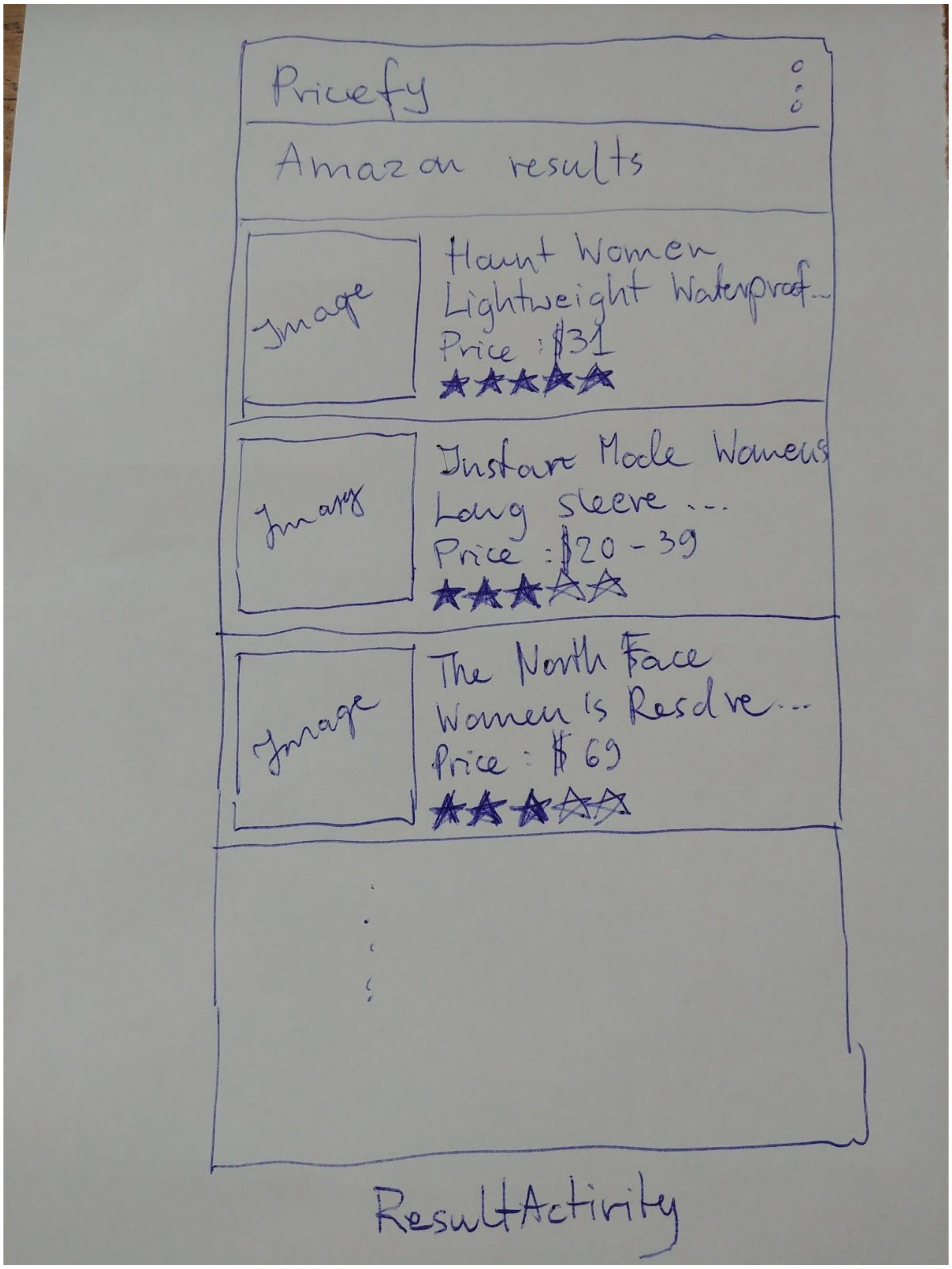


ProgressCircleActivity screen - this activity enters the stack after returning from native Android camera activity. It has some transparency set. This activity is on stack until the app has finished processing:

- Compressing image
- Sending image to Google Vision API
- Waiting for image recognition
- Receiving Json result from Google Vision API and saving it in local database

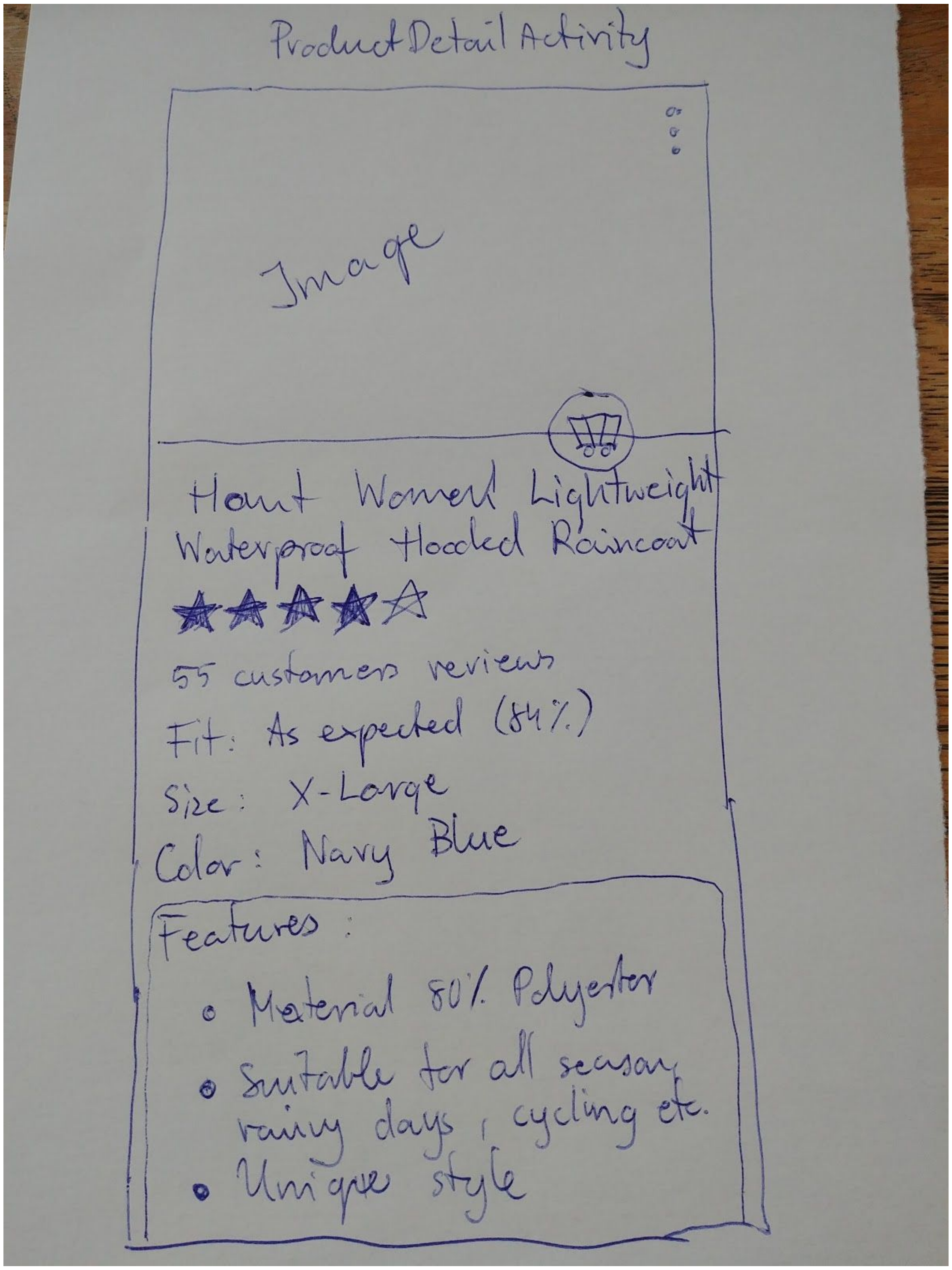
After performing those actions the activity finishes and **MainActivity** (Screen 1) is shown with new item in the list.

Screen 4


Pricefy
Amazon results
Image
Haunt Women Lightweight Waterproof...
Price : $31
Image
Instant Mode Womens Long sleeve ...
Price : $20 - 39
Image
The North Face Women's Resolve...
Price : $69
ResultActivity

ResultActivity screen **-** this screen displays results of item search on Amazon website. The search parameters are tags returned by Google Vision API.
Item view has thumbnail image, item title, price and “five stars” rating component.

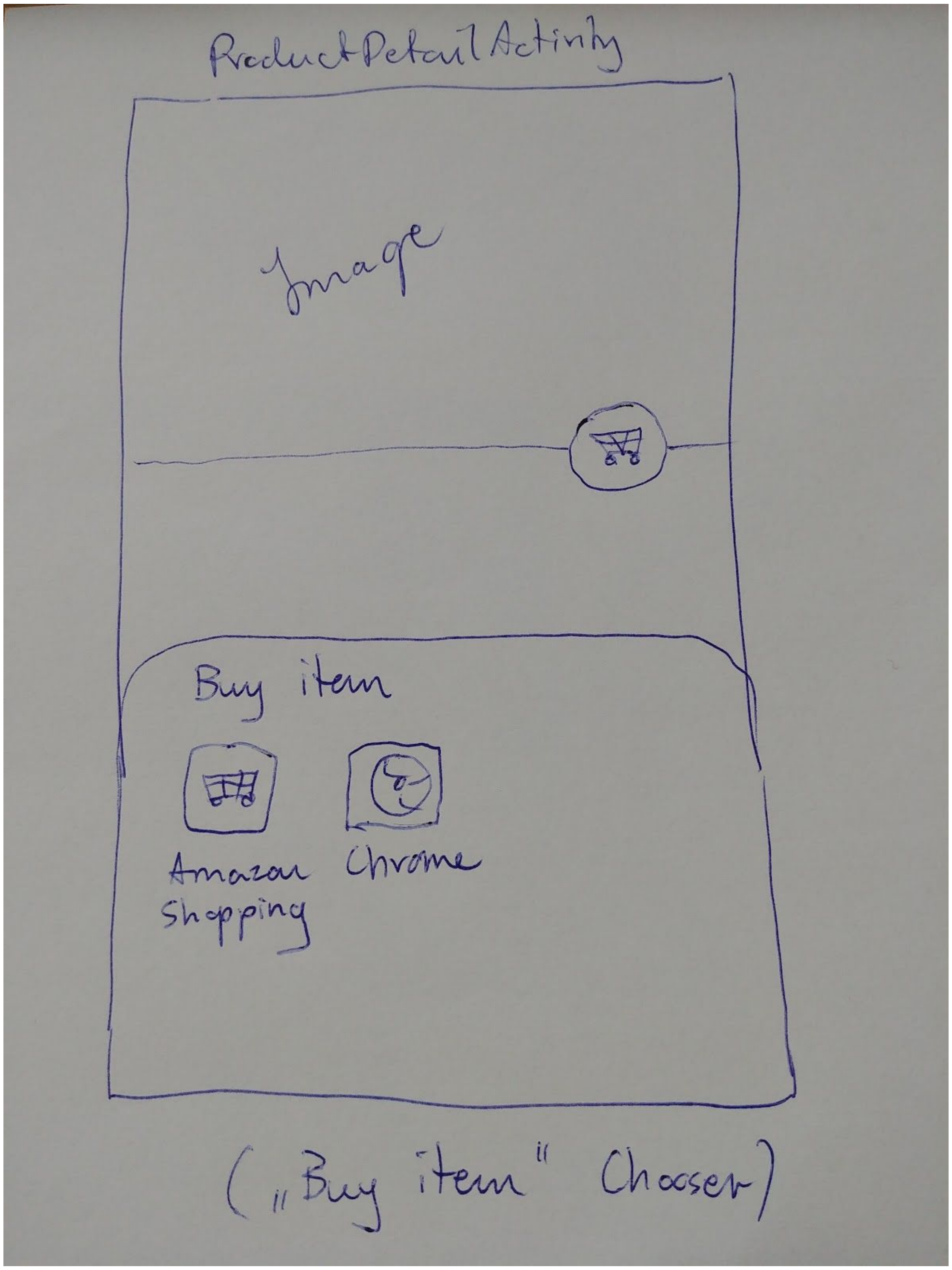
Screen 5



ProductDetailActivity screen - it is a detail view for item selected in **ResultActivity** screen (Screen 4). It contains the same information as previous screen and also has some additional, more detailed data like number of customers reviews, size, color and other features.

It has floating action button which opens chooser with apps that can handle item purchase request (Screen 6).

Screen 6



ProductDetailActivity screen - this screen presents expanded chooser. There will usually be two applications inside - Amazon Shopping app or Chrome app. Tapping one of those will cause redirection to Amazon website (or Amazon Shopping app screen) with selected item. Users will be able to purchase selected item.

Key Considerations

How will your app handle data persistence?

The app will store data in local database (SQLite). The Room library will be the interface to get access and manage database.

Describe any edge or corner cases in the UX.

1. If user has not taken picture of product yet main screen will display empty view with message “No items searched”.
2. Camera screen will use native Android camera activity.
3. Floating action bar in main screen will launch Camera activity.
4. After taking a picture transparent screen will be displayed with custom progress bar in the center.
5. Tapping floating action bar in ProductDetailActivity screen will launch browser if no Amazon Shopping app is installed.
6. Recognizing image depends on active network connection (request to Google Vision API). If it is not available the corresponding message will be displayed.
7. Any info/error messages will be displayed using Snackbar component.

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

1. Constraint Layout library - to build and manage layouts in proficient way
2. Google Vision API library - for image recognition and getting tags describing image
3. Room library - to create local database and perform database actions
4. Butterknife library - to handle and bind View components across layout files and activities.
5. Architecture Component Lifecycle library - to base the app architecture on MVVM design pattern.
6. Retrofit2 library - to prepare and make http calls and handle responses
7. Espresso library - to create basic UI tests
8. Jsoup library - to parse html retrieved from Amazon website

9. Picasso library - to load and cache images both from network and local Uri.

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

The only Google service used will be Google Vision API:

- com.google.apis:google-api-services-vision

which uses as internal dependencies:

- com.google.api-client:google-api-client-android
- and com.google.http-client:google-http-client-gson

Next Steps: Required Tasks

Task 1: Project Setup

Subtasks:

- Create Android project with empty activity
- Add needed libraries to build.gradle
- Configure libraries (resolve possible dependency conflicts)
- Set up min and compile sdk
- Build project and run it on emulator

Task 2: Implement UI for Each Activity and Fragment (Implement View layer)

Subtasks:

- Build UI for MainActivity
- Use Collapsible Toolbar for app bar
- Use RecyclerView for displaying images
- Use GridLayoutManager to display pictures in two or three columns
- Use floating action button to launch Camera activity
- Use Picasso to load images

- Build ProgressCircleActivity (custom progress bar)
- Use Activity as a basis for the screen (not Dialog component).
- Add transparency to Activity background
- Implement circle progress bar in the layout center

- Build UI for ResultActivity
- Use Collapsible Toolbar for the app bar
- Use RecyclerView for displaying list of products found in Amazon store
- Use Picasso to load images

- Build UI for ProductDetailActivity
- Use Collapsible Toolbar for app bar
- Use AppBarLayout for implementing toolbar when expanded
- Use CoordinatorLayout to combine the above
- Add floating action bar to open “Buy item” chooser
- Use Picasso to load image

Task 3: Implement taking photo

- Invoke native Camera app
- Implement handling taken photo in MainActivity
- Compress image using Bitmap class methods
- Save image in file
- (Use IntentService for compressing and saving image)
- Save Uri to the file in local database

Task 4: Implement Google Vision API

- Read image from local Uri
- Initialize Vision API client (use CLOUD_VISION_API_KEY)
- Execute request to Google Vision API
- Handle response

Task 5: Implement model layer

- Implement model classes
- Create Room database
- Create DAO interfaces to get/write data
- Create Repository class as a single point for accessing data

Task 6: Implement ViewModel layer

- Implement ViewModels for each activity
- Create ViewModel factory and retrieve ViewModels from that factory
- Initialize ViewModels in Activities (in `onCreate()` method)
- Attach observers to ViewModels' methods

Task 7: Implement Jsoup client

- Create Jsoup requests with `field-keywords` parameter
- This parameter value should be words retrieved from Google Vision API as recognized tags describing image
- Parse retrieved Html document
- Save parsed data in local database

Task 8: Implement widget

- Create application's widget. This includes creating xml files (provider) and java classes (widget service, factory and provider)
- Widget will present list of recent photos taken by user with a short description (Title, Date/Time when it was taken)
- Tapping element in the widget's list will launch ResultActivity (Screen 4) showing list of items found in Amazon store

---

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**"