# Flutter and iOS Application Comparison

**An Empirical Metric Analysis of Performance and User Experience**

## Bachelor Thesis


| | |
|---|---|
| submitted by: | Philip Krück |
| Date of Birth: | 04.11.1998 |
| Matriculation Number: | 3938 |
| Company Supervisor: | Jan Jelschen |
| First Reviewer: | Dr. Oliver Becker |
| Word Count: | < 12.000 (text + footnotes) |
| Degree Program: | B.Sc. Business Informatics (A Track 2018) |
| University: | Hamburg School of Business Administration |
| Submission Date: | 09.04.2021 |
| Partner Company: | apploft GmbH |



HSBA HAMBURG SCHOOL OF BUSINESS ADMINISTRATION

# Contents

# Chapter 3

# Methodology and Study Design

This chapter explores the employed methodology for testing both the performance hypothesis $H_P$ and the usability hypothesis $H_U$ (mentioned in Section 1.3). The goal is to explicitly show the reasoning for the chosen methods as well as provide specific method implementation details to aid transparancy about the obtained results discussed in Chapter **??**.
Generally, the employed methodology to achieve the research aim (Section 1.3) is a replication of a relevant subset of functionality of an existing iOS app using Flutter. Thereby, the original app acts as a baseline with which the Flutter clone can be comparatively evaluated. Based in this comparison, the research question - whether the Flutter framework can match native performance and provide equivalent usability - is answered. Instead of creating artificial use cases, taking advantage of an existing app provides realistic instances for performance as well as user interface testing.
The first section in this chapter (Section 3.1) details the decision process for selecting the baseline testing app as well as its feature reduction for further comparison. The subsequent two sections (Sections 3.2 and 3.3) explore the specific methods and reasoning for the performance and usability comparison respectively.

## 3.1 Baseline App Testing Decision Process

The procedure for choosing the case study app is based on a filtering process of 4 steps (i.e. constraints):

1. The app is built and maintained by apploft.

2. The app includes common application **features**.

3. The app uses modern iOS framework technologies.

4. The app conforms to the human interface guidelines (HIG) by Apple (Apple Inc. 2021a).

The reasoning behind selecting the above filtering constraints is detailed in the following paragraphs.

**1. Creation and Maintainance by apploft**

This constraint was opposed on the filtering process such that both a contact person (apploft employee) is available for code specific questions and the original source may be referenced. Thereby, functionality may be implemented in a similar manner to facilitate comparability between both apps. E.g. a particular algorithm could be implemented similarly in the Flutter application to produce equal runtime complexities and thus be retracted as a confounding variable explaining performance differences. Furthermore, access to the original source code provides the ability to reduce unnecessary features of the original app for proper comparisons against the Flutter clone.

**2. Inclusion of Common Application Features**

On the one hand, including only common features constrains the tested functionality to fit within the scope of this thesis. On the other hand, it allows for the ability to generalize the results to a majority of iOS applications[1].

**3. Use of Modern iOS Frameworks**

Constraining the baseline app to be built with modern iterations of iOS framework technology ensures a fair comparison against the replica app built with the recently released Flutter framework.

**4. Conformance to HIG**

Conforming to Apple's HIG ensures the original app looks native to the iOS platform. Building a matching Flutter clone is especially interesting regarding the usability hypothesis $H_U$ as it would stretch the realm of possibility for Google's framework. In addition, providing a recognizable UX for iOS users would keep participants in the usability study (detailed in Section ...) focused on noticing differences instead of being distracted by an ambiguous UI.

1. This assumes that the observed apps for selecting features are indicative of the archetypal iOS app.

Based on the above constraints, a small study was conducted looking at 15 apps developed by apploft (constraint 1) from 9 different iOS App Store categories. The goal is to find common application features (constraint 2) among them. As for the purposes of this study, a **feature** is defined as either

(a) a generalizable UI component which is non-trivial, or

(b) an underlying technical attribute influencing the user experience.

Trivial UI components (a) such as buttons or text weren't considered **features** as they are omnipresent throughout every app. As for (b), a technical attribute has to influence the user experience to be incorporated as the purpose of this thesis is testing Flutter's value as a UI framework (see Section 1.3). For example, networking can be viewed as a **feature** if fetched data is displayed via the UI, but is not a **feature** if the sole purpose of networking within an app is to extract analytics data.

Furthermore, a **feature** had to appear at least twice. The results are summarized in Table (...).

Continuing the filtering process, as per constraint 2 uncommon **features** are excluded which appear in less than 50% of the observed apps. This reduces the list of features to the following (see Table . . . ):

- Networking
- Login/Authentication
- Tab navigation[2]
- Stack navigation[3]
- Keyboard interaction
- Vertically scrolling collections[4]
- Horizontally scrolling collections[5]

2. ...
3. ...
4. ...
5. ...

- Webview component[6] integration

Out of the 15 initially tested apps, 5 include all of the above **features**. Kickdown (see Section ...) is chosen among the remaining contestants for the baseline testing app. It was most recently release (Feb 2021) and is therefore built with modern iOS technologies (constraint 3) and complies to the most recent iteration of the **Human Interface Guidelines** (constraint 4). *@Jan: I believe Kickdown has the smallest code base. Is this worth measuring and including as a fifth constraint?*

*@Jan: Would it be intristing to see a graphic of the filtering process using a funnel as an analogy?*

The login and signup mechanism is removed from the original app for baseline testing. In terms of hypotheses evaluation, testing the signup mechanism would yield no further insight as opposed to normal networking, textfield and button interaction which is already available in other places of the app.

The Flutter app is implemented as closely as possible to the original application to avoid an indiscriminate comparison as detailed in Section 4.

## 3.2 Performance Comparison

The methodology chosen to test the performance hypthesis $H_P$ (Section 1.3) is a quantitative comparison along multiple common profiling metrics.

### 3.2.1 Selected Performance Measurement Variables

The selected variables for performance measurement include CPU, GPU and memory usage during specific user activities (Section 3.2.2). On the hand, the chosen metrics are the underlying causes of more ephemeral metrics such as page load speed. On the other hand, they can be easily measured using software tooling (Section 3.2.3).

### 3.2.2 Measurment Process

The measurement process for the individual metrics is further split into specific **user actions** which are executed and tested on both the iOS and Flutter app separately:

6. ...

- **app start:** The app is freshly installed on the test device, opened and measured util the visible postings are loaded.
- **scrolling:** On the postings overview screen, the posting cards are scrolled fully to the bottom and subsequently back to the beginning.
- **detail view:** From the postings overview, a posting is tapped to navigate to the detail view. Afterwards the back button is tapped to navigate back to the overview.
- **image gallery:** The image gallery of a posting is opened from the detail view of a posting and the first 10 images are viewed by swiping.

For each **user action**, the least efficient value for the particular metric is selected and averaged over 5 consecutive tests. Furthermore, 2 testing rounds are devised on separate devices. The iPhone 12 and iPhone SE are chosen as the upper and lower bounds of hardware performance respectively. The lower bound is defined in this case as per Apples recommendation to set the deployment target to the current operating system version (iOS 14 at time of writing) minus one (iOS 13) which lists the iPhone SE as the oldest supported device (Apple Inc. 2021b). iOS 13 is also the deployment target of the original Kickdown app. To reduce measurement bias, the device is restarted before each measurement to ensure that all irrelevant background processes are cancelled.

### 3.2.3 Profiling Tools

**Xcode Instruments** (Apple Inc. 2019) - a part of the **Xcode** IDE tool set - are used for profiling the individual metrics. It provides multiple preconfigured profiling trace instruments. For the purposes of this thesis, the **Time Profiler** tool (see Figure ...) is used for CPU, the **Core Animation** tool (see Figure ...) for GPU, and **Allocations** tool (see Figure ...) for memory usage quantification over time.

*Should the units be explained here or does it make more sense to mention it when discussing the results?*

### 3.2.4 Evaluation Process

To better understand the data gathered, it is subsequently visually depicted using exploratory data analysis (EDA) (Tukey 1977). Furthermore, a statistical ANOVA test is conducted to

calculate statistical differences between the two groups (I don't know if I should actually do this - haha).

## 3.3 User Experience Comparison

Coming soon...

# Bibliography

Apple Inc. 2019. *Instruments Help.* https://help.apple.com/instruments/mac/current/.

———. 2021a. *Human Interface Guidelines.* https://developer.apple.com/design/human-interface-guidelines/ios/overview/themes/.

———. 2021b. *Supported iPhone models.* https://support.apple.com/guide/iphone/supported-iphone-models-iphe3fa5df43/13.0/ios/13.0.

Tukey, John Wilder. 1977. *Exploratory Data Analysis.* Addison-Wesley.