Fakulteten för teknik och samhälle
Datavetenskap

Examensarbete
15 högskolepoäng, grundnivå

Exploring End User's Perception of Flutter Mobile Apps

Utforska slutanvändares upplevelse av Flutter Mobilapplikationer

Ola Dahl

Examen: Kandidatexamen 180 hp
Huvudområde: Datavetenskap
Program: Applikationsutveckling
Datum för slutseminarium: 2019-02-07

Handledare: Mia Persson

Abstract

When developing mobile applications, developers need to make a decition: either develop multiple native applications for different operating systems or developing one app that is cross-platform compatible. Many technologies for creating cross-platform applications have emerged over the years, and new techologies are released every year. One such technology is Flutter, which is a mobile application SDK (Software Development Kit). Flutter promises the ability to build native applications on iOS and Android that achieve native performance.

To answer if this statement is true, from a user's perspective, two visually identical mobile applications were developed. One application was developed using the Flutter SDK and one using the native Android SDK. The applications were then evaluated by users who were asked about their perception of using the applications. Users were also asked if they preferred any of the applications.

The result show a clear difference between the applications in user's perception of speed, but not in their perception of appearance. Users perceived the speed of the native application to be faster than the speed of the Flutter application, but perceived the apperance of both applications to be equal. A majority of users said that they preferred the native application and only 10% said that they preferred the Flutter application.

Sammanfattning

Vid utveckling av mobila applikationer måste utvecklare göra ett val: antingen utveckla flera nativa applikationer för olika operativsystem eller utveckla en applikation som kan köras på flera plattformar. Många teknologier för att skapa plattformsoberoende applikationer har uppstått genom åren, och nya teknologier släpps varje år. En sådan teknologi är Flutter, som är ett SDK (Software Development Kit) för mobila applikationer. Flutter lovar möjligheten att bygga nativa applikationer för iOS och Android som har nativ prestanda.

För att svara på om detta påstående är sant, från en användares perspektiv, utvecklades två visuellt identiska mobila applikationer. En applikation utvecklades med Flutters SDK och en med Androids nativa SDK. Applikationerna utvärderades därefter av användare som svarade på hur deras upplevelse var av att använda applikationerna. Användarna svarade också ifall de föredrog någon av applikationerna.

Resultatet visar att det finns en skillnad mellan applikationerna i användarnas upplevelse av hastighet, men inte i deras upplevelse av utseende. Användarna upplevde att hastigheten var snabbare hos den nativa applikationen jämfört med Flutter-applikationen, men upplevde utseendet av applikationerna som lika. En majoritet av användarna sa att de föredrog den nativa applikationen och endast 10% sa att de föredrog Flutter-applikationen.

Contents

1 Introduction

The smartphone market has consistently grown ever since the introduction of the first iPhone in 2007. The number of units shipped worldwide has gone from 173 million in 2009 to 1.4 billion in 2015 and a forecast of 1.8 billion units sold in 2018 [27]. The main reason for the success of the smartphone is the popularity of mobile applications (hereafter referred to as "app"). The app market today consists of more than two million apps and downloaded billion of times from app stores such as the Google Play Store and Apple App Store [25].

The development of an app is usually done with a Software Development Kit (SDK) provided by the creator of the mobile operating system. The apps made with these SDKs are referred to as native apps. Native apps are highly optimized and provide a user experience tailored for the operating system they are running on [19]. However, a challenge for app developers is that these SDKs use their own programming language and application programming interface (API) [19]. This demands that developers have to write an app for each platform they want to target. This approach poses many challenges for developers such as in-depth knowledge of each platform and their SDKs. This increases development and maintenance cost, which often is not a viable option for many developers [6].

Apps can also be developed using cross-platform development approaches. This approach enables developers to implement the app using a single codebase that can run on multiple platforms [9]. The cost of development is often reduced and offers other benefits such as faster development and release overheads [20].
Cross-platform approaches are not without its flaws however. Studies have shown that users have an overall more negative perception of apps developed using cross-platform approaches[12][11]. Mercado et al. used natural language processing to classify 787,228 user reviews for 50 Android and iOS apps into four quality concerns: performance, usability, security, and reliability. The result showed that apps using cross-platform approaches received a higher rate of user complaints than other apps [12]. Malavolta et al. did a similar study with 3,041,315 user reviews for 11,917 apps on the Google Play Store. The findings concluded that users perceived native apps better concerning performance and presence of bugs, and when dealing with platform-specific features [11].

One crucial aspect of the user perception of an app is user satisfaction. The ISO 9231-11 standard defines satisfaction as the "Freedom from discomfort, and positives attitudes towards the use of the product"[28]. To provide good user satisfaction, you need a user interface that the user feel comfortable interacting with. Users often feel comfortable in apps that resemble native apps [9]. When choosing a cross-platform technology, it is therefore imperative that it provides a native look-and-feel.

Flutter is a new cross-platform technology that promises high-performance apps that feel natural on different platforms [26]. It provides widgets for Android and iOS that delivers a highly native experience. This new technology could therefore be a good candidate to create cross-platform apps that provide high user satisfaction.

This study explores the user satisfaction of Flutter apps. Two prototype apps was developed. One app was developed using the Flutter SDK and the other using the native Android SDK. A user evaluation was performed where the user satisfaction of both apps was measured. The results show that native apps still provide higher user satisfaction than Flutter apps.

1.1 Research Question

This study aims to evaluate a user's satisfaction while interacting with an app developed in Flutter. The goal is to determine how the user satisfaction compares to that of native apps. If the user satisfaction of a Flutter app is equal or comparable to that of a native application, this will provide valuable feedback to a developer or company thinking about developing an app in Flutter. The aim is therefore to answer the following question:

> Are there any differences in the user's perceived experience, with respect to the quality attribute user satisfaction, between a Flutter and native Android application?

2 Previous Research

As Flutter is a new technology, there has not been any scientific studies on the user perception of Flutter apps. There have however been studies on the user perception of cross-platform apps. The findings from the most relevant studies are presented below.

Malavolta et al. [11] investigates the end user perception of hybrid mobile apps on the Android platform with respect to native apps. The authors presented an empirical study where they analyzed 11,917 apps and their corresponding 3,041,315 user reviews. They developed a data-extraction tool to classify the type of an app. The tool analyzes the information in the Android Package file and can get a series of information that is used to classify if the app is a hybrid app or a native app. The tool also extracted the size of the app. The review data from these apps was then extracted and grouped into the categories *perceived value*, *perceived performance*, and *perceived bugginess*.

The findings showed that hybrid apps were most present in a category the authors are calling *data-intensive apps*. These are apps that focus on the collection of data and performing simple interactions with data items. The categories with the lowest number of hybrid apps were those that require closer integration with the Android platform such as *Photography* and *Music & Audio*. Further, the authors concluded that in terms of *user perceived value*, native apps had a slight advantage and a more significant advantage in apps that are not data-intensive. In terms of *user perceived performance*, native apps also have an advantage with 11.30 points in favor of native apps. The highest difference observed in the study were in terms of *user perceived bugginess* that was summed up to 18.42 points with a higher value for hybrid apps. The authors conclude that there is still room for working on hybrid development frameworks and that they need to improve their performance and testing practices.

Mercado et al. [12] did a similiar study where natural language processing was used to classify 787,228 user reviews of 50 apps on the Google Play Store and Apple App Store. The reviews were classified into complaints of the four quality concerns *performance*, *usability*, *security* and *reliability*. The included apps in the study were 47 cross-platform apps and three native apps. The cross-platform apps were made with different tools and were divided into the app types Hybrid and Interpreted/Generated. The native apps were included in the study to serve as a baseline in evaluating cross-platform apps.

The main findings showed that users had a more negative perception of the performance, reliability, and usability of apps developed with hybrid approaches compared to apps developed using the native approach. Integrated/generated apps had a lower complaint density than hybrid apps in all quality concern, except security. The authors conclude that hybrid apps tend to have more user complaints than both native and integrated/generated apps.

The authors also focussed in particular on the iOS Facebook app. This app changed the development approach from a hybrid app to a native app. The results show that after the change in the development approach, the user perception improved for the performance and reliability concerns. The user perception of security and usability declined after the change. The authors argue that the decline in the perception of usability could be an effect that when the app matures, it gets more features. This could make the app harder to use. The decline in the perception of security could be due to a change in permission handling of iOS 6, making security more visible to the user.

When developing mobile applications there are many aspects to consider. Heikötter et al. [9] did a comparison between different cross-platform platforms and compared them to native apps. From the infrastructure perspective *look and feel*, *GUI design* and *application speed* were important criteria when evaluating development approaches. It is argued that a native look and feel is of importance and that users seek apps that resemble native apps.

The result shows that the majority of cross-platform apps suffer from bad performance, both in start-up time and in runtime. Cross-platform apps were generally slow when interacting with them. The authors further mean that the app appearance is an aspect where cross-platform apps fall short in particular. Further improvements are needed to alleviate this aspect.

Further studies have invistigated tried to determine the user perception of apps based on user reviews. Ali et al. [1] investigates how hybrid apps compare to native apps, in terms of user-perceived ratings. The study is based on a dataset of 15,512 hybrid apps and examines if the choice of a cross-platform tool influences how users perceive it. The article introduces a metric called Aggregated User-perceived Rating (AUR). AUR combines the number of reviewers for the app, the average star rating for the app, the average number of reviewers for all apps in the dataset, and the average number of star ratings for all apps in the dataset. Essentially, if an app has few reviews, it can't be trusted to accurately measure how users perceive it.

The findings show that the AUR scores for hybrid apps were very close to the AUR scores for native apps. Hybrid apps even scored higher in some categories. This is interesting as it does not align with the findings from other articles. It could be that the measurement techniques that the article uses are different from other articles. Most other articles used classification of reviews given by users. This article based its results only on the star ratings.

Further findings showed that there was a difference in AUR scores depending on the cross-platform tool used. Apps made with PhoneGap was best perceived by users, followed by Titanium and AdobeAir. The result is in line with findings in other articles and the findings in [24].

With the the purpose of evaluate different cross-platform development approaches, Xanthpoulos et al. [24] performed a comparative analysis between the cross-platform app types *web*, *hybrid*, *interpreted* and *generated* apps. Interesting criterion used in the analysis was the *User-perceived performance* and *User interface look and feel* compared to a native app. The authors evaluated if the performance, as perceived by the user is low, medium or high compared to a native app.

The findings show that web apps have low user-perceived performance, hybrid and interpreted apps have medium user-perceived performance and generated apps have high user-perceived performance. Interpreted apps could have comparable performance to generated apps. This is however dependent on the implementation of the interpreter.

Regarding the user interface look and feel, interpreted apps delivered the best result while web apps delivered the worst result.

The authors point out the importance of conducting tests in real-life conditions. Aspects such as the evaluation of user interface responsiveness and application speed are given as examples for future research.

To investigate if there is a difference between development approaches, Que et al. [22] compared the user interface and performance of hybrid and native apps.

It is concluded that the user interface is one of the most important aspects of an app. Native apps have the capability to achieve good appearance and provide excellent user interactions. However, the authors argue that hybrid apps also are capable of creating a good appearance that homologous to the native platform in some cases.

To evaluate the performance, two functionally identical Android apps were developed using the native Android SDK and using the Cordova hybrid app framework. The apps are tested using a cloud testing platform, which uses over 200 different real Android devices when running the tests.

The findings show that the performance of the hybrid app is weaker in every aspect. Notable findings were that the hybrid app consumed 106% more CPU resources and used 73% more memory space. Another interesting finding was that the number of device features implemented in the app (such as GPS, camera, and accelerator) had a substantial effect on the memory occupancy in the hybrid app. When adding the number of device features in the native app, the memory occupancy increased only slightly. When adding the same device features in the hybrid app, the memory occupancy increased drastically.

It is clear from previous research that much development has been done in the field of cross-platform apps. One interesting aspect shown is that users indeed can tell the difference between development approaches (see e.g. [1, 11, 12]). Conclusive for all aforementioned studies was that native apps had an advantage in both user perception and in performance.

Many aspects are evaluated such as security, bugginess, and reliability. Two aspects that occur frequently, however, is the performance and look and feel an app (see e.g. [9, 23, 21]). All the aforementioned studies show that native apps generally have better performance, both in user-perceived performance and in actual performance. Good app appearance is also an aspect that many authors mean is lacking in cross-platform apps (see e.g. [9, 21]).

As Flutter is promising native performance and a native user interface, it would be relevant to evaluate if any improvement has been made in terms of user perception of performance and app appearance.

3 Method

In this chapter, we present the method used to conduct the research. Moreover, a motivation for the use of this method is presented and a discussion of how the method was implemented. Then follows a discussion regarding the limitations of the research.

3.1 Mixed Method

This study used an explanatory mixed method, to achieve a stronger understanding of the problem. Creswell [5] explains mixed methods as the collection of both quantitative (close-ended) and qualitative (open-ended) data. It uses established methods for data collection, data analysis and interpretation of both quantitative and qualitative data. Both data types are integrated into the analysis where qualitative data explains the quantitative data.

Creswell presents a number of reasons why a mixed method is a good research methology:

- It draws from the strength of both quantitative and qualitative methods, minimizing the limitations of both approaches.
- It explains quantitative results with qualitative data collection and analysis.
- It augments a survey by incorporating the perspective from individuals.

The reasons above align with the aim of the study and a mixed method was therefore considered suitable to achieve the results that answer the research question.

3.2 Study Content

The study involved different parts. The contents of these parts are discussed here.

3.2.1 Application Development

To evaluate both systems two Android prototype apps were developed. One app (App A) was developed using the native Android SDK, and the other (App B) using the Flutter SDK. App A and App B were identical in terms of functionality and user interface. The purpose of the apps was to evaluate the apps from the user perspective to determine if there was a difference in user satisfaction.

The apps contained four parts *buttons*, *input*, *list* and *weather app*. The first three parts included easy tasks where the user interacted with commonly used user interface elements. The fourth part provided a concrete example of how an app made in these systems behave. It was important to provide a concrete example because users interact with an app in this way in the real world.

The reason for developing the apps in-house and not evaluating already existing apps has multiple explanations. One is that Flutter is a cutting edge technology and not many apps have been developed yet. Therefore it was hard to find suitable apps to evaluate. Another explanation is that most apps only contain a subset of all available interface elements. Therefore, it would cover a smaller set of interaction opportunities. When developing the apps in-house, there was better control of the implementation and minimized the number of error sources.

3.2.2 User Satisfaction

User satisfaction is part of a broader concept called usability. The usability of a product is very important, especially for mobile apps because of the way they are delivered to customers. Mobile apps are published on the Apple App and Google Play Store. The success of the apps are strongly dependent on the reviews it gets by its users. Good usability in mind is therefore crucial for an app to achieve commercial success.

The ISO definition of usability consists of three distinct aspects [28]:

1. Effectiveness. Accuracy and completeness with which users achieve specified goals.
2. Efficiency. Resources expended in relation to the accuracy and completeness with which users achieve goals.
3. Satisfaction. Freedom from discomfort, and positive attitudes towards the use of the product.

However, usability is most often used when comparing different designs of user interfaces. The purpose of a usability metric is to measure the success rate, the time a task requires, the error rate, and the users' subjective satisfaction [16]. To benefit from these measurements, a different design needs to be used.

The purpose of this study was to compare the user perception of apps based on the development tool: the Flutter SDK and the native Android SDK. Therefore, the design of the apps had to be equal. If the apps had different designs there was a risk of a shift in user perception based on the design. The apps were therefore created as equal as possible from a design perspective.

Chin et al. [4] mean that user satisfaction may be a critical measure of a system's success. Even though a system is evaluated positively on every performance measure, it may not be used due to dissatisfaction with the system and its interface. When there is little change in design, user satisfaction is regarded as a good metric to assess users' perception of systems [4].

3.2.3 Measuring User Satisfaction

To measure user satisfaction, one needs to evaluate the user interface in some way [14]. Nielsen et al. [14] recommend using a heuristic evaluation technique. This is done by simply looking at the interface and trying to coming up with an opinion about the interface. This way of evaluating interfaces is shown to be effective when the result of more than three evaluations is aggregated to a single result [14].

To operationalize the measurement of user satisfaction simple satisfaction questions can be used. Nielsen [15], gives the example:

> "On a 1-7 scale, how satisfied were you with using this website (or application, intranet, etc.)? The average of the scores will give the average satisfaction measure."

The study used this way of measuring user satisfaction. To generate data, a user evaluation was performed in conjunction with a survey. Test users were instructed to interact with each part in both App A and App B. They were told to notice any thoughts and feelings they had while interacting with the app. After each part was finished the user answered questions for that part in the survey.

One advantage of performing the survey in conjunction with each part was that the questions were related to the interaction the user just had. This limited the possibility that the user perception was affected by any other interaction in the app. In total, the user evaluated 12 different screens containing different aspects of the app.

3.2.4 Survey

The survey questions were created following [7] and kept short and simple. They were formed after the recommendation by Nielsen [15], with questions measuring satisfaction. Chin et al. [4] also recommends this way of measuring user satisfaction, and propose that different questions could be used to cover different aspects of the system. As seen in previous research, the perceived appearance and perceived performance of cross-platform apps is frequently mentioned as a weakness. Therefore, this study evaluated the user perception of these two aspects.

In the first part of the survey, quantitative data was generated. It used a scale from 1-7 with questions regarding the user's perception of the appearance and speed of the user interface.

The users also answered if they had a preference for any of the apps. The second part generated qualitative data. Users were able to explain what aspects affected their interaction. This gave further insight into what aspects of the apps that were perceived differently, and also confirmed the quantitative result.

A covering letter was included with survey and can be seen in Appendix A. It was formed after the recommendations described in [7], and described the aim of the study and instructed the participants how to perform the evaluation. The participants were informed that their participation was voluntary and that all questions would be treated anonymously.

Ten participants in ages between 22-66 were involved in the evaluation. In a usability study, it is generally recommended to include five test users in order to achieve accurate results [13]. Nielsen [13] means that there is little additional benefit in using a large number of users. As user satisfaction is a subset of usability it can be argued that a similar number of users would be appropriate. In a survey of 217 participants, the average number of test users was eleven [13]. Ten participants were therefore considered sufficient in this study.

The occupations of the participants were varied. Seven of the participants were students and three had full-time employment. None of the participants had a background in computer science or any knowledge of how mobile apps is developed.

3.2.5 Pilot Study

A pilot study is performed to ensure that the questions asked will get the result that the query constructor intended. It is performed on a small set of people that can test if the questions are understandable [7].

In this study, two persons evaluated the preliminary questions. The questions were modified together with these persons, to clarify the format of the survey. The questions were modified to focus on the comparison between the apps, rather than each app individually. For each task, both apps were used before answering the questions. This made the questions easier for the respondents to understand, as a direct comparison between the apps was more apparent to the users. A finished version of the survey was then developed. The pilot study was performed over two days 25-11-2018 – 26-11-2018.

3.3 Limitations

The following limitations was made:

- The study only focused on the user satisfaction aspect of mobile apps.
- The study only evaluated the user satisfaction of the attributes appearance and speed.
- The study only compared the Flutter application in the context of Android.

To measure user perception many metrics can be used. Usability is most often used when comparing different designs of user interfaces. The purpose of a usability metric is to measure the success rate, the time a task requires, the error rate, and the users' subjective satisfaction [16]. To benefit from these measurements, a different design needs to be used. When there is little change in design, user satisfaction is regarded as a good metric to assess users' perception of systems [4]. Therefore this study only focused on the user satisfaction aspect of mobile apps.

As seen in previous research there are many attributes in an app that could affect user satisfaction. However, user perception of appearance and speed are two aspects that frequently are mentioned. For example, Heitkötter et al. [9] argue that the look and feel, and application speed is two very important criteria for the success of an app. This points to the perceived appearance and speed is important, and worth investigating.

The study also only compared Flutter apps in the context of Android. Even though the main benefit of using Flutter is the ability to create apps for both Android and iOS simultaneously, iOS apps were not included in this study. The reason was that the development tools required was not available at the time of writing the thesis.

4 Background

This section presents a background of different cross-platform approaches and an introduction to Flutter.

4.1 Cross Platform Development

The number of cross-platform technologies has seen strong growth in the last couple of years. The motivation for using these technologies is that they offer easier and more cost-efficient development [24]. They differ from traditional development technologies in that developers can implement their app once, and execute it on multiple platforms [9]. There are four main categories of apps created using cross-platform technologies: *web*, *hybrid*, *interpreted* and *cross-compiled* apps [24].

4.1.1 Web apps

The first approach to creating cross-platform apps started with web apps. These apps take advantage of the fact that every modern mobile operating system ships with a web browser. Web apps are developed using HTML5, CSS3 and JavaScript [24]. The apps are mobile optimized and hosted on a remote server. The user can then access the app using a standard web browser. Because the code conforms to web standards, the app can deliver an equal experience on multiple platforms [10]. This is possible due to broad compatibility with the WebKit rendering engine. The WebKit rendering engine is an open-source project led by Apple and Google that is the most complete implementation of the HTML5 standard [10].

4.1.2 Hybrid apps

Hybrid apps are a variant of a web app that combines web technologies and native functionality [9]. Hybrid apps are developed as web apps that are wrapped in a native container. The advantage of using this approach is that the app can be distributed through an app store but developed as a single code base. The hybrid app is installed on the device and can access hardware functionality through specialized APIs [21], expanding the features which the app can provide. However, hybrid apps are often inferior in performance compared to native apps. The user interface is reusable but does not provide a native look and feel which many users seek [20].

4.1.3 Interpreted apps

Interpreted apps are an approach where native code is generated through an interpreter. The source code is interpreted at runtime on different platforms, and native code is generated for the specific platform [20]. The user interface is implemented using native interface components for each platform which are made available through an abstraction layer [20]. These apps have the advantage that they can be distributed through an app store and also provide a reusable native user interface [21]. The main disadvantages are that the development is dependent on the feature set provided by each framework. This means

that platform-specific features, such as user interface components, are only available if supported by the selected framework. Interpreted apps may also suffer in performance due to the run-time interpretation of the code.

4.1.4 Cross-Compiled Apps

Cross-compiled apps convert source code to native binaries at compile time. Developers can write source code in a common language (such as Java or C#) and the cross-compiler generates executable code for each specific platform [24]. Developers can thus reuse a large part of the code base. The advantage of this approach is that cross-compiled apps achieve overall high performance. They can also use all available user interface components. The main disadvantage is that the user interface components are not reusable and need to be written for each platform [20].

4.2 Flutter

This section introduces Flutter and discusses where it comes from, how the framework is structured and what makes Flutter different from other cross-platform frameworks.

4.2.1 Introduction to Flutter

Flutter is a cross-platform framework for developing apps that can run on Android, iOS and Google's next-generation operating system Fuschia [23]. It was created by the Google Chrome browser team and publicly released in 2016. The reason for creating Flutter was to evaluate if rendering could be more efficient if traditional layout models were ignored. This resulted in a different way of thinking about layout. Instead of letting the layout be determined by a large set of rules, each widget in the layout would specify its individual set of rules [17].

4.2.2 Rendering

The way Flutter does rendering is different from other cross-platform frameworks. Other frameworks rely on either WebViews (PhoneGap, Cordova, Ionic) or native OEM widgets (React Native). The app code in these frameworks is written in Javascript. For the app code to communicate with the platform, it will need to implement a Javascript bridge. The bridge will do context switches between the app code and the platform, as seen in Figure 1. This enables cross-platform apps to access device functionality such as native widgets [23].

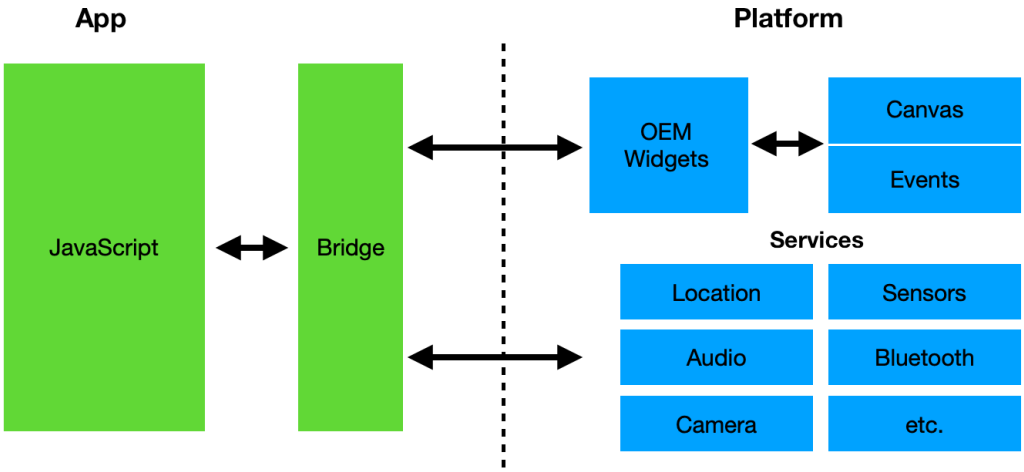

Figure 1: Rendering in React Native

This context switch is not performed instantaneously however. If platform resources are accessed frequently, such as OEM widgets, the app can have performance problems [17]. Flutter tries to solve this problem by using an ahead of time (AOT) compiled language called Dart. The communication between app code and platform can therefore be done without a context switch [23]. The source code is also compiled to native code, as can be seen in Figure 2. Widgets are also part of the app, reducing the number of times the app need to communicate with the system.

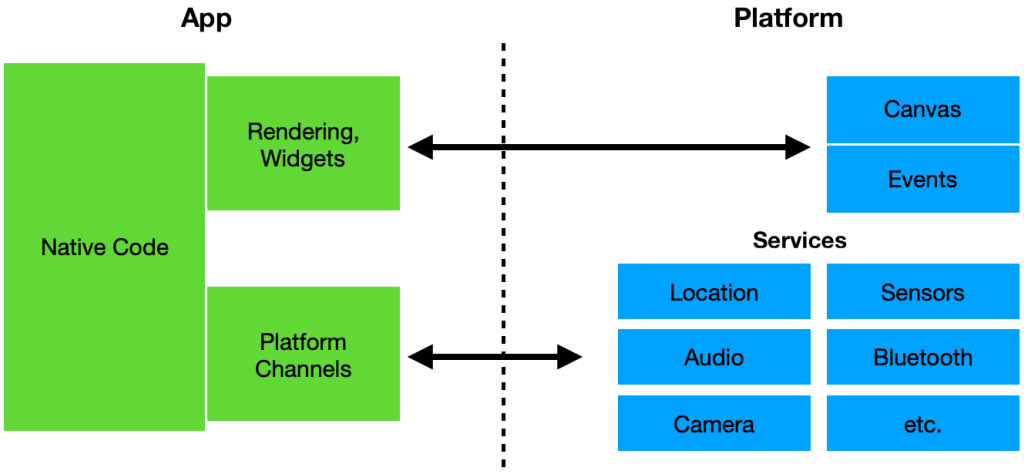

Figure 2: Rendering in Flutter

4.2.3 Dart

Flutter apps are using the Dart programming language. Dart is developed and maintained by Google. It was developed as a successor to Javascript and incorporates many of the features in the ES7 Javascript standard [23].

As mentioned before, Dart is compiled ahead of time. This allows for faster execution and avoids the Javascript bridge. The code still needs an interface to communicate with the platform code. This is however much faster than the Javascript bridge. Flutter also do not utilize the platform OEM widgets, reducing the number of times communication with the platform occurs [18].

Dart can also be compiled using a just in time compiler. This is done during development and allows for rapid development cycles. Flutter take advantage of the just in time compiler using a feature called *hot reload*. Hot reload allows code changes to be reloaded and injected on the device very fast. This allows for very efficient development of Flutter apps. The downside is that the app will be less efficient during development [18].

4.2.4 Widgets

Flutter doesn't use the native OEM widgets, but instead provide its own widgets. The widgets are not rendered by the system. Instead, the app itself includes the renderer. Flutter only requires a canvas to render the widgets on the device screen. It also needs access to services (camera, location) and events (timers, touch events) [17].

This allows Flutter to decide when and how the widgets will be rendered. It also allows the widgets to be more customizable and extensible. The downside of this approach is that the app size increases, as it needs to host both the widgets and renderer within the app [23].

4.2.5 Layout

The layout is implemented differently in Flutter compared to other frameworks. The layout rules are what define the position and size of widgets that are rendered to the screen. The rules are moved from the application scope down to the widgets themselves [17]. Position is controlled using *position widgets*. The position widgets are not visible on the screen. Its purpose is to control the position of another widget. For example, the position of widgets can be controlled using a *Center* and *Column* widget, as seen in Figure 3.

```
Center(
  child: Column(
    children: [
      Text('Hello, World!')),
      Icon(Icons.star, color: Colors.green)
    ]
  )
);
```

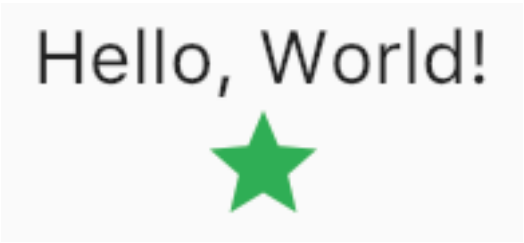

Figure 3: Rendering in Flutter

5 Result and analysis

In this section, the results from the user evaluation will be presented.

Apps

In this study, two prototype apps were developed. One app (App A) was developed using the Flutter SDK, and the other (App B) was developed using the native Android SDK. Both apps include the same functionality and design.

The apps are divided into four parts. The first three parts are *Buttons*, *Input* and *List*. These parts include a number of simple UI components. The purpose of these three parts is to evaluate if there is a difference in user satisfaction between systems while interacting with simple UI components.

The fourth part, *Weather-app* is a more concrete example of how a basic app can look and behave in these systems. It includes the ability to search for cities and get the weather information for these cities. The purpose of this part is to evaluate if there is a difference in user satisfaction between systems while using an app that is familiar to many users.

Figure 4 and Figure 5 show the start screen of both the Flutter and native Android app.

Figure 4: Start screen in the Flutter app.

Figure 5: Start screen in the Native app.

Questions

Ten persons participated in the user evaluation. After the participants had finished each part in both App A and App B, they answered questions about their interaction. The questions were the same for all parts and included a quantitative part and a qualitative part. Following are the questions asked for each part:

- How satisfied were you with the appearance of App A?
- How satisfied were you with the appearance of App B?
- How satisfied were you with the speed of App A?
- How satisfied were you with the speed of App B?
- Do you prefer any of the apps?
- Comment. Explain what factors (if any) that affected your interaction.

5.1 Respondents

This section presents background information for the respondents that participated in the study.

5.1.1 Gender

There were four men and six women that participated in the study (see Figure 6 below).

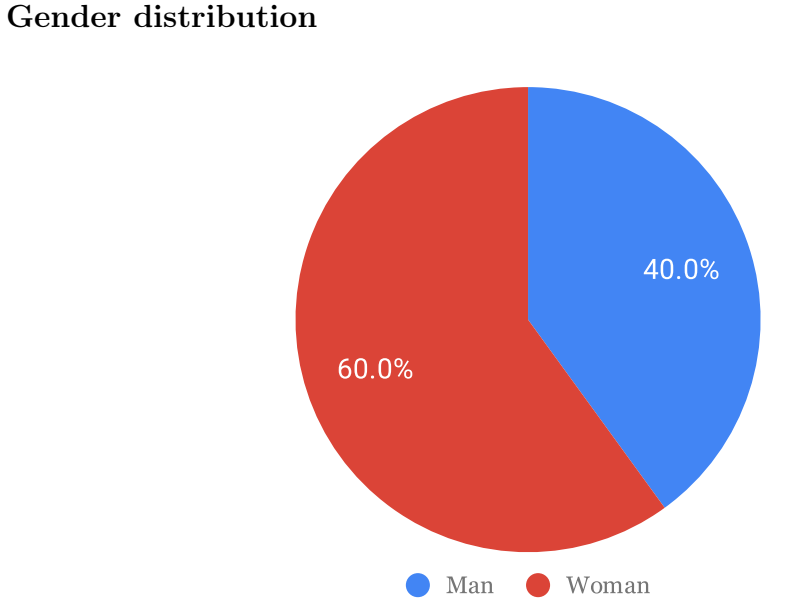

Figure 6: Gender distribution in the study.

5.1.2 Age

A majority of the respondents were in the age span 20-30. Two were in the age span 30-40, and one respondent was over 60 (see Figure 7 below).

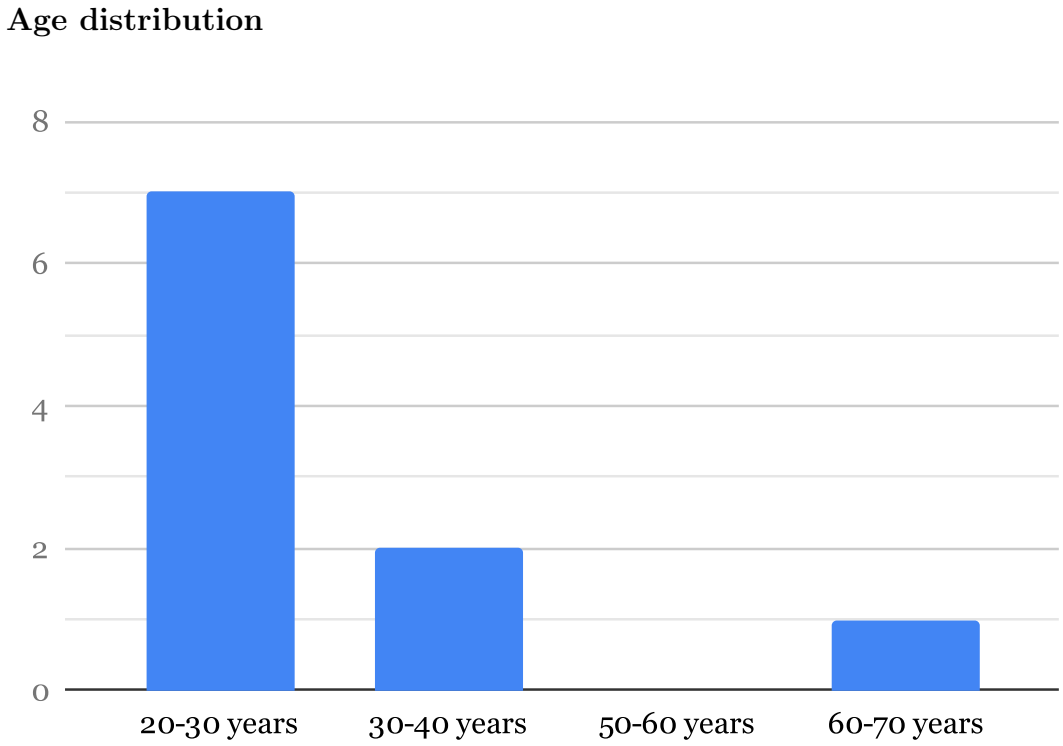

Figure 7: Age distribution in the study.

5.1.3 Smartphone Usage

The respondents were asked how often they use a smartphone. The smartphone usage is important information as the experience of smartphones could affect the perception the respondents have of the apps.

Seven of the respondents use a smartphone a couple of times every day. Three of the respondents use a smartphone a couple of times per hour. All participants are thus frequent users of smartphones.

5.2 User Evaluation

This section presents the result from the user evaluation. The result presented here is the answers for the four evaluated parts *Buttons*, *Input*, *List* and *Weather-app*. The questions *How satisfied were you with the appearance of App A/App B?* and *How satisfied were you with the speed of App A/App B?* was asked for both apps. It also shows which app the users preferred. After follows an analysis, which compares the satisfaction scores and how it relates to users comments and preference.

5.3 Buttons

5.3.1 Questions about appearance

The user was asked to focus on the appearance of the buttons and the text. The result shows a similar distribution between apps with a slight advantage to the native app (App B). Figure 8 below shows how users answered. For the Flutter app (App A), 2 answered neither bad or good, 7 answered good and 1 very good. For the native app (App B), 1 answered neither bad or good, 7 answered good and 2 answered very good.

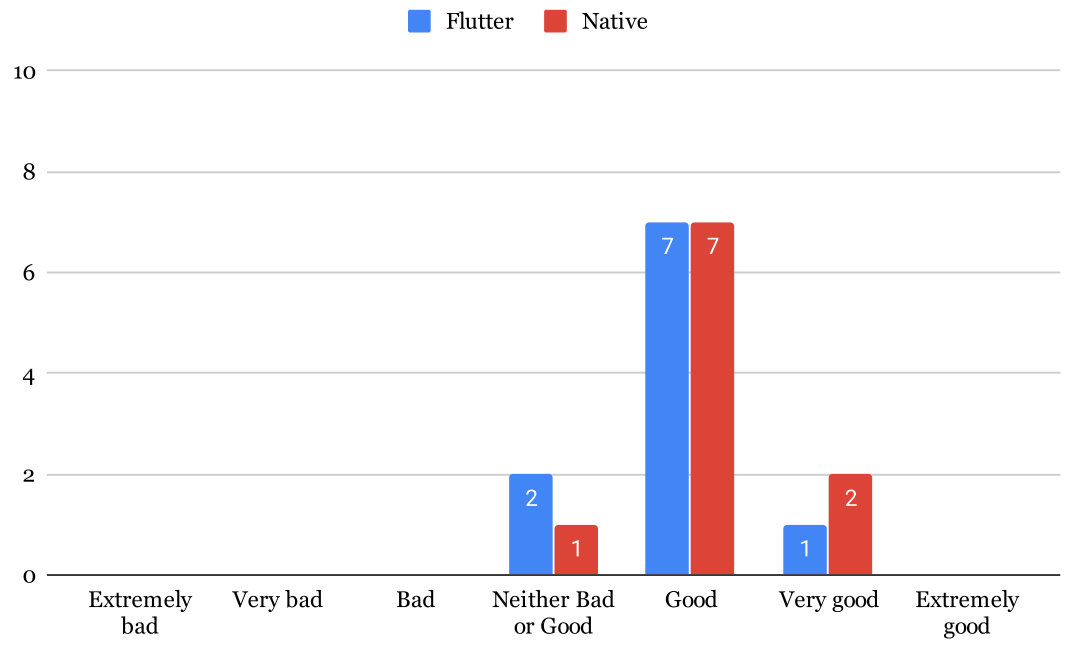

Figure 8: Satisfaction of appearance for part one

5.3.2 Questions about speed

The user was asked to focus on how fast they felt the button responded to their interaction. The result shows an advantage to the native app (App B). Figure 9 below shows how users answered. For the Flutter app (App A), 1 answered *slow*, 5 answered *neither slow or fast* and 4 answered *fast*. For the native app (App B), 3 answered *neither slow or fast*, 2 answered *fast* and 5 answered *very fast*. It is clear from the results that the users thought that the native app responded faster, with five users answering they felt the native app to be very fast, and no user for the Flutter app.

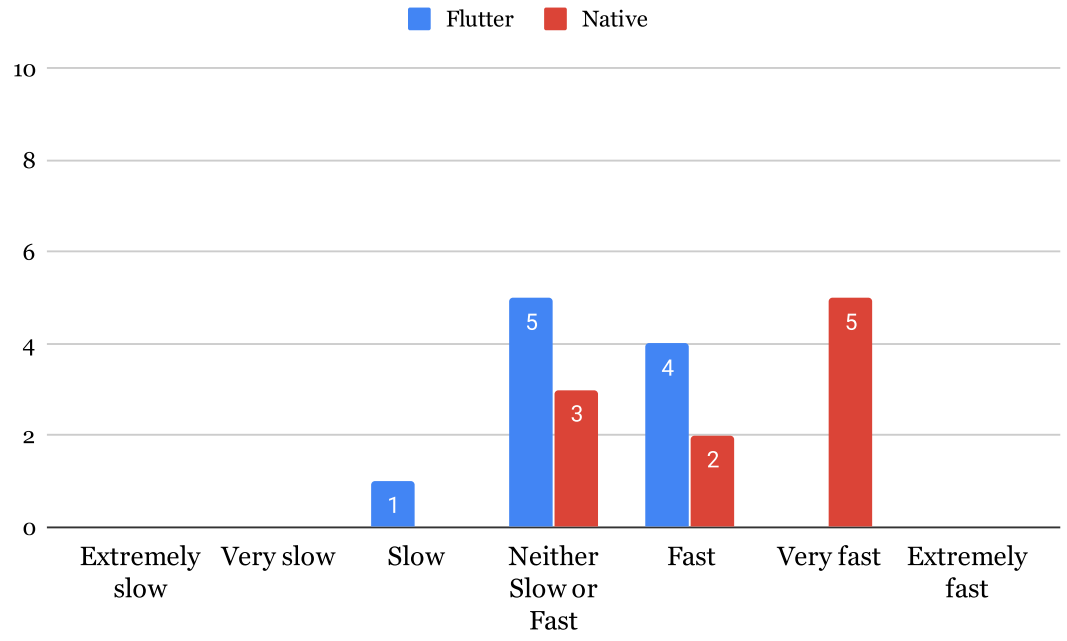

Figure 9: Satisfaction of speed for part one

5.3.3 App Preference

It is clear that the users prefer the native app. When asked *Do you prefer any of the apps?*, 7 users said they prefer the native app (App B), 1 user said they prefer the Flutter app, and 2 users answered they had no opinion. Figure 10 below shows how users answered.

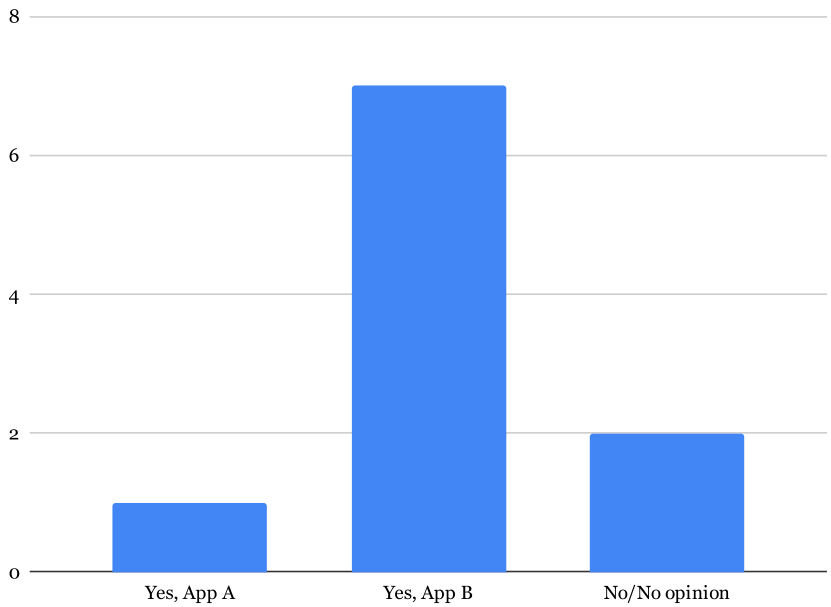

Figure 10: App preference for part one

5.4 Input

5.4.1 Questions about appearance

The users were asked to focus on the appearance of the components such as text fields, checkboxes, and dialogs. The result shows that the users perceived the appearance of the apps equally. Figure 11 below shows how users answered. For the Flutter app (App A) and the native app (App B), 6 answered *good* and 4 answered *very good*.

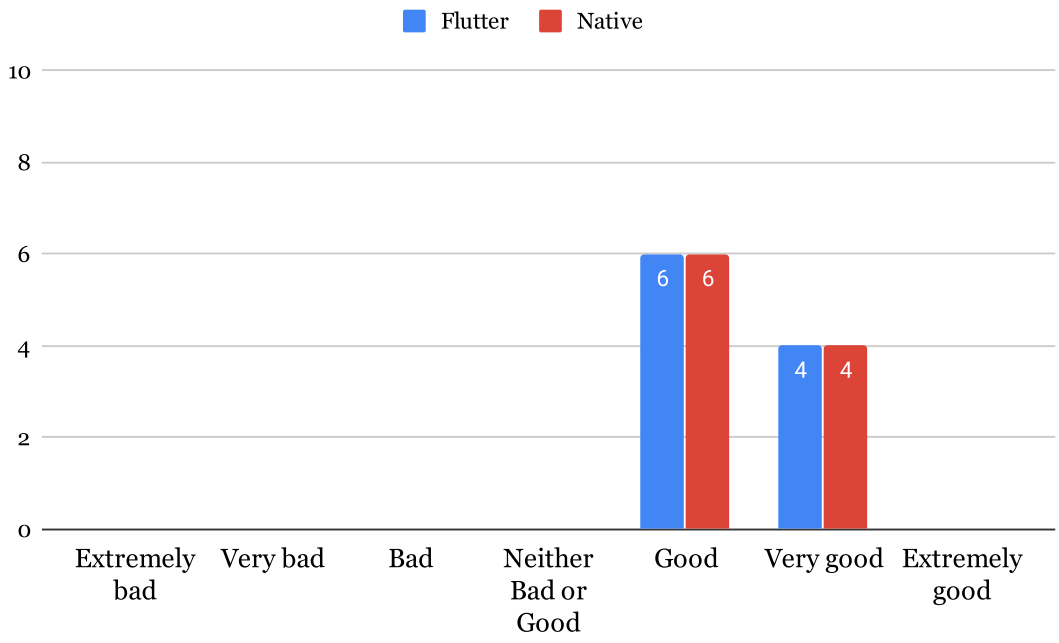

Figure 11: Satisfaction of appearance for part two

5.4.2 Questions about speed

The users were asked to focus on how fast they felt that components such as text fields, checkboxes and dialogs responded to their interaction. The results show an advantage for the native app (App B). Figure 12 below shows how users answered. For the Flutter app (App A), 3 answered *slow*, 3 answered *neither slow or fast*, 3 answered *fast* and 1 answered *very fast*. For the native app (App B), 1 answered *neither slow or fast*, 5 answered *fast* and 4 answered *very fast*. Again, the results show that users perceived the native app to be faster.

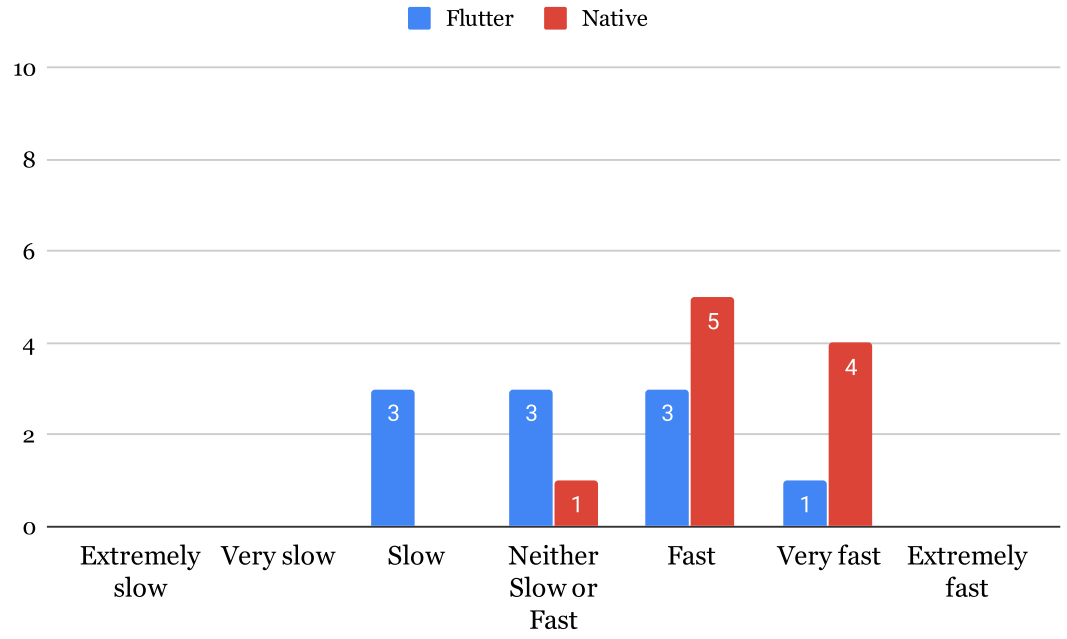

Figure 12: Satisfaction of speed for part two

5.4.3 App Preference

It is clear that again the users prefer the native app. When asked *Do you prefer any of the apps?*, 7 users said they prefer the native app (App B), 1 user said they prefer the Flutter app, and 2 users answered they had no opinion. Figure 13 below shows how users answered.

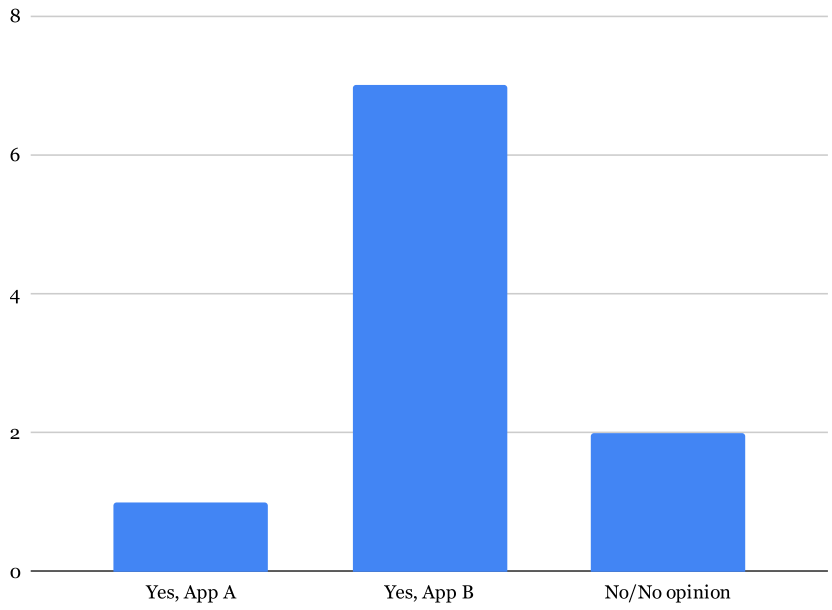

Figure 13: App preference for part two

5.5 List

5.5.1 Questions about appearance

The users were asked to focus on the appearance of the list items and the feedback box when pressing an item (called a Snackbar). The results show a slight advantage to the native app (App B). Figure 14 below shows how users answered. For the Flutter app (App A), 2 answered *neither bad or good*, 6 answered *good* and 2 answered *very good*. For the native app (App B), 2 answered *neither bad or good*, 5 answered *good* and 3 answered *very good*. The results are almost equal. There was only one person more that thought the perception of the appearance of App B was very good.

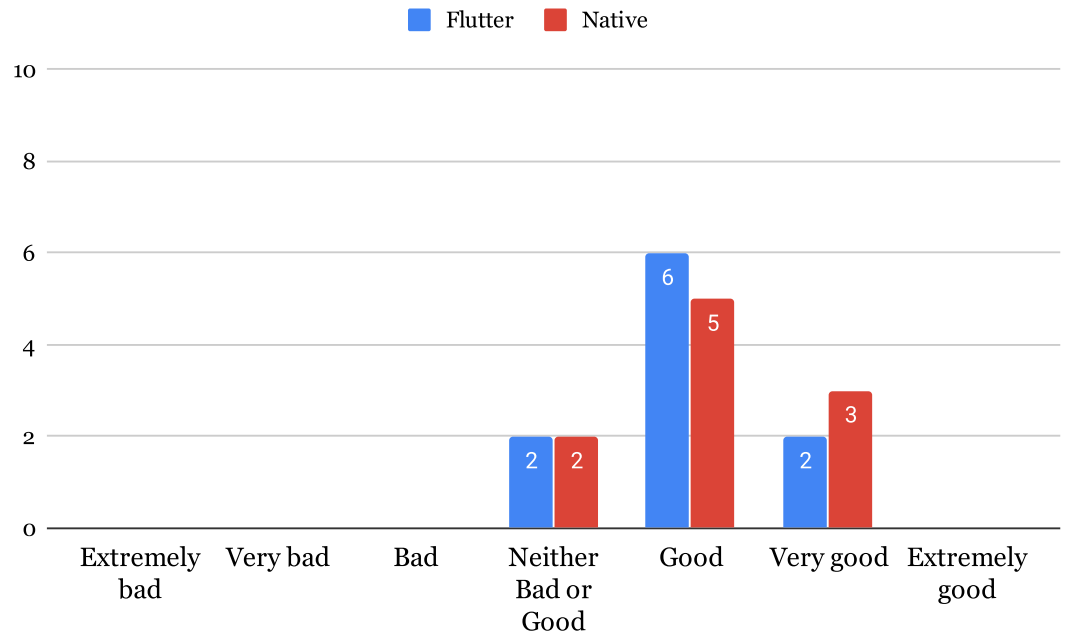

Figure 14: Satisfaction of appearance for part three

5.5.2 Questions about speed

The users were asked to focus on how fast they perceived scrolling the list was, and how fast they felt list item responded to their touch. The result shows a similar distribution between the apps, with only a slight advantage to the native app (App B). Figure 15 below shows how users answered. For the Flutter app (App A), 1 answered *slow*, 2 answered *neither slow or fast*, 5 answered *fast* and 2 answered *very fast*. For the native app (App B), 2 answered *neither slow or fast*, 5 answered *fast* and 3 answered *very fast*.

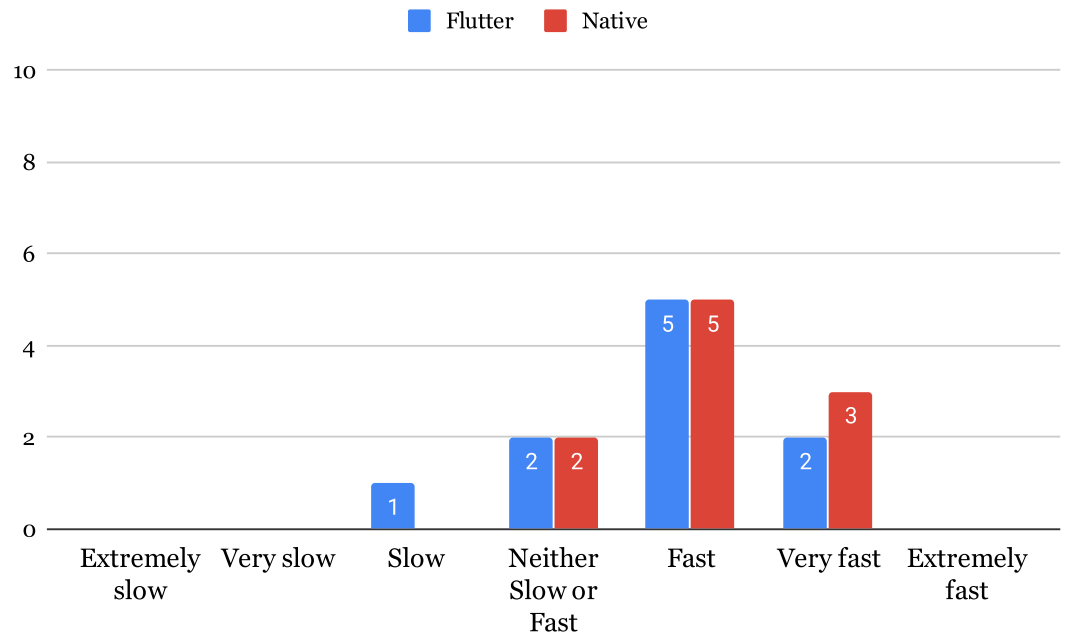

Figure 15: Satisfaction of speed for part three

5.5.3 App Preference

The users had no clear opinion of what app they preferred. When asked *Do you prefer any of the apps?*, 8 answered that they had no opinion, 1 user answered they prefer the native app (App B), and 1 said they prefer the Flutter app. Figure 16 below shows how users answered.

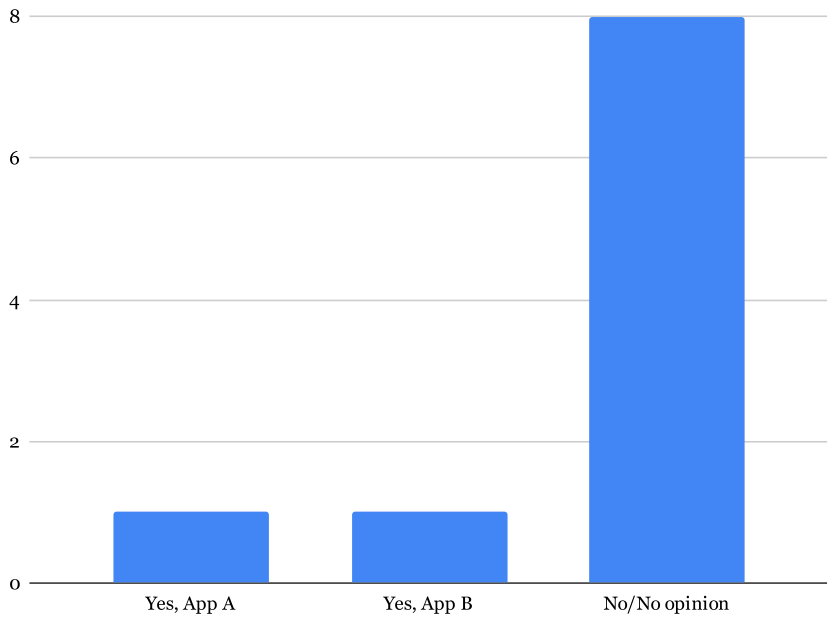

Figure 16: App preference for part three

5.6 Weather App

subsubsectionQuestions about appearance The users were asked to focus on the appearance of the different components such as the weather-card and the different buttons. The result shows an even distribution of opinion between the apps, with only a slight advantage to the native app. Figure 17 below shows how users answered. For the Flutter app (App A), 8 answered *good* and 2 answered *very good*. For the native app (App B), 7 answered *good* and 3 answered *very good*.

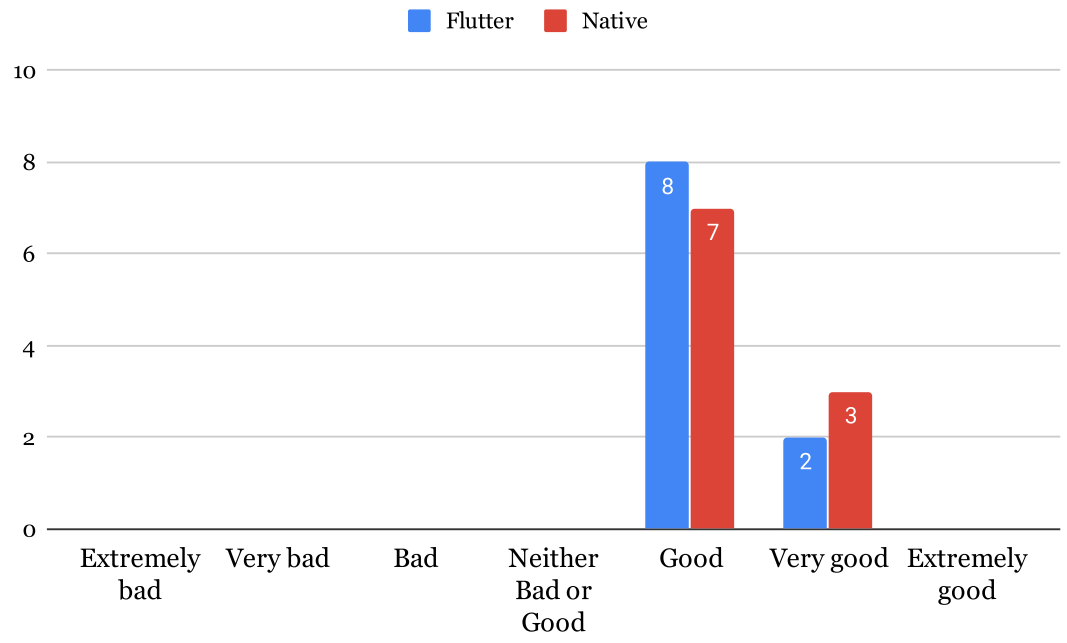

Figure 17: Satisfaction of appearance for part four

5.6.1 Questions about speed

The users were asked to focus on how fast they felt it was to search and add a city, and how fast screen transitions felt. Figure 18 below shows how users answered. For the Flutter app (App A), 2 answered *neither slow or fast*, 7 answered *fast* and 1 answered *very fast*. For the native app 5 answered *fast* and 5 answered *very fast*. The results show an advantage to the native app (App B).

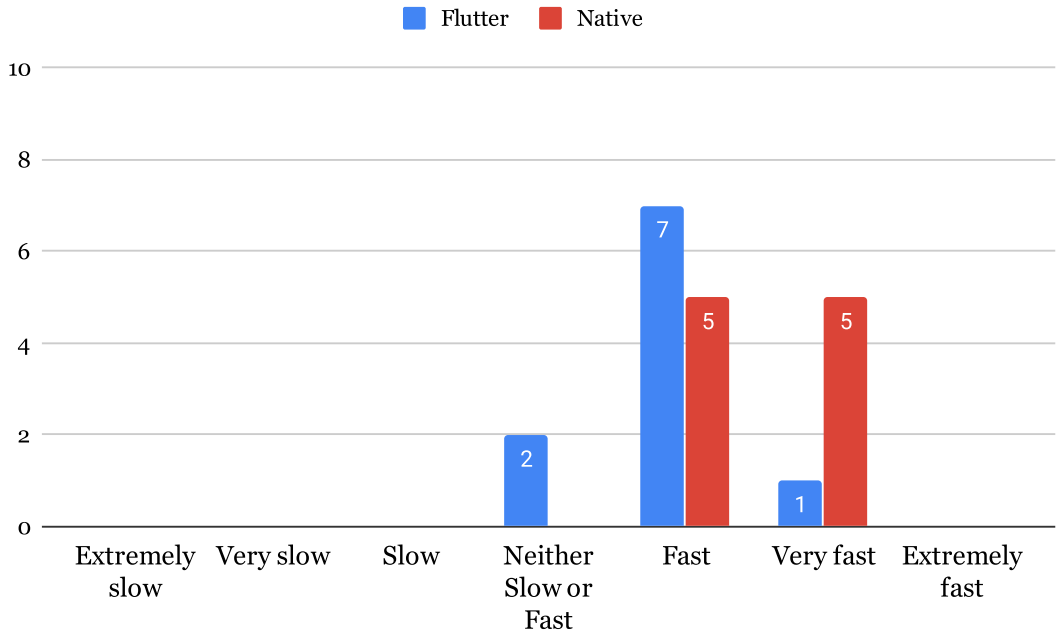

Figure 18: Satisfaction of speed for part four

5.6.2 App Preference

The users preferred the native app. When asked *Do you prefer any of the apps?*, 6 answered that they prefer the native app (App B), 3 answered that they have no opinion, and 1 answered they prefer the Flutter app (App A). Figure 19 below shows how users answered.

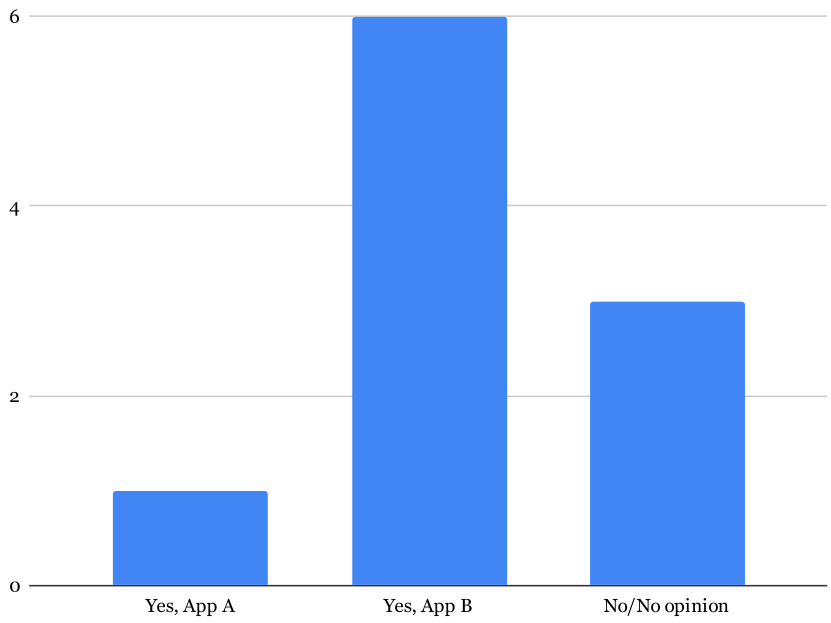

Figure 19: App preference for part four

5.7 Analysis

The result shows that there is a difference in user satisfaction between the Flutter and native app. Nielsen [15] recommends averaging the satisfaction score in order to determine the difference in user satisfaction. In this study, this is done by assigning a value to the answers where the answer *extremely bad/extremely slow* have a value of 1, and *extremely good/extremely fast* have a value of 7.

We can see in figure 20 that the user satisfaction in terms of appearance is similar between Flutter and native. The average satisfaction score is generally higher for the native app but not by a large amount. The satisfaction score for input is equal between both apps for instance. Combining the result from all parts gives the Flutter app and native app a satisfaction score of 5.1 and 5.2 respectively.

A difference in user satisfaction in terms of speed is more evident. Figure 21 shows the average satisfaction score in terms of speed. The native app has a consistently higher value. The satisfaction score for input has a difference of 1.1 points with an advantage for the native app. Combining the result from all parts gives the Flutter app and native app a satisfaction score of 4.6 and 5.2 respectively.

Part	Flutter	Native
Buttons	4.9	5.1
Input	5.4	5.4
List	5.0	5.1
Input	5.2	5.3
Summary	**5.1**	**5.2**

Figure 20: Satisfaction of appearance

Part	Flutter	Native
Buttons	4.3	4.7
Input	4.2	5.3
List	4.8	5.1
Input	4.9	5.5
Summary	**4.6**	**5.2**

Figure 21: Satisfaction of speed

The users had the ability to comment on what affected their decision. We can see that the qualitative result match the quantitative result. Most users expressed that they perceived the appearance to be similar but that the responsiveness of the native app was better. The difference was that users felt that the Flutter app did not respond as fast as the native app. Many users could not articulate precisely what they perceived differently, only that the native app felt "smoother" or that it "worked better." Some of the comments are listed below:

- "App B felt much faster. The appearance is very similiar but a bit better in App A actually. But App B feels more alive."
- "I thought that I recognized that I had pressed the button in App B. I was a bit unsure if it had worked in App A. Most buttons I was a bit unsure if I had pressed in App A."
- "Both apps felt fast but App B felt smother. The appearance was very similar but when you switched pages it felt smother in App B."
- "Thought it was similar but worked a little better in App B".
- "It felt like everything is floating better in App B. But they worked similarly."

- "I did not feel any difference. Equally good."

Most problems was small details such as how fast a button respond to touch or how it felt when switching between pages. Some of these aspects could be remedied by implementing a function in a different way. For example, a custom page transition could be implemented to improve the experience when switching pages. Other aspects such as the responsiveness of button presses are harder for the developer to change. These aspects will probably require the Flutter SDK to improve either the response time of event listeners or how it handles animations.

The result also shows the consequence of an app with lower user satisfaction. When combining the answers of what app users preferred 70% said that they preferred the native app, while only 10% preferred the Flutter app. This result shows that the user's opinion of an app is dependent not only of the appearance of the app but also how it behaves. The combined result is shown in figure 23.

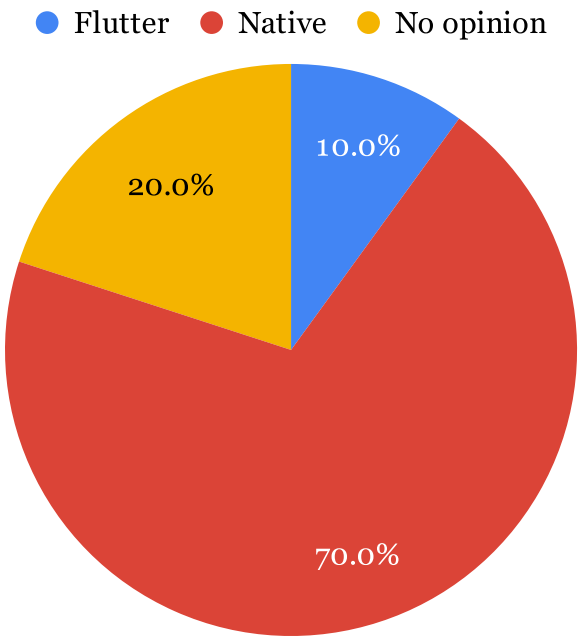

Figure 22: Combined result of app preference.

6 Discussion

This section discusses how the results relate to the findings in previous work. A discussion of the used method will then be performed, followed by a motivation for the type of apps developed.

6.1 Previous research

The result of our study aligns with the findings in previous work. Many of these studies have not focussed on the Flutter framework, but on other cross-platform technologies. Mercado et al. [12] observed a change in the complaint density in user reviews, when Facebook made a change in their development approach from developing a hybrid app, to developing a native app. The complaint density regarding the user's perception of performance and reliability decreased after changing to a native development approach. This aligns with the findings in this study where the user's perception of speed was weaker in the Flutter app compared to the native app. This could be related to the user's perception of performance as observed in [12].

The gap between the appearance and performance of cross-platform apps has also been observed by Que et al. [22]. They conclude that cross-platform development approaches can archive great appearance using the platform's widgets, but will suffer in performance degradation. When performing performance tests of hybrid and native apps, the hybrid app was slower in start-up time, had higher CPU occupancy ratio and higher memory occupancy. This aligns with the findings in our study where we observe little difference in the user's perception of the appearance, but find a difference in the user's perception of speed.

Weaker performance has also been observed in an actual Flutter app. Bizotto [2] built a simple stopwatch app and analyzed the CPU and memory usage. He then analyzed the native stopwatch app for iOS. The result shows that the Flutter app utilize the CPU twice as much and use three times as much memory of the device compared to the native app. The conclusion is that these problems in performance are noticeable by users and that the Flutter needs to improve the runtime profile. This result suggests that users perception of speed in our study could be due to actual performance problems with the Flutter SDK.

6.2 Method Discussion

In this study, a mixed method was used. Creswell [5] explains that mixed methods are suitable when you want to explain quantitative results with help of qualitative data. In this sense, a mixed method has been successful, as we did get an understanding of the user satisfaction of these apps. We also got an insight into what factors affected the perception of users. The implementation of the method however, could have been done differently.

One aspect that could have been done differently is the way the user evaluation was performed. Instead of letting users evaluate both apps, half of the users could have evaluated the Flutter app and half the native app. This could have given a different result, as a user would not have compared the interaction they just had, with that of the other framework. However, it can also be hard to formulate your interaction when not comparing it with anything. Therefore it was decided to let the users evaluate both apps.

Another approach could have been to instead of letting users comment about their interaction in text, data gathering could have been done using semi-structured interviews. It can be hard to formulate an abstract concept such as the perception of the appearance of an app. By using interviews further insight could have been gathered with follow-up questions. The reason for not using interviews was that of time constraints. The user evaluation was already long, and semi-structured interviews would have increased that time further.

6.3 Motivation for apps

To ensure the best possible result, the apps were developed as simple as possible. There were multiple reasons for this. One reason was to limit the number of factors that would affect the result. If the apps had a large number of functions, it could be hard to determine what factor affected the user perception. By focussing on simple components, it is more likely that the achieved result will show the user perception of that particular component.

Another important reason is the difference in experience when developing in-house Flutter apps compared to the in-house development of native Android apps. The difference is several years versus a couple of months. To limit the effect of variance in experience, the apps were developed as simple as possible. The reasoning was that more complex apps could affect the result of the study due to the quality of implementation. When developing simple apps, it is unlikely that the quality of the apps is affected due to less experience in developing Flutter apps.

7 Conclusion and future research

In this study two mobile apps were developed, one using the Flutter SDK and one using the native Android SDK. The apps were then evaluated by users who were asked to interact with different components that were implemented in both apps. The users were then asked to answer on a scale of 1-7 what their perception was of the appearance and speed of the apps. These questions were asked to get the user satisfaction of the apps, as explained by Nielsen [15]. The users were also asked to answer if they preferred any of the apps and to motivate what aspects that affected their perception.

The results show that user satisfaction is higher for the native app. Users perceived the appearance of the apps to be similar with a satisfaction score of 5.1 for the Flutter app and 5.2 for the native app. However, users perceived the speed of the native app to be better with a satisfaction score of 4.6 for the Flutter app and 5.2 for the native app.

The result indicate that the perceived speed of an app is important, as 70% said they preferred the native app. When asked to motivate what factors affected their perception, users explained that while both apps looked and worked similarly, the native app was perceived to be faster. One user said: "It felt like everything is floating better in App B. But they worked similarly." App B here is the native app.

The quality of cross-platform development technologies is constantly improving, and Flutter is an interesting new alternative. It is a technology that makes it easy for developers to create native-looking apps that can be executed on both the Android and iOS platform. However, this study has shown that users perceive the speed of Flutter apps to be slower than the speed of native apps. Further improvements of the Flutter framework is therefore needed in order for the user perception of Flutter apps to be equal to that of native Android apps.

7.1 Future research

A number of common interface components were tested in this study. Some of the components evaluated were buttons, checkboxes, radio buttons, alert dialogs, and input fields. This is only a fraction of all the available interface components that can be implemented in modern apps. To get a full understanding of the user perception of Flutter apps compared to native apps more components need to be evaluated. In future studies, more components could be implemented and evaluated by users.

Even though Flutter is a cross-platform framework, this study has focussed only on Flutter apps running on the Android operating system. Future studies should also evaluate apps made for the iOS operating system. As this is one of the main benefits of developing apps using Flutter, it would make sense to also gather information of the user perceptions of Flutter apps running on the iOS operating system and evaluate if it differs from the perception of native apps on iOS.

Finally, as Flutter apps become more prevalent an analysis of user reviews on the Google Play and Apple App Store could be performed, as have been done with other cross-platform apps in previous research. Such a study could focus on user reviews of apps made using Flutter and use natural language processing to evaluate the user perception of these apps compared to native apps.

References

[1] Ali M, Mesbah A. Mining and Characterizing Hybrid Apps. In: Proceedings of the International Workshop on App Market Analytics; 2016 Nov 14-14 Seattle, WA, USA. New York, NY, USA: ACM; 2016 [cited 2018 Nov 1]. p. 50–56. Available from: http://doi.acm.org/10.1145/2993259.2993263

[2] Bizzotto A. How fast is Flutter? I built a stopwatch app to find out. [Internet]. freeCodeCamp.org. 2018 [cited 2019 Jan 29]. Available from: https://medium.freecodecamp.org/how-fast-is-flutter-i-built-a-stopwatch-app-to-find-out-9956fa0e40bd

[3] Brooks F. Essence and Accidents of Software Engineering. ResearchGate. [cited 2019 Feb 14]. Available from: https://www.researchgate.net

[4] Chin JP, Diehl VA, Norman KL. Development of an Instrument Measuring User Satisfaction of the Human-computer Interface. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems: 1988 May 15-19 Washington, D.C., USA. New York, NY, USA: ACM; 1988 [cited 2018 Nov 25]. p. 213–218. Available from: http://doi.acm.org/10.1145/57167.57203

[5] Creswell JW, Creswell JD. Research Design: Qualitative, Quantitative, and Mixed Methods Approaches. SAGE Publications; 2017.

[6] Dalmasso I, Datta SK, Bonnet C, Nikaein N, Antipolis S. Survey, comparison and evaluation of cross platform mobile application development tools. In: 2013 9th International Wireless Communications and Mobile Computing Conference; 2013 Jul 01-05 Sardinia, Italy. IEEE; 2013. Available from: IEEE Xplore.

[7] Ejlertsson G. Enkäten i praktiken - En handbok i enkätmetodik. Lund: Studentlitteratur, 2005.

[8] Frøkjær E, Hertzum M, Hornbæk K. Measuring Usability: Are Effectiveness, Efficiency, and Satisfaction Really Correlated? In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems; 2000 Apr 01-06 Hague, Netherlands. New York, NY, USA: ACM; 2000 [cited 2018 Oct 11]. p. 345–352. Available from: http://doi.acm.org/10.1145/332040.332455

[9] Heitkötter H., Hanschke S., Majchrzak T.A. (2013) Evaluating Cross-Platform Development Approaches for Mobile Applications. In: Cordeiro J., Krempels KH. (eds) Web Information Systems and Technologies. WEBIST 2012. Lecture Notes in Business Information Processing, vol 140. Springer, Berlin, Heidelberg.

[10] Malavolta I. Beyond native apps: web technologies to the rescue! (keynote). In: Proceedings of the 1st International Workshop on Mobile Development - Mobile!; 2016 Oct 31-31 Amsterdam, Netherlands. New York, NY, USA: ACM; 2016 [cited 2018 Nov 4]. p. 1–2. Available from: http://dl.acm.org/citation.cfm?doid=3001854.3001863

[11] Malavolta I, Ruberto S, Soru T, Terragni V. End Users' Perception of Hybrid Mobile Apps in the Google Play Store. In: 2015 IEEE International Conference on Mobile Services; 2015 Jun 27-Jul 02 New York, NY, USA. New York, NY, USA: IEEE; 2015 p. 25–32. Available from: IEEE Xplore.

[12] Mercado IT, Munaiah N, Meneely A. The Impact of Cross-platform Development Approaches for Mobile Applications from the User's Perspective. In: Proceedings of the International Workshop on App Market Analytics; 2016 Nov 14-14 Seattle, WA, USA. New York, NY, USA: ACM; 2016 [cited 2018 Oct 8]. p. 43–49. Available from: http://doi.acm.org/10.1145/2993259.2993268

[13] Nielsen J. How Many Test Users in a Usability Study? [document on the Internet]. Nielsen Norman Group. [cited 2019 Apr 17] Available from: https://www.nngroup.com/articles/how-many-test-users/

[14] Nielsen J, Molich R. Heuristic Evaluation of User Interfaces. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems; 1990 Apr 01-05 Seattle, WA, USA. New York, NY, USA: ACM; 1990 [cited 2018 Oct 23]. p. 249-256. Available from: IEEE Xplore.

[15] Nielsen J. User Satisfaction vs. Performance Metrics [document on the Internet]. Nielsen Norman Group. [cited 2018 Nov 28]. Available from: https://www.nngroup.com/articles/satisfaction-vs-performance-metrics/

[16] Nielsen J. Usability Metrics. Nielsen Norman Group. [cited 2019 Apr 17] Available from: https://www.nngroup.com/articles/usability-metrics/

[17] Leler W. What's Revolutionary about Flutter [document on the Internet]. Hacker Noon; 2017 Aug 25 [cited 2018 Nov 21]. Available from: https://hackernoon.com/whats-revolutionary-about-flutter-946915b09514

[18] Leler W. Why Flutter Uses Dart [document on the Internet]. Hacker Noon; 2018 Feb 26 [cited 2018 Dec 6]. Available from: https://hackernoon.com/why-flutter-uses-dart-dd635a054ebf

[19] Ohrt J, Turau V. Cross-Platform Development Tools for Smartphone Applications. Computer. 2012; 45(9):72–9. doi: 10.1109/MC.2012.121

[20] Raj CPR, Tolety SB. A study on approaches to build cross-platform mobile applications and criteria to select appropriate approach. In: 2012 Annual IEEE India Conference (INDICON); 2012 Dec 07-09 Kochi, India. New York, NY, USA: IEEE; 2013. p. 625–629. Available from: IEEE Xplore.

[21] S.Thakare B, Shirodkar D, Parween N, Parween S. State of Art Approaches to Build Cross Platform Mobile Application. IJCA. 2014;107(20):22–3. doi: 10.5120/18868-0389P

[22] Que P, Guo X, Zhu M. A Comprehensive Comparison between Hybrid and Native App Paradigms. In: 2016 8th International Conference on Computational Intelligence and Communication Networks; 2016 Dec 23-25 Tehri, India. New York, NY, USA: IEEE; 2017. p. 611–4. Avaialable from: IEEE Xplore.

[23] Wu W. React Native vs Flutter, cross-platform mobile application frameworks. Information technology [thesis]. Helsinki, Finland: Metropolia University of Applied Sciences; 2018.

[24] Xanthopoulos S, Xinogalos S. A Comparative Analysis of Cross-platform Development Approaches for Mobile Applications. In: Proceedings of the 6th Balkan Conference in Informatics; 2013 Sep 19-21 Thessaloniki, Greece. New York, NY, USA: ACM; 2013 [cited 2018 Oct 8]. p. 213–220. Available from: http://doi.acm.org/10.1145/2490257.2490292

No author

[25] App stores: number of apps in leading app stores 2018 [document on the Internet]. Statista; 2018. [cited 2018 Dec 7]. Available from: https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/

[26] Technical Overview, What is Flutter? [document on the Internet]. [cited 2018 Nov 15] Available from: https://flutter.io/technical-overview/

[27] Smartphone shipments worldwide 2009-2018 | Statistic [Internet]. Statista. [cited 2018 Dec 7]. Available from: https://www.statista.com/statistics/271491/worldwide-shipments-of-smartphones-since-2009/

[28] ISO 9241-11:1998(en), Ergonomic requirements for office work with visual display terminals (VDTs) - Part 11: Guidance on usability [document on the Internet]. Organisation Internationale de Normalisation (ISO); 1998 [cited 2018 Dec 7]. Available from: https://www.iso.org/obp/ui/#iso:std:iso:9241:-11:ed-1:v1:en

Appendix

Appendix A

Mobila applikationer (mobila appar) används idag av de flesta personer i vårt samhälle för att hjälpa till med många av de dagliga uppgifter vi utför i våra liv. Många vet dock inte att appar kan utvecklas på en mängd olika sätt. Dessa appar är ofta funktionellt ekvivalenta från användarens perspektiv, men kan ha skillnader i användarens upplevelse av dem.

Denna undersökning utvärderar två olika system för att utveckla appar, med fokus på hur användaren upplever interaktion och användning av appar gjorda i bägge systemen. Målet är att identifiera ifall det finns skillnader i upplevelse vid anvädning av appar gjorda i dessa system.

Denna enkät består av fyra stycken delar, där varje del innehåller en rad enkla uppgifter som utförs i båda apparna. När du utför uppgifterna, försök att notera vilka känslor och tankar du har vid interaktionen med appen. Du kommer att få svara på vilken generell inställning du har till varje app och om du har någon preferens för någon av apparna. Därefter har du en möjlighet att beskriva dina tankar för apparna. Försök att nämna alla tankar du har, ingen detalj här är för liten!

Alla svar kommer att behandlas konfidentiellt!

Ditt deltagande i undersökningen är naturligtvis frivilligt, men är betydelsefullt för undersökningens kvalitet. Svaren kommer att databehandlas anonymt och således inte kopplas till dig som person. Inga namn samlas in och ingen möjlighet att koppla svaren till dig som person finns därför.

Jag hoppas att du vill medverka i undersökningen för att öka kunskapen kring användares inställning till appar gjorda i dessa system.

Tack på förhand för din medverkan! Lund, November 2018

Ola Dahl