

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”**

[Description](#)

[Intended User](#)

[Features](#)

[User Interface Mocks](#)

[Screen 1](#)

[Screen 2](#)

[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.](#)

[Next Steps: Required Tasks](#)

[Task 1: Project Setup](#)

[Task 2: Implement UI for Each Activity and Fragment](#)

[Task 3: Your Next Task](#)

[Task 4: Your Next Task](#)

[Task 5: Your Next Task](#)

GitHub Username: devmil

PaperLaunch

Description

PaperLaunch is a side launcher reachable from anywhere in your Android device.

It can be configured to contain apps and shortcuts that can be launched.

PaperLaunch focuses on appealing design in the spirit of Googles Material Design and on speed.

You can group your apps into folders or even subfolders and you can launch the desired app with only one touch.

From placing the finger down to do the launch gesture over navigating through the folder structure to launching the desired app or shortcut you don't have to lift your finger.

And even if you have accidentally started PaperLaunch you have a neutral zone that doesn't launch anything to minimize accidentally launched apps.

Intended User

This is an app for people that have a set of apps that they want to start from anywhere. So people that have a handful apps and are willing to take some time configuring their device to fit their needs.

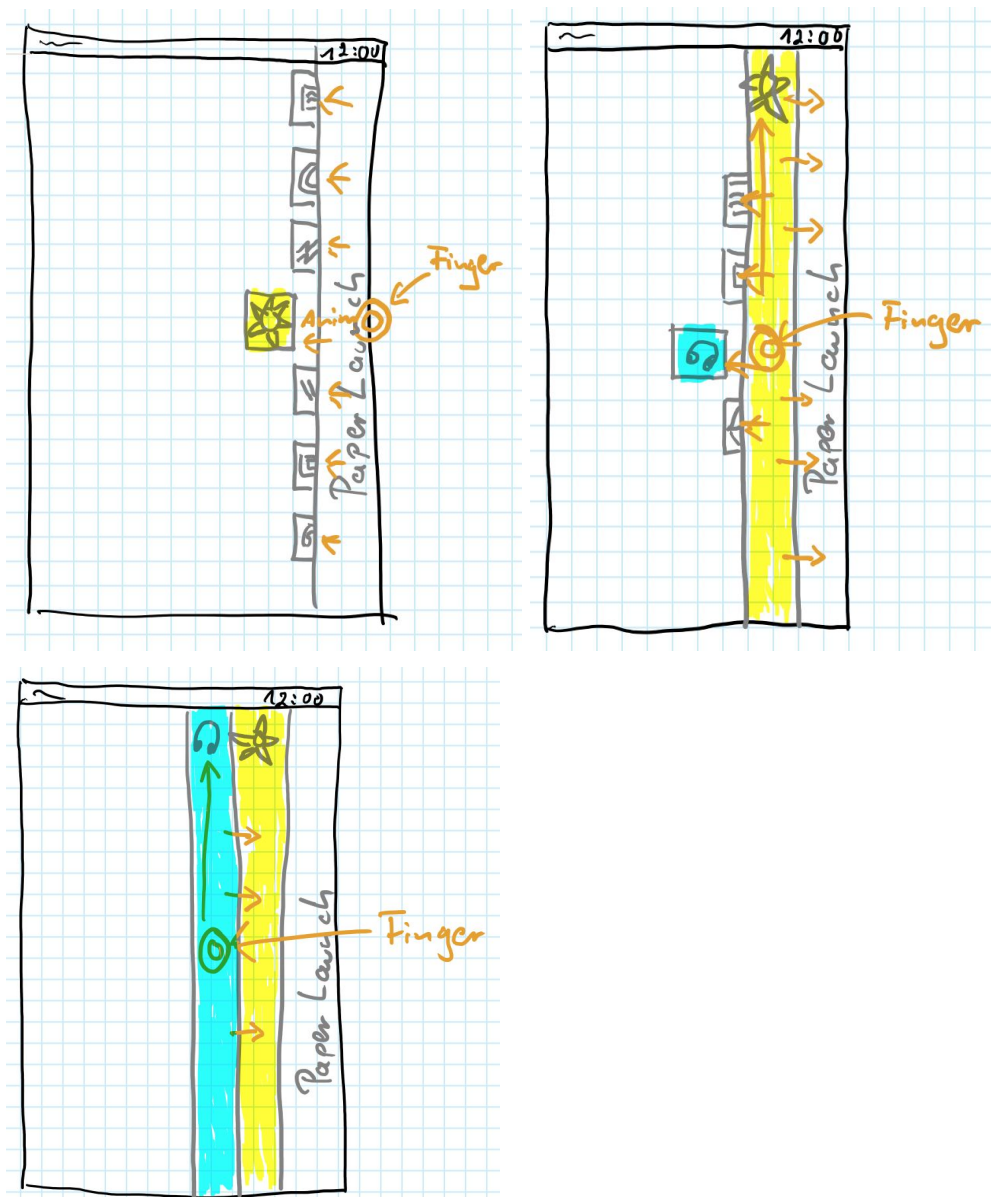
Features

- Launchable from anywhere in Android (swipe gesture from the side)
- Configurable Apps, Shortcuts and folders
- Appealing design based on Material design (hence the name of the app)

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.

Launcher Screen



This is the main launcher screen (overlay).

Behavior:

The vertical position of the finger focuses the element on the same y level in the current layer. Moving the finger vertically changes the focus and moving the finger over the focused item selects it. The focused item moves from under the layer handle next to it.

All non-selected items of the current layer move under the layer handle and the selected item animates to a full height layer handle for the next layer.

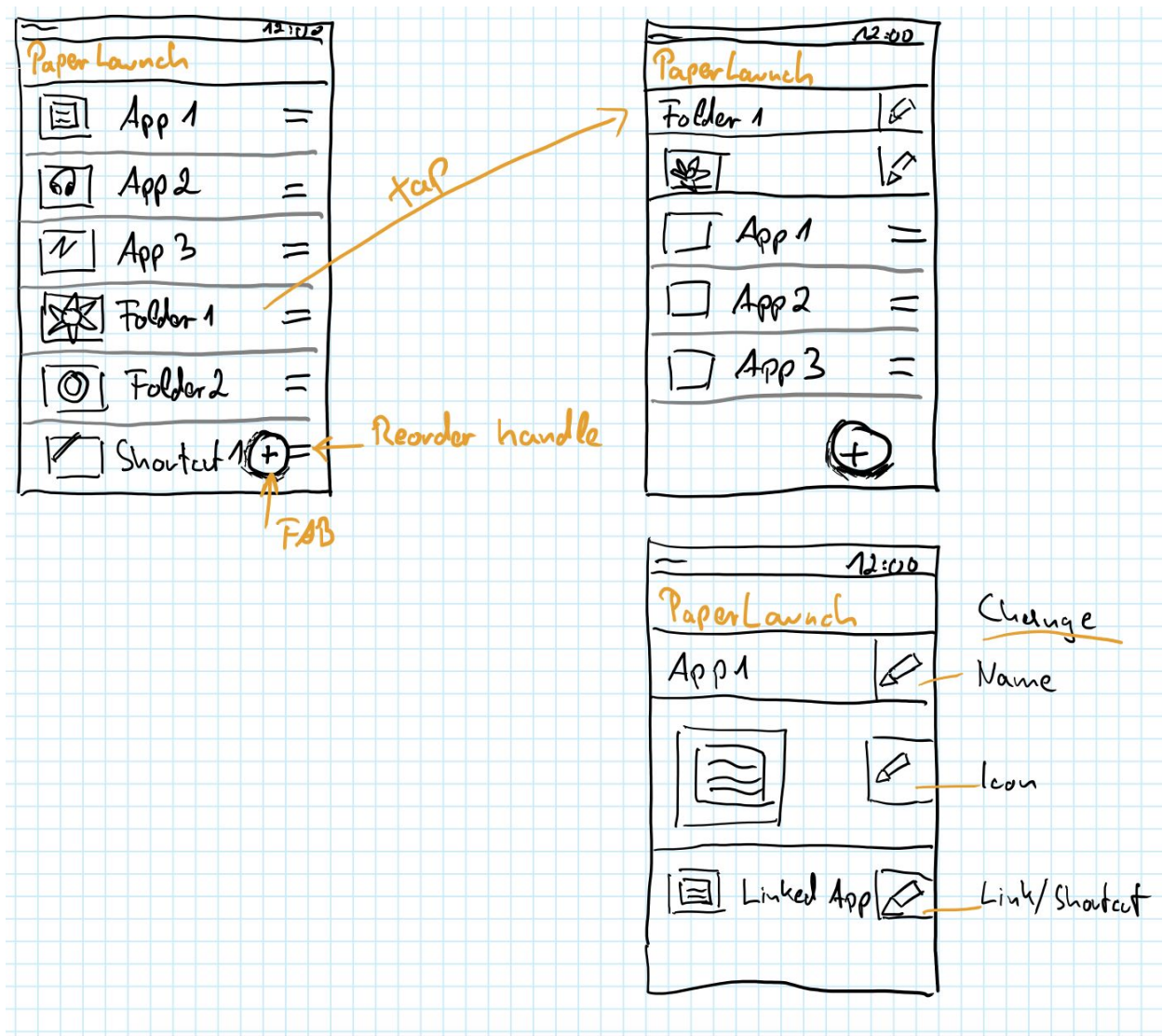
If the selected item is a folder the items of this folder animate from under the layer handle half way out and the focused item (if there is one depending on the finger position) moves completely out of the layer handle.

If the selected item is an app or a shortcut there are no more items to show.

If the finger gets released while an app or shortcut is selected the app or shortcut gets launched and the launcher disappears.

The finger can also be moved back and will release the selected item layer by layer as it passes by.

Configuration screens



The configuration screens allow the user to configure the structure of the launcher.

Each layer can be edited and will present a list of subitems that can be rearranged (using the drag and drop handle on each item)

Long tapping on an item activates action items in the toolbar ("delete", "add app/shortcut" and "add folder" for instance).

The add functionality is also exposed through a FAB which will expose a new submenu (app/shortcut or folder) when tapped.

Tapping on an item will open the configuration for the appropriate item.

For folders a similar configuration screen is shown where the items of this folder, its name and image can be edited.

For apps or shortcuts the name, icon and link (app or shortcut) can be changed.

Preferences screen

The settings screen is a default Android preferences screen and contains these preferences:

- Width of the launcher layers
- Sensitivity for the trigger gesture
- Position (left or right)
- Animation speed
- Haptic feedback

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

Key Considerations

How will your app handle data persistence?

The data will be stored in a SQLite database as this allows a dynamic depth of the folder structures.

Describe any corner cases in the UX.

Moving fast between the levels should not result in a lag due to animations taking place. Also this operation must not result in weird UI changes (due to animations cut off).

This might be handled by ensuring a relatively fast animation speed.

If not then there has to be a "fast option" for animations to complete faster if the layer is about to change and it has to catch up.

The goal is to have a very small (if any) lag between the user executing the launch gesture and the actual launch of the Launcher UI.

There are two major variants to implement this launcher:

- a. Directly putting the visuals to the window
This has the advantage that all required visuals are already there and no activity has to be started to get the launcher going.

The big disadvantage is that all the visuals have to be in memory forever. Therefore is this approach only the last option if Variant b doesn't work well

- b. The launch gesture launches the actual launcher Activity.

The advantage is that the system can release any resources the Launcher uses for its visuals.

The disadvantage might be that the launch time is too high. This has to be analyzed.

The launcher has to put its gesture detector in place triggered (besides other triggers) by a boot receiver to be "always ready".

I have seen other launchers holding an ongoing notification to make sure that they survive. This has to be analyzed and decided if this is necessary for the Launcher to be reachable all time.

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

RXAndroid

for all Tasks that have to happen asynchronously (loading images)

EventBus

For event communication between the preferences and the running launcher

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

- Create a new Android Studio project
- Configure the dependencies for all already known components

Task 2: Create the persistency layer

- Create a Database helper to manage the launcher structure
- Implement a model that represents the structure and can be read and written to the database

Task 3: Configuration Activities

- Create a configuration Activity that can visualize manipulate the launcher structure

- Search for ways to implement a list with drag and drop reorder option
- Create Fragments for
 - Launcher root editing
 - Subfolder editing
 - App/Shortcut editing
- Create a dialog to select activities or create shortcuts

Task 4: Create the Launcher logic

- Create a debug Activity to temporary display the launcher visuals
- Implement the launcher logic
 - building the visuals for the current layer
 - animating the focus / selection of items
 - handling traversing through the layers
 - Launching the selected item on release

Task 5: Integrate the Launcher logic with the system

- Create a service that installs a gesture detector and keeps itself active (by setting the appropriate parameters and / or installing an ongoing notification)
- When the gesture detector detects the launch gesture start the Launcher UI
 - Preferred: By launching an Activity
 - If no other way: By adding the visual elements to the window

Task 6: Synchronization logic

- Create a service that handles app installation state changes.
Mark apps as unavailable as an uninstall is detected

Task 7: Make the UI tablet friendly

- Use the fragments to make a master detail configuration layout for tablets

Task 8: Create a preferences Activity

- Create an Activity to edit the preferences for PaperLaunch

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**"