Bachelor Degree Project

Survey on the state of cross-platform mobile development frameworks

Author: Linus Hvenfelt
Supervisor: Alisa Lincke
Examiner: Arianit Kurti
Semester: HT 2022
Subject: Computer Science

Abstract

Mobile application development has grown in the past few years, and instead of native development, some developers have moved to a new strategy; cross-platform mobile development using frameworks. There are many frameworks that all have their use case, but whether or not these frameworks are ready to be used in production applications is hard to decide on. This research aims to find the strengths and weaknesses of cross-platform mobile development frameworks, and how they can be improved to better suit the needs of developers. To gather data in this topic a survey was created to analyze developer experiences on frameworks in key areas such as front-end design, platform maturity and more. The results reveal that there are many areas that can be improved, but frameworks are a great tool for smaller teams and are being used extensively already to create applications for multiple platforms.

Keywords: cross-platform mobile application development, development frameworks, software development

Preface

I'd like to thank my supervisor, Alisa Lincke for guiding me and helping me along the way through the process of writing this thesis. I would also like to thank Jads, Anton and Andrey Breslav for their feedback on the survey questionnaire, as well as the interview participants:

- Federico Curzel - https://curzel.it/
- Brian R Clow - https://brianrclow.com/
- Alexander May - https://de.linkedin.com/in/may-dev
- Anonymous

And finally, this whole ordeal could not have been done without the immense help and support of my family.

Contents

1 Introduction

This is a 15 HEC bachelor thesis in computer science that is focused on cross-platform mobile development frameworks. Applications on our smartphones and desktop devices are used generously in our day-to-day life, providing us with all sorts of services and experiences. Nowadays, cross-platform mobile development frameworks serve as a way of allowing application developers to develop applications more efficiently for multiple platforms at once. As these tools grow, some growing pains are expected. Android and iOS change at a rapid pace, and it's a constant game of catch up for the software developers behind frameworks to implement the latest features but also keep the frameworks bug free.

1.1 Background

Two dominant platforms exist on the mobile market, one being Android and the other iOS. Besides these platforms, we also have computers running Windows, Mac, and Linux but also the web being represented by different platforms both on desktop and mobile.

Most of these platforms have their own unique way of implementing applications, or programming languages to create them for. This is how the idea of a cross-platform development framework arose, to allow a developer to deploy an application written in a single codebase to multiple platforms. More frameworks have started to be developed and released during the last 5 years, and the usage of them is only growing. But a lot of companies still have their doubts about using a framework, and instead have separate teams creating separate applications for each platform. The question then arises; why? Aren't the frameworks mature or viable enough yet? What parts of the frameworks are good enough, and which ones are not?

This thesis surveys developers on their experiences and opinions about a framework they used in a project, and what they believe to be its strengths and weaknesses.

1.2 Related work

In 2019 an article [1] researchers reported on the industry's perspective on cross-platform mobile development frameworks. They surveyed 101 developers online and found the most common issues and problems that the participants had with frameworks. The most common issues according to developers at the time were the following:

- Overall loss in performance compared to native apps

- Suboptimal User Experience (UX)
- Immature frameworks (too cutting edge, too much risk, etc.)
- Suboptimal options for creating good User Interfaces (UI)
- Immature communities (too new/low activity, etc.)

The survey was received in May of 2018 and published in 2019, which is as of writing about 3+ years ago. 3 years in Computer Science is a long time as React Native was launched in 2015 and Flutter in 2017. This work is based on some of the findings (framework's disadvantages) of that survey [1] and further explores whether there have been any changes in the frameworks to overcome the issues/problems they presented, or if they have stayed the same.

In another study from 2017 [2] 8 different apps were created for Android, iOS, and Windows Phone. Some were made natively, and some used Ionic, PhoneGap and NativeScript frameworks. They researched the advantages and disadvantages of programming the applications natively and with the frameworks and came to conclusions at the time about which alternative gave the best product. A similar study was done in 2022 [3] where an app was made in different frameworks and together with a company was put through certain criteria, to research what frameworks would suit the company's needs the best.

In summary, recent surveys [1] [2] [3] have explored how well frameworks were performing at the time, for what purposes they exist, which one to choose for a specific project in a company, what common issues were perceived and more.

1.3 Perceived issues with frameworks

From the results found in the related work, relevant issues according to previous research have been found, which will now be presented. There are also other not-before-mentioned challenges in using a cross-platform mobile development framework.

Platform specific bugs might occur, forcing the usage of platform specific code outside of the shared codebase. Other problems include loss in performance compared to native applications [4] and suboptimal user experience, causing cross-platform development to feel too immature as a concept and therefore too risky to use at large.

Another type of platform specific bug is where the application has different results on two different platforms, often referred to as platform quirks. These problems could for example be how the UI is handled, or something else related to the native APIs such as camera, permissions, and location. Usually

this leads to the developer having to write extra code detecting the platform and handling the functionality differently, but in the worst-case scenario they might have to go and write platform specific native code instead to handle the issue. Being forced to do this completely disregards the point of the framework in the first place, only having to use a single codebase.

When it comes to creating an application for iOS devices, writing some code in Xcode is often obligatory. For iOS devices, when the time comes around to launch a beta or alpha version on Testflight or launching your application to the iOS App Store, problems often arise. Regardless of whether it is using a cross-platform framework or not, if you want to launch an application to the iOS app store- you need to own a Mac computer. Sending compiled builds into Testflight (the beta testing program for iOS apps) or the app store, you need to send the build directly from Xcode using the built in functionality. Furthermore, the problem lies within the fact that Xcode can only be run on a Mac computer, thus you must own a Mac to be able to launch your application into the iOS ecosystem.

Android on the other hand, is fully buildable on a Mac, Windows, or Linux system. You can send your Android project to the Google Play Store the same way on all platforms. This leads to the subjectively best platform to develop these applications being a Mac, as you have access to both sides.

1.4 Problem formulation

The field of cross-platform mobile development frameworks is rapidly changing with the development of the mobile platforms themselves, and the frameworks. Previous studies on cross-platform mobile development frameworks have found the common issues [1] [5], the advantages and disadvantages of frameworks [2] [6]. However, these studies may not reflect the current state of frameworks due to the rapid change of the technology. Moreover, this information is outdated because new frameworks appear every year while others are disappearing or stopped from being supported. But the results serve as good data that can and will be compared to my own survey results later in this thesis.

As a reference, iOS 11 was relevant in 2017, and currently iOS 16.4 is relevant. On the Android side, Android 8 Oreo was launched in 2017 and now we are up to Android 13. More relevant and new research can benefit this field of development. New work and analysis are therefore always needed to make sure the frameworks are up to par with the application developers' standards.

Therefore, the purpose of this thesis is to provide a more up to date understanding of the current state of cross-platform mobile development frameworks. Their positive and negative aspects as well as the common issues from an application developer's perspective.

1.5 Research questions

Based on this information, the following research questions have been defined:

RQ1: What are the common use cases for a cross-platform mobile framework currently and how can frameworks be improved to better suit them?
RQ2: Compared to previous research, how have the negative and positive aspects changed?

With RQ1, the thesis aims to provide a general overview of the framework's usability by application developers. More precisely, the usability aspect refers to what type of mobile application the framework has been used to create, the developer's experience level, the company type, and size, etc. The output of this research question is relevant information on currently developed frameworks.

The second part of the first research question is concerned with a more detailed investigation of the common positive and negative aspects of each framework and identifying the possible improvements of the found common issues to better suit the developer's needs. Identifying these issues were based on both the survey results and the issues presented by interviewees. The findings of RQ1 provide the recommendation for cross-platform framework developers to improve the current frameworks or consider the recommendation in developing new cross-platform mobile frameworks.

The second research question is concerned with comparing this thesis' results with previous research [1]. The found aspects are compared to the previous studies to report which are already fixed and which are new or remain an issue. Therefore, to be able to compare the previous study [1] outcomes with RQ2.2 outcome, some of the survey questions from previous survey [1] were included in the survey in this work. These questions have focus on issues and problems surrounding cross-platform frameworks and are used to compare the possible changes in the current frameworks' version (such as improvements, new issues, old issues).

These research questions aim to provide a current overview of the most used cross-platform mobile frameworks and their positive and negative aspects from an application developer's view. The research questions identified are

focused on providing information on the usability of the frameworks, comparing the common positive and negative aspects to previous studies, and identifying improvements to better suit the developer's needs. The data gathered around these research questions can help further develop the field of cross-platform mobile development frameworks by providing relevant information on currently developed applications, highlighting common positive and negative aspects, and recommending improvements for the frameworks.

1.6 Motivation

This thesis will help to further advance the field of cross-platform mobile development frameworks. The result will show the current state of frameworks as of the last couple of years, and present concrete evidence on what key areas developers using frameworks believe to be good and bad. The results will also show how the negatives and positives have shifted, what old problems have been less of an issue and what potentially new issues are. By presenting feedback from developers in the field, we can find areas of improvement for the frameworks. This result can be used to then solidify the quality of a framework further by the developers or open-source community.

The results of the research questions will provide a good basis for software developers to further improve the quality and experience for application developers using frameworks for their future applications. The focus lies in what issues frameworks individually and generally have, and how these issues should be addressed.

Frameworks are built on the idea of using a single codebase, which is supposed to increase efficiency by for example not having to use two separate teams or developing an app twice. The efficiency of this approach has the potential to be immensely powerful, as more and more pressure is put on application developers to achieve better results, faster.

1.7 Scope/Limitation

This thesis is not about comparing native development (Java/Kotlin for android or Swift for iOS) to frameworks. It will only be focusing on cross-platform mobile development frameworks. Some frameworks allow the deployment to web as well as desktop environments, and whilst the survey does contain the question asking if the users' project was deployed to web and desktop, I will not be going deeper into detail in this area.

Profiling question data were used for providing descriptive information about the dataset for RQ1. The only part in which these results are presented are in the results for the first part of RQ1, seen later in the thesis at 5.1.1.

My research does not aim to solve the question about "what is best," but instead to simply present what the frameworks have been used for by the survey participants, and to what result. The outcome of the survey is used to find key areas of improvement for some of the frameworks.

The data I gathered is from many different time periods, and whilst this thesis's main goal is to find current issues, answers from projects made from all years are used in the graphs and data analysis. Approximately 23,6% of the data in this thesis are from before 2019, which means that the findings are quire recent.

1.8 Target group

Framework developers and researchers in this area are the focus group in this thesis, because the research outcome of the survey is based on the experience and opinion of the end-users (application developers) of frameworks. The application developers (end-users) themselves could also be interested in the identified pros and cons of the frameworks when selecting a framework for application development, to find one that suits their specific needs.

1.9 Outline

Chapter 2 is about describing the research methods and how they were used, as well the recruitment for the survey, reliability, validity, and ethical considerations. Chapter 3 includes the necessary theoretical background to gain an understanding of cross-platform mobile development. Chapter 4 explains the research project implementation, and chapter 5 presents the results from the different research methods and relates them to the research questions. Chapter 6 is further analysis surrounding the research questions and the results. In chapter 7 conclusions for every research question will be presented, and what future work can be done.

2 Research Method

The research methods used in this thesis are a survey and an interview with a focus group. Many studies on cross-platform mobile frameworks have utilized a survey method to collect a large amount of quantitative data to analyze it and find common areas, aspects, and issues of the frameworks. The advantages of this method are that they can be analyzed using statistical techniques to draw meaningful conclusions, allow to collect data from a large and diverse sample, which can increase the generalizability of the findings, and usually it uses a standardized questions and response options, which can reduce measurement error and increase the reliability of the data. Therefore, a survey method was chosen to find the common areas, issues, problems, potential improvements of frameworks.

Many studies on cross-platform mobile frameworks have utilized a survey method to collect a large amount of quantitative data to analyze it and find common areas, aspects, and issues of the frameworks. The advantages of this method are that they can be analyzed using statistical techniques to draw meaningful conclusions, allow to collect data from a large and diverse sample, which can increase the generalizability of the findings, and usually it uses a standardized questions and response options, which can reduce measurement error and increase the reliability of the data. Therefore, a survey method was chosen to find the common areas, issues, problems, potential improvements of frameworks. The interview method was chosen due to its ability to give more in-depth information and provide qualitative data, which can't be provided in a survey focused on shorter but more concise answers. Using two different methods also helps by confirming each other, trends and responses from the survey data can be presented to the interviewee and see if they agree or not.

At first a literature review was conducted on IEEXplore and Google Scholar, to find and analyze the latest trends and topics in research surrounding cross platform mobile applications and frameworks. By reading abstracts and results I gathered a sizable number of articles surrounding the topic that I found interesting and could be useful for my thesis and finding a research gap. Previous positive and negative aspects were also collected from these articles and compared to the data gathered in the survey, to compare and solve RQ2.

2.1 Data Collection Methods

2.1.1 Reasoning for data collection methods

To gather information on the usage and perspectives of developers regarding cross-platform mobile development frameworks, this study utilized two primary data collection methods: surveys and interviews. Surveys were chosen

as a method of data collection due to their ability to gather a large amount of data in a short period of time and their ease of distribution through various online platforms. The use of surveys is also supported by previous studies in the field of software engineering [7] [8] [9]

Surveys were deemed appropriate for this study as they allowed for the collection of quantitative data on the types of frameworks used, the frequency of usage, and the perceived advantages and disadvantages of each framework. Additionally, survey responses allowed for the identification of trends and patterns in framework usage and provided a broad understanding of the current state of cross-platform mobile development frameworks.

Interviews were chosen as a complementary method of data collection to provide a more in-depth understanding of developers' experiences and perspectives on cross-platform mobile development frameworks. Interviews allowed for the collection of qualitative data on the challenges faced by developers when using cross-platform frameworks, as well as the identification of potential areas for improvement. The use of interviews is also supported by previous studies in the field of software engineering. [10] [11]

By utilizing both surveys and interviews, this study aimed to provide a comprehensive understanding of the current state of cross-platform mobile development frameworks, their positive and negative aspects, and potential areas for improvement.

2.1.2 Survey tool

The surveying tool used for this research was Google Forms, a free online survey platform by Google. The survey consisted of a variety of different questions, the majority using a Likert scale with response options ranging from 1 (strongly disagree) to 5 (strongly agree), as well as some scales ranging from 1 to 7. The possible responses were based on a Likert scale response sheet [12]. For each question written a suitable response range was chosen. A scale from left to right was used for 1-5 answers whilst radio buttons were used for more complicated questions where every option was written out.

The survey data was analyzed using Google Sheets, which is a spreadsheet program by Google. Google Sheets allowed for visualizing and creating diagrams of the survey data, creating averages, and analyzing the data.

2.1.3 Interviews

The interviews were conducted using Zoom, an online video conferencing platform. The purpose of the interviews was to gather in-depth information about the participants’ experiences related to their answers in the survey.

Each interview lasted 30-45 minutes and the participants were asked to elaborate on their survey responses as a start and were then presented with broader questions to get their general view on all frameworks.

2.2 Pilot test

To make sure the survey was formed in an effective way to prevent bias and reach a certain standard, a pilot test was done on a couple of colleagues. Participants were asked to go through the survey and reach out with feedback related to any confusion in question formulation, how easy it was to understand, and general feedback. From the pilot test a lot of questions were rephrased to make the questions more standardized and use the same type of Likert-scale answers to prevent confusion, as before there were a lot of different answers instead of simply strongly disagree to strongly agree.

2.3 Recruiting Survey and Interview Participants

The survey was released into the wild, posted to multiple online forums such as LinkedIn groups and the LNU Slack. Most of the answers came from different subreddits (a forum dedicated to a specific topic) of the online discussion website Reddit.com, such as r/xamarindevelopers, r/FlutterDev, r/androiddev and more. It was also shared in public programming communities on the online VoIP and text platform Discord. Due to the survey simply being shared as a link and that no personal information was collected by me or Google Forms, it's not possible to identify where the answers came from, and if they were shared along via other means.

Survey participants were asked if they were interested in participating in a short interview about their answers, and these people were contacted later by email where they could book a time slot for an interview.

2.4 Reliability, Validity and Ethical Considerations

The survey was spread out over a lot of different platforms to gain insight from various types of developers. The topic of bias was brought up as a concern from early on, as people tend to have a love or hate relationship with framework that they use. This is one of the reasons we decided to ask the questions out of the perspective of a project they've worked on. This makes the participants think of experiences they had instead of relying on their more subjective based opinion that they have developed over time, hopefully helping in preventing bias. The questions were also asked in a way to prevent bias [13], as well as how we present the answers, in this case from negative (left) to positive (right).

The pilot test served as a great way to help preserve content and face validity. A lot of feedback was gathered on the way questions were asked and the predefined options. These were altered a lot according to the feedback, to make sure the participant had a good understanding of the questions. It is important to note however that this data only provides a handful of peoples' opinions, and their opinions does not represent the whole population of developers. As this survey was released "into the wild" and shared on many different platforms, identifying the type of developer who answered the survey may be difficult, but the profiling questions gave good insight into this.

Cronbach's Alpha was calculated based on all the Likert scale questions, and the result is an alpha of 0,89. An alpha above 0.9 is often defined as an internal consistency of "excellent" quality, whilst 0.8 to 0.9 is considered "good". Due to this, the validity of this survey can be considered quite high. The profiling questions also allows us to get perspective on what type of user is answering the questions, such as experience in the field and job experience. Although it is important to keep in mind that this survey is only a sample of people and does not represent the whole population. The best conclusion can be made on all frameworks, but going into specific frameworks the answers become less in quantity as they are spread out across multiple different frameworks.

As this thesis is using both a quantitative and qualitative method, this increases the validity further. Comparisons can be made between the data gathered from the survey and the interviewees' qualitative answers, to see if the results line up or not.

On the topic of ethical considerations, a small amount of personal information based on the project the participant worked on was collected in the first survey questions. The user was asked if they agreed to their answer being used in this thesis, and then were asked about their highest level of education, how long they have been developing and previous native development experience. After that came more project-based questions, such as how big the team was, what platforms it was released to, how long it took to work on and etcetera.

Based on this, there was not much personal information gathered. We saw no reason to ask for things such as age or gender, as these aren't relevant to the research questions. The data is stored by me locally and only in my GitHub project for analyzing the data. Google Forms gathers some personal information, but the user accepts these when using any Google product and accepting the cookies.

3 Theoretical Background

In this chapter, fundamental theoretical background is presented in the field of mobile application development and cross-platform mobile frameworks as well as the challenges in using a framework.

3.1 Mobile Application Development

Mobile application development stems from the need for applications for different platforms, starting with the first generation of smartphones. The first iPhone released in 2007 started a revolution in the industry that would only grow and grow the following years, to the point of 86% [14] of the world's population today owning a smartphone. Today, iOS applications are developed in the language Swift and Android applications are built on Kotlin or Java. The respective platforms have their own application stores, the Apple App Store and Google Play Store where developers can upload their applications to for the masses to download and use.

When developing natively, the developer must pick a side to develop on. If they wish to release their application to both mobile platforms or any other platform, they must remake the same application in two separate programming languages, for example one in Kotlin for Android and one in Swift for iOS. This causes a lot of extra work, having to do the same thing twice. Large companies even tend to have two separate development teams, working on delivering the best possible application to each platform.

3.1.1 Native development on Android

The Android SDK (Software Development Kit) is a collection of tools, libraries, and resources used to develop Android applications [15]. It includes the Android Studio IDE (Integrated Development Environment), which provides a graphical interface for designing, coding, and testing Android applications. Android Studio is based on the IDE IntelliJ IDEA and is the official IDE for Android development. It offers features such as code completion, debugging, and performance analysis tools, and includes an emulator for testing apps on virtual Android devices. The Android SDK also provides a set of APIs that allow developers to access device features such as camera, GPS, and sensors, as well as access to Google Play services for integrating with Google services such as Maps, Firebase, and Google Sign-In. The Android SDK is constantly evolving, with new versions released frequently to provide updates, bug fixes, and new features.

3.1.2 Native development on iOS

Xcode is the Integrated Development Environment (IDE) for iOS development, and is the primary tool used to create iOS apps [16]. It provides a graphical interface for designing, coding, and testing apps, as well as for submitting them to the App Store. Xcode includes features such as code completion, debugging, and performance analysis tools, and offers an iOS Simulator for testing apps on virtual iOS devices. Xcode also includes an Interface Builder tool, which provides a visual interface for creating user interfaces using a drag-and-drop system. In addition to the tools, Xcode also provides a set of APIs and frameworks that allow developers to access device features such as camera, GPS, and sensors, as well as Apple services such as Apple Pay, iCloud, and Push Notifications. Xcode is constantly evolving, with new versions released frequently to provide updates, bug fixes, and new features.

3.1.3 Cross-platform mobile development

Cross-platform development has become an increasingly popular approach in mobile application development due to the need for developers to create apps for multiple platforms [17]. By using a cross-platform mobile development framework, it allows developers to use a single codebase to create apps for different platforms, rather than having to develop separate codebases for each platform. Furthermore, this allows developers to write their code once and then compile it or deploy it for each platform, reducing the time and effort required to develop and maintain multiple codebases. This can lead to faster development cycles and lower development costs, as well as an easy approach to maintain consistency across platforms [17].

3.2 Cross-platform mobile development frameworks

A cross-platform mobile development framework is a framework developed under the previously mentioned circumstances. A developer usually codes in a single programming language and the code is then able to be compiled and seamlessly deployed to multiple platforms. As an example, the Google developed framework Flutter uses the Dart programming language for the shared codebase, and the framework Xamarin uses C# for it’s codebase.

There are numerous amounts of frameworks using this fundamental concept. These frameworks vary widely in popularity and functionality, but the core difference being the different approach that they use, that will be explained in the next sections.

The most popular frameworks as of 2019-2022 are Flutter, React Native, Cordova, Ionic and Xamarin [18] that can be seen in figure X below. Although some new smaller frameworks are on the rise, the approach to the they use is like that of their predecessors.

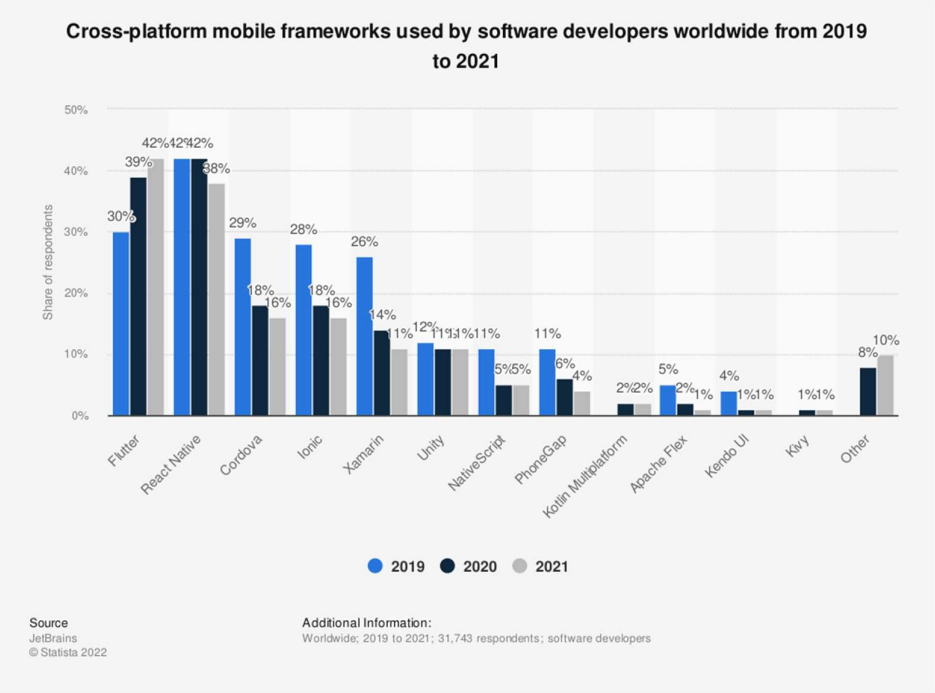


Figure 3.1 Illustrating popularity of cross-platform mobile frameworks from 2019 to 2021 [18]

3.2.1 Web-based frameworks (hybrid approach)

Web-based frameworks also referred to as the hybrid approach [19] use web technologies, such as HTML, CSS, and JavaScript, to create cross-platform mobile applications. In the hybrid approach, these web-based applications are packaged in a native wrapper, typically a WebView component. The WebView provides a container for the application to run within, allowing it to access native device features whilst also sharing the WebView component across multiple platforms. This is also very powerful, as this lets the application in theory to be launched to any device that has support for a browser. This can range from not only mobile devices, but also bridges the gap to browser-based applications or desktop applications for Windows, MacOS and Linux.

This approach allows web developers to leverage their existing web development skills and tools to create cross platform applications. Whilst

some platform specific code will most likely have to be written, the amount of the codebase that can be shared is still the primary positive factor to this approach.

Ionic

Ionic is a popular web-based framework that allows developers to build cross-platform mobile applications using web technologies like HTML, CSS, and JavaScript [20]. It uses Angular, a popular front-end web framework, as its main technology stack, and is built on top of Capacitor, which provides access to native device features. As previously described, Ionic uses a hybrid approach where a WebView is used to render the frontend on the platform it's deployed on.

Ionic is known for its ease of use, as it provides a vast library of pre-built UI components, making it easier for developers to create visually appealing and consistent mobile applications [20]. Additionally, Ionic provides an extensive set of plugins and integrations with popular tools such as Firebase, Stripe, and Google Maps, making it easier to add functionality to applications [20]. With its focus on web technologies and its wide range of features, Ionic has become a popular choice for developers who want to build cross-platform mobile applications quickly and efficiently and that come from a web-development background.

React Native

React Native is a JavaScript framework that allows developers to create native mobile applications for both iOS and Android platforms. It uses a similar approach to hybrid development, but instead of using a WebView to render UI components, it uses actual native components that are assembled with JavaScript [21]. This allows the application to have a more native feel and better performance compared to traditional hybrid applications. React native can therefore be defined as a middle ground between hybrid and cross-compiled approach, as it has access to native components alongside the WebView.

React Native has gained a lot of popularity due to its ability to create applications with near-native performance while still allowing developers to write code in JavaScript, a language that is familiar to many web developers. The framework is supported by a large and active community, providing access to many third-party libraries and resources that can help speed up development. React is also backed and developed by Meta and used for their various applications including the Facebook app [21].

3.2.2 Cross-compiled frameworks

Cross-compiled frameworks utilize a single codebase written in a specific programming language, which is then compiled to native code for each platform. This approach allows developers to use their knowledge of a specific programming language and its associated ecosystem to build cross-platform apps without relying on web technologies. Cross-compiled frameworks often provide a set of APIs that abstract away platform-specific details and allow developers to write code that works on multiple platforms.

Examples of cross-compiled frameworks include Xamarin, Flutter and Kotlin Multiplatform. One of the primary advantages of cross-compiled frameworks is that they generally offer better performance compared to hybrid frameworks, as the resulting apps are compiled to native code. Additionally, cross-compiled frameworks can provide a more native look and feel compared to hybrid frameworks. However, there may still be some platform-specific code that needs to be written, and the development workflow may be different compared to traditional web development.

Xamarin

Xamarin is a cross-platform development framework created by Microsoft that enables developers to build native mobile applications for Android, iOS, and Windows using C#. Xamarin allows developers to create a single codebase that can be shared across multiple platforms, and it provides full access to the underlying native APIs and UI components of each platform.

One of the key differences between Xamarin and other cross-platform development frameworks is that it uses C# as its primary programming language. This allows developers to leverage their existing .NET skills and experience and provides access to the vast library of .NET libraries and tools. Additionally, Xamarin provides a fully integrated development environment within Visual Studio, which streamlines the development process and allows for seamless testing and debugging.

Another important feature of Xamarin is its use of platform-specific UI components. Unlike other frameworks that rely on HTML and CSS for UI development, Xamarin enables developers to create platform-specific UI components using the same programming language as the rest of their code. This allows for a more native look and feel across each platform.

Flutter

Flutter is an open-source mobile application development framework created by Google that uses the Dart programming language, which is largely based on JavaScript. Unlike hybrid web-based frameworks, Flutter utilizes a unique

approach called "Ahead-of-Time" (AOT) compilation, which compiles the code directly to native machine code for the target platform, rather than relying on a WebView component. This allows Flutter apps to run with better performance, faster startup times, and access to platform-specific features. Flutter also comes with a rich set of pre-built widgets, making it easy for developers to create beautiful and responsive user interfaces.

Kotlin

Kotlin is a programming language that has become the new standard in Android native development, which was formerly written in Java. Kotlin has an extension of the language called Kotlin Multiplatform, which is the framework for providing cross-platform development in Kotlin. Just like the other cross-compiled frameworks it allows developers to write platform-specific code while still having access to the shared codebase, making it a powerful tool for cross-platform development. The platform also provides interoperability with native code, making it easy to integrate Kotlin code with existing applications. Kotlin Multiplatform is a relatively new platform, but it has gained a lot of attention from developers due to its flexibility and ease of use.

4 Research project – Implementation

For this thesis, three different methods of data collection were used. First and foremost, there was a literature review to explore the current research surrounding the main topic of cross platform mobile application frameworks. This was followed up by a survey to gather quantitative data and could be considered explorative. This explorative quantitative data was then used to sharpen the research questions and served as a base for qualitative interviews held with some survey participants.

4.1 Survey questions and measures

In a previous study [1] the following issues could be picked on the questionnaire for the question "If any, which of the following issues do you relate to cross-platform development?":

- overall loss in performance compared to native apps
- Suboptimal User Experience (UX)
- immature frameworks (too cutting edge, too much risk, etc.),
- suboptimal options for creating good user interfaces (UI)
- immature communities (too new/low activity, etc.)
- hard to integrate with device APIs
- hard to test/debug
- security issues
- other

Based on these identified issues, specific categories were created, forming the basis for the survey questions. The categories for the survey questions in this study were defined as follows:

- Profiling (personal information) Q1-Q10
- Framework maturity (external sources, documentation, help) Q11-Q16
- Frontend design (prebuilt, custom) Q16-Q19
- Platform capabilities (APIs, sensors, hardware) Q20-Q24
- Coding Q25-Q26
- Personal opinions and future potential Q27-Q28

A total of twenty-six questions were formulated to gather data from the participants. Most of these questions were designed as single-choice questions, allowing participants to select an option on a Likert scale ranging from 1-5 or 1-7. The profiling questions, which collected demographic or background information, used pre-defined options related to the question, such as years of development experience or highest level of education.

The complete set of survey questions can be found in Appendix A1 along with a picture of the survey in appendix A2.

Note that no "other" option was provided to the participants for most questions, as the survey primarily aimed to gather quantitative data. The interviews, on the other hand, provided an opportunity for more in-depth discussions and qualitative data collection.

4.2 Interviews

The participants were asked to first explain further about their experiences with a framework they had used, based on their answers to the survey. This allowed me as an interviewer to gather more in-depth information on what the interviewees' experiences had been.

Further, the discussion was based around the following questions:

- What problems did you have with the framework?
- What can be improved in the framework?
- What do you think about cross-platform mobile development frameworks being used at large? Are they ready and a viable option?

These interview questions provide qualitative data to further strengthen the survey data on what needs to be improved in frameworks. The interview answers are more qualitative as I can go into further depth when asking about these frameworks, and the feedback for the frameworks becomes broader.

The survey was created using Google Forms. In the survey, mostly Likert-scale type questions were used and there was a total of 26 questions directly related to answering the research questions. More about this can be read in chapter 2.1.2 and 2.1.3.

Interviews were conducted with willing survey-participants, the final number of participants being four. These interviews were held on Zoom during November to December of 2022. Each interview lasted approximately 30-40 minutes, with a variety of people using different types of frameworks.

5 Results

In this section, the results of the literature review, survey and interviews will be summarized and presented.

5.1 Survey

After three weeks of sharing the survey on a variety of different online platforms, a total of 77 responses were gathered. The goal was to gather at least 50 responses based on similar studies of this nature that gathered data ranging from 30 to 100 responses . As the goal was reached, the survey was locked, and the data extracted to be analyzed.

5.1.1 Survey summary

The common use cases for cross-platform mobile frameworks can be answered with the survey questions 1 through 10.

According to the survey results, cross platform mobile application frameworks are being used at a large scale for a multitude of different projects, both at hobby scale but also at larger company scale. The most used frameworks of the survey participants were in order Flutter, Xamarin, Ionic, React Native and Kotlin Multiplatform.

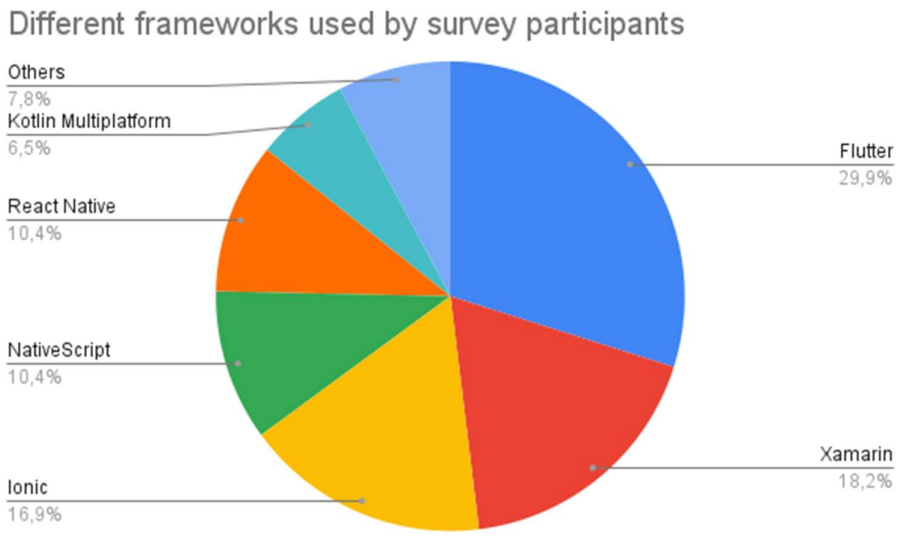


Figure 5.1 Illustrating a circle diagram of different used frameworks by survey participants.

The survey participants are generally well educated, 55,1% of them having a bachelor's degree, 16,7% a master's degree, and 1,3% a PhD. Most people

have been developing applications for more than 10+ years 35,9%, 5-10 years 29,5%, 2-5 years 25,6% and lastly 9% of people being rather new at 0-2 years of experience. 66,2% of people had previously worked with native development, and the other 33,8% had not done so.

The type of applications is very varied, with 24+ different categories being chosen. The majority were business applications coming in at 15,4% and tools at 10,3% of apps.

Around 50% of the participants started working on the application project during the years 2020 to 2022, giving a lot of recent data. The rest consisted of projects from 2019, 2018 and 2017 with only a handful 13,4% being even older than that.

The development teams of these apps were rather small, with 1 person representing 35,9% of apps, 2-3 people 39,7%, 3-10 people at 17,9% and 10-50 the remaining 6,4%. Therefore, the data only represents what could be defined as smaller teams.

5.1.2 Framework improvements

In appendix B1, averages for all the Likert-scale questions are presented across the different frameworks used. Based on this, we can conclude that Flutter was the best performing framework amongst the survey participants and the most popular one. Followed by Flutter is Kotlin Multiplatform with a score of 4.08, although only 5 projects used Kotlin and therefore the results may not be very accurate. NativeScript and React Native had 8 projects each and performed about the same with NativeScript in the lead. After these frameworks comes Ionic, but in dead last is Xamarin. Xamarin had 14 projects and was the second most popular framework but performed the worst by far on almost all questions.

Kotlin and NativeScript are rather new frameworks on the market, and this can be seen by the lack of access to educational material, documentation and framework maturity asked about in Q13 to Q18. But on Q18 asking if the participant believes the framework to be a mature framework, React Native falls behind almost as far back as Xamarin, and NativeScript is the winner even though they lack the elements of a mature framework asked about in previous questions. This indicates that there are other parts of a framework that people believe makes a framework mature.

The results of all questions can be found in appendix B2, to show what questions performed the best to worst. By also calculating the average answer

for each question answers, we can generate a list of the biggest perceived issues across all different frameworks. All these averages can be seen in appendix B3, and presented here are the worst performers:

Question	Average	Placing
Did you come across any platform quirks?	3,132	1st
Did you consider the used framework to be a mature framework?	3,565	2nd
Online resources (stackoverflow, YouTube)	3,639	3rd
The framework debugging tools allowed me to debug with ease	3,703	4th
How easy was it to work with the framework?	3,709	5th

Figure 5.1 Table of averages of the worst performing questions, showing which questions had the lowest average score.

Platform quirks are by far the worst problem according to the survey participants, all the frameworks performing quite poorly on this question and is an area that needs great improvement in the future for each framework to become more viable to the developer.

The second biggest issue is that the developer considered the framework to not be mature enough, which can also be seen in the bad performance of the first 5 questions, online resources is the third biggest issue and is directly related to framework maturity.
Following this is a lack of good experience with the framework debugging tools, which can be based on lack of experience or simply the debugging tools not being easy enough to use for the developer. But beware that the experience with this question can also be based on IDE, development platform and many more variables.

On place number 5 is simply an indication that the framework was perceived as hard to work with in general. This was asked as one of the last questions to get a general overview of the developers view on the framework, and this being negative is an indication that the framework is hard to work with. But this can also be an indication that cross-platform mobile development frameworks are quite hard to use. Flutter performed better than the rest on this question, with an average of 4,2 whilst the next runner up is NativeScript at 3,8.

5.2 Interviews

A total of 4 interviews were conducted, following up the survey results. The four participants were each user of the following frameworks:

- Interviewee 1 – Kotlin Multiplatform
- Interviewee 2 – NativeScript
- Interviewee 3 – Xamarin
- Interviewee 4 – Ionic

Although these are the frameworks the participants used, all of them have experience with multiple different frameworks, and therefore gave very varied responses and could respond in many perspectives. The interviews gave qualitative data to further in-depth answer research question 1, more specifically on how frameworks can be improved to better suit the common use cases of frameworks.

All interviewees proved the fact that cross-platform frameworks are being used to build a variety of different types of applications that are being used by up to thousands of people. Everyone seemed to quite enjoy using a framework and saw the upsides of using a shared codebase to unite projects that are cross-platform.

They all unanimously agreed on the fact that frameworks are a great way to build applications quicker as opposed to a shared codebase. And for smaller teams lacking development teams for both platforms, cross-platform frameworks are a great way of bridging the gap between multiple platforms. Although, multiple interviewees showed concern in using a framework for applications of a larger scale, or for a larger development team.

Another main problem that all the interviewees mentioned in one way or another, has to do with setup time for a framework and preparing the environment. Developing natively comes with a setup time on its own-preparing emulators for both platforms, on Android downloading the Android SDK, and for iOS it involves downloading the IDE Xcode. Installing a framework causes further time spent on preparation, having to download the SDK for the framework as well as setting it up for the IDE you want to use. And some frameworks allow for greater support in some IDEs than others, therefore even though you're used to a specific IDE, you might generally have a better time learning and using a new IDE that is more compatible with the framework.

One example that stood out was from the Xamarin developer, who comes from an iOS development background using Swift. To be able to use Xamarin, one

must use Microsoft Visual Studio. But it appears that Visual Studio is a subpar experience on Mac, and therefore it almost forced this interviewee to use the Windows version to be able to use all the functionality that he wished for.

All developers interviewed agreed that forcing the use of Xcode to launch an application the iOS app store was a huge hassle, as this involved having to use a Mac. This is a problem that occurs if you develop natively as well, and such can't be considered an issue really related to cross-platform development. But due to the shared codebase between the platforms, there's a certain expectation that it should be easier to compile to iOS, but that is not the case. All developers wished there were some sort of compromise- where the developer does not need to use Xcode to launch an application built with a cross platform mobile development to the app store.

Platform quirks were generally a large issue, especially when it comes to design. Depending on the platform you expect a certain behavior, and sometimes this behavior was as expected, or it simply defaulted to the behavior of one of the platforms. As an example, Ionic tended to default to explorative iOS behaviors.

To summarize the general opinions of the interviewees, all developers agreed that cross-platform mobile frameworks have their use cases. Frameworks serve its purpose as another tool in the developer's arsenal, but all frameworks have their weaknesses and strengths. As there's multiple frameworks, one might fit better for the type of application a developer is building. One of the best use cases in general though is for a small developer team having to develop for multiple platforms, as it saves a lot of time and is generally stable and mature enough for an application of not too large of a scale.

The answer to the question if frameworks are the future of cross-platform mobile development, the answer was always "it depends," with reference to the previous paragraph as everything has its own use case. Although frameworks are popular, other technologies such as PWA are becoming more mature and ready to use, which sometimes leads to some websites not having to use an app at all, but instead relying on PWA.

6 Analysis and discussion

In this chapter analysis surrounding the research questions will be presented.

6.1 Usage of frameworks

The usage of frameworks is prevalent. From the results of the survey, we can see that a large number of different applications are being built, of different scales and teams. Using a framework does not seem like a new and catchy thing, but rather simply being used as a tool where needed in the application development industry.

One question that comes to mind from the survey results is why aren't frameworks being used by larger companies for projects of bigger scale? A survey by JetBrains from 2022 [22] had a similar result, where this question was asked: "How many developers work on your mobile application on both iOS and Android simultaneously (including yourself). The answer was 32% just me, 42% saying 2-4 and then decreasing dramatically with 5-7 people as low as 8%, and the rest even less. If we compare this to the results of this thesis' survey, we get comparable results that are the following: 35,9% of teams were only one person, 39,7% 2-3 people, 17,9% 3-10 people, and lastly 6,4% 10-50 people. Although the questions are not asked in the same way, it still proves a small team size is common.

All the interviewees agreed that they saw using a framework in a larger company as bringing potential hassle, because most of them believed that the framework wasn't built to scale up in the way that is needed for a large team, where many people are working on the same project at the same time. But the truth is that there are many big companies out there using frameworks, although we don't know to what extent and how well it's functioning. Examples of these are Google Classroom, Google Pay, Reflectly and Alibaba using Flutter [23], Facebook, Microsoft Teams and other Microsoft products, Shopify, Wix and Tesla using React Native [24]. This rather proves the point that frameworks are being used, and very ready to be used at larger companies if these companies even considered using them.

But the interviewees agreed that using a framework is the perfect fit for a small team having to produce a cross-platform application. In the long run it will probably be worth it even though platform quirks and other issues might occur along the way.

6.2 Positive areas of frameworks

6.2.1 Performance

Loss in performance is an issue that was raised in a 2019 study [1], but according to this thesis' survey, does not seem to be a truly relevant problem anymore. The survey question "The framework allows me to build an app with high performance" gave the following responses that can be seen in figure 6.x.

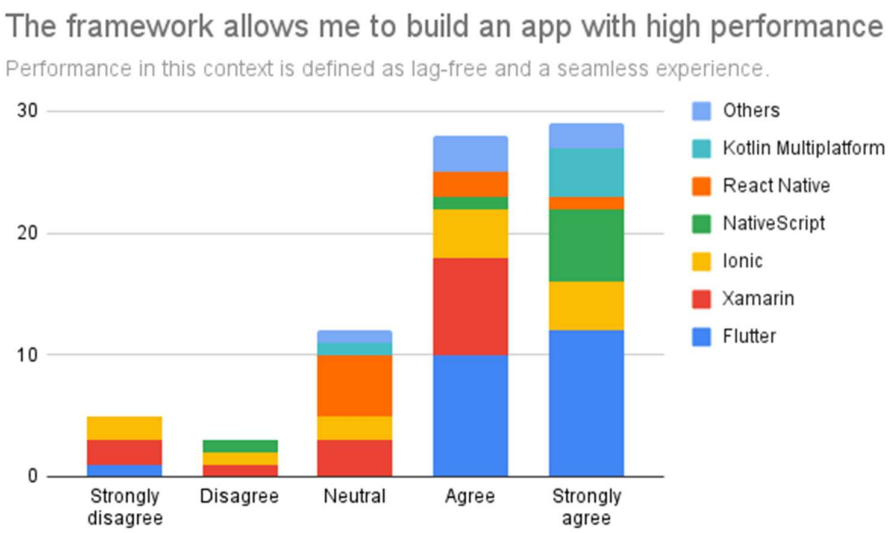


Figure 6.2 Illustrates a bar diagram of survey result performance question, divided into different frameworks.

This indicates a very high trust in the framework, being able to build an app with high performance. On average this is an average score of 3,95, and a 4 would be considered "agree" in this question.

6.2.2 Frontend design

The same kind of results can be seen in the questions surrounding user experience, with the 2019 study [1] presenting suboptimal UX and suboptimal options for creating good UI, whilst this thesis' survey participants indicate a rather good experience with frontend design:

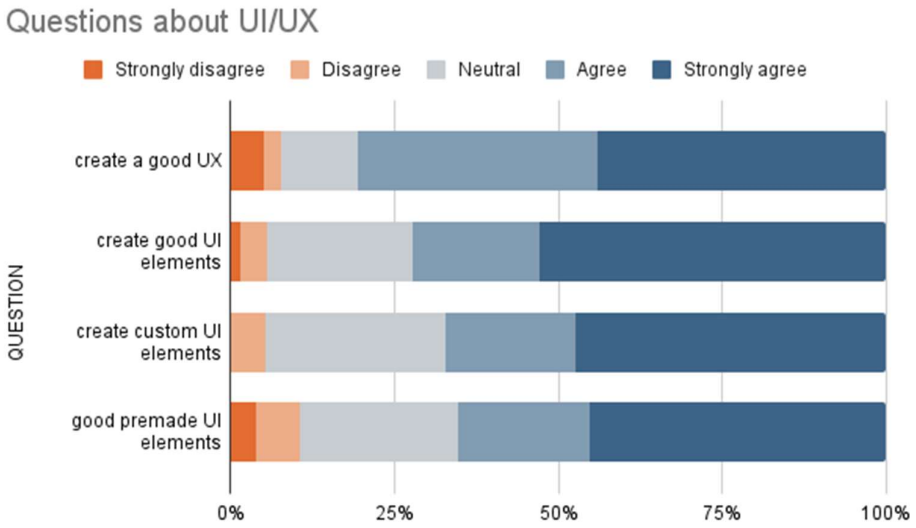


Figure 6.3 Illustrates a bar diagram of UI/UX questions overall.

Although, in interviews most participants mentioned having some kind of problem in the front-end design part of the framework they used. The Ionic participant especially had this problem, saying that the UI of their application could break completely very easily sometimes, and that the Ionic abstraction layers caused problems. The same goes for the Kotlin user, who said that he wouldn't even try to create a UI in Kotlin as it stands. And whilst Jetpack Compose is being introduced to Kotlin, they believed that it wasn't ready enough yet to be used in a production application. The Xamarin user also had concern with UI implementation- stating that an option for other Xamarin users is to use paid, very highly priced third party addons that bring a better UI experience.

6.2.3 Shared codebase

Finally, an honorable mention to the main function of them all, the shared codebase. It's by far the best feature according to the interviewees, as it really does help create applications more efficiently compared to having to write two separate applications. The time spent debugging and solving platform quirks is for most interviewees worth it in the long run.

6.3 Framework criticisms and solutions

This section further discusses the results in 5.1 and 5.2, combining the quantitative data from the survey with the qualitative from the interviews,

along with my personal analysis to create a broader perspective on framework issues.

6.3.1 Platform quirks

More than 50% of users came across a platform quirk occasionally or even more frequently. Platform quirks lead to further development having to be done, and sometimes these issues must be resolved by going into the native development platform that the platform quirk is appearing on. This can become a big issue, but 66,2% of survey participants said they have previously worked with native development, indicating some sort of capability of solving such an issue.

But let's not disregard the fact that this is the biggest issue perceived by the survey participants. Platform quirks are by far one of the biggest gripes with cross platform mobile development according to the developers I interviewed and is by far one of the hardest issues to fix. The platforms, mostly iOS and Android, are everchanging. They continue to push out updates constantly. Both platforms, but especially Android, run on a giant number of devices of different sizes and types, and therefore bringing a platform quirk free framework is a very difficult task. Yet, some frameworks have an easier time decreasing the platform quirks than others.

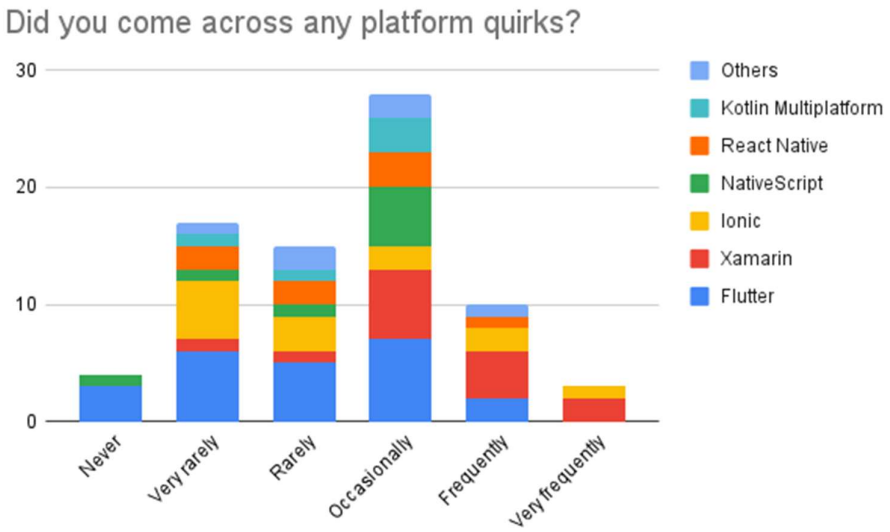


Figure 6.4 Illustrates a bar diagram of the platform quirks question, divided into different frameworks. In this case, more is bad and less is good.

In the front-end side of development frameworks that are using the web-based approach have an upper hand, as they don't have to rely on the type of

device as much. The same also goes for Flutter, which uses its own rendering engine. Flutter was the best performing framework in the survey, and had the least amount of platform quirks according to survey data. Although the truth is rarely that black and white, it is nonetheless a pattern that needed to be mentioned.

6.3.2 Framework maturity

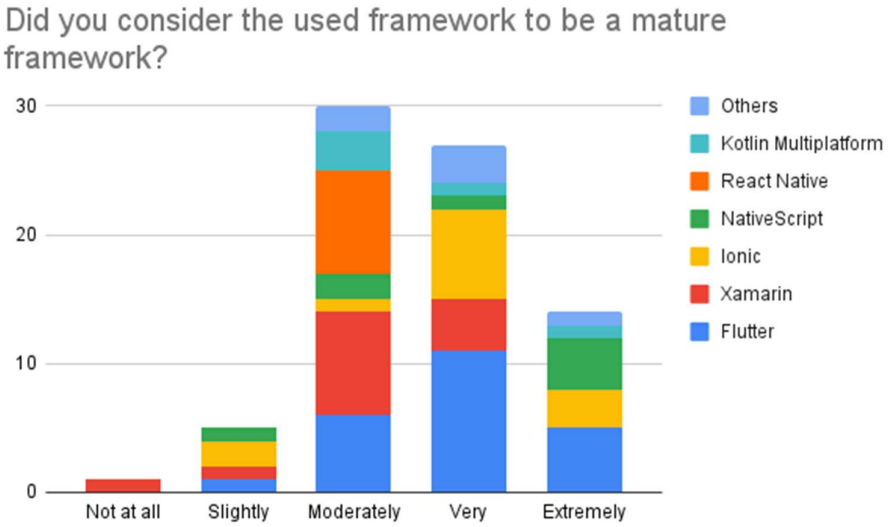


Figure 6.5 Illustrates a bar diagram of the framework maturity question, divided into different frameworks.

Based on the averages, this question was second in placement for the most negative answers, but looking at the available options, this doesn't appear to be a very badly performing metric. Most participants believed their framework was "moderately" mature, following "very" and "extremely" mature.

But let us put this statistic into perspective. Xamarin got a score of 2,3 out of 5 on maturity, the worst average of them all, over all questions. Xamarin was released in 2011, and yet it still has the worst results of all frameworks in this survey. It seems that Xamarin really has lost its way and is one of the clear losers that is in big need of an update, seemingly on all fronts.

But to summarize, in figure 6.x, most of the data lies from moderate to extremely mature. Only an exceedingly small number of participants chose "not at all" or "slightly." Therefore, the participants of this survey believed that the framework they used was quite mature, even though they generally seemed more doubtful when not asked as directly as this question was.

A framework doesn't mature just because it's old. Xamarin is a fitting example of this. What keeps a framework relevant and mature is it working for the newest technologies, and it continuously being worked on and maintained. One interviewee talked about how he saw new functionality released for a platform and got excited to use it but couldn't because it wasn't yet implemented in the framework he used. It does take longer for a new feature to land in a framework (in most cases) than native, as it must be implemented by the framework first. And if it isn't a priority, it can take a very long time.

Another part of the framework maturity lands in the hands of the users. Users can help by providing code for the open-source projects and providing bugs/feedback. This then also needs to be considered by the developers of the framework, and not just ignored. If there exists a large community, there will be community members that ask questions on StackOverflow, create YouTube tutorials, and talk about the framework on different forums. This all helps with the maturity of a framework.

6.3.3 Other issues

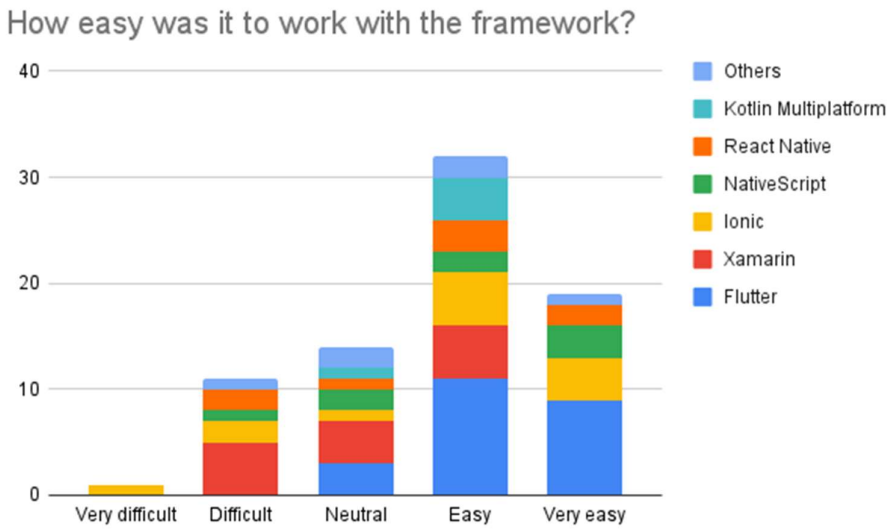


Figure 6.6 Illustrates a bar diagram of how easy it was to work with the framework according to survey participants.

Usability has a great impact on the general performance of a framework. If the developer doesn't enjoy using the technology, it will be a much bigger pain having to build in it. Although the average of this question is quite low,

the graph shows that most developers tend to think that the framework they worked with was either “easy” or “very easy” to work with.

Something that came up in interviews is the setup time of frameworks, which was previously talked about under interview results. All interviewees would see the framework as much easier to use if it weren’t for the large setup time often involved. It usually wasn’t as straight forward as hoped to be, and they all had very different experiences on getting started depending on what platform they used. Mac is the logical choice for development because of the access to both Android and iOS, whereas on Windows you only have access to Android. But the Mac experience is usually subpar compared to Windows. If more time and effort were put on improving the coding experience for both platforms, the ease of use for the framework would surely rise.

Debugging is also a crucial part of software development, and cross-platform mobile development is no exception. However, according to the survey results and interviewees, the current ones are subpar. One of the main issues with debugging cross-platform mobile applications is the lack of proper integration with native debugging tools. For example, Xamarin provides integration with Visual Studio's debugging tools, but the debugging experience can be inconsistent across different platforms. This relates to the Windows/Mac inconsistencies talked about in the last paragraph.

Kotlin Multiplatform has relatively limited debugging tools, as it’s a very new framework not much work has been put into the field of creating a robust debugging tool yet. Despite these limitations, there are still some debugging tools available for frameworks. For example, React Native has integration with the mobile app debugger Flipper, which provides deeper analysis tools further helping the application developer [25].

To summarize, more focus needs to be put on the basic functionality expected by developers, such as testing suites, debugging tools and being able to develop on the operating system you prefer. A framework is only a good tool if it’s a tool worth using over doing it the native way, and if the hassle of even using it is too big- it isn’t worth it.

7 Conclusions and Future Work

This thesis sets out to find the main problem areas in cross-platform mobile development frameworks, to help software developers further develop frameworks for application developers. The results of the survey and interviews have shown that the issues have changed over time, and that the focus should be on (in order) platform quirks, framework maturity and online resources, debugging tools and ease of use of the framework. Whilst the positivity of the survey participants on frameworks was high overall- issues still exist both on the fundamental levels of it being harder to develop for both platforms on Windows, and on the more general side where frameworks have specific issues that need to be solved. Although problems were identified, finding specific solutions for these issues is hard without further research into the fundamentals of a framework. All that this thesis can do is point to the issues.

7.1 Answers to research questions

RQ1: What are the common use cases for a cross-platform mobile framework currently and how can frameworks be improved to better suit them?

According to the survey results presented, cross-platform mobile frameworks are being used by developers to build a wide variety of applications across multiple platforms. The frameworks help build applications of small and larger scale, although smaller scale applications and smaller team sizes are preferred by developers using frameworks.

To answer the second part of the question, a summary is needed of what the participants thought. Positive aspects include a shared codebase, good availability of frontend design, performance somewhat on par with native applications, consistency across platforms and the entire developing cycle being more streamlined only having to build an application once instead of multiple times for different platforms. The negatives are platform quirks such as applications not behaving the same on two platforms, frameworks being too immature with too little online resources, debug tools and testing suites. Other problems include frameworks being hard to work with, problems such as bad documentation, subpar educational material available and framework setup.

Therefore, the negatives presented are the biggest underlying problems that need to be addressed to better improve the frameworks. Platform quirks is a game of constant trying to catch up as platforms are constantly changing with new updates. The best way forward is making sure to have an active

development group that works on fixing issues, but also a community that can provide fixes and bug reports. Platform quirks are fundamental issues that plague frameworks that will not go away for good, but more action needs to be taken to resolve them.

On the topic of framework maturity, frameworks will mature over time with more users using them and the communities growing. That leads to more educational material being made surrounding them, questions being asked online and more people contributing to the open-source aspects of the frameworks. The setup time for the Android and iOS environment for both platforms as well as frameworks could be dramatically improved, but a lot of problems lie on Apple for locking iOS development to only Mac and Xcode.

RQ2: Compared to previous research, how have the negative and positive aspects changed?
The problems have shifted, most notably with frontend design not at all perceived as a problem compared to previous research [1]. Platform quirks remain a problem but are now seen as a bigger issue than before. With the growth of many smaller frameworks, the issues are also more spread across different frameworks than they were before. Framework maturity is also a persisting problem, although time has passed since the previous study it does not seem to have affected people's opinion on frameworks maturity in general. Issues with testing and debugging also remain.

7.2 Future work

A lot more work can be done in the field of identifying concrete issues for specific frameworks that need to be improved on. There are a few major questions that can be asked:

- What are the major overhauls needing to be done to Xamarin to get it up to par with the other frameworks?
- In what ways does Kotlin Multiplatform need to grow and mature as a framework to challenge the other big frameworks?
- Does NativeScript have the potential to be the big next framework?
- In what ways can frameworks improve to better suit the need of large development teams building larger and more complex applications?

There's also a discussion to be had about what type of approach is currently performing the best, whether the web-based hybrid approach or cross-compiled approach is the way going forward.

8 References

[1] A. Biørn-Hansen, T.-M. Grønli, G. Ghinea and S. Alouneh, "An Empirical Study of Cross-Platform Mobile Development in Industry," *Wireless Communications and Mobile Computing,* vol. 51, no. 5, pp. 1-12, 2019.

[2] T. Vilček and T. Jakopec, "Comparative analysis of tools for development of native and hybrid mobile applications," in *2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 2017.

[3] V. Ionzon and S. Jägstrand, "A Company Case Study: Examining criteria in cross-platform evaluation frameworks," 2022.

[4] D. You and M. Hu, "A Comparative Study of Cross-platform Mobile Application Development," in *CITRENZ 2021*, Wellington, 2021.

[5] A. Ahmad, K. Li, C. Feng, S. M. Asim, A. Yousif och S. Ge, "An Empirical Study of Investigating Mobile Applications Development Challenges," *IEEE Access,* vol. 6, pp. 17711-17728, 2018.

[6] T. A. Majchrzak, A. Biørn-Hansen och T.-M. Grønli, "Comprehensive Analysis of Innovative Cross-Platform App Development Frameworks," i *Hawaii International Conference on System Sciences*, Mānoa, 2017.

[7] B. Kitchenham and S. M. Charters, "Guidelines for performing systematic literature reviews in software engineering," *Keele University,* vol. 2, no. 3, pp. 1-26, 2007.

[8] T. Dybå and T. Dingsøyr, "Empirical studies of agile software development: A systematic review.," *Information and Software Technology,* vol. 50, no. 9-10, pp. 833-859, 2008.

[9] M. Mader and G. Schermann, "Survey-based research in software engineering: An empirical analysis.," *Information and Software Technology,* vol. 53, no. 7, pp. 753-764, 2011.

[10] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering.," *Empirical Software Engineering,* vol. 14, no. 2, pp. 131-164, 2009.

[11] C. Seaman and A. Powell, "Interviewing techniques for software engineering research.," Springer, Boston, 2008.

[12] S. Brown, "Likert Scale Examples for Surveys," Iowa State University Extension, 2010.

[13] F. J. Fowler, Survey Research Methods, Thousand Oaks: SAGE Publications, 2013.

[14] J. Degenhard, “Forecast of the number of smartphone users in the World from 2013 to 2028,” Statista, 2023. [Online]. Available: https://www-statista-com.proxy.lnu.se/forecasts/1143723/smartphone-users-in-the-world. [Accessed 26 02 2023].

[15] “Meet Android Studio,” Android Developers, [Online]. Available: https://developer.android.com/studio/intro. [Accessed 26 02 2023].

[16] “Xcode 14 Overview,” Apple Developer, [Online]. Available: https://developer.apple.com/xcode/. [Accessed 26 02 2023].

[17] “The Six Most Popular Cross-Platform App Development Frameworks,” Kotlin Help, [Online]. Available: https://kotlinlang.org/docs/cross-platform-frameworks.html. [Accessed 26 02 2023].

[18] Statista, “Cross-platform mobile frameworks used by software developers worldwide from 2019 to 2021,” [Online]. Available: https://www-statista-com.proxy.lnu.se/statistics/869224/worldwide-software-developer-working-hours/?locale=en. [Accessed 28 March 2023].

[19] A. Biørn-Hansen, T.-M. Grønli and G. Ghinea, “A Survey and Taxonomy of Core Concepts and Research Challenges in Cross-Platform Mobile Development,” *ACM Computing Surveys,* vol. 51, no. 5, pp. 1-34, 2018.

[20] “Introduction to Ionic,” Ionic Framework, [Online]. Available: https://ionicframework.com/docs. [Accessed 26 02 2023].

[21] “React Native · Learn once, write anywhere,” Meta Platforms Inc, [Online]. Available: https://reactnative.dev/. [Accessed 26 02 2023].

[22] JetBrains, “Miscellaneous Tech - The State of Developer Ecosystem in 2022,” [Online]. Available: https://www.jetbrains.com/lp/devecosystem-2022/miscellaneous/#how-many-developers-work-on-your-mobile-application-on-both-ios-and-android-simultaneously-including-yourself-two-years. [Accessed 3 April 2023].

[23] Flutter, “Showcase - Flutter apps in production,” Google, [Online]. Available: https://flutter.dev/showcase. [Accessed 3 April 2023].

[24] Meta, “Showcase - React Native,” [Online]. Available: https://reactnative.dev/showcase. [Accessed 3 April 2023].

[25] Kotlin, “The Six Most Popular Cross-Platform App Development Frameworks,” [Online]. Available: https://kotlinlang.org/docs/cross-platform-frameworks.html#kotlin-multiplatform-mobile. [Accessed 6 April 2023].

A Appendix

A1 – Survey questions

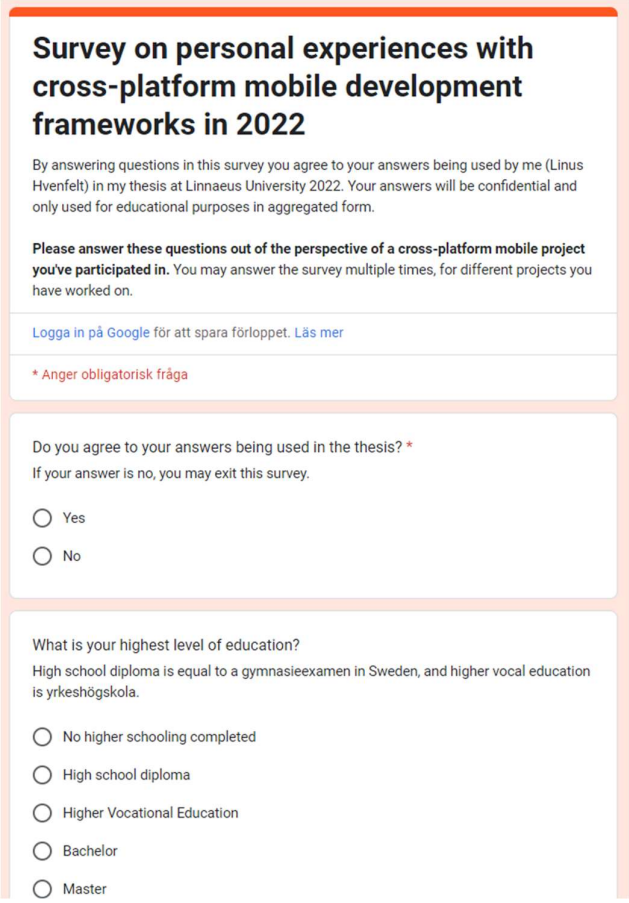
N	Question	Associated RQ	Type	Predefined options
Q1	What is your highest level of education?	RQ1	Single Choice	*1) No higher schooling completed (2) High school diploma (3) Higher Vocational Education (4) Bachelor (5) Master (6) PhD*
Q2	How long have you been developing applications for?	RQ1	Single choice	*(1) 0-2 years (2) 2-5 years (3) 5-10 years (4) 10+ years*
Q3	Have you previously worked with native development?	RQ1	Single choice	*(1) No (2) Yes*
Q4	In your project, what cross platform mobile development framework did you use?	RQ1	Single choice	*(1) React Native (2) Flutter (3) Kotlin Multiplatform (4) Ionic (5) Xamarin (6) NativeScript (7) PhoneGap (8) Unity Xcode (9) Other*
Q5	What type of application was created?	RQ1	Single choice	*33 different categories from Google Play Store, and other*
Q6	What year did you start working on this project?	RQ1	Single choice	*(1) 2022 (2) 2021 (3) 2020 (4) 2019 (5) 2018 (6) 2017 (7) 2016 (8) 2015 (9) 2014 (10) 2013*
Q7	At the time of this project, how much previous experience did you have working with the framework?	RQ1	Single choice	*(1) No prior experience (2) less than 6 months (3) 6 months to a year (4) 1-2 years (5) 2-3*

				years (6) 3-5 years (7) 5+ years
Q8	For how long was the project in development before release?	RQ1	Single choice	*(1) less than 6 months (2) 6 months to a year (3) 1-2 years (4) 2-3 years (5) 3-5 years (6) 5+ years*
Q9	How big was the development team?	RQ1	Single choice	*(1) 1 person (2) 2-3 people (3) 3-10 people (4) 10-50 people (5) 50-100 people (6) more*
Q10	What platforms was the application launched to?	RQ1	Multiple choice	*(1) iOS (2) Android (3) Windows (4) Mac (5) Web (6) Other*
Q11	How did you perceive these different aspects of the framework to be? Stability (not prone to crashing)	RQ2	Single choice	*(1) Very poor (2) Poor (3) Acceptable (4) Good (5) Very good*
Q12	… Documentation	RQ2	Likert scale	*1-5 scale, strongly disagree to strongly*
Q13	… Educational material	RQ2	Likert scale	*1-5 scale, strongly disagree to strongly*
Q14	… Online resources (stackoverflow, YouTube)	RQ2	Likert scale	*1-5 scale, strongly disagree to strongly*
Q15	… Libraries/packages/addons	RQ2	Likert scale	*1-5 scale, strongly disagree to strongly*
Q16	Did you consider the used framework to be a mature framework?	RQ2	Single choice	*(1) Not at all (2) Slightly (3) Moderately (4) Very (5) Extremely*
Q16	The framework allowed me to create a good user experience	RQ2	Likert scale	*1-5 scale, strongly disagree to strongly agree*
Q17	The framework allowed me to create good UI elements with ease	RQ2	Likert scale	*1-5 scale, strongly disagree to strongly*
Q18	The framework allowed me to create custom UI elements with ease	RQ2	Likert scale	*1-5 scale, strongly disagree to strongly*

Q19	The framework had good premade UI elements to use	RQ2	Likert scale	*1-5 scale, strongly disagree to strongly*
Q20	My application stayed consistent across the multiple platforms	RQ2	Likert scale	*1-5 scale, strongly disagree to strongly*
Q21	Did you come across any platform quirks?	RQ2	Single choice	*(1) Never (2) Very rarely (3) Rarely (4) Occasionally (5) Frequently (6) Very Frequently*
Q22	My framework allowed good access to device APIs and sensors	RQ2	Likert scale	*1-5 scale, strongly disagree to strongly agree*
Q23	The framework allows me to build an app with high performance	RQ2	Likert scale	*1-5 scale, strongly disagree to strongly*
Q24	I could easily integrate my application with a backend/API	RQ2	Likert scale	*1-5 scale, strongly disagree to strongly*
Q25	The framework allowed me to create good tests	RQ2	Likert scale	*1-5 scale, strongly disagree to strongly*
Q26	The framework debugging tools allowed me to debug with ease	RQ2	Likert scale	*1-5 scale, strongly disagree to strongly*
Q27	How easy was it to work with the framework?	RQ2	Single choice	*(1) Very difficult (2) Difficult (3) Neutral (4) Easy (5) Very easy*
Q28	Would you consider using this framework again for your next project?	RQ2	Single choice	*(1) Definitely not (2) Probably not (3) Possibly (4) Very probably (5) Definitely*
Q29	Do you consider cross-platform mobile application development frameworks to be the way of the future?	RQ2	Single choice	*(1) Strongly disagree (2) Disagree (3) Neither agree or disagree (4) Agree (5) Strongly agree*

Table of all survey questions

A2 – Image of survey



Survey on personal experiences with cross-platform mobile development frameworks in 2022

By answering questions in this survey you agree to your answers being used by me (Linus Hvenfelt) in my thesis at Linnaeus University 2022. Your answers will be confidential and only used for educational purposes in aggregated form.

Please answer these questions out of the perspective of a cross-platform mobile project you've participated in. You may answer the survey multiple times, for different projects you have worked on.

Logga in på Google för att spara förloppet. Läs mer

* Anger obligatorisk fråga

Do you agree to your answers being used in the thesis? *

If your answer is no, you may exit this survey.

- ◯ Yes
- ◯ No

What is your highest level of education?

High school diploma is equal to a gymnasieexamen in Sweden, and higher vocal education is yrkeshögskola.

- ◯ No higher schooling completed
- ◯ High school diploma
- ◯ Higher Vocational Education
- ◯ Bachelor
- ◯ Master

Image of survey on Google Forms

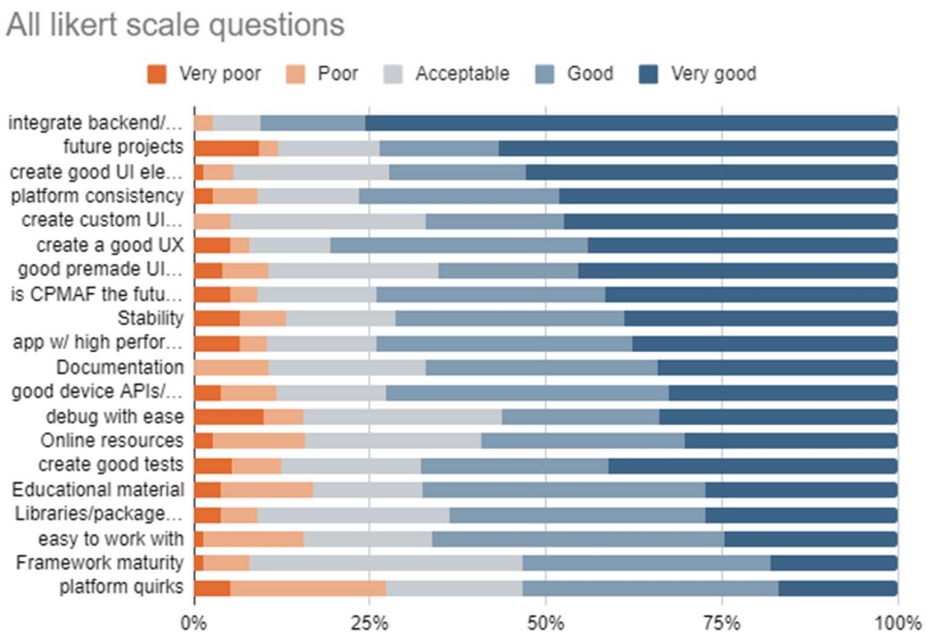
B Appendix

B1 – Survey results

	Flutter	Xamarin	Ionic	React Native	NativeScript	Kotlin Multiplatform
Q13	4,391	2,929	3,846	3,875	3,625	4,600
Q14	4,565	3,357	3,750	4,000	3,500	3,600
Q15	4,217	3,286	3,769	4,125	3,625	3,400
Q16	4,174	3,143	3,692	4,250	3,375	3,200
Q17	4,130	3,429	3,846	3,875	3,500	3,800
Q18	3,870	3,071	3,846	3,000	4,000	3,600
Q19	4,609	3,357	3,846	4,250	4,500	4,200
Q20	4,696	3,083	4,167	4,375	4,286	4,500
Q21	4,478	3,000	4,077	4,375	4,625	4,500
Q22	4,478	3,071	4,077	4,125	3,857	4,250
Q23	4,391	3,286	4,154	4,250	4,250	4,600
Q24*	3,536	2,369	3,244	3,188	3,292	3,167
Q25	3,696	3,643	4,077	3,875	4,875	4,200
Q26	4,391	3,214	3,538	3,500	4,500	4,600
Q27	4,682	4,286	4,692	4,625	5,000	5,000
Q28	4,067	3,400	3,444	4,000	4,125	4,250
Q29	4,091	3,214	3,300	3,286	4,125	4,200
Q30	4,261	3,000	3,692	3,625	3,875	3,800
Participants	23	14	13	8	8	5
AVG	**4,262**	**3,230**	**3,837**	**3,922**	**4,052**	**4,081**
Placing	1st	6th	5th	4th	3rd	2nd

Table of the average answer to all questions, separated by frameworks.

B2 – All likert scale questions



All likert scale questions, sorted by good and very good top to bottom

B3 – All questions averages

Question	Average	Placing
Stability (not prone to crashing)	3,878	9th
Documentation	3,795	8th
Educational material	3,737	6th
Online resources (stackoverflow, YouTube)	3,639	3rd
Libraries/packages/addons	3,763	7th
Did you consider the used framework to be a mature framework?	3,565	2nd
The framework allowed me to create a good user experience	4,127	
The framework allowed me to create good UI elements with ease	4,184	
The framework allowed me to create custom UI elements with ease	4,176	
The framework had good premade UI elements to use	3,976	
My application stayed consistent across the multiple platforms	4,155	
Did you come across any platform quirks?	3,132	1st
My framework allowed good access to device APIs and sensors	4,061	
The framework allows me to build an app with high performance	3,957	
I could easily integrate my application with a backend/API	4,714	
The framework allowed me to create good tests	3,881	10th
The framework debugging tools allowed me to debug with ease	3,703	4th
How easy was it to work with the framework?	3,709	5th

Averages for all questions.