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

Abstract

To be written... (max: 300 words)

Contents

Chapter 1

Introduction

Mobile platforms are dominated by two players - Apple and Google with their respective operating systems iOS and Android. Cumulatively, they form a duopoly in the smartphone operating systems market with a combined usage share of 15.2% for iOS and 84.8% for Android in 2020 according to to the International Data Corporation (IDC 2021).
To develop a mobile software application (*app*) for both target platforms, the corresponding development environments and technologies are utilized for each platform. This leads to a doubling of cost, development time, and the need for knowledge of two different application development paradigms. As a result, cross-platform frameworks such as Xamarin (Microsoft Corp. 2021), React Native (Facebook 2021), and Ionic (Ionic 2021) have been created.

The fundamental principle behind these frameworks is the provision of a unified tech stack operating on a single code base leading to increased development speed while also providing the ability to deploy for both mobile operating systems.

Generally, cross-platform frameworks utilize webviews[1] or a software bridge to communicate with the underlying host platform plugging into native interfaces. In both cases, communication channel delays may occur during app runtime leading to reduced execution speed. In addition, cross-platform frameworks deliver abstract interfaces over multiple native interfaces leading to a decreased subset of functionality and especially impaired UI customizability. These inherent architecture attributes (further see Section 2.1) explain both impeded performance (Ebone et al. 2018 and Corbalán 2019) and user experience (Mercado et al. 2016 and Angulo and Ferre 2014) compared to native technologies.

However, over the past 2 years one particular cross-platform technology with certain distinct

1. Webviews are UI component in both iOS and Android to display web content such as *HTML*, *CSS* and *JavaScript*

attributes has risen strongly in popularity (JetBrains s.r.o. 2021): Flutter (Google Inc. 2020) is a newly developed open-source UI toolkit by Google. It has a nonconventional approach to cross-platform development in that every app ships with the framework's rendering engine (Section ...). Thereby, Flutter bypasses host platform communication by avoiding the underlying system app SDK for terms of UI rendering. Flutter apps are compiled in native binary format which can be directly executed by the mobile computing device (see Section 2.2). Furthermore, the framework provides a *"[...] collection of visual, structural, platform, and interactive widgets"* (Google Inc. (2021b)) for UI customization. It seems that Flutter addresses the exact same issues that are generally criticized about cross-platform solutions.

1.1 Motivation

As a digital agency specialized on native iOS and Android development, **apploft GmbH** (the partnering company of this thesis) (apploft GmbH 2021) is highly interested in Flutter. The implications of using this framework could be wide-ranging, including the extension of the services portfolio to clients with lower budgets. For example, startups which are unsure about product/market fit (Andreesen 2007) and held by tight budget constraints are especially keen on reaching the maximum number of potential customers with their app. Flutter could be utilized for a fast iteration of a product deployable to both mobile platforms. Worth noting are the benefits of serving smaller customers like startups which include higher growth potential for long-term cooperation and risk diversification in apploft's client portfolio.

The possible updside of implementing Flutter exceeds the acquisition of small clients. Instead of focusing on platform customization with native tooling, more effort could be directed into developing unique custom features. Furthermore, infrastructure setup, package development and app updates would only be necessary for one codebase.

Since the aforementioned economic incentives aren't necessarily company-specific, they are relevant for mobile application developers at large. Scientifically, this thesis may be the basis for future work on Flutter's architecture attributes and their contribution to runtime performance and UI rendering of frontend toolkits, in general.

1.2 Problem Statement

To the best of the author's knowledge, there are no peer reviewed articles comparing the performance or usability to native apps[2] since Flutter was first released in March 2018 (Google Inc. 2021a), This leaves an especially interesting gap in the literature, since both aspects are the topmost perceived challenges of cross-platform frameworks considered from an industry-perspective (Biørn-Hansen 2019).

1.3 Thesis Objective

The aim of this thesis is derived based on the initially stated problems with cross-platform frameworks and the current lack of research on Flutter's lofty marketing claims to solve those drawbacks. Specifically, Google's assertion that Flutter can match *"native performance"* and the framework can be utilized for building *"expressive and flexible UI"* (Google Inc. 2020) will both serve as inductively derived hypotheses that shall be empirically verified or falsified individually by this thesis:

H_P: The Flutter framework yields comparable **performance** to native mobile frameworks for app development.

H_U: The Flutter framework can be utilized to produce comparable **user experiences** as native UI toolkits for mobile devices.

1.4 Methods

An architypal native mobile app has been chosen as a case study for the evaluation of both hypotheses H_P and H_U. Based on typical mobile application features (explained in detail in Section 3.1), *Kickdown* (Kickdown GmbH 2021) - an online car auction app - was chosen for this thesis. The app has already been developed for iOS by apploft and released to the App Store in February of 2021 (link to appstore). For the purposes of comparison, a Flutter equivalent has been developed mimicking the relevant subset of UI and functionality of the original app (see Section 3.1). Both the original and Flutter replica app are used for subsequent hypothesis testing.

2. A search for relevant articles has been conducted using Google Scholar, Sci-hub and IEEE Xplore.

1.4.1 Performance comparison

The assessment of the performance hypothesis H_P has been conducted by profiling specific performance metrics including CPU, GPU and memory usage for particular use case flows in the original and Flutter application. The profiling results have been analyzed using common statistical techniques. A further explaination of the measurement process is detailled in Section 3.2.

1.4.2 Usability comparison

The analysis of the usability hypothesis H_U has been evaluated by the means of semi-structured expert interviews. Specifically, study participants have been be asked to evaluate both the Kickdown iOS and Flutter application clone along various metrics.

A detailed explanation of the interview process is given in Section 3.3.

1.5 Scope & Limitations

The feature set of the implemented app is representative for most, but not every type of app (see Section 3.1). Therefore the results cannot be generalized to every type of possible app. However, they should be seen as indicators of Flutters value as a cross-platform framework for the archetypal mobile app. Furthermore, the deductively chosen methodology yields the potential of finding adjacent hypothesis which may be further explored by other researchers. Additionally, the usability study doesn't provide statistical significance due to its qualitative nature. Nevertheless, the depth of detail in expert interviews is much greater compared to quantitative methods, and unthought of considerations may be suggested by the interviewees. This research attribute is especially compelling given the current research state on the Flutter framework as mentioned above.

1.6 Thesis Structure

- TODO: Describe structure of thesis and summarize each chapter. Do this once the other chapters are actually written.

Chapter 2

Flutter

consider naming it theoretical background

- introduce native technology as a baseline

- create graphic with over of mobile development approaches

- revisit sections for PWA, Hybrid apps, runtime-based and interpreted apps of Springer article

or name it overview of cross-platform development approaches

2.1 Mobile Developement Tiers

2.1.1 Web Apps and Progressive Web Apps

Web apps are applications run and rendered in the browser. The underlying technologis are HTML, CSS and JavaScript with popular frameworks used for the development process such as Angualar (ref), React.js (ref) and Vue.js (ref). Web apps rely on the Browser and an internet connection. They do not have direct access to hardware capabilites such as the camera or the file system. Progressive Web Apps (PWAs) function similarly to web apps but provide additional capabilities such as offline use, locally cached data, push notifications, and the ability to add them to the home screen.

2.1.2 Hybrid Apps

Hybrid Apps utilize the same technologies as web apps, but they are rendered in a platform web view. Additionally they provide the ability to interact with native APIs through a platform bridge (see 2.1). Popular framework choices for building Hybrid Apps include Ionic and Cordova

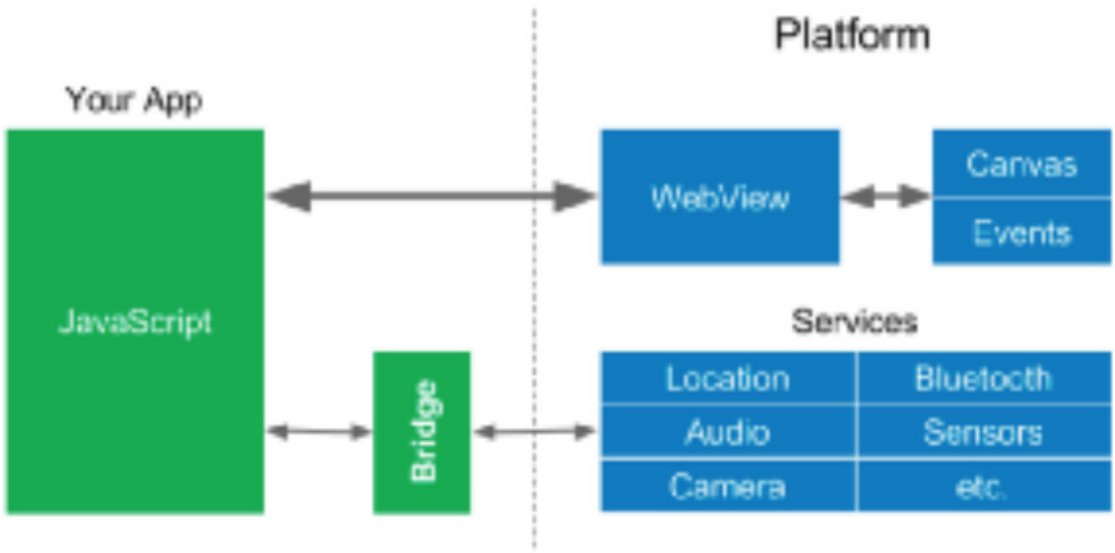

Figure 2.1: Hybrid architecture (image to be replaced).

Hybrid apps unlike web apps are shippable to official stores.

(ref: https://www.freecodecamp.org/news/a-deeply-detailed-but-never- definitive-guide-to-mobile-development-architecture-6b01ce3b1528)

2.1.3 Web Native Apps

Web Native Apps which are built with frameworks such as React Native (ref) or Native Script (ref) utilize the OEM components instead of web views. This is achieved by using JavaScript calls to a platform bridge which transpiles the JavaScript code into native platform code (see 2.2). The views are thus in Web Native Apps rendered with native technologies.

2.1.4 Cross Compiled Apps

Cross Compiled Apps utilize a compiled programming language such as C# (used for Xamarin) to generate byte or machine code which is then executed on the host platform. This machine code performs similar actions as web native apps in that platform APIs and OEM widgets are called (see 2.2). Flutter can also be classified as a cross compiled architecture. However, Flutter ships its own widgets and rendering engine instead of relying on the host platform (see 2.3). These technicalities will be explained in Section 2.2.

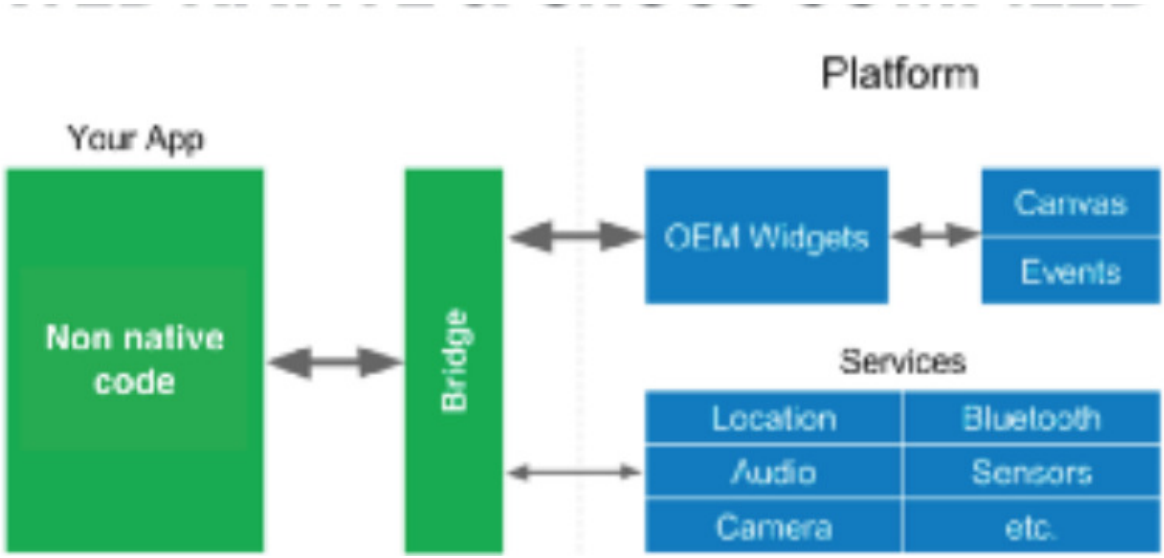

Figure 2.2: Web Native and Cross Compiled Architecture (image to be replaced).

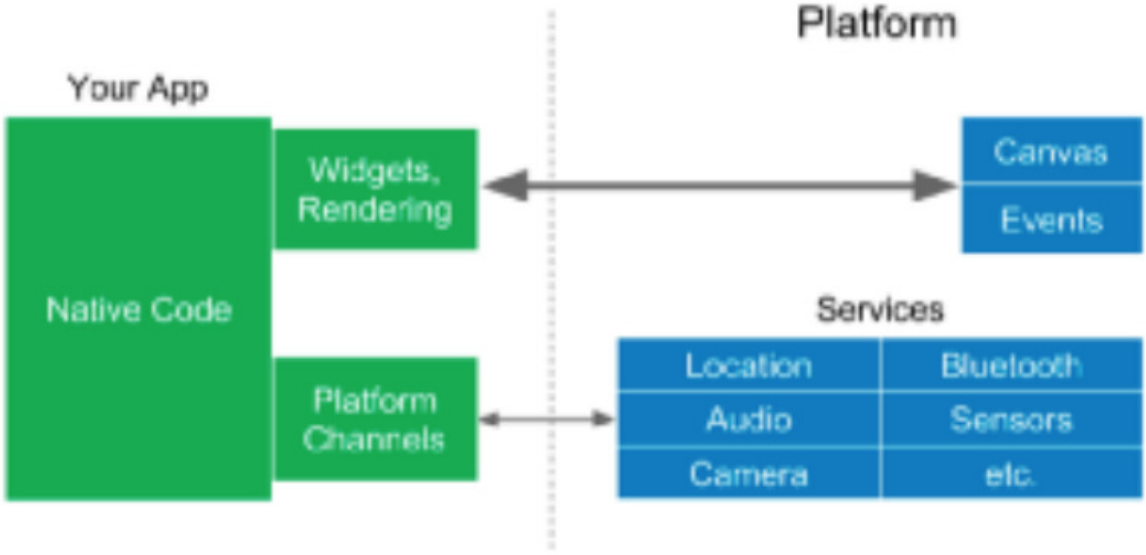

Figure 2.3: Flutter Architecture (image to be replaced).

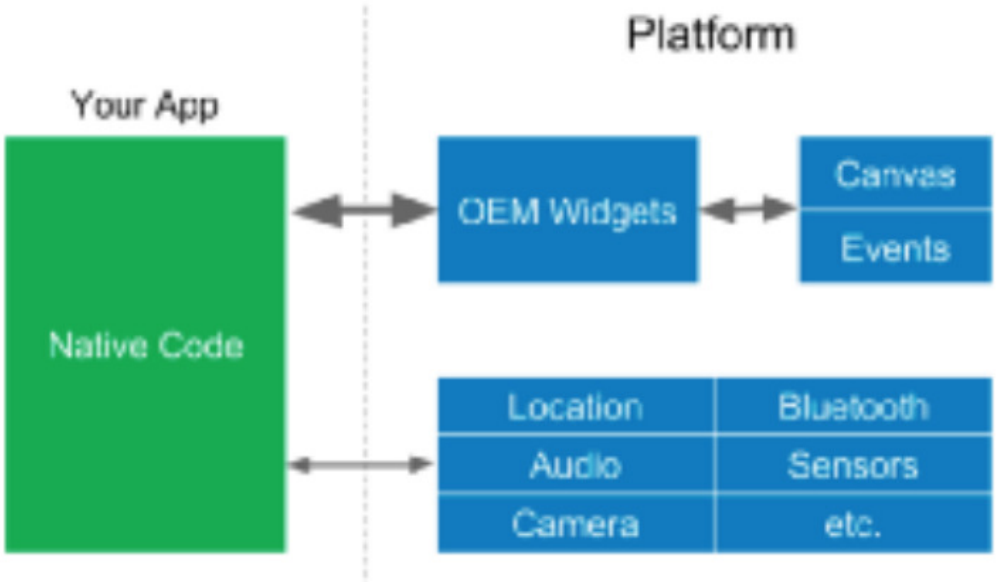

Figure 2.4: Native Mobile Architecture (image to be replaced).

2.1.5 Native Mobile Applications

Native applications directly communicate with the OEM components and APIs which are part of the operating system and thus tightly integrated into the platform. Making full use of hardware and software integration, native apps yield the highest performance and user experience.

2.2 Flutter Framework Architecture

Flutter's framework architecture is composed of three distinct layers: framework, engine and embedder. The framework acts as the top most layer which app developers interact with. It features the widgets, animation and gestures API as well as platform specific UI components delivered through the Material (Android) and Cupertino (iOS) package.

Below the framework layer lies the engine which unlike the framework isn't written in Dart, but in C and C++. Presumably, this design choice has been made to easily allow the production of native binaries. This act of cross-compilation is Flutter's technical value proposition to increase execution performance as stated in Section 1. The engine layer includes Skia - a 2D graphics rendering engine which is also used in the Chrome web browser.

At the bottom lies the embedder layer. Its sole purpose is to integrate the Flutter application into the platform-specific environment by providing native plugins and event loop interoperation. Additionally, app package formatting is provided in the same way as for native

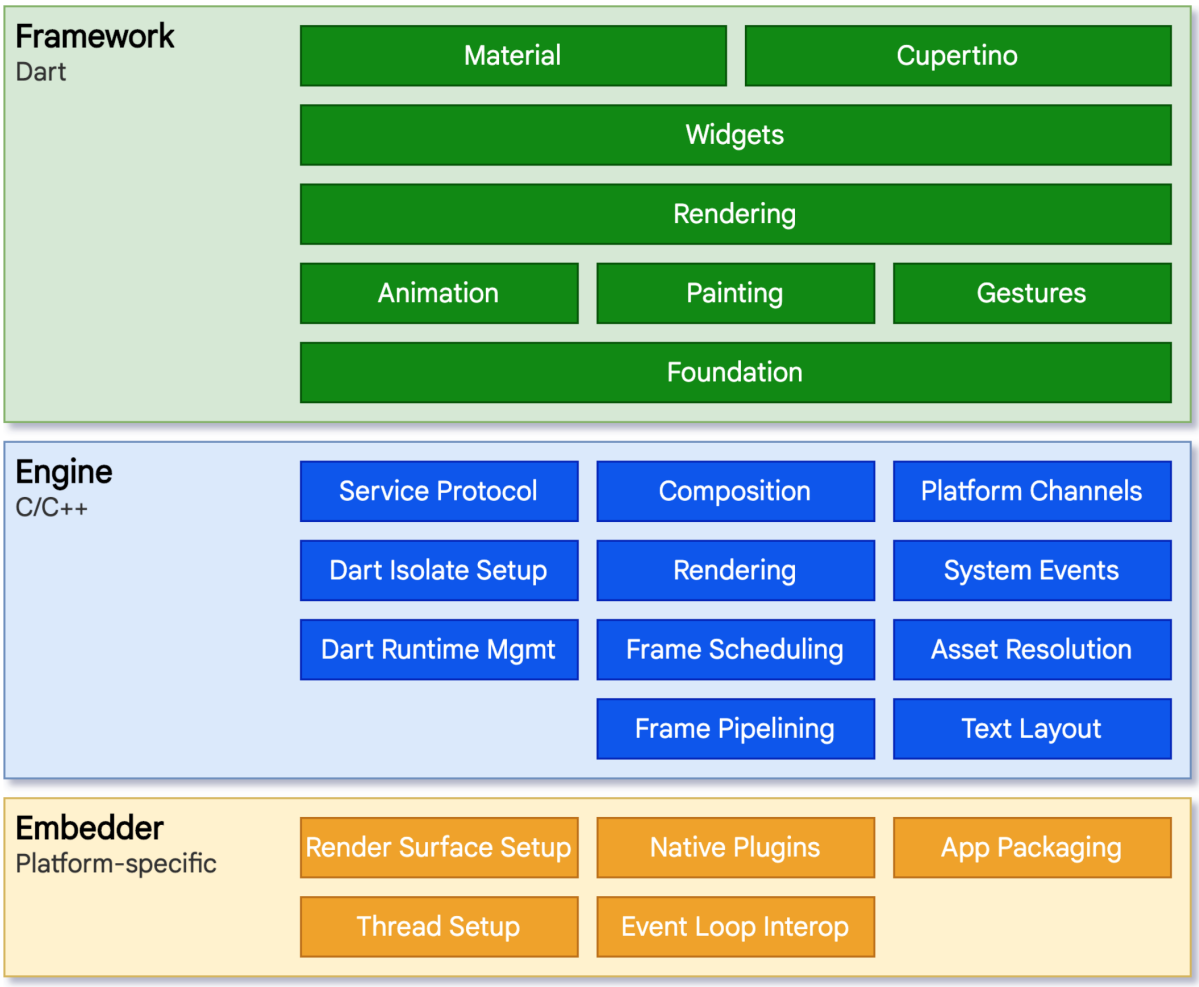

Figure 2.5: Flutter's Layered Architecture

apps. The host operating is thus not able to differentiate between a Flutter and a natively written app. Flutter aims to minimize application size as it's already shipping the entire framework including the rendering engine with each app which amounts to approximately 4.5 mb (https://flutter.dev/docs/resources/faq). Common app platform plugin functionality like camera access or webview interaction are extracted in separate packages. Networking, animations or other platform agnostic functionality is also bundled in packages.

(ref to include https://flutter.dev/docs/resources/architectural-overview).

2.2.1 Programming Language and Compilation

Flutter apps are written in Dart an multiparadigm programming language syntactically similar to Java. (https://dart.dev/overview#language)

During Flutter development, apps run in the Dart Virtual Machine with JIT compilation. This offers stateful hot reload which allows reload changes without needing to fully recompile

the app leading faster development iteration. For release purposes, Flutter applications are Ahead-of-Time compiled into native machine code including the Intel x86 and ARM instruction sets. (https://flutter.dev/docs/resources/architectural-overview).

2.2.2 Rendering and UI State

Flutter apps are fundamentally composed of widgets which declare structural, stylistic and layout elements for building the user interface. Each widget may have 0, 1 or multiple children which in turn create a tree structure of parent-child relationships. For example, a Column which has the purpose of laying out its children vertically has both a Text and an Image as its children. Both the Text, and Image widgets have no children and represent the leaf nodes in the tree. The Column is an invisible structural layout element while Text and Image are stylistic. [1] The entry point of every Flutter app is either a 'MaterialApp' or 'CupertinoApp' widget which also marks the root of the tree. Based on this tree structure Flutter can determine where and how elements should be drawn on screen, and instruct its graphics engine accordingly. This concept alone is not yet sufficient to enable modern mobile applications with complex UI changes or animations based on asynchronous events like user interaction. Widgets are mappings from state to a UI representation. When the state changes during runtime Flutter creates a new widget tree, diffs it against the old one and then redraws only its changes to the screen. This declarative approach simplifies UI development in the sense that the developer does not need to keep track of UI state which can grow exponentially with the increase of UI components on screen.

2.2.3 Method Channels

To utilize platform functionality like camera access, geolocation or other sensor data, Flutter communicates with the platform's native APIs via method channels (see architecture diagram). Common functionality is already provided by Flutter and third party packages, but custom platform channel functionality may be implemented as required. (reference Flutter doucmentation...)

2.3 Flutter's Limitations

Unlike other cross-platform frameworks, Flutter doesn't use OEM components, but its own components instead (widgets). When the underlying OEM component of the host platform changes

1. Technically both Text and Image do have an implicit single child widget which is created by the framework.

with an OS update, Flutters' framework needs to updated to resemble this new functionality. Flutter always has to continuously replicate these changes in their framework layer. As Google is using Flutter itself in multiple production apps (cite something), the company has an incentive to swiftly replicate OS features. Additionally, since Google also owns Android, coordination may be easier and not as surprising. The benefit though is that once the new widget is rebuilt in Flutter it can even be shipped to older operating systems.

Chapter 3

Methodology and Study Design

- mehtod triangulation to explore problem space by conducting quantitative

3.1 Feature Selection

- What are the necessary features that were selected and why? Typical mobile application features include the visual and interactive representation of information, horizontal and vertical scrolling, communication with a remote API, different means of navigation between screens as well as animations and transitions. - Why did I choose reduce functionality of original app for means of testing hypotheses?

3.2 Performance Comparison

On the one hand, the chosen are the underlying causes of more ephemeral metrics such as page load speed. On the other hand, they can be easily measured using software tools as explained in Section. - restate hypothesis - find measurement units (CPU, GPU, memory, etc.) - why are the measurement units relevant? - think of individual click journeys which will be measured that make use of the features mentioned in section above - try to find a capture-replay - which measurement technology will be utilized - Vorgehen zur Messung / Erhebung - repeat measurements multiple times (accuracy)

- calculate statistical figures between measurements (variance)

3.3 User Experience Comparison

- Contrarily to the verification of the performance hypothesis, the usability hypothesis will not be quantitatively, but qualitatively evaluated. - semi-structured expert interview - interviews will be conducted by author with individuals - interviewees are employees of apploft - every single interviewee has dedicated multiple years of work experience in the mobile application space as either a UX/UI designer, software engineer or project manager - interviews will be conducted via video call - goal is to get input on subjective user experience - guiding questions developed to support interview process based on commonly known UX framework

Chapter 4

Application Design and Implementation

- Thesis is not an [Ingenieurswissenschaftliche Arbeit] and thus application design and implementation will be explained from a perspective a necessary requirement to execute both the performance and UX comparison. - goal from development perspective was to get implement Flutter as close as possible to the original app in terms of UI - used as many out of the box cupertino widgets as possible for implementation - only deviated from above guideline when absolutely necessary - this was the case when implementing the following features - image gallery (zooming ability), image caching, custom modal iOS 13 presentation)s - Naturally, software implementation complexity may impede performance if unscalable algorithms or memory-inefficient datastructures are chosen. In order to prevent this bias to be introduced into the experiment, the Flutter app was implemented as closely as possible to the original iOS app. Todo: explain in what way.

- should I explain the general app architecture which was used?(MVVM + Services)

Chapter 5

Performance Comparison

- Use mean as measurement of tests for comparison purposes - Use multiple devices? - also measure startup time after initial download - CPU, idle-state RAM occupancy (PreRAM), busy-state memory occupancy (RAM), ComputedRAM - explain Xcode profiling tools - explain what the individual benchmarks actually mean - do benchmarking on multiple devices to be representative for all iOS devices -> find num of iOS devices which support currentIOS-1 - How is the device prepared before taking a measurement? - mean, min, max, std deviation - perform Anova or Tukey post-hoc test to determine statistical significance b/w both groups - visualize data using R to examine it properly (boxplot, etc.) - The performance analysis is conducted by comparing X tests for each performance metric, on both the Flutter and iOS app - TODO: Explain which tools etc. were used for comparison (Xcode Tools + Flutter Profiling) - look at specific actions within app (like scrolling, opening page) for comparison - use "Instruments" from Xcode - read Springer paper to explain potential performance differences -> Time profiler tool for CPU usage -> Core Animation tool for GPU FPS -> Allocations tool for memory usage

Chapter 6

User Experience Comparison

6.1 Target Group

- apploft employees -> app experts

Chapter 7

Summary

- What are my key findings? What is their significance and implications

7.0.1 apploft

- Byproduct of analysis was the development of a decision guide when choosing between the development of a native and Flutter mobile app

- maybe point to future research into Flutter - Reference back to goal in introduction - What are the limitations of my research - Future Research questions - Outlook

Bibliography

Andreesen, Marc. 2007. *Product/Market Fit.* https://web.stanford.edu/class/ee204/ProductMarketFit.html.

Angulo, Esteban, and Xavier Ferre. 2014. "A Case Study on Cross-Platform Development Frameworks for Mobile Applications and UX." *Proceedings of the XV International Conference on Human Computer Interaction,* no. 27 (September): 1–8.

apploft GmbH. 2021. *apploft.* https://www.apploft.de/.

Biørn-Hansen, Andreas. 2019. "An Empirical Study of Cross-Platform Mobile Development in Industry." *Wireless Communications and Mobile Computing.*

Corbalán, L. 2019. "Cloud computing and big data." Chap. A study of non-functional requirements in apps for mobile devices. Springer International Publishing.

Ebone, A., Y. Tan, and X. Jia. 2018. "A Performance Evaluation of Cross-Platform Mobile Application Development Approaches." *IEEE/ 5th International Conference on Mobile Software Engineering and Systems (MOBILESoft)* (November).

Facebook. 2021. *React Native.* https://reactnative.dev/.

Google Inc. 2020. *Flutter Homepage.* https://flutter.dev/.

———. 2021a. *Flutter SDK Releases.* https://flutter.dev/docs/development/tools/sdk/releases.

———. 2021b. *Widget Catalog.* https://flutter.dev/docs/development/ui/widgets.

IDC. 2021. *Smartphone Market Share,* December. https://www.idc.com/promo/smartphone-market-share/os.

Ionic. 2021. *Ionic Framework.* https://ionicframework.com/.

JetBrains s.r.o. 2021. *Cross-platform mobile frameworks used by software developers worldwide in 2019 and 2020.* Technical report. Statista. https://www.statista.com/statistics/869224/worldwide-software-developer-working-hours/.

Kickdown GmbH. 2021. *Kickdown.* https://www.kickdown.com/.

Mercado, I. T., N. Munaiah, and A. Meneely. 2016. "The impact of cross-platform development approaches for mobile applications from the user's perspective." *Association for Computing Machinery Journal.*

Microsoft Corp. 2021. *Xamarin.* https://dotnet.microsoft.com/apps/xamarin.