DEGREE PROJECT IN COMPUTER ENGINEERING,
FIRST CYCLE, 15 CREDITS
STOCKHOLM, SWEDEN 2020

User Experience Influenced Model For Comparing Application Development Tools

CELINE MILEIKOWSKY

SEBASTIAN PORLING

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Abstract

There are many possible tools to develop mobile applications with. Choosing a development tool is done by considering many different factors, and the choice is currently done, in many cases, arbitrarily. For this project, a decision model is designed to ease the process of choosing a development tool.

A survey was conducted to examine how people using different smartphone platforms discover and download applications. 94 responses were collected, showing that approximately 50% of Android-users found mobile applications by using search engines or browsers. The corresponding number was approximately 30% for iOS-users.

A usability test was conducted to discover the differences in user experience between Progressive Web Applications and native applications. 18 usability tests were conducted comparing the same product developed as a Progressive Web Application and a native application. A majority of the participants had a technical background. Both Android and iOS devices were included in the tests.

The results indicated that end-users notice when an application is not natively developed. The effect on the user experience is combined with other technical differences and applied to the decision model. This model was designed to predict if a native application, a Progressive Web Application or a React Native application is the most favourable to develop for a specific scenario.

The final model could, according to consultants at the stakeholder Slagkryssaren AB, with good accuracy predict when the different development tools should be used. The model could be used as a discussion tool in the first stages of the development process of an application.

Keywords

Progressive Web Application, User experience, Native application, Hybrid application, Usability testing, React Native, Decision model, Weighted Sum Model, Multi-Criteria Decision-Method

Sammanfattning

Det finns många möjliga verktyg för att utveckla mobila applikationer. Valet av utvecklingsverktyg görs genom att överväga många olika faktorer, och görs idag i många fall högst godtyckligt. För det här projektet designades en beslutsmodell som förenklar processen av att välja ett utecklingsverktyg.

En undersökning gjordes för att undersöka hur användare av olika smartphone-plattformar upptäcker och laddar ner applikationer. 94 svar samlades, svaren visade att ungefär 50% av Android-användare hittade mobila applikationer genom internetsökningar eller webbläsare. Denna siffran var ungefär 30% för iOS-användare.

Ett användarbarhetstest utfördes för att finna skillnader i användarupplevelse mellan progressiva webbapplikationer och native-applikationer. En majoritet av deltagarna hade en teknisk bakgrund. Både Android- och iOS-enheter testades.

Resultatet tydde på att slutanvändare la märke till när en applikation inte utvecklades som en native-applikation.
Effekten på användarvänligheten, kombinerat med tekniska skillnader mellan verktygen, tillämpades på beslutsmodellen. Modellen designades för att förutse om en native-applikation, en progressiva webbapplikation eller en React Native-applikation är mest fördelaktig att utveckla i ett specifikt scenario.

Den slutgiltiga modellen kunde, enligt konsulter på uppdragsgivaren Slagkryssaren AB, med god precision avgöra när de olika utvecklingsverktygen bör nyttjas. Modellens användning blev som ett diskussionsverktyg i de första stadierna av processen med att välja utvecklingsvektyg.

Nyckelord

Progressiv webapplikation, Användarupplevelse, Nativeapplikation, Hybridapplikation, Användbarhetstestning, React Native, Beslutsmodell, Viktad summa-modell, Multikriteriebeslutsmetod

Acknowledgements

We would like to thank Slagkryssaren AB for allowing us to conduct this study at their facilities and sponsoring us with goodie-bags. We would especially like to thank Oskar, our supervisor at Slagkryssaren AB, for providing understanding of the business perspective of this project. We would also like to thank Fredrik Kilander, our academic supervisor, and Anders Sjögren, our examiner, for providing guidance through every predicament on the way. Finally, we would like to thank all the participants in the market survey and the usability tests.

Authors

Celine Mileikowsky <celinemi@kth.se> and Sebastian Porling <porling@kth.se>
Electrical Engineering and Computer Science
KTH Royal Institute of Technology

Place for Project

Stockholm, Sweden
Slagkryssaren AB

Examiner

Anders Sjögren <as@kth.se>
Kista, Sweden
KTH Royal Institute of Technology

Supervisor

Fredrik Kilander <fki@kth.se>
Kista, Sweden
KTH Royal Institute of Technology

Contents

1 Introduction

This chapter gives a short introduction and background of the area. It provides the problem to be solved and what the purposes and goals are for the conducted study. It introduces the research methodologies used, along with the stakeholders and this study's delimitations. Finally, it goes through the structure of the report as a whole.

1.1 Background

The market for mobile applications is constantly evolving and changing, with new research being made in both performance and usability. With a growth in the market for mobile internet devices, the market has also diversified. This has lead to many different devices for mobile internet access, including a multitude of smart phones, being available. For a mobile application to penetrate the market properly, it needs to be available across all popular mobile device platforms.

Progressive web applications provide this important cross-platform compatibility [1] without the need to develop several applications. Progressive web applications (PWA) are web applications which are engaging since they can be added to the home screen, leading to easy access for the device user. They are also fast and reliable since, just like native applications, they can be accessed without an internet connection [1]. The benefit of developing a PWA instead of a native application lies in its reachability and cost-efficiency compared to developing separate native applications for several platforms. Native applications refers to applications written for the top two mobile platforms: iOS, using Swift or Object-C; and Android, using Kotlin or Java.

Another solution to the platform issue is using a hybrid application, such as React Native. React Native is a mobile application framework which combines React, a JavaScript library, and native platform capabilities to develop applications for the most common platforms.

With a PWA or hybrid application, only one application needs to be developed, and that application can be run on a plethora of platforms. However, this cross-platform compatibility comes with some compromises in functions. A PWA on an

iPhone can for example not access push-notifications or advanced hardware, and on all platforms, the graphical design might differ from what is the norm for the platform as PWA doesn't use native components from iOS or Android.

1.2 Problem

A problem that has arisen with the diversification of the market is compatibility across the different mobile device platforms. Applications that work on one platform might not be functional on another, and companies are often forced to launch several different applications for different devices. A PWA can solve this problem, as a developer only has to develop one application, and that application can be run on all platforms. There are, however, some performance issues with a PWA. To find out to what extent these compromises in performance can be noticed by the end-user, user experience for PWA and native applications was tested.

How the choice of development tool affects the user experience was investigated. This effect was combined with the technical advantages and drawbacks of the different application development tools to determine which type of application was beneficial to develop in a specific case.

This study investigated the difference of user experience in PWAs and native applications to answer the following questions:

- Do users notice a difference between progressive web applications and native applications, and does the difference affect the user experience?

With these findings, combined with other factors, a model was built to answer the question:

- Can a model accurately decide whether a PWA, a React Native application or a native application is most favourable?

1.3 Purpose

Several studies have been made on mobile applications and web applications, but few have taken PWA into account. The purpose of this study was to define how using a PWA affects the user experience, and weigh that together with other factors

into a decision model. The model was to be used as a tool to determine if a PWA, a React Native application or a native application should be developed.

1.4 Goal

The goal was to develop a model for software developers that eases the process of choosing between different application development options. The model was to use the customers' and end-users' preferences and demands as a basis and produce the most suitable option for a mobile application development tool.

1.4.1 Benefits, Ethics and Sustainability

The impact of this study was analyzed using the four pillars of sustainability [2]. The UN Sustainable Development Goals [3] were reviewed for the planning of this study.

Human sustainability The data that was collected from the usability tests and surveys was to be handled in a way that it could not be accessed or used by any other parties than the authors of this study. The data was to be presented in a way that anonymizes the test subjects.

Social sustainability Products being available on all mobile platforms would improve the mobile connectivity aspect of the UN Sustainable Development number 9 Industry, innovation and infrastructure [3]. This effect, however, was not massive as the technology for these development techniques already existed.

Economic sustainability PWAs were at the time of writing overall easier, cheaper and faster to develop and launch than making two native applications, according to IT-consultants at the company Slagkryssaren AB. Making a model that helps the developers to choose a PWA at appropriate cases was therefore advantageous for the customer, and possibly the end-user.

Environmental sustainability This study did not have any foreseeable effects on environmental sustainability. The application development would not

change the environmental impact of the application in any noteworthy way.

1.5 Methodology

The research questions for this study were *"Do users notice a difference between progressive web applications and native applications and does the difference affect the user experience?"* and *"Can a model accurately decide whether a PWA, a React Native application or a native application is most favourable?"*.

These questions could be divided into two research sections: User experience and Designing a model.

1.5.1 User experience

To understand the user experience differences between PWAs and native applications the differences between the development methods had to be investigated, along with the experienced differences when using the applications. To get this understanding end-users tested applications made with both tools and provided feedback. The opinions of experts within the area combined with research and literature previously done on similar subjects was investigated.

1.5.2 Designing a model

In order to build a decision model, there was a need to establish which criteria were important to include in the model. When these criteria were established, how well the different development tools perform on said criteria had to be determined. A method for making a decision had to be decided on and implemented. The model was to be evaluated and improved upon until it produced accurate outcomes. The outcome was deemed accurate when input from previous use-cases where a specific development tool was chosen seemed to align with the recommendation from the model.

1.6 Stakeholders

The created model was designed for the company Slagkryssaren AB. Slagkryssaren AB is a tech agency that specializes in mobile platforms, back-end programming,

cloud computing, artificial neural network technologies and software architecture. This model would aid the process of deciding which development tool to use on new mobile application projects, so that their client's needs are met. Slagkryssaren AB supplied their expertise and experience when it came to choosing what criteria were important, how they usually made the decision between the different technologies and how the business perspective of the decision looked.

1.7 Delimitations

The most restrictive resource for this study was time. Only PWA, React Native and native development were included in the final results. There are at the time of writing many more hybrid solutions for making mobile applications, and there was no time to compare all of them. There was also no time to compare the user experience of more than two of these development methods, as this would have required the development of an app made with all three development methods. Such an app was not available at the time of this project.

The smartphones that were looked into were running the two most popular operating systems in Sweden [4], Android and iOS. The device used in the test was not the test subjects' own device, so there was a risk of them having to deal with a different layout than they are used to.

The usability tests were limited by demography and number of participants, as each test required a lot of time for planning, execution and analysis. The tests were limited by being performed in Stockholm only. The tests were not performed online, so our participants had to be present physically during office hours, which limited the number of participants with day-time occupations.

1.8 Outline

- Chapter 2 addresses the background of the different technologies, usability testing and decision-making methods.
- Chapter 3 explains in detail how the study will be carried out. This includes how to gather relevant information, how to test user experience and how to

design and implement a decision-method.

- Chapter 4 addresses the different steps in implementing the decision-method, show what was found in the literature and case study, and show the time-line of the user experience tests.
- Chapter 5 presents the result and explain the reasoning for choosing the different criteria.
- Chapter 6 presents the discussion of this study. This shows what could have been done differently and what future work could be performed.
- Chapter 7 presents the conclusion of the project and suggestions for future work in the field.

2 Theoretical Background

This chapter consists of all the background information needed to follow and understand what is included in this study. It gives an overall description on PWAs and React Native applications and their limits. How someone can evaluate usability, like usability testing, is described, and an overall description of decision-making and the Multi-criteria Decision-making method Weighted Sum Method is presented. Finally, it talks about the work that has previously been done in the area.

2.1 PWAs, React Native Applications, Native Applications and their differences

A mobile application is a software application designed to run on mobile devices such as smartphones. The two smartphone platforms which held the majority of the market share at the time of writing were iOS and Android [5], and three application development techniques were most commonly used [6]: Native, Web-based and Hybrid.

Native applications are mobile applications developed to function on a specific platform. If a developer wants to release an application on several platforms, a native application has to be developed separately for each platform.

Web-based applications utilise web servers to store data. This results in very low storage size on the user's device but only allows for access when an internet connection is available.

Hybrid applications are a mix of native and web-based applications. The idea is to write the majority of the code in a way that it can be compiled to run on several platforms. This is done by using a framework like Apache Cordova, Xamarin or React Native. Hybrid development allows for faster and more economic development than native development but produces applications which feel and function more like native applications than web-based applications.

2.1.1 Progressive Web Application

Progressive web applications (PWAs) work similarly to web-based applications, but by storing some data on the user's device they can be accessed offline [7]. PWAs can also access more hardware and functions than web-based applications. PWAs are typically not downloaded through the platforms own application distribution service, but rather accessed through the web browser of the device. Accessing the device hardware through the web browser adds another level of abstraction. This means a PWA is not able to access hardware to the same extent as native applications.

The functionality of PWAs is significantly better on Android devices than on iOS devices [7]. The term PWAs was originally coined by Google in 2015 [8], and Safari did not have support for the technology until 2018 [9, 10]. Some functionalities are still, at the time of writing, unavailable on iOS devices. A few of these features can be found in table 2.1.

Screen orientation & lock	Bluetooth
Credentials	Push messages
Background sync	Local notifications
Advanced Camera Control	VR and AR
Vibration	Recording media

Table 2.1: Selection of functions that work on Android but not on iOS for PWA. (VR stands for Virtual Reality, AR stands for Augumented Reality)

Some functions remain unavailable on both platforms. The lack of advanced sensors and contact information are two major limitations, inhibiting some uses for PWAs. The lack of Near-Field Communication (NFC) support, for example, makes contact-less payments impossible. Some functions have been experimented with, but have not appeared as officially available. A selection of these features can be found in table 2.2.

Proximity sensors	Contacts
Geofencing	NFC
Task scheduling	Ambient light

Table 2.2: Selection of features that do not work on either iOS or Android for PWA.

2.1.2 React Native

React Native developed applications have access to more functions than PWAs. A drawback of using React Native is the poor performance aspect. Due to its single-threaded nature, JavaScript is not optimal for heavy computing, therefore applications heavy in graphics tend to perform poorly when developed as React Native applications or PWAs compared to native applications [11].

When developing a React Native application, a frame is first built. Then, for each platform, this frame is built upon to make the application act like a platform-native application. About 95% of the code is commonly built as cross-platform, and the remaining is platform-specific [12], but this number can vary depending on the application. This means developing a React Native application is more budget-friendly than developing two or more native applications. However, the development of React Native applications demands knowledge of the different platforms. This differs from a PWA, where knowing JavaScript is enough to develop one cross-platform application.

2.2 Evaluating usability

Performing usability tests is a resource-efficient way of testing a product's usability, since not so many tests are required to find a majority of the issues with the product. After just 15 tests, as much as 90% of the problems with the product can normally be found [13].

There are several ways to test a solution's usability [14]. One could, for example, conduct Remote Moderated Research, where the test subjects use the application during their own time and then report back to the researcher how they feel about the product they have tested. This method is useful when it is vital to the test that the product is tested during a specific time or activity, in a specific place or for a longer time period [14, p. 353]. With this method, there is however a delay in receiving feedback from the test subjects.

There is also the option of A/B testing where the test subjects, in a controlled environment, get to test out two (or more) versions of the same product. After the tests, the subjects rate the solutions and communicate their preference. This

method is good for choosing between two alternatives. This could, for example, be two different graphical elements, or two different wordings. A/B testing is most useful when a choice between two or more product has to be made, but it does not answer the question of why the subject chose a certain product version. [14, p. 12].

Group discussions, such as focus groups, are also an option for usability testing. Several subjects get to sit together and review the products. When there are plenty of test subjects to utilize, this is an efficient way to get several people's opinions at the same time. However, for a first-hand experience test, it demands that there are enough devices to test on and resources to capture all participants' thoughts and feelings about the product. [14, p. 228].

Comparative usability testing is another alternative for conducting a usability test. A subject gets to test two (or more) products in front of the researcher, and the researcher notes the subject's reactions to the products. This method is resource-efficient and only demands one device to test on. It is also relatively robust, as the tests are performed separately on different subjects, small errors will not affect the outcome of the study as a whole [15]. The test subjects would test both applications, to maximize the input from the subjects on each version.

2.2.1 Conducting a Usability test

In order to conduct a usability test, one needs to design a number of tasks or scenarios for the test subjects to perform. The subject will perform these tasks under the supervision of the moderator [16]. The moderator is there to introduce the applications and to help the subject to proceed if they get stuck. A person, referred to as a recorder, takes notes from the test. This is useful since the moderator might be too busy with the test subject to properly record what is happening. Test subjects are encouraged to think out loud, giving the researchers qualitative data in real-time. This also allows the moderator to ask questions during the test, to further understand how the test subject's reactions to the product. The tests are documented, using a camera angled at the device being tested and/or a camera aimed at the participants' face. This is done to conserve the qualitative data, allowing for later analysis. The setup for the tests looks like

the example in figure 2.1.

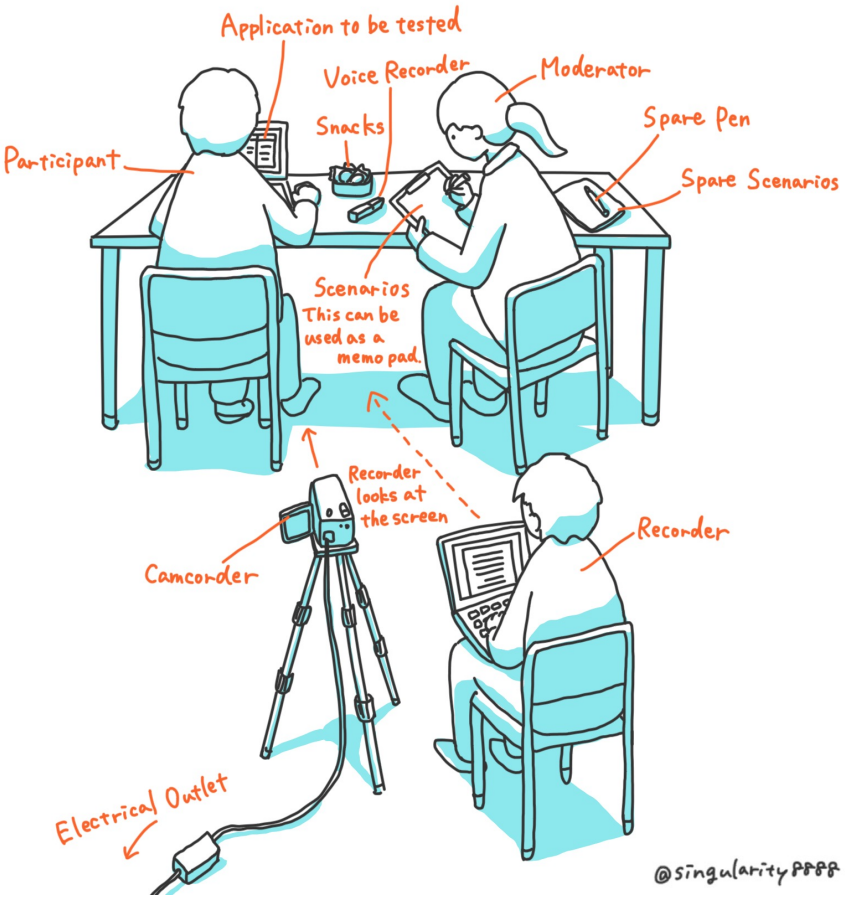


Figure 2.1: *Diagram of conducting a simple usability test, illustrated by Yukari Shingyouchi [17].*

To collect quantitative data efficiently, one can utilize forms and questionnaires in their usability tests. The subject assesses the products after, or during, the tests. The System Usability Scale (SUS) is an easy technique that is reliable even on small sample sizes [18]. It uses 10 items that can each be rated on 5 steps from Strongly agree to Strongly disagree, shown in figure 2.2.

Every odd question in the SUS is asked in a way such that a ”1” corresponds to a negative response while a ”5” corresponds to a positive response, while the even questions are vice versa. Because of this, calculating the SUS Score is done in the following way: For every odd-numbered question, subtract 1 from the score. For every even-numbered question, subtract the score from 5. Sum the scores from all questions, then multiply the total with 2.5. The score will then get a grading that is based on known results shown in table 2.3.

	Strongly disagree				Strongly agree
1. I think that I would like to use this system frequently					
	1	**2**	**3**	**4**	**5**
2. I found the system unnecessarily complex					
	1	2	3	4	5
3. I thought the system was easy to use					
	1	2	3	4	5
4. I think I would need support of a techncal person to be able to use this system					
	1	2	3	4	5
5. I found the various functions in this system were well integrated					
	1	2	3	4	5
6. I though there was too much inconsistency in this system					
	1	2	3	4	5
7. I would imagine that most people would learn to use this system very quickly					
	1	2	3	4	5
8. I found the system very cumbersome to use					
	1	2	3	4	5
9. I felt very confident using the system					
	1	2	3	4	5
10. I needed to learn a lot of things before I could get going with this system					
	1	2	3	4	5

Figure 2.2: *SUS questions [18].*

The user experience questionnaire (UEQ) seen in figure 2.4 is a fast and reliable way to measure both usability aspects and user experience aspects [19]. The UEQ can be separated into 6 categories: Attractiveness, Perspicuity, Efficiency, Dependability, Stimulation and Novelty. All questions in the questionnaire affect the score of one of these categories. If the result from one of the categories is not needed for the test, it's corresponding questions can be left out of the questionnaire. A UEQ comparing two different versions of a product could produce a result as shown in figure 2.3.

SUS Score	Grade	Adjective Rating
>80.3	A	Excellent
68 - 80.3	B	Good
68	C	Okay
51 - 68	D	Poor
>51	F	Awful

Table 2.3: Grading system for SUS.

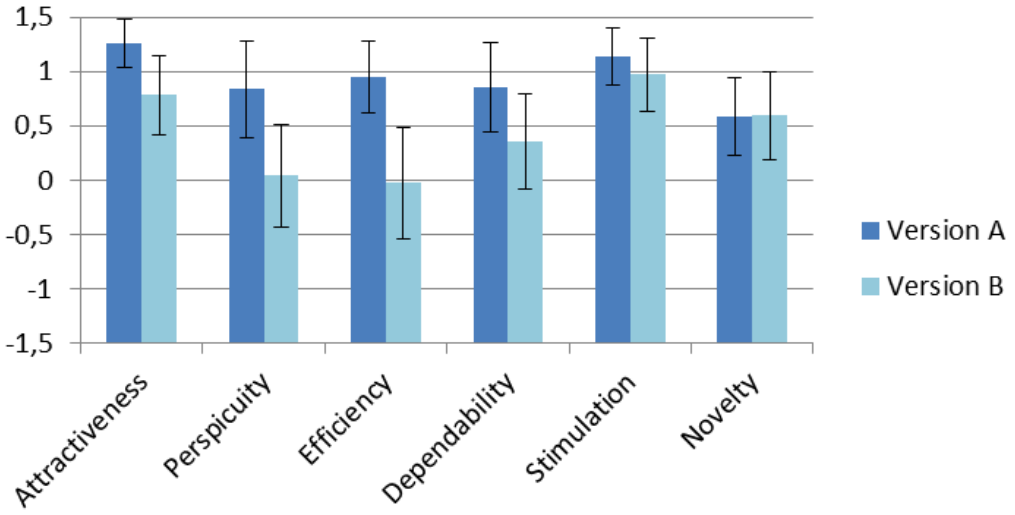


Figure 2.3: *UEQ result of a comparison between two versions of a product. The mean of each product on each category presented in blue bars, the confidence interval marked in black [20].*

2.3 Decision making

Decision-making is the practice of choosing an alternative based on available information. When the decision is based on many criteria, the process of choosing the optimal alternative falls under Multiple-criteria decision making.

2.3.1 Multiple-criteria decision making

Multiple-criteria decision making (MCDM), a collective term for different ways to optimally choose an alternative of something based on multiple, possibly conflicting, criteria [21]. The criteria of an MCDM can be factors such as cost, convenience, quality etc. In most decisions, the criteria have varying importance. This is represented by giving the criteria different weight, such that a criterion with high weight is more crucial to the final decision than one with a low weight.

English version

obstructive	o o o o o o o	supportive
complicated	o o o o o o o	easy
inefficient	o o o o o o o	efficient
confusing	o o o o o o o	clear
boring	o o o o o o o	exiting
not interesting	o o o o o o o	interesting
conventional	o o o o o o o	inventive
usual	o o o o o o o	leading edge

Figure 2.4: *Example of UEQ questions [19].*

A very common method for the MCDM is the Weighted Sum Model (WSM) [22, p. 6]. With WSM, all criteria are given weight by the decision-maker, and each alternative is given a score on each criterion. For each alternative available, the weight of every criterion is multiplied by the score. These numbers are summed, producing a WSM-score for each alternative. The alternative with the highest WSM-score is the optimal solution. The WSM uses relatively simple mathematics, but it requires the data in the model to be quantifiable.

Another popular method is the Analytical Hierarchy Process [23], which works by doing a pair-wise comparison of each criterion. One flaw with this method is that it's not very good at dealing with many criteria, as the decision-maker has to remember what they decided on previous comparisons and why, or else they risk making contradictory rankings [22, p. 11].

2.3.2 Weighted Sum Method Example

An example of a WSM: Charlie is choosing a car. They have three alternatives to choose between: A Kia Picanto, an Audi A8 and a Rolls Royce Phantom. Charlie has the following criteria to take into consideration: Price, Comfort, Performance and Prestige. The weights for each criterion and the score for each alternative is shown in table 2.4.

The weights indicate how important the different criteria are to the decision-maker. Charlie has set the highest score on the criterion Price, meaning they think that Price is the most important criterion.

	Price	Comfort	Performance	Prestige
Weight	0.50	0.25	0.20	0.05
Kia Picanto	10	3	2	1
Audi A8	6	6	8	5
Rolls Royce Phantom	1	10	10	10

Table 2.4: Weighted sum model matrix example.

The score for each criterion shows how well an alternative fulfils a criterion. A higher score indicates that the alternative is more superior in a criterion, for example, the Rolls Royce Phantom has a score of 10 on Comfort, indicating that it is a very comfortable car compared to the other alternatives.

Formula (1) is used to calculate the WSM-score. *a* refers to the alternatives, *n* is the number of criteria, *m* is the number of alternatives and *w* is the score of each alternative on a criterion.

$$A_i^{WSM-score} = \sum_{j=1}^{n} w_j a_{ij}, \ for\ i = 1, 2, 3, ..., m \qquad (1)$$

Car model	WSM-score
Kia Picanto	6.2
Audi A8	6.35
Rolls Royce Phantom	5.5

Table 2.5: Example WSM score for the example in table 2.4

As seen in table 2.5, in this scenario, the most favourable option for Charlie would have been the Audi A8, with the Kia Picanto coming in as a close second.

2.4 Related Work

Comparisons between web applications and native applications have been performed before [24]. This study, however, did not take PWAs or hybrid solutions into account. This study also did not take other factors apart from responsiveness into account when measuring the user experience.

PWAs have been compared to native applications in a study [25]. This study compares the performance aspect of the two techniques and also explored the UX

differences. However, this study's UX assumptions were based on a study [26] performed on native applications versus web applications, not PWAs.

Hardware responsiveness of PWAs has been tested in one study [27]. This study focused on Android and only researched the response time of the camera and GPS of a device. Another study [28] compared performance in launch-time, installation size and rendition time consumption.

A study has been made on the usability of Web applications and Hybrid applications [29], but did not include PWAs.

The usability of PWAs versus native applications has been investigated in a study comparing web applications, native applications and PWAs [30]. This study only performed a qualitative study on an Android device.

3 Method

In this chapter, the methodologies used for the project are presented. This includes the research methodology, the project methodology, which methods and tools that were used to answer the research question and the technical methodology.

The methodology gives a clearer understanding of how the project was conducted and why the different parts of the project were included.

3.1 Research methodology

The research questions for the project were as follows.

- Do users notice a difference between progressive web applications and native apps, and does the difference affect the user experience?
- Can a model accurately decide whether a PWA, a React Native application or a native application is most favourable?

The first question, regarding the user experience, was divided into two sub-tasks:

- Understand how the choice of tool affects user experience (UX). A usability study was performed to understand what affect on UX using different development tools has.
- Understand how users find mobile applications. This sub-task was performed by sending out a market research survey.

The second question was divided into two sub-tasks:

- Understand the limitations of Development tools. A literature study and discussions with consultants at Slagkryssaren AB were performed to gather data on the limitations of the development tools.
- Understand the factors involved in choosing a development tool.

The factors for choosing a development tool were explored in discussions with consultants at Slagkryssaren AB.

The study followed the four-step method of Pólya. This method was chosen since it divided the main problem into smaller, more perspicuous problems. This method allowed for an iterative work process, which would produce deliverable products that could have been improved on in increments. The model contains four steps: Understand the problem, Devise a plan, Implement the plan, and Evaluate and control [31]. The four-step method and its parts can be seen in figure 3.1.

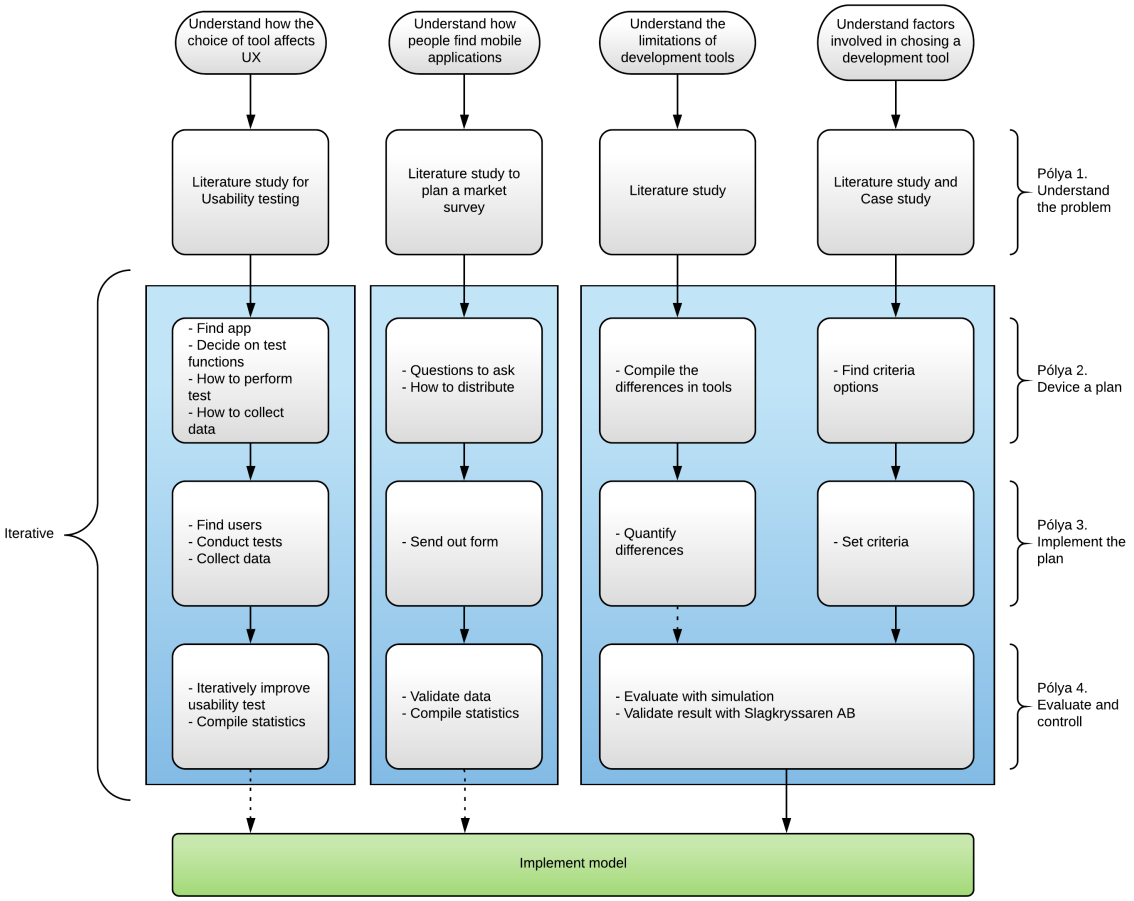


Figure 3.1: *Implementation of Pólya's four-step method.*

3.2 Project methodology

The project methodology is a process which guides the project from a plan to a finished product. Resource management, cost efficiency and prioritization are some of the factors to consider when entering a new project, and following a well-defined project method would increase the chances of producing deliverable results with the resources available.

3.2.1 The project triangle

The project triangle, shown in figure 3.2, describes how the quality of the project correlates with the time, cost and scope [32]. The cost and time of this project were predetermined, so the only mobile variable was the scope. This means that the scope of the project could have been extended or cut as needed. This could have been applied to the usability tests, as more or fewer tests could have been conducted depending on the time left. Extending the scope could, for example, mean introducing more technologies into the model. The scope could also have been extended, improving the quality, if more tests had been conducted.

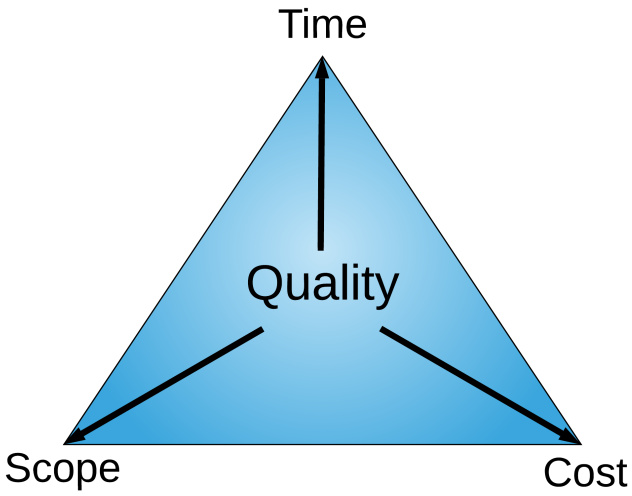


Figure 3.2: *The project triangle*

3.2.2 MoSCoW

MoSCoW is a method used to prioritize during project management [33]. The term MoSCoW is an acronym for the four prioritization categories: Must have, Should have, Could have and Won't have. The MoSCoW was chosen as a prioritization method for this project due to its ability to fraction the project into a core assignment and possible improvements. The MoSCoW for this project can be seen in figure 3.3.

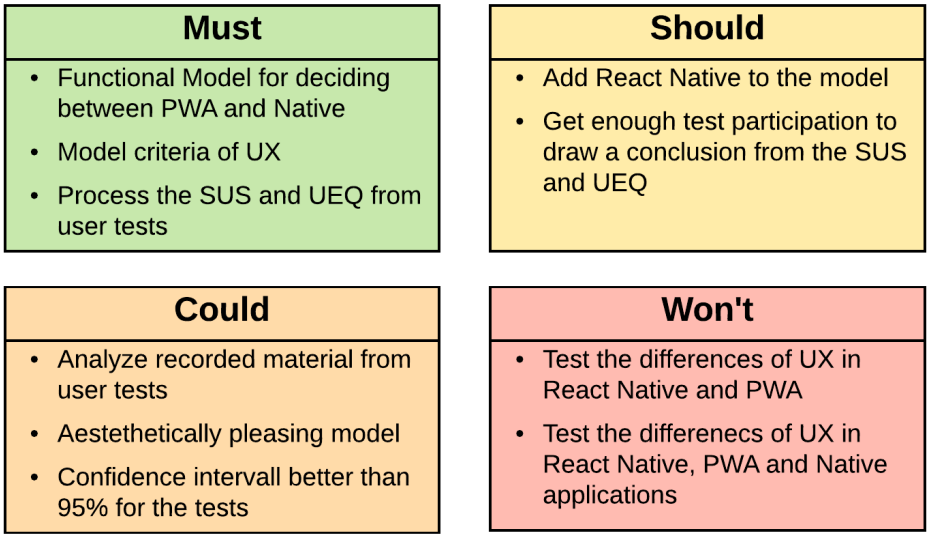


Figure 3.3: *MoSCoW chart of this project*

3.2.3 Connecting the dots

With the research and project methodology explained and established it was easier to get hold of the overall picture of this study. The scope and iterations were limited by the time and the cost, two factors which this study could not affect. When the tasks of the MoSCoW category ”Must” had been fulfilled, depending on how much time was left, more tasks from the categories ”Should” and ”Could” could have been added to the project.

3.3 Literature Study

To understand and explore the differences between the different types of mobile applications, extensive research had to be made. Mobile applications that use more of a hybrid approach, such as PWA or React Native, tend to follow trends. This means that if the technology does not get enough attention it might go extinct. This was at the time of writing happening to many JavaScript frameworks such as Angular, according to consultants at Slagkryssaren AB. The information available on this subject was both opinion- and research-based.

For the MCDM, the design practices and evaluation processes were investigated. To conduct usability testing, it was a requirement that these tests should

follow typical practices currently in use. This includes how to design, conduct and validate usability tests. Typical places where information was gathered were:

- Blogs, forums and similar websites
- Research articles and technical literature
- Opinions from developers and experts within interaction design

3.4 Case study

Since the model was to be used by Slagkryssaren AB, the development of the model was done in close co-operation with the company's consultants. These consultants have extensive experience in the field of developing applications and choosing application development tools. This inductive approach was more suitable than a deductive one, as there was no decidedly right or wrong conclusion when choosing an application development tool.

The case study was conducted by holding discussions with the consultants at Slagkryssaren AB. These discussion meetings were to be unstructured in an attempt to get richer, more qualitative data. A combination of planned and improvised discussions were held to answer questions emerging during the progression of the study. The meetings were held approximately once a week and lasted around 40 minutes each.

3.5 Mobile application discovery survey

The survey was conducted to explore two hypotheses which emerged during the discussions with Slagkryssaren AB, one of them being that people tend to find information about applications mostly on the platforms application store, rather than using web searches. The other hypothesis was that Android users tend to look for information about applications through web searches, while iPhone users tend to go to the App Store first. This hypothesis was based mainly on the fact that smartphones running Android usually have a Google search bar pre-installed on the home screen. It would, therefore, be easier for them to just tap the search bar to find information, rather than opening the App store application which one

has to do on iOS devices. These hypotheses were explored to understand the consequences of releasing an application which can not be found on the platform's app store, which is the case for PWAs.

3.6 Usability testing

To see the differences in user experience, usability tests were conducted. Usability testing was chosen as a method for measuring UX due to its relative simplicity and time efficiency.

During the tests, PWA and native applications had the main focus. React Native and PWA both use JavaScript but React Native can access native user interface components, meaning it is like a mixture of both types. To be able to perform the usability tests with all three techniques we would have needed a similar application developed with all methods, something that was not available at that moment, and outside of the scope of this study to develop.

To measure the differences in quality, usability, attractiveness and other UX-related criteria, validation of the tests was done using surveys. Several measuring methods for this exist, such as the System Usability Scale (SUS), the Usability Experience Questionnaire (UEQ), Net Promoter Score and the Standardized User Experience Percentile Rank Questionnaire. These surveys are popular for measuring how good the different applications are in certain criteria [34, 35]. The surveys can only receive quantifiable data. A better overview of the process of the tests is shown in figure 3.4.

The users tested two versions of the same application on the same platform they normally used. One of the application versions was a PWA and the other a native application.

The subject got a list of six tasks to perform on the application, and after using each version of the application the subjects filled in a UEQ and a SUS form. The results from these SUSs and UEQs formed the base for the quantitative data from the usability tests. During the tests, the moderator encouraged the subject to keep talking by asking questions about what the subject was doing. To fully understand the test subject's experience, it was useful to have them think out loud. This means

the subject says out loud what they are thinking and trying to do while testing the application, giving the researcher an insight into what the subject has troubles with and misunderstands in the product. Thinking out loud gave qualitative data from the users [36].

After both applications had been tested, the subject was asked more in-depth questions about their experience of the applications and their thoughts and feeling. This gave richer, more nuanced comparisons between the different applications, and qualitative information which could later be processed to further define how the criterion for UX should be scored in the model.

The recorded material from the tests was analyzed. The subjects' feelings about the applications were interpreted through their spoken words, their facial expressions and verbal cues. This data was then quantified by processing the recorded material and transcriptions, in search of patterns in the subjects' behaviour. The material was only showed to the test conductors, and all material collected was only used for the purpose of the thesis. After this, the material was deleted.

The results from the UEQ and SUS were plotted and compared between the two applications. With this data, a confidence interval could be calculated and compared to the preferred confidence level. This result formed the basis for the criterion of UX.

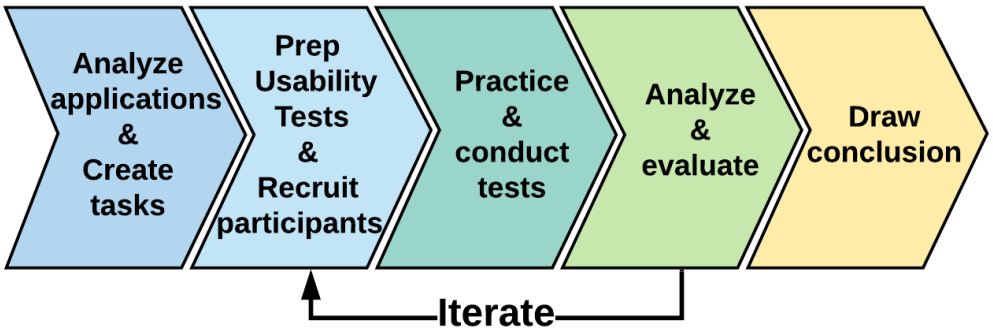


Figure 3.4: *Process of performing a usability test*

3.7 Multi-Criteria Decision-Method

To build a model which took usability into account, the comparative usability test was conducted. This inductive approach would produce qualitative data. The different solutions tested in a comparative usability test could be tested either separately on the different test subject, or together on the same subjects. In this study, both solutions were tested on all test subjects. A drawback with this method is that there could emerge a pattern that the user would like a particular version if they are tested in the same order [15]. To counteract this effect the subjects tested the applications in varying order so that the bias would have the same impact in both directions, hopefully cancelling out in the end. This was done since the scope of the study was comparatively small, meaning the benefit of more test subjects was vital.

To make the results as comparable as possible, we chose to have all the test subjects perform the same set of tasks. The tests were conducted on the same OS that the test subject normally used, to minimize confusion due to an unknown platform [37]. For the model, it was also relevant to include application exploration as a criterion. To acquire quantitative data for the finding and downloading habits of the population, a cross-sectional survey was conducted. This gave data from a more diverse population than is possible from qualitative methods, such as interviews, in the span of the study.

The model itself was based on the MCDM method. The MCDM uses a mathematical function to calculate the best alternative when faced with a decision involving many different criteria. The decision-makers decide which criteria are important and rank them. The possible alternatives are then scored on each criterion. The alternative with the highest score is then the best alternative for that decision-makers ranking.

Several MCDM methods could have been applied to this study. WSM was chosen due to its simplicity of calculation. Since the WSM demands the data to be quantifiable, all qualitative data gathered was translated into quantitative data. Mobile application discovery was researched with an online survey, producing quantified data.

For the tests, quantification was done with surveys and questionnaires where test subjects quantified their own experience with the application, in combination with analyzed facial expressions and behaviour from the recordings.

The criteria were based on input from results of our test and survey, scientific articles, software development websites and discussions with consultants at Slagkryssaren AB.

When a set of criteria was decided, scores were given to each alternative development tool in the model. The scores were set based on previous literature study. These scores were presented to the consultants at Slagkryssaren AB and were iteratively improved upon based on their feedback.

When scores were set for the alternatives, the next iterative phase began. Use cases were run through the model, and the result was assessed. If the outcome was seen as unreasonable or unlikely to be chosen, the scores and criteria were reassessed. This iterative process was to be repeated until reasonable results are acquired, or until there was no more time to reiterate.

3.8 Technical method

In order for this study to be carried out, there was a need to include some tools. Testing different ways of calculating the decision-making model and analyzing the data from the survey, UEQ and SUS was done using *Google sheets*. Gathering the data for the tests and survey was done using *Google forms*. The recording of the usability tests was done using an *Asus* laptop and a *GoPro Hero 4 Session*. All the gathered data and other media were stored on *Google drive*. The devices that were used for the tests are *Apple iPhone 11 Pro* and *Oneplus 5*. The model was implemented using the javascript framework *Vue.js*, and stored on *GitHub*.

4 Survey, Conduct, Design and Implement

This chapter shows how the different iterations for this study were conducted. How the survey was made is explained, and how the decisions for criteria that should be used in the decision-model were made. It then talks about how the usability tests were performed and analyzed. Finally, it goes through the evaluation and implementation of the decision-model.

4.1 Conducting the survey

The questions that were included in the mobile application discovery survey can be seen in table 4.1.

How old are you?
What type of smartphone do you primarily use?
You just heard about a new smartphone application (app). Where do you turn when you want to find information about the app?
You just heard about a new smartphone application (app). Where do you turn when you want to download the app?
When you use the internet, how much time do you spend on apps versus web sites?
How often do you think about your security and integrity when installing/downloading mobile applications?

Table 4.1: Questions asked on the Internet usage survey.

The survey was implemented in Google Forms, and then distributed on the social media platform Facebook and posted on digital bulletin boards of The Royal Institute of Technology KTH. Posting on Facebook was done through the researchers own personal Facebook accounts, to try to reach people outside of the IT-field and in a wider age range. Posting to KTH bulletin boards was done to maximize the amount of people responding, as other students would be likely to answer a survey to help with the research. The survey was offered in both English and Swedish. After the data was collected, the answers were analyzed. This was done by comparison of how participants using Android answered versus how participants using iOS answered.

4.2 Designing the decision model

Since the model was to be used in the case of choosing a cross-platform implementation, only the options of developing either one PWA, two native applications or two applications using React Native were investigated. The two native platforms were Android, using the browser Google Chrome, and iOS, using the browser Safari. Different browsers may have different support for PWA functions, but Google Chrome is the most commonly used browser for Android, and Safari is the most commonly used for iOS, so they were chosen for this study [5].

4.2.1 Deciding on criteria

The criteria in the model were based on the literature study and brain-storming sessions with Slagkryssaren AB. During these sessions, the business perspective, the developer perspective, the customer perspective and the end-user perspective were taken into account and discussed. The first set of criteria that were decided upon can be seen in table 4.2.

Life span	Reachability
Micro transactions	Budget
Hardware support	Time-frame
Security	Customer skill set
Graphics	UX

Table 4.2: The first criteria selection.

A few modifications were made to this list as the implementation progressed.

The criteria of *Life-span* and *Customer skill set* were later changed to be multiple-choice questions. These were hard to rank since the possible alternatives were few, for example, the *Life-span* as it was defined in this context could only have two possible outcomes: A permanent app or a temporary app. They also only affected the budget-aspect of the decision.

The criterion of *Graphics* was partially combined with *Hardware support* to form *Performance*, and the remaining *Hardware support* parts formed the new

criterion *Functions*.

Micro-transactions was removed as a criterion, as the effect for each choice on the income for a micro-transaction business model was hard to define. The fee for using micro-transactions through App Store and Play Store was at the time of writing 30%, which gives PWAs a benefit as a PWA bypasses the app stores [38]. Payment made in a PWA could be done on a multitude of payment services, but if a workaround like this was implemented, distributing an application through an app store becomes problematic. Also, for the end-user, there is an added convenience to purchasing applications and doing micro-transactions through an app store which were hard to quantify.

The criterion of *Maintenance* was added to include the benefit of native applications from a maintenance and longevity standpoint, according to developers at Slagkryssaren AB. *Time-frame* was renamed *Time to develop* to better define the criterion.

The scores were decided based on discussion points from meetings with developers, literature studies, the statistically significant results from the UEQ, the first iteration of the qualitative data and the results from the mobile application discovery survey. The scores were set in relation to each other, meaning the best alternative on each criterion was given the score 10 and the other alternatives were given scores relative to this. The score for PWA and native application was set first, with the React Native being added afterwards.

Detailed reasoning for the scores were as follows:

Budget: The Budget score was based on accounts from developers at Slagkryssaren AB. According to them it was approximately 30% more expensive to develop a cross-platform solution than to develop one native application. This means developing two native applications was approximately 35% more expensive than developing one PWA, and therefore the scores were set to 10 for PWA and 7 for native. React Native was given the score 9. This was based on the fact that up to 95% of the code written is cross-platform [12], and the rest is platform specific. It was therefore slightly more expensive than a PWA, but better than a native application.

UX: The UX score for the PWA was based on the usability test results. From the statistically significant results out of the UEQ, the PWA scored approximately 6:10 compared to the native application. The score for the React Native was based on information from developers at Slagkryssaren AB.

Functions: The Functions score was intentionally set very low for PWA and high for the native application. If the decision-maker had many demands for functions not available on PWA, this would affect the weight, and if the decision-maker ranked the criterion Functions high it would give the native development method the desired advantage in that situation. Likewise, had the decision-maker few functions needed and/or no high ranking for the criterion, the weight would be very low, and so the scores could differ much without affecting the total outcome greatly. The most important feature for a PWA compared to a website on mobile was the offline mode and the home screen installation. Many functions were at the time of writing unavailable, so the PWA was given a score of 2 and native was given a 10. React Native was given a score of 9, based on the fact that almost all native functions can be accessed through a React Native application when using wrappers.

Reachability: The Reachability score was based on a number of factors. PWAs have an advantage in reachability, as they can be downloaded through most mobile browsers [39], and reached as websites without modifications through the others. PWAs could also be uploaded to Google Play store and, with some added difficulty, to Apples App store [40]. A PWA would also be available on desktop computers, laptops and tablets without the need to launch a separate website. The advantage of native applications in comparison is that there are at the time of writing less hoops to jump through to get your application on the app stores, and therefore they got a 5. React Native was estimated to have a similar reachability as native, but with the added benefit of having a Javascript-base allowing for easy translation into a website.

Maintenance: According to developers at Slagkryssaren AB, native applications demand less maintenance as the environment is less volatile. In their experience, Javascript libraries and frameworks get deprecated or become

outdated more frequently than code written in languages for native applications. This would mean a significantly larger workload to maintain the PWA, giving it a lower score than the native application. React Native has the dependencies of Javascript, combined with the added need for maintenance from OS updates, new device releases and two different platforms to handle debugging on. With this in mind, PWA was given 6, native 10 and React Native 4.

Time to develop: The Time to develop score was based on information from developers at Slagkryssaren AB. Developing one PWA is much less time consuming than developing two native applications, and it can be launched through a browser without delay of app stores. React Native is slightly more time consuming than a PWA to launch, as it demands some extra code to be added to the Javascript-base to function on both platforms and for the applications to be approved on the app stores. Due to this, PWA was given the score 10, native application the score 5 and React Native application the score 7.

Performance: The Performance score was based on literature studies and discussions with developers at Slagkryssaren AB. PWAs do not have the same direct link to the hardware of the unit as a native application does since they are run on a browser [25, p. 12], meaning the abstraction level is higher for a PWA than for a native application. This results in worse performance. This effect would, however, be preventable, as a PWA in some cases can be made to have almost native-like performance [25, p. 43]. The React Native is a bit more complex to analyze. A well-implemented React Native application can have performance very similar to a native application [41]. However, according to Slagkryssaren AB a React Native application will on average be slightly worse than a native application, but not by as much as a PWA. With this in mind, the native application was given the score 10, the PWA the score 7 and the React Native application 8.5.

4.2.2 Scoring and implementation

The first version of the model was implemented with Google Spreadsheet. The input from the decision-maker was collected using Google Form.

In this version of the model the criteria and corresponding scores found in table 4.3 were set.

Criterion	PWA	Native	React
Budget	10	7	9
UX	6	10	8
Functions	2	10	9
Reachability	10	5	8
Maintenance	6	10	4
Time to develop	10	5	7
Performance	7	10	8,5

Table 4.3: Scores set for the first version of the model.

The model used three types of input from the user. Firstly, the user was presented with the functions which are currently unavailable for a PWA on iOS and/or Android. They were instructed to select the functions they needed for the application (see list in figure 4.4). This affected the weight for the criteria *Functions*, the more functions selected, the higher the weight. The effect was calculated by summation of the selected functions' corresponding number of unavailable platforms, dividing the sum by the total 20 and added by one. This produced a number between 1 and 2, depending on how many functions were selected.

Second, a number of multiple-choice questions were asked.

- *Competences*: Which competencies are available in the customers business?
  - Web: *Multiply Budget with 1.5*
  - Android and/or iOS: *Multiply Budget with 0.5*
  - All of the above: *No effect*
  - None of the above: *No effect*
- *Website*: Does a website with the same functions already exist?
  - Yes: *Multiply Budget with 1.3*

Function	no. platforms unavailable
App store presence	1
Integrated camera function	1
Bluetooth	1
NFC	2
Notifications	1
Surrounding sensors [1]	2
Vibration	1
Contacts	2
Scheduling	2
Background sync	1
High accuracy GPS	2
Geofencing	2
VR/AR	1
Advanced screen manipulation [2]	1

Table 4.4: Functions and how many platforms out of iOS and Android they remain unavailable on when choosing PWA.
[1]*Surrounding sensors includes ambient light and proximity sensor*
[2]*Advanced screen manipulation includes orientation, full screen and wake lock*

  - No: *No effect*

- *Short-lived*: Is the app short-lived?
  - Yes: *Multiply Maintenance with 0.5*
  - No: *No effect*

These questions affected the weights of certain criteria. *Competences* and *Website* affected the criterion Budget. If the customer business had competences within web-programming, the weight was increased, as this gave further advantage to PWAs and React Native applications. Same with the website, a website can be converted to a PWA easily, and to a React Native application with slightly more work, which would save time and money. If the customer held competences in native application programming, the weight was decreased, as this made native

applications less expensive in the long run compared to PWAs and React Native applications.

The question of *Short-lived* affected the criterion of Maintenance. If the application was short-lived, the disadvantage of the higher demand for maintenance on a PWA and React Native application was lessened. The risk of a utilized function or library being deprecated would, however, still be present.

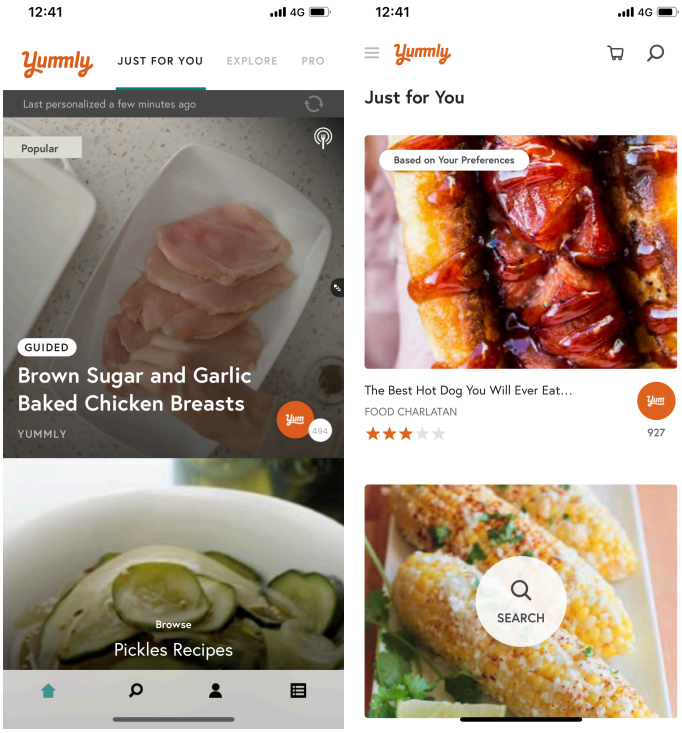
After that, the user ranked the criteria on importance. The criterion with the highest ranking was given the most weight, one to 100. The last step of the process multiplied in the numbers given by the multiple-choice questions and the function-selection on the relevant criteria, and then normalized so that the weights summed to 1, according to WSM standards.

4.3 Conducting comparative usability tests

The test subjects were recruited through a form shared on the social media platform Facebook and posted on bulletin boards at the campus of KTH. The test subjects were compensated with a goodie-bag and free coffee. The first tests were conducted at the office of Slagkryssaren AB. The tests were later moved to the campus of The Royal Institute of Technology KTH.

The product chosen for the usability test was the recipe application Yummly. Yummly has both a native Android application and a native iOS application, and a PWA. Yummly was chosen for the test since it had a good amount of interaction elements, and it would be possible to put the test participants in realistic scenarios. The product itself was relatively intuitive and relatable for most people. Since the application was not developed for the Swedish market, it would be unlikely that the test participants had used the application before. Yummly had several interaction elements, some of which were included in the tests including searching, filtering and navigation of menus. An example of the differences in the views can be seen in figures 4.1a and 4.1b.

Before each test, the subjects answered some background questions, shown in table 4.5. These were used both to gather more information for the statistical



(a) Native iOS (b) PWA iOS

Figure 4.1: *Screenshots of the different versions of the Yummly application.*

review of the test results and to make the subjects more open and ready to talk. The subject also filled in a consent form. The consent form made it possible to record and process the recorded material for analysis of the qualitative data. The consent form can be read in appendix A.

What is your main occupation?
How much time do you spend on your smartphone on an average day?
How much of that time do you spend on applications versus websites?
What kind of mobile applications do you mostly use? Work, chat, social media, finances, shopping etc.?

Table 4.5: Background questions for the usability tests.

For the user tests, the Nielsen Norman Group's article on task scenarios for usability tests was used [42]. The scenarios were designed to be relatively realistic, actionable and with an appropriate level of vagueness for the test subjects to have to think when they performed the tasks. The scenarios in table 4.6 were included in the first tests.

1. Find the recipe “Avocado vegan fudge brownie”
2. Use the app to find recipes with chicken which take less than 15 minutes to cook
3. You are making dinner. Use the app to make a shopping list for: a dish with fish, a dish with bacon, and a dessert
4. One can add their dietary preferences/allergies in the app. Try adding that you eat gluten-free and vegan, and find recipes for chilli. Browse around and find a recipe you find interesting.
5. It’s possible to save recipes through a so-called “Yum’s” list. Create a “Yum’s” list called Pastries and add a few recipes to the list
6. Add “milk” and “eggs” to the shopping list
7. You’re in the store and have now picked up eggs, update your shopping list.

Table 4.6: The first list of tasks.

The tests were recorded with a camera, showing the face and recording the voice of the test subject. This allowed for richer, more nuanced qualitative data to be gathered from the usability tests, and also allowed for tests to be revisited and analyzed a posteriori. This was useful since only two people were conducting the tests, and neither has extensive experience in usability testing. After three tests, another camera was added, recording the subjects’ from behind. This allowed the researchers to see what the subject was reacting to on the device.

Pilot tests were performed first to assess how well the thought out scenarios worked. The pilot tests were timed to evaluate how time-consuming the different parts of the test were. These pilot tests averaged at 30-40 minutes in total. Some scenarios were removed and edited to minimize time consumption, without affecting the quality of the results. The order of the tasks was also changed to better mimic how the application would be used in real life. Scenario number 6 was put first for this reason. Scenario number 1 was removed because scenario 2 and 4 filled the same function.

The SUS, UEQ and consent forms were initially printed on paper to ease the process of filling them in. Since the computer was used for facial recording, the subject was forced to lean over to reach the touch-pad. After the pilot tests, this

was changed to have the subjects fill in the SUS and UEQ on the computer instead. This saved both time and the amount of paper used for the tests. The data from the SUS and UEQ was stored digitally to be used for statistics. The tests were saved separately depending on the device used and the type of application tested. This would ease the statistical analysis between the platforms. The aspect of Novelty from the UEQ was left out of this test, as it was not within the scope of the research question.

4.4 Analyzing the usability tests

The results from the SUS and the UEQ were compiled and then evaluated on the statistical significance of the data. The qualitative data recorded in text from the usability test was processed and compiled to form the first qualitative base for the test results.

4.5 Evaluating the Decision-making model

The first version of the model was tested by iteratively running scenarios through the model and evaluating the results. Some of these scenarios were constructed for the purpose of testing if the model gave reasonable results in ”ideal” native application or PWA scenarios, and some scenarios were taken from real-life applications implemented either as native applications, React Native applications or PWAs.

After a few rounds of testing, the score for Functions was changed from 1 to 2 on PWA. The motivation for this was that even in the cases where few functions were chosen as necessary for the application, the native got a huge advantage in the score. Changing the score for PWA evened out the playing field a bit, without affecting the cases where native applications had a clear advantage. It also represents the available functionality better, a PWA is similar to a website on mobile devices but has some distinguishing functions such as offline mode and home-screen installation.

When the model seemed to be acting reasonably, it was implemented as a Javascript application. After that a consultant from Slagkryssaren AB was invited

to test the model out. The consultant tried a few scenarios, and checked if the application gave the same answer as he would have chosen. He noted that the model seemed to give an accurate recommendation compared to how he provided input. The consultant's overall feedback of the model was positive, noting that it would be a useful tool for the earlier stages of the development process. It gave a good ground for discussion and could give the developing team another opportunity to consider what could be necessary for the project. At this point, it was decided that the model was finished.

5 Result

This chapter first goes through the results of the mobile application discovery survey. After that, the results from the usability tests are presented and lastly the decision-model that has been designed and implemented is shown.

5.1 Mobile application discovery survey

A total of 94 people, 28 iPhone users and 66 Android users, participated in the internet survey. The age of the participants ranged from 20 to 73 years, with the median of the participants' age being 27 years. The average age was 32.7 years. In both cases of device used, only one person answered that they use internet searches to download applications. How Android users found information about applications can be seen in figure 5.1. The same data for the iPhone users can be seen in figure 5.2.

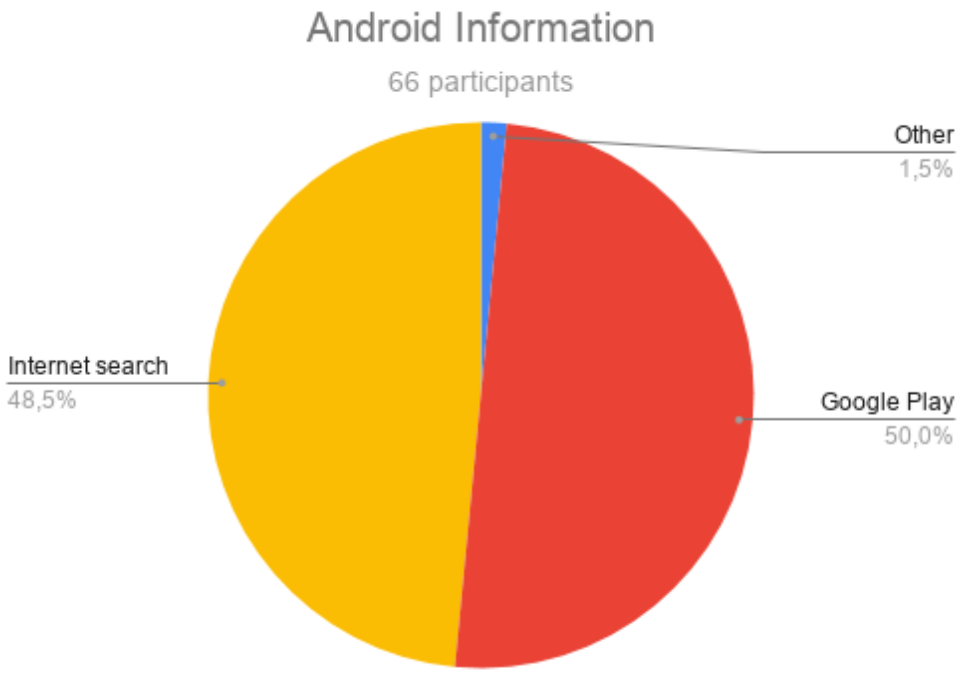


Figure 5.1: Android users' answers to "You just heard about a new smartphone application (app). Where do you turn when you want to find information about the app?".

The users on average answered that they spent more time on applications than browsing the web, but no statistical conclusion could be drawn from the questions

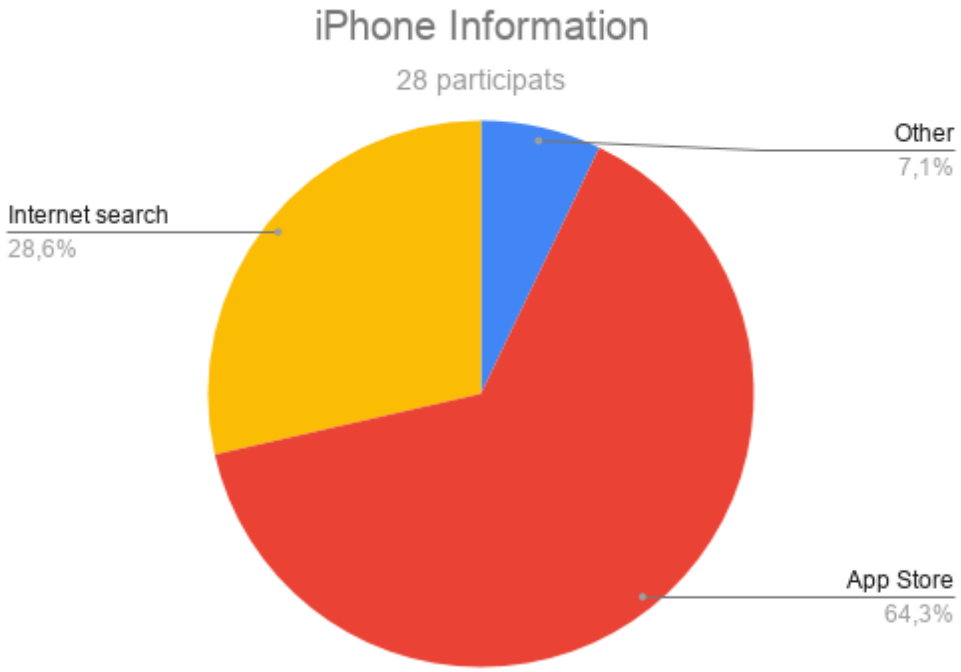


Figure 5.2: iPhone users' answers to "You just heard about a new smartphone application (app). Where do you turn when you want to find information about the app?".

on web browsing vs. application usage or security, as can be seen in appendix C.

5.2 Usability test

The findings from the usability tests regarding user experienced is analyzed through the UEQ, SUS and the qualitative data gathered from the tests. There were a total of 18 tests, 5 iPhone users and 13 Android users.

5.2.1 UEQ

Two of the five categories of the UEQ got statistically significant results on a confidence level of 95%: Attractiveness and Efficiency. Dependability and Stimulation had statistically significant results on a confidence level of 85%. Due to this low value, in the first calculation, only the categories of Attractiveness and Efficiency were chosen to form the basis for the criteria UX in the model.

Perspicuity had no significant difference on any reasonable confidence level.

To quantify the results from the UEQ, the score for PWA was divided by the score for Native in the different categories. This provided a relative ratio between PWA and Native which was used to set the score for the criteria. The relative ratio score for the categories Attractiveness and Efficiency can be seen in table 5.1 The average for the two statistically significant categories was 0.62.

	Relative Ratio Score
Attractiveness	0.68
Efficiency	0.56
Total Average:	0.62

Table 5.1: Relative ratio score from the 95% statistically significant UEQ results

The overall result from the UEQ, with the categories of low statistical significance included, can be seen in table 5.2. The total average was 0.77. The whole result without the relative ratio score can be found as a graph in figure 5.3.

	Relative Ratio Score
Attractiveness	0.68
Perspicuity	1.14
Efficiency	0.56
Dependability	0.76
Stimulation	0.73
Total Average:	0.77

Table 5.2: Relative ratio score from all UEQ results

5.2.2 SUS

The result from the SUS, seen in figure 5.4 indicated that Android users preferred the Native application, whereas iPhone users preferred the PWA. However, these results were not statistically significant and were therefore not included in the calculations for the model.

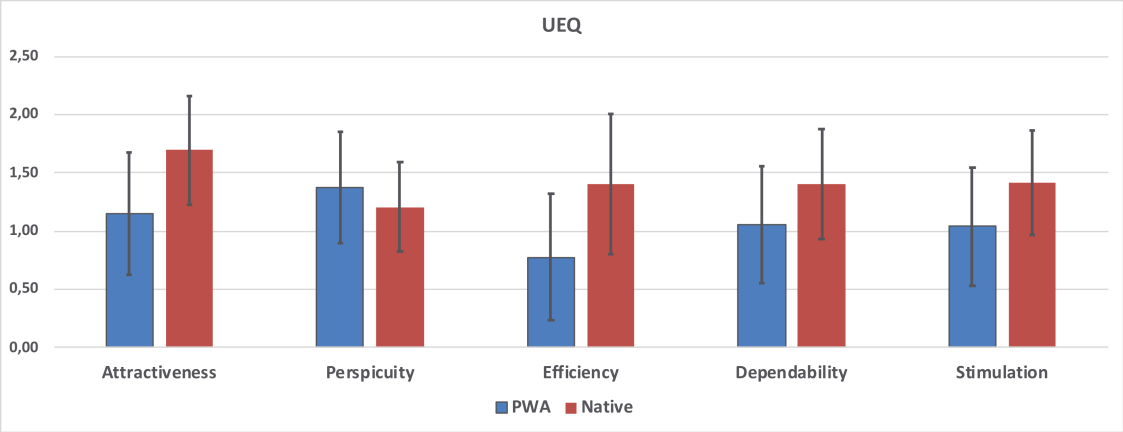


Figure 5.3: UEQ results from Usability Test

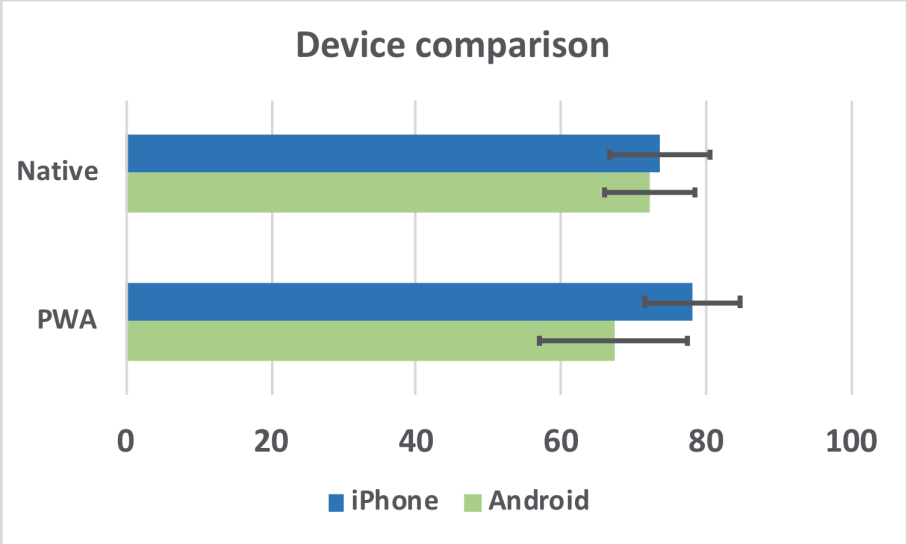


Figure 5.4: A SUS-Score comparison between iPhone and Android for PWA and Native.

5.2.3 Qualitative data

The result from the qualitative questions after the Usability tests, and the responses and reactions recorded in written form at the test sessions, produced the first iteration of qualitative data.

10 subjects said that they experienced the PWA to be slow or slower in comparison with the native application. 2 subjects expressed that the native application was slow.

3 subjects said that they found the PWA to be easier to navigate. No such opinion was expressed about the native application.

3 subjects said the PWA felt like a web page or felt ”web-like”.

5 subjects had visible trouble with the search bar in the PWA due to poor responsiveness. 1 subject had similar problems with the native application. 2 subjects vocally expressed problems with the hitbox-size of the native applications buttons.

6 subjects noted the native applications was more attractive, 2 subjects preferred the aesthetic of the PWA.

The iPhone users more often preferred the layout of the PWA, 3 out of 5 subjects using iPhones preferred the PWA on the aspects of user interface.

3 subjects preferred native even though they liked the user interface of PWA more, due to the poor responsiveness of the PWA.

5.3 Decision Model

The decision model was implemented with *Vue.js*. The dependencies added for this implementation were *Chart.js* for creating bar-charts, a slider component for the ranking, *Vuetify* for the styling, *Vue-router* and *Vuex* for functionality like saving state and swapping between different views. The code for the implementation can be found on the following GitHub repository:

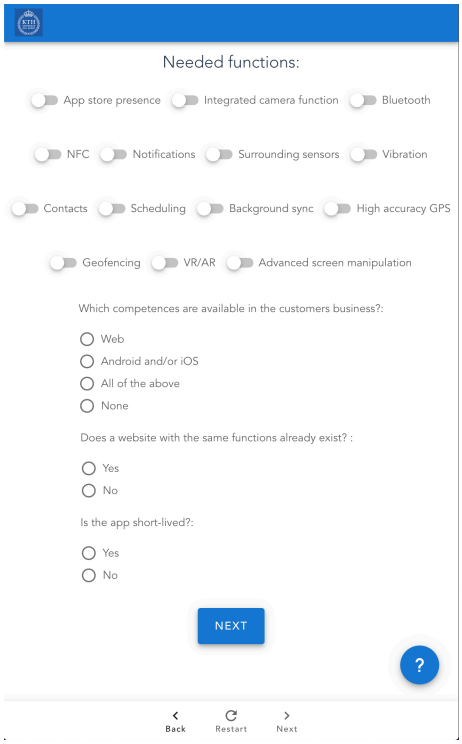
https://github.com/sebastian-porling/mobile_application_mcdm.

The application is up and running on the following website:

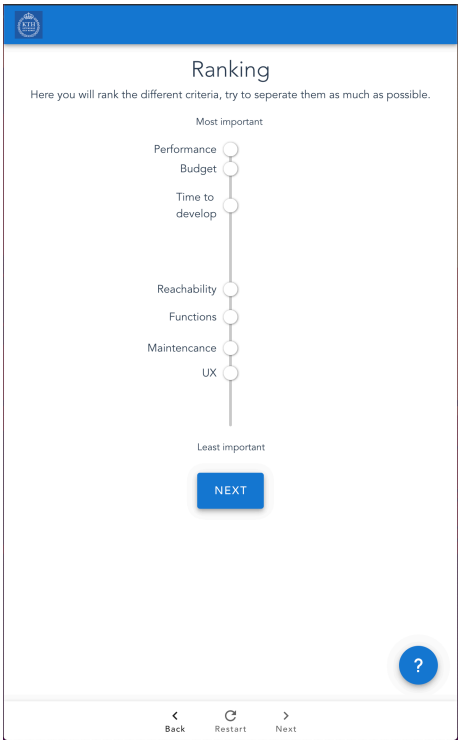
https://mcdm-application-development.web.app

The implementation consisted of three parts. The first view seen in figure 5.5a was a form, which gathered the functions that were needed for the application and some follow up questions. The second view seen in figure 5.5b was used for ranking the different criteria. The slider made the ranking more perspicuous. The ranking could have been implemented in many ways, like filling in a number for each criterion, but this version made it easier to see the relationship between the criteria. The final view seen in figure 5.5c was the result of the calculation. It gave the recommended alternative for the decision-maker, showed how much better the alternative is compared to the other alternatives and gave a bar chart

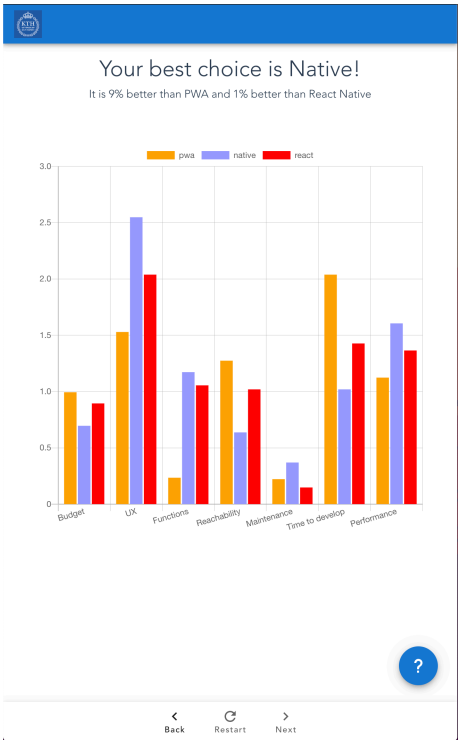
that showed how much the score was affected by the different criteria. A list of functions which will not be available while using PWA was presented.



(a) The form



(b) Ranking module



(c) Result view with graph

Figure 5.5: *Screenshots of the different views for the implementation.*

6 Discussions

This chapter presents the research questions and methodically goes through how well they were answered. After this, a discussion on how each part of the project could have been done differently to improve the accuracy of the results takes part.

6.1 Discussion

As can be seen in the project triangle, the scope of this project was the only non-constant variable for affecting the quality. To get useful results from the usability tests, it was planned to perform 20 tests at least. Only 18 tests were conducted, as it was harder to find test subjects than we thought, and also to have time to properly analyze the material from the tests and implement the model. However, after 18 tests were conducted, the time planned for the usability tests was up. It was apparent that some parts of the data would not be improved with just a few more tests, but would require a much larger study.

6.1.1 Mobile application discovery survey

As presented in Chapter 3.1 the task of finding mobile applications was answered using a survey. The mobile application discovery survey was filled in by 94 persons. The ages ranged between 20 and 73, a more realistic outcome of the survey could probably be reached if it had included those below 20 as well. The survey was distributed in social media and on a KTH bulletin board only, limiting the reach of the participants. A survey spread through other types of media as well would increase the scope and improve the results. It would especially have been useful to reach those in the age range younger than 20, as mobile usage habits could differ for those people compared to those older than 20. This means that even though the validity of the test result is good, the reliability might not be optimal. A possible way to achieve this would have been to post the survey on other social media platforms more commonly used by the age-group, or send out the survey on platforms for schools.

After the survey was conducted, the following questions were found to be poorly

worded:

- *"You just heard about a new smartphone application (app). Where do you turn when you want to find information about the app?"*
- *"You just heard about a new smartphone application (app). Where do you turn when you want to download the app?"*

A better formulation would have been as follows:

- *"You are searching for new smartphone applications (apps). Where do you turn when you want to find information about the apps?"*
- *"You are searching for new smartphone applications (apps). Where do you turn when you want to download the apps?"*.

This might have improved the accuracy of the survey as the desired statistics was on how people find *new* applications, and the way the questions were asked it could be interpreted as the question being about known applications.

For the data collected to be useful in any statistical context it would have required that some more information on the participants was collected. It would be useful to know how knowledgeable the subject is on new technology, how much the subject uses their smartphone, level of education and occupation. Knowing this, one could compensate for a skewered participant population compared to the general population.

6.1.2 Usability test

As presented in Chapter 3.1 the task of UX was investigated through usability testing. The results of the usability tests suggest that there was a difference in how a user perceives a PWA and a native application, and that overall the native application was preferred. Two out of the five included aspects of the usability test UEQ had results which were statistically significant on a 95% confidence interval, even on the relatively small amount of participants used in this survey, indicating that the difference is substantial enough to affect the users experience of the application.

The usability tests were conducted during office hours, and the participants were

recruited through university campus channels. This resulted in the participants almost exclusively being students at a technological institute. The tests were conducted in Slagkryssaren ABs office in the beginning but were later moved to the campus of the university. This was done to ease the process of recruitment, as holding the tests in the office resulted in added travelling time for the student participants. It also opened up for the possibility of having drop-in times for participating, instead of having to book a time in advance.

The age range was also limited to young adults between 20 and 30 years old. This means the participants were likely more experienced and comfortable using new mobile software than the average person. An improved result could have been achieved if there was a wider demographics of participants, which would demand that the tests be performed outside of office hours and in several different locations. Recruitment could have been performed on several different platforms, targeting harder-to-reach demographics. The participants were compensated for their time in a goodie-bag consisting of candy. Monetary compensation could have attracted more participants, but there were no funds for such compensations in this study.

There were some technical limitations to the usability test. The iOS device used was a relatively new iPhone 11 Pro, whereas the Android device was an older OnePlus 5. There were no newer Android devices available to test on, so the iPhone used for the tests had noticeably better performance than the Android device. Due to the low performance of the Android device, it was possible the PWA was given a disadvantage compared to how it would perform in real life on most users' personal devices. This did, however, give a good insight into how a PWA might perform on a lower-end or older device. An ideal scenario would have been to use one newer and one older device from each platform, with comparable performance.

There were also some technical difficulties to take into consideration. During one of the tests, the WiFi-connection was down, forcing the moderator to intervene which might have affected the test subject's impression of the PWA negatively. Since the application needed to access information from the internet, a stable, monitored internet connection would have prevented these problems. For a few

of the tests, some parts were not captured on camera from behind, meaning the device's screen was not shown. This was due to the difficulty in recording and charging the camera simultaneously, resulting in the camera running out of battery.

Since the test conductors were each responsible for one of the two platforms, one person got to practice moderating and the other recording more. This affected the quality of the written material and the instruction-giving for the test subjects. This could have been avoided if the same person had been moderator or recorder for every test, but since the conductors were not experienced with the other device this was not optimal for this test. Screen recording and eye-tracking technology could have eliminated the need for a recorder, which would improve the accuracy of the material. With recorded material mainly in the form of notes made by a human, there is always the risk of a bias towards one of the products. With the limited time frame and finances of this study, it was not possible to gather and process the amount of material that would be produced by implementing these technical solutions.

Due to time limitations, the recorded material could not be processed in detail. Instead, the written material compiled at the tests was used as a base for the qualitative data, and when there were uncertainties in the written material the video recordings were reviewed for clarity.

The test conductors had limited experience in the field of Human-computer interaction. There is a risk that useful material from the usability tests was overlooked or misinterpreted due to lack of experience.

6.1.3 Decision model

As presented in Chapter 3.1 the tasks deriving from the research question of implementing a model were investigated through conducting literature and case studies. That information would then be used for the process of designing, implementing and evaluating the model. The decision model was implemented with WSM. It is possible a version of AHP would have produced different results, as it would force a more polarizing opinion from the decision-maker. For an AHP implementation, the criteria would have to be re-evaluated, as the complexity of

the AHP increases with every new criterion added.

Some studies on the performance aspect of PWAs exist, but it was hard to decide on a clear number of how good or bad it was compared to native applications. The performance of an application, regardless of the development method, is highly dependent on how the application is coded. A well-coded PWA could perform better than a native application with poor implementation. For the model used in this study, what would be considered a well-implemented native application was taken into consideration when setting scores and compared to a well-implemented PWA. How well this model translates to real life was uncertain, as trying to compare how well applications are implemented was a very complex task.

The Budget criterion was highly simplified. Many factors not included in the model could affect the economical aspects of choosing a development method, including the revenue model of the business and the effect of improved search engine optimization.

Overall it was difficult to quantify the scores for the model. This resulted in the scores being set partly on a trial-and-error basis. A possible solution to this would have been to use a Knowledge-Based System (KBS) to build the model. A KBS uses artificial intelligence to predict which solution is most advantageous, comparing an input to the input from several other solved scenarios. This could give a solution which is correct with a certain probability, which would increase when more scenarios are run through the model and evaluated. The KBS bypasses the need for quantification, eliminating the risk of a faulty score affecting the outcome of the model [43].

The decision model for React Native specifically was limited by the complexity of the available functions. React Native can behave similarly to a native application if implemented with the proper wrappers, but different functions have different limitations and opportunities. We decided to instead just raise the score on Functions to a level agreed on by the developers at Slagkryssaren AB to be an appropriate approximation of a React Native applications functionality.

7 Conclusions

In this chapter, the final conclusions on the projects findings are presented. Finally, in the section Future Work, research which could be conducted in the future on the topic of this project is discussed.

7.1 Conclusion

Do users notice a difference between progressive web applications and native apps, and does the difference affect the user experience?

According to our findings yes, to a certain extent. The biggest difference between PWAs and native applications was at the time of writing the limitations on functions available, but even when functionality is not taken into consideration we would still argue that PWA was shown to be worse from an end-users perspective. We say this because the difference between the native application and PWA was significantly noticeable when it came to response time, at least on the older device tested. It was possible we could have gotten a different result if we had tested with a different application. When trying to find an application to perform the tests on, another product called Pinterest was considered. However, the PWA for Pinterest did not work on the iOS device available so it was not selected for this test. The PWA on Pinterest was more similar to the native application and would otherwise have been a good choice for the tests. Our conclusion from the usability tests is that when the device running the application has outdated hardware the PWA would not perform as well as a native application. However, it was also possible that the PWA for Yummly was simply not as well-implemented as its native counterpart.

Something to take into account is that from the survey we noticed that half or more of users will probably try to find the application through the phones app store first. If this was a trend among the population in general this could mean that users would need to do more searching in order to find the PWA than a native application. There also seems to be a trend that iPhone users primarily find information about applications through app store. This aligns with our hypothesis, however, this was not a big enough survey to convey a definite

conclusion on the matter.

Can a model accurately decide whether a PWA, a React Native application or a native application is most favourable?

Yes and no. After presenting the model to Slagkryssaren AB the consultants expressed the opinion that the model seemed to give accurate results based on the input given. The recommendation from the model in itself was not a definite truth, but the recommendation in combination with process of filling in the form gives a good ground for discussion. This could be helpful to challenge a fixed idea of what application to implement, either within the development team or when discussing with a customer.

The subject of choosing a development tool is subjective in its nature, it was therefore not unexpected that the model would be used more as a tool for discussion and prioritization than a definite decision-making tool.

7.2 Future Work

A usability test including PWAs, native applications and hybrid applications such as React Native could further improve the reliability of a decision model, as the UX criteria for React Native in this model was based entirely on opinion from developers at Slagkryssaren AB.

The score for the criteria Maintenance was also based on opinions from developers. An improvement would, therefore, include basing this number on some kind of statistics, analyzing how much more maintenance a PWA needs compared to other applications. For the Budget criterion to be more precise, research on how the revenue is affected by where an application is downloaded and which payment methods the end-users can utilize is necessary.

A larger study on application exploration could also be enlightening in the choice of application development technique. It would be useful to compare how different demographics find new applications, to find out the true impact of an application being present or absent on the most popular app stores.

References

[1] *Progressive Web Apps*. URL: https://developers.google.com/web/progressive-web-apps (visited on 02/14/2020).

[2] *The four pillars of sustainability*. 2017. URL: https://www.futurelearn.com/courses/sustainable-business/1/steps/157438 (visited on 02/19/2020).

[3] *The four pillars of sustainability*. URL: https://sustainabledevelopment.un.org/?menu=1300 (visited on 02/19/2020).

[4] *Mobile Operating System Market Share Sweden*. 2020. URL: https://gs.statcounter.com/os-market-share/mobile/sweden (visited on 02/18/2020).

[5] *Mobile Operating System Market Share Global*. 2020. URL: https://gs.statcounter.com/os-market-share/mobile (visited on 02/18/2020).

[6] *Understanding the 3 Types of Apps: Native, Mobile, and Hybrid*. URL: https://www.charterglobal.com/mobile-app-development/ (visited on 02/18/2020).

[7] Saccomani, Pietro. *Native Apps, Web Apps or Hybrid Apps? What's the Difference?* 2019. URL: https://www.mobiloud.com/blog/native-web-or-hybrid-apps/ (visited on 02/18/2020).

[8] Filip Rakowsky Kaja Grzybowska, Piotr Karwatka and Kwiecien, Aleksandra. *The PWA Book*. 2019. URL: https://divante.com/pwabook/chapter/02-The-history-of-PWAs.html (visited on 02/18/2020).

[9] Firtman, Maximiliano. *Progressive Web Apps on iOS are here*. 2018. URL: https://medium.com/@firt/progressive-web-apps-on-ios-are-here-d00430dee3a7 (visited on 02/18/2020).

[10] *iOS 11.3 is available today*. 2018. URL: https://www.apple.com/newsroom/2018/03/ios-11-3-is-available-today/ (visited on 02/18/2020).

[11] Gimeno, Alberto. *Node.js multithreading: What are Worker threads, and why do they matter?* 2019. URL: https://blog.logrocket.com/node-js-multithreading-what-are-worker-threads-and-why-do-they-matter-48ab102f8b10/ (visited on 02/18/2020).

[12] Ganguly, Dev Shankar. *3 Big Benefits Of React Native Development*. 2018. URL: https://hackernoon.com/top-3-reasons-of-choosing-react-native-for-cross-platform-app-development-702b1cffff63 (visited on 02/14/2020).

[13] Six, Janet M. and Macefield, Ritch. *How to Determine the Right Number of Participants for Usability Studies*. 2016. URL: https://www.uxmatters.com/mt/archives/2016/01/how-to-determine-the-right-number-of-participants-for-usability-studies.php (visited on 02/14/2020).

[14] Martin, Bella. *Universal methods of design 100 ways to research complex problems, develop innovative ideas, and design effective solutions*. eng. Beverly, Mass.: Rockport Publishers, 2012. ISBN: 1-61058-199-7.

[15] Ross, Jim. *Conducting Qualitative, Comparative Usability Testing*. 2017. URL: https://www.uxmatters.com/mt/archives/2017/03/conducting-qualitative-comparative-usability-testing.php (visited on 02/14/2020).

[16] Brooke, Sophia. *How to Conduct a Usability Test in Six Steps from Start to Finish*. 2018. URL: https://uxplanet.org/how-to-conduct-a-usability-test-in-six-steps-from-start-to-finish-4082e8d57858 (visited on 02/19/2020).

[17] Shingyouchi, Yukari. *Diagram of Simple Usability Testing*. 2019. URL: https://voynetch.com/638 (visited on 02/14/2020).

[18] *System Usability Scale (SUS)*. URL: https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html (visited on 02/14/2020).

[19] *User Experience Questionnaire*. URL: https://www.ueq-online.org/ (visited on 02/14/2020).

[20] Martin Schrepp Andreas Hinderks, Jörg Thomaschewski. “Applying the User Experience Questionnaire (UEQ) in Different Evaluation Scenarios”. In: (2014). DOI: DOI : 10 . 1007 / 978 - 3 - 319 - 07668 - 3 _ 37. URL: https : / / www . semanticscholar . org / paper / Applying - the - User - Experience - Questionnaire - (UEQ) - in - Schrepp - Hinderks / 355d945d3818531a1425dff948c9a5dd4ad5d54e/figure/2.

[21] *Multi-Criteria Decision Analysis*. 2011. URL: https : / / projects . ncsu . edu/nrli/decision-making/MCDA.php (visited on 02/18/2020).

[22] Triantaphyllou, Evangelos. *Multi-criteria Decision Making Methods A Comparative Study*. eng. 1st ed. 2000.. Applied Optimization, 44. 2000. ISBN: 1-4757-3157-4.

[23] Vargas, Ricardo Viana. *Using the analytic hierarchy process (ahp) to select and prioritize projects in a portfolio*. 2010. URL: https : / / www . pmi . org / learning / library / analytic - hierarchy - process - prioritize - projects-6608 (visited on 02/18/2020).

[24] Randleff, Veronica. “Native app vs Web app: Multi-criteria decision-making for optimised mobile solution”. MA thesis. KTH, School of Electrical Engineering and Computer Science (EECS), 2018.

[25] Yberg, Viktor. “Native-like Performance and User Experience with Progressive Web Apps”. MA thesis. KTH, School of Electrical Engineering and Computer Science (EECS), 2018, p. 50.

[26] Jobe, William. “Native Apps Vs. Mobile Web Apps”. In: *International Journal of Interactive Mobile Technologies (iJIM)* 7 (Oct. 2013), pp. 27–32. DOI: 10.3991/ijim.v7i4.3226.

[27] Fransson, Rebecca and Driaguine, Alexandre. *Comparing Progressive Web Applications with Native Android Applications : An evaluation of performance when it comes to response time*. 2017.

[28] Biørn-
Hansen, Andreas, Majchrzak, Tim A., and Grønli, Tor-Morten. “Progressive Web Apps: The Possible Web-native Unifier for Mobile Development”. In: Jan. 2017, pp. 344–351. DOI: 10.5220/0006353703440351.

[29] Koziokas, Panagiotis, Tselikas, Nikolaos, and Tselikis, George. “Usability Testing of Mobile Applications: Web vs. Hybrid Apps”. In: Sept. 2017, pp. 1–2. ISBN: 978-1-4503-5355-7. DOI: 10.1145/3139367.3139410.

[30] Andrade Cardieri, Giulia de and Zaina, Luciana Martinez. “Analyzing User Experience in Mobile Web, Native and Progressive Web Applications: A User and HCI Specialist Perspectives”. In: *Proceedings of the 17th Brazilian Symposium on Human Factors in Computing Systems*. IHC 2018. Belém, Brazil: Association for Computing Machinery, 2018. ISBN: 9781450366014. DOI: 10.1145/3274192.3274201. URL: https://doi.org/10.1145/3274192.3274201.

[31] Melvin, George. *The Four-step Method of Polya*. 2015. URL: https://math.berkeley.edu/~gmelvin/math110f15/polya.pdf (visited on 02/18/2020).

[32] *What is Iron Triangle of Projects?* URL: https://www.visual-paradigm.com/project-management/what-is-iron-triangle-of-projects/ (visited on 02/18/2020).

[33] Connection, Agile. *MoSCoW for UX*. 2016. URL: https://www.interaction-design.org/literature/article/making-your-ux-life-easier-with-the-moscow (visited on 02/18/2020).

[34] Rauschenberger, Maria. “Efficient Measurement of the User Experience of Interactive Products. How to use the User Experience Questionnaire (UEQ).” In: *International Journal of Interactive Multimedia and Artificial Intelligence* 2.1 (2013), pp. 39–45. DOI: 10.9781/ijimai.2013.215. URL: http://www.ijimai.org/journal/sites/default/files/files/2013/03/ijimai20132_15_pdf_35685.pdf.

[35] Kortum, Philip and Peres, S. Camille. “The Relationship Between System Effectiveness and Subjective Usability Scores Using the System Usability Scale”. In: *International Journal of Human–Computer Interaction* 30.7 (2014), pp. 575–584. DOI: 10.1080/10447318.2014.904177. eprint: https://doi.org/10.1080/10447318.2014.904177. URL: https://doi.org/10.1080/10447318.2014.904177.

[36] Nielsen, Jakob. *Thinking Aloud: The #1 Usability Tool*. 2012. URL: https://www.nngroup.com/articles/thinking-aloud-the-1-usability-tool (visited on 02/14/2020).

[37] Schultz, David. *10 usability tips tricks for testing mobile applications*. 2006. URL: https://interactions.acm.org/archive/view/november-december-2006/10-usability-tips-tricks-for-testing-mobile-applications1 (visited on 02/18/2020).

[38] Saxena, Prateek. *How Much Money Can You Earn Through an App? Read Here*. 2018. URL: https://appinventiv.com/blog/how-much-money-can-you-earn-through-your-mobile-app/ (visited on 02/28/2020).

[39] *Progressive Web App Browser Support*. URL: https://vaadin.com/pwa/learn/browser-support (visited on 02/28/2020).

[40] Himango, JudahGabriel. *I built a Progressive Web App and published it in 3 app stores. Here's what I learned*. URL: https://www.freecodecamp.org/news/i-built-a-pwa-and-published-it-in-3-app-stores-heres-what-i-learned-7cb3f56daf9b/ (visited on 02/28/2020).

[41] Johansson, Erik and Söderberg, Jesper. *Evaluating performance of a React Native feature set*. 2018.

[42] McCloskey, Marieke. *Turn User Goals into Task Scenarios for Usability Testing*. 2014. URL: https://www.nngroup.com/articles/task-scenarios-usability-testing/ (visited on 02/28/2020).

[43] Brent, E.E. "Knowledge-based systems: A qualitative formalism". In: (1986). DOI: DOI:10.1007/BF00988401.

Appendices

Appendix - Contents

A Consent forms

Usability test evaluation

CONSENT FORM for participants – 18 YEARS AND OLDER.

You are invited to participate in a usability test evaluation being conducted by the student evaluators listed on the bottom of the page. In order for us to be allowed to use any data you wish to provide, we must have your consent.

In the simplest terms, we would like to ask you questions about these products. We would like to observe and evaluate your actions and thoughts about these products to be analysed at KTH Royal Institute of Technology for a bachelor thesis. This will involve:

- Recording your actions by notes
- Recording your expressed thoughts by notes
- Recording your voice and facial expressions

The researchers and teaching staff will be able to look at our records (in written word) for the purposes of this project. You may stop participating at any time and ask for your information to be deleted at any time.

You may consent to participate in this usability test evaluation by *checking off all the boxes that reflect your wishes and signing and dating the form below.*

❑ I agree that any response I make may be recorded in writing to be analysed, discussed and seen by the evaluators of this usability test evaluation and the teaching staff of KTH The Royal Institute of Technology.

❑ I agree that my answers and reactions, in written form, may be included in the report for this bachelor thesis.

NAME [please print]	
Signature	
Date	

Student evaluator
Contact information: Celine Mileikowsky
Email: celinemi@kth.se
Number: +46 733 18 61 44

Student evaluator
Contact information: Sebastian Porling
Email: porling@kth.se
Number: +46 761 89 35 07

Examiner
Contact information: Anders Sjögren
Email: as@kth.se

Användarvänlighetstestsutvärdering

Samtyckesformulär för deltagare – 18 ÅR OCH ÄLDRE.

Du är inbjuden för att delta i en utvärdering av ett användarvänlighetstest vilket kommer bli utfärdat av studentutvärderarna som är listade på slutet av denna sida. För att vi ska få använda den data som du kommer förse oss med måste vi ha ditt samtycke.

Vi kommer fråga dig frågor kring produkterna. Vi kommer också observera och utvärdera dina handlingar och tankar kring dessa produkter, samt analysera dessa på KTH Kungliga Tekniska Högskolan för ett kandidatexamensarbete. Detta innefattar:

- Transkribering av dina handlingar och tankar
- Inspelning av din röst och ansiktsuttryck

Utforskare och personal kommer få möjlighet att kolla på våra dokument (i skrift) för ändamålet att publicera i rapporten för detta kandidatexamensarbete. Ditt deltagande i studien är helt frivilligt. Du kan när som helst avbryta ditt deltagande utan närmare motivering, samt få din data raderad.

Du kan samtycka till att delta genom att kryssa i rutorna som motsvarar dina önskemål och fylla i formuläret nedan:

❑ Jag samtycker till att mina svar kan komma att nedtecknas för analys och diskussion av utvärderare för detta användarvänlighetstest samt personal på Kungliga Tekniska Högskolan.

❑ Jag samtycker till att mina svar och reaktioner, i skriftlig form, kan komma att publiceras i rapporten för detta kandidatexamensarbete.

NAMN	
Underskrift	
Datum	

Studentutvärderare
Kontaktinformation: Celine Mileikowsky
Email: celinemi@kth.se
Nummer: +46 733 18 61 44

Studentutvärderare
Kontaktinformation: Sebastian Porling
Email: porling@kth.se
Nummer: +46 761 89 35 07

Examinator
Kontakt information: Anders Sjögren
Email: as@kth.se

B SUS Data

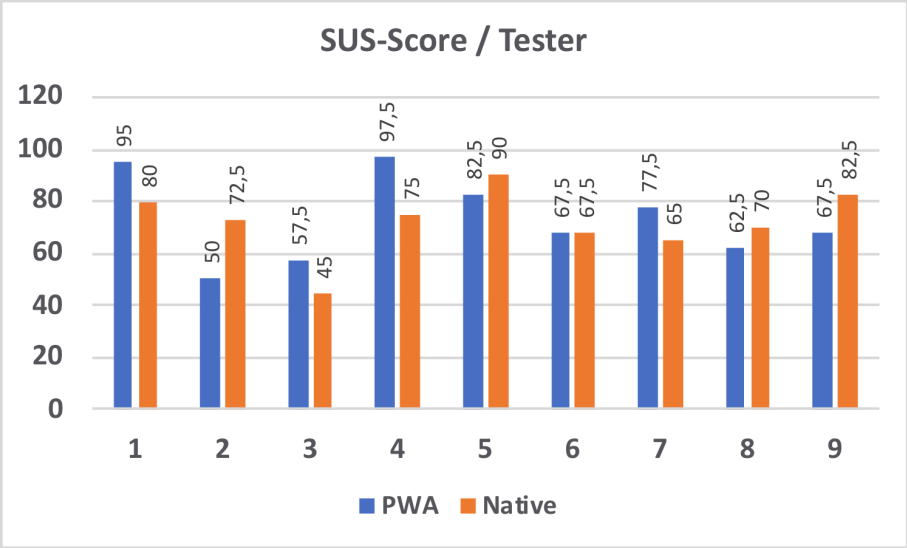


Figure B.1: SUS results from Usability Test

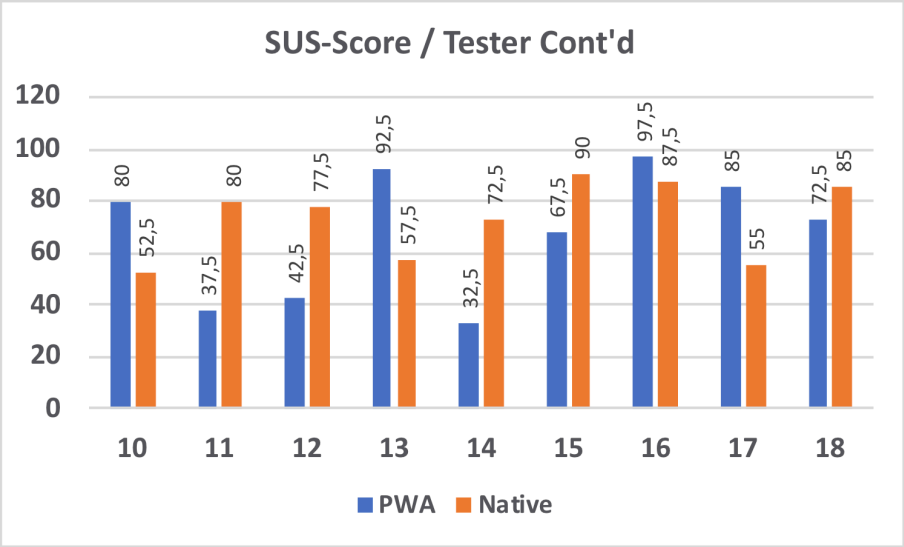


Figure B.2: SUS results from Usability Test

C Survey Data

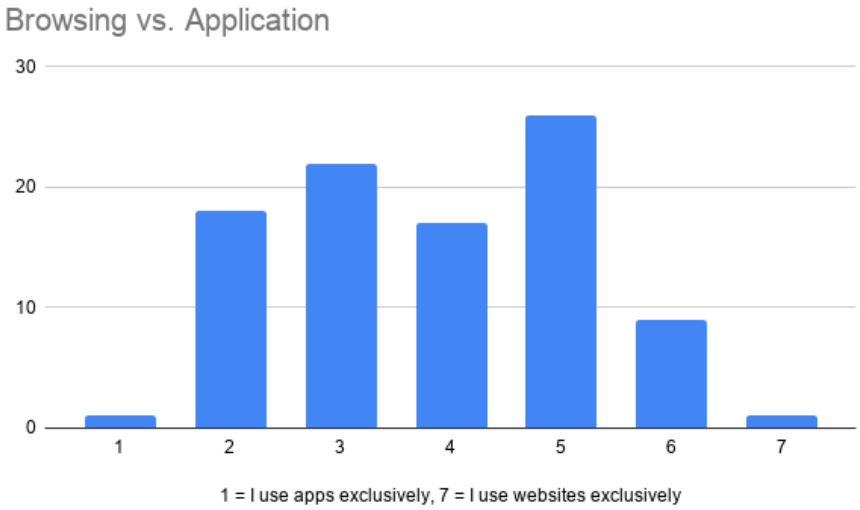


Figure C.1: Survey results for the browsing vs applications question.

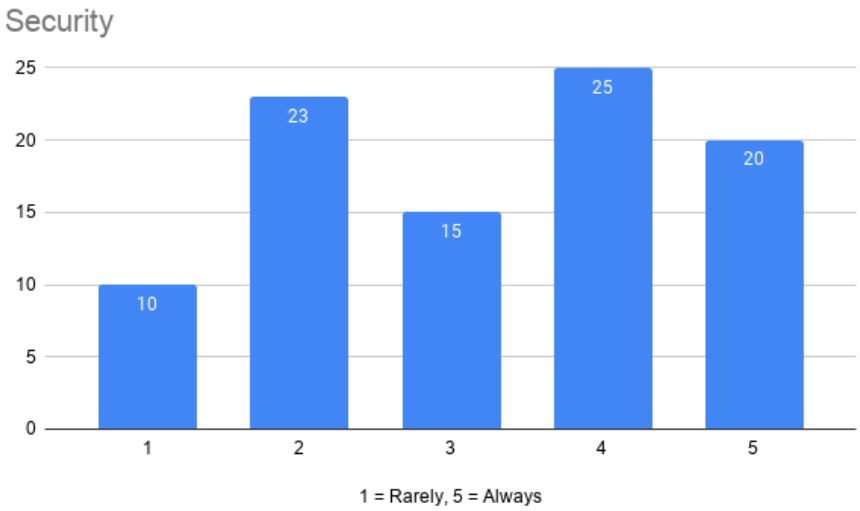


Figure C.2: Survey results for the question regarding security.

c = 0,05	App vs. Browser usage	Security
Mean	-0,148	0,234
Std	1,359	1,323
Confidence	0,274	0,267

Table C.1: The result from survey in regards of application usage and security awareness.