Mittuniversitetet
MID SWEDEN UNIVERSITY

An analysis and comparison of the Native mobile application versus the Progressive web application

William Berggren

Mid Sweden University
Bachelors of Science in Computer Engineering
Credits: 15 hp
Semester/year: Spring 2023
Supervisor: Muhammad Waseem Akhtar
Examiner: Patrik Österberg
Course code/registration number: DT099G

At Mid Sweden University, it is possible to publish the thesis in full text in DiVA (see appendix for publishing conditions). The publication is open access, which means that the work will be freely available to read and download online. This increases the dissemination and visibility of the degree project.

Open access is becoming the norm for disseminating scientific information online. Mid Sweden University recommends both researchers and students to publish their work open access.

I/we allow publishing in full text (free available online, open access):

☒ Yes, I/we agree to the terms of publication.

☐ No, I/we do not accept that my independent work is published in the public interface in DiVA (only archiving in DiVA).

Sundsvall, Sweden, 2023-09-20 .....
Location and date

DT099G .....
Programme/Course

William Berggren .....
Name (all authors names)

20010720 .....
Year of birth (all authors year of birth)

Abstract

This thesis explores the comparison of progressive web applications (PWA) with native mobile applications. This thesis sheds light on an issue that is relevant today and will be even more so in the near future due to the constant advancement of technology and the growing dependence of human life on the use of mobile applications. The use of the mobile has increased significantly over the past couple of decades with the mobile application being integrated into the human lifestyle. This pattern of growth indicates that the development of the applications needs to be adjusted for an effective and more secure approach. Although the native application is currently the leading app, the PWA puts on a more extensive challenge than before due to its effectiveness and simplicity. By developing one PWA and one native mobile application, this study aims to identify the key differences. The focus of the analysis was to demonstrate the three critical aspects, speed, security and time of development with the results showing that the PWA performs better in terms of speed when being optimized, but the native mobile application has a larger access base for feature implementation. These results enrich the ongoing discussion with further understanding for the two most popular strategies of crafting a mobile application, with information about how the future of the app scene possibly evolve.

Keywords: Mobile App Development, UX, Swift, React.

Sammanfattning

Det här arbetet utforskar jämförelsen mellan progressiva webbapplikationer (PWA) och native mobila applikationer. Arbetet belyser ett problem som är aktuellt idag och kommer att vara ännu mer så i närmaste framtid på grund av den ständiga teknikutvecklingen och människans beroende av mobila applikationer. Användningen av mobilen har ökat kraftigt under de senaste årtiondena med mobila applikationer som integreras i människans livsstil. Denna tillväxt indikerar att utvecklingen av applikationer behöver justeras för en effektivare och säkrare metod. Även om native applikationen för närvarande leder, utmanar PWA mer än tidigare på grund av dess effektivitet och enkelhet. Genom att utveckla en PWA och en native mobilapplikation syftar denna studie på att identifiera de huvudsakliga skillnaderna. Analysens fokus var att visa de tre kritiska aspekterna, hastighet, säkerhet och utvecklingstid, med resultaten som tyder på att PWA presterar bättre i form av hastighet när den optimeras, men native mobila applikationen har en större tillgänglighet för implementering av funktioner. Dessa resultat berikar den pågående diskussionen med ytterligare förståelse för de två mest populära strategierna för att skapa en mobilapplikation och ger information om hur appscenen möjligen kan utvecklas i framtiden.

Table of Contents

Terminology

Acronyms/Abbreviations

PWA	Progressive Web Application
DOM	Document object model
EB	Exa Byte
IOS	Iphone Operating System
UI	User Interface
DSR	Design Science Research
CSS	Cascading Style Sheets
URL	Uniform Resource Locator
HTML	hyper text markup language
FPS	Frames per seconds
CPU	Central processing unit
App	Application
AWS	Amazon Web Services
ROI	Return On Investment

1. Introduction

With the constant advancements of the digital and technological elements of society, applications operated with mobile devices has been observed as one of the most essential components for revolutionizing human interactions, daily tasks and structures of social frameworks.

This thesis that has been proposed by the company Knightec [2], has been done as an in-depth study concerning the comparison of the two approaches of developing applications suited for mobile devices, the progressive web application (PWA) and the native application.

1.1. Background and motivation

With the current technological advancements and emphasis on efficiency, the everyday use of mobile applications has increased in all kinds of settings and use-cases. Humans have never been this well connected with the internet and a large amount of the daily tasks for individuals have been transformed to be done in manners with the help of digital technology, primarily based on the mobile device accompanied with its applications.

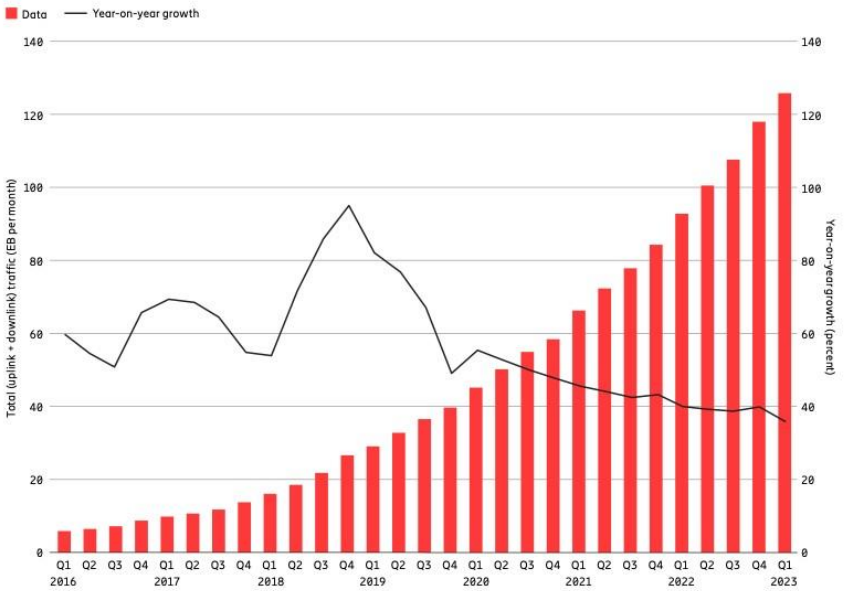

Figure 1: Yearly data traffic growth (measured in EB), the Figure is taken from [7], where the line of year-on-year growth stands for the yearly data traffic percentage growth, and the orange staples visualize the monthly data traffic growth.

It can see in figure 1 provided by the company Ericsson [7], where the amount of monthly global mobile network data traffic Exabyte (EB) has been measured, it is represented that the monthly data use with mobile devices has had a severe increase over the last few years, and from 2020 to 2022, the amount of EB has doubled.

The increment in the global data traffic proves that the usage of the mobile device has reached an all-time high, and there are no signs of decrease in the growth according to figure 1. This growth in data usage proves the major responsibility that tech companies and service providers have for managing and protecting the data being transmitted in society. With a greater use of the mobile device comes a larger extension of functions based on mobile applications. From communication, finance, health and fitness to education, the mobile device has become a priority and a dependency in a multitude of aspects for human beings.

As the popularity and the widespread use of the mobile application grows rapidly on a daily basis, additionally the industry for the mobile application grows with new developing methods, a wider economy standpoint to rely on and developing of smarter, efficient, and more contemporary with the advanced level of technology and needs of the applications that will be utilized by the current society. Currently, two of the biggest ways of developing an application based for the use with mobile devices are the PWA and the native application.

Comparing these two ways of constructing an application, the Native mobile application is currently the most prominent, however, in the past few years the PWA has been seen to be expanding in the use by companies and overall in the mobile application industry, due to its efficiency and simplicity [1].

The native mobile application is designed for exclusive deployment on mobile devices only with the need of downloading it from a proprietary application store. The PWA is designed to run directly from a website and be accessed via a mobile device without the need to download it.

The motivation for using this subject lies in the opportunity of getting a large set of knowledge and a better insight of the status of which type of

application currently is the leading concept and the way to go for developing in the mobile application industry, in terms of strength and weaknesses.

1.2. Overall aim and problem statement

The overall aim is to develop and evaluate two types of applications suited for mobile devices, one Progressive web application (PWA) and one Native application with the exact same design and feature functions. Furthermore, by analyzing and comparing the challenges and benefits in the processes of developing the apps, the project aims to identify and explicate differences based on the following chosen factors, speed, security and time of development.

To investigate how a PWA would stand against a Native application in terms of the designated factors and in general, multiple problem statements have been used for structuring the theoretical and technological standpoints to delve into. The following statements that have been provided by the company Knightec [2] will shape the groundwork of the investigation, and the intention of the project is to thoroughly analyze and present the most optimal answer possible. The goal of this project is to find the answers of the following technical questions.

- How much time does it take to develop a PWA versus a native mobile application?
- Can a PWA be faster and more secure than a native mobile application?
- What can a Native Web App do that a PWA cannot?
- Will we still have the need for an app store in ten years?

1.3. Scientific goals

In pursuit of answering the problem statements outlined in chapter 1.2 best at hand, the following objectives has been made.

- Create one PWA and one native application based on the Iphone Operating System (IOS) with the same implemented mobile feature access. The features should be access to the device's camera, access to see if the device is “in-call” and access to the device's contact book.

- Implement one list consisting of 10 000 elements, with functions to add, delete and sort elements.

- To implement a UI for the applications, for the user to be able to use the functions of each application.

- Do performance tests for the applications long list of 10 000 elements.

1.4. Scope

The scope of this thesis is to develop two applications, one PWA and one native application based on the iPhone Operating System (IOS). These applications are built to appear identical with the same features and design. The primary focus of the project is to analyze how the Progressive Web Application differs against the native application in terms of possibilities, limitations and developing process by implementing a set of native access features. This work will explore the technical viewpoint of the status of the Progressive web app, in terms of how it behaves with feature access in relationship with the native app based on performance and user needs.

The thesis will not focus on developing the two applications with the intention of fitting business needs or creating full-scale products aimed at market deployment by any company. The applications developed will serve the needs of being a template with implemented functions, determining if it is possible to implement the same feature access for the apps with the same overall performance.

1.5. Outline

Following are the outlines of the thesis.

Chapter 2 describes the research and technological theory of the thesis work.

Chapter 3 is about the methodological standpoint of the work, what methods have been used and how they have fulfilled their purpose.

Chapter 4 covers the Pre-study of the work, with chosen approaches and requirements of the thesis.

Chapter 5 is about the implementation of the PWA and the native mobile application.

Chapter 6 covers the results of the thesis work.

Chapter 7 describes the discussion and analysis of the work.

Chapter 8 covers the conclusion of the work, with answered problem statements and possible future work.

2. Theory

This chapter covers the background information needed and serves as a foundation of knowledge for the reader to fully understand how the technologies, concepts and methods used in the report function. The more in-depth theoretical view of the components will cover the rest of the project chapters relevant for the analysis of PWA: s and Native applications.

2.1. XCode

XCode [3] is a development environment suited for the development of various Operating System software's created and maintained by Apple [4]. The program was released in 2003 and supports numerous programming languages such as C, C++, Java, Python, Swift, AppleScript, Ruby and ResEdit.

2.1.1. Instruments

Instruments [25] is a built-in performance testing and analyzing tool, built mainly for the use of IOS applications. The tool includes a set of templates for specifying measurement used for tests, including CPU, memory usage, network analysis and much more.

2.2. Visual studio code

Visual Studio Code [5] is a source editor tool that functions on the operating systems Windows, Linux and MacOS. The program has a built-in system that supports the programming languages C++, C#, Python, Java, PHP, .NET, Go, JavaScript, TypeScript and the JavaScript runtime environment “node.js”.

2.3. Angular

Angular [6] is a framework and a platform for developing single page applications by using the programming languages HTML (HyperText Markup Language) and TypeScript. The platform operates with fundamental and additional functions which can be imported as a selection of TypeScript libraries and utilized by the creator of the project.

An application created with the Angular framework is designed to be composed of various elements, mainly arranged “NgModules”(An

Angular component). The NgModules constructs the structure of the application program by gathering all the correlated code and placing them into working sets. A program using Angular can have numerous “feature modules”, but it's certain that it contains modules that use “bootstrap”.

The different components of the Angular framework operate in different manners to specify functions called “views”, that based on the logic and data of the specific application, outlines an assortment of UI elements that the framework can choose from and adjust grounded in the needs of the program. The components in Angular also uses a feature called “services” to make the source code more effective and sustainable by inserting these services as a dependency into the component and organize the application.

2.4. Programming library

Libraries in terms of computer programming [21] is a set of prewritten code with components, that can be utilized when programming different tasks to make the work process more effective.

2.5. Delegates

Delegates [26] is a type of design pattern used in coding for allowing two objects to send information between each other. The delegator object communicates with and updates the delegate object about events that occur during the operation of the program.

2.6. React

React (also known as ReactJS) [7] is a JavaScript library that is built by utilizing User interface components. React was created and had its first edition “0.3.0” in May 2013. The library broadens and organizes the JavaScript language with a collection of features for crafting a web application with respective UI functions.

2.6.1. Theme Provider

A Theme Provider [22] based on the library React, is an UI theme specification used for being able to control the design structure of the

chosen file, with specifications about the fonts, sizing, colors and spacing and so on.

2.7. Vue.js

Vue.js [13] is an open-source framework for JavaScript aimed at application user interfaces. The framework uses a model-view-view model for front end architectures. The framework is specified to be declarative rendering and composing of components and by enchaining the Hypertext Markup Language (HTML) with attributes known as directives. A wider set of functionalities is being offered to the applications by using the built-in or specified directives.

2.8. Matplotlib

Matplotlib [24] is a library based on the programming language python that creates graphs and plots from given script code. It uses the pyplot module that facilitates the plotting process with formatting axes, font properties and line style control. It includes the plotting of scatter graphs, graphs, histograms, bar charts and so on. An example of a table generated using Matplotlib can be seen in figure 2.

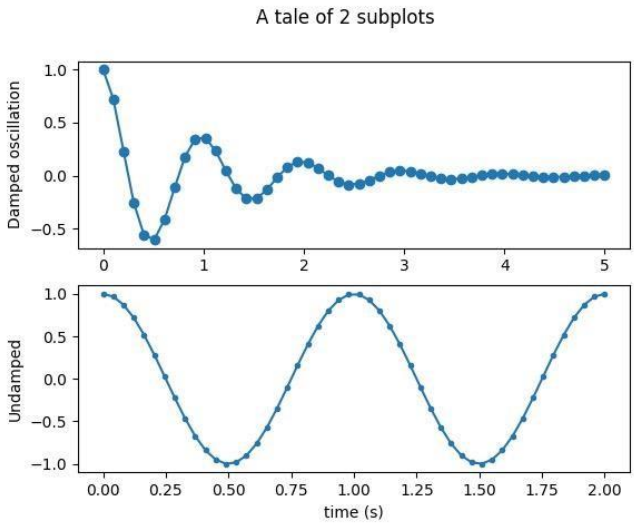

Figure 2: Example of an matplotlib table

2.9. Native mobile application

Native mobile applications are apps developed to specifically only function on the mobile device, with the need of downloading it from the built in application store based on the mobile operating system. According to the web services (AWS) [8], the application is set to work on mobile devices with user experience based on the interface with potential to be designed with features only available for mobile devices.

The overall development of native mobile applications requires knowledge of multiple programming languages since the development process of applications based on different platforms varies. For instance, an application created for IOS will be built with a programming language suited for IOS, for example Swift. And an application built for Android will be using a language made for the Android operating system, for example Java [8].

2.10. Progressive Web Application

Progressive Web Applications [14] are built with programming languages including HTML, CSS, JavaScript. It's constructed utilizing a single codebase to be able to be executed on all devices, with adjustments of resolutions to fit the device capabilities.

The PWA operates on a web site through supporting browsers with the possibilities of being installed to the home landscape of the device.

PWA:s acts autonomously, meaning that they could be launched when associated files are being opened and they have the possibilities to be operated when the device is offline and to access hardware capabilities.

2.11. Web worker

Web workers [31] utilized with JavaScript runs code scripts in background threads instead running it directly in the main thread of the program. By separating the code into smaller threads set in the background, larger resource demanding tasks could be executed without directly affecting the main thread. The web worker communicates with the main thread of the program using a system of messages that duplicates the web worker's tasks instead of sharing them.

2.12. SMART goals

The SMART goals framework [11] consists of a set of principles to follow when creating goals for research or study. The aim of the framework is to help students pursue their ambitions and set up a structured way of analyzing problems, paving a way for setting up goals that are logical for the research with the best possible approach of achievements.

2.13. Design Science Research

The Design Science Research or also known as DSR is an methology [10] used in computer science and information systems with the intention of being used in technological development processes, evaluations, and theoretical considerations.

The DSR methodology includes a series of different working steps that includes identification of problem, objectives of a solution, construction, demonstration, evaluation and communication. These steps of the DSR together form a strategy for developing and later analyzing the working product, design, or instantiation of model [10]. The general guidelines of the DSR methodology can be seen in figure 4.

Guideline	Description
Guideline 1: Design as an Artifact	Design-science research must produce a viable artifact in the form of a construct, a model, a method, or an instantiation.
Guideline 2: Problem Relevance	The objective of design-science research is to develop technology-based solutions to important and relevant business problems.
Guideline 3: Design Evaluation	The utility, quality, and efficacy of a design artifact must be rigorously demonstrated via well-executed evaluation methods.
Guideline 4: Research Contributions	Effective design-science research must provide clear and verifiable contributions in the areas of the design artifact, design foundations, and/or design methodologies.
Guideline 5: Research Rigor	Design-science research relies upon the application of rigorous methods in both the construction and evaluation of the design artifact.
Guideline 6: Design as a Search Process	The search for an effective artifact requires utilizing available means to reach desired ends while satisfying laws in the problem environment.
Guideline 7: Communication of Research	Design-science research must be presented effectively both to technology-oriented as well as management-oriented audiences.

Figure 4: Guideline of DSR, taken from [10].

2.14. Amazon Web Services

Amazon web services (AWS) [27] is an online cloud service available all around the globe. The service offers various data features that serve different needs, for example management of databases, analytics, storing of data objects, mobile development, and Internet of Things. The cloud service operates on a flexible and on-demand service delivery, which makes the service adjustable with pricing and accessible features based on the requirements and needs of the customer.

2.14.1. Amazon S3

Amazon s3 [28] is an available feature in AWS with the intention of serving as a platform for storing object information. The objects will be stored in so called "buckets" containing different types of data, for example source code. Users can create unique buckets for their projects and upload their specific data as objects.

2.14.2. AWS Command Line Interface (CLI)

AWS CLI [29] is used as a tool based on open-source code that makes it possible for the user to engage with AWS features through commands. The commands are supported in different command-line interfaces, such as windows command prompt, Linux bash, PowerShell and Mac terminal. AWS CLI gives the user access to all the component resources regarding each AWS feature.

2.14.3. Amazon CloudFront

The CloudFront [30] service available in AWS delivers dynamic and static web information, such as JavaScript, HTML, CSS and images across the internet. The web information is being delivered through edge locations and when users want to interact with information in specific CloudFront distributions, CloudFront is directing the user to the edge location with the shortest delay for performance optimization.

2.15. Related work

Research has shown that the topic about how native mobile applications can be compared and analyzed to PWA: s is not overwhelming, but there can be found studies that's related to this thesis.

The study found named “Impact of Progressive Web Apps on Web App Development” [9] provides a reference point for this thesis and with the information based on this study, will enhance a better understanding about the subject, providing knowledge and clarification for a better-informed assertion.

2.15.1. Example of a related work

By analyzing the related study work [9], similarities and differences regarding this thesis can be observed. One major resemblance can be seen in that the study also analyzes the PWA with benefits, disadvantages and in context of when the application can be used as a better alternative than other apps. The study also compares the native mobile app with the PWA with factors as for example performance.

The main difference that can be seen in the study is the comparison of applications. The related study comparison consists of three applications, native mobile application, PWA and Standard Web Application. Another difference in the related work is that the study compares the different parameters of building the applications, rather than implementing a set of features as can be seen in this study.

3. Methodology

This chapter will cover the methods used to delve into the theme of the subject, PWA's vs native mobile applications. With the chosen methods being adapted and custom-made for the most optimal approach of the process, to transition from theoretical studies to having an actual result consisting of constructed and evaluated applications. By setting up a logical methodology based on the topic, the problem statements presented in chapter 1.2 and the scientific goals outlined in chapter 1.3 have the potential to be achieved and answered satisfactorily.

3.1. Theoretical studies

To better understand and be able to attack the given problems stated in the problem statement chapter 1.2, theoretical studies will be used in different ways to collect required information. The theory studies will be grounded in a set of vast research to gather as much knowledge about the core principles, methodologies, theories, development environments, programming languages and frameworks that could be a solution for attacking the problem statements and scientific goals. By using this strategy for the theoretical study, it ensures that the later chosen method specified to each process of the thesis is correctly serving the needs.

3.2. Design Science Research (DSR)

The method being used for structuring the work procedure will be the Design Science Research (DSR) methodology. With the help of a large scale of background research about the methodology, how it works and areas its suitable for in context of this thesis combined with brainstorming potential ideas surrounding it, the method will be adjusted according to the needs. Each step of the DSR method can be seen in figure 5.

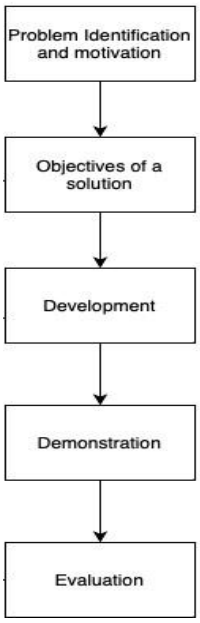

Figure 5: Overview of the Design Science Research methodology used in the thesis

The DSR has been chosen as a scientific method by theoretical study found [10] to get a broader and more precisely working process throughout the thesis with theoretical and technological perspectives with explicated answers, for the problem statements and scientific goals.

With the development of the PWA and the native application, they both will be crafted alongside with each problem statements by the principles of the DSR.

3.2.1. Problem identification and motivation

By following each step of the DSR methodology seen in Figure 5, the first phase will be to comprehend the given problem statements seen in chapter 1 and define what the possibilities of the research's intended outcome could be. This will involve analyzing different theoretical studies, as mentioned in chapter 3, web-based articles and whatever information that could lead to a better understanding of the given problem statements in terms of, the current state and overall performance processes of the PWA and native mobile application.

Alongside the information gained by the theoretical study research, knowledge, feedback, exchange of ideas and important eye openers

about the subject will be obtained from colleagues, the supervisor/supervisor at the school and company Knightec [2] where I am conducting and composing my thesis for.

3.2.2. Objectives of a Solution

The second phase of the work progress will be to objectify well-structured solutions to being able to answer the problem statements satisfactorily. The problems stated in the chapter 1 section 2, together with the knowledge based analyzes based on them, will create a baseline of information for setting up the scientific goals that can be seen in chapter 1 section 3.

The method used when optimizing the objectives of the solutions is the SMART goals principles. The SMART goals principles will be utilized during the process of the phase with specification, to set a specific and plain goal. The goals will be measurable, with the need to be measured with, for example, a performance test. Attainable, the goals will be reachable with a logical standpoint and not out of reach. The questions will be relevant for the subject PWA: s compared to Native mobile applications and being able to answer the problem statements. Finally, the goals will be adjusted according to the given time frame for the bachelor thesis, as preliminary 2.5 months, to be able to be completed.

3.2.3. Development

When the scientific goals have been outlined and prepared to be tackled, the stage of transforming the objectives into achieved goals will be grounded by the implementation of the two applications, the PWA and the native mobile application.

The development phase will not start with actual implementation of the applications. The phase will start with a detailed analysis of the technical viewpoints available for the later implementation, with research, prior knowledge, and personal experience. This method will be used to understand what it takes in terms of technological tools and frameworks to develop the applications based on the architecture and potentially eliminate any bad choice, that further down the line may slow down the working process.

To even more optimize the chances of choosing technologic tools best suited for the achieving the scientific goals more efficiently, a smaller test will be performed by creating simpler programs built with each of the relevant framework, libraries and programming languages. By using this method, the efficiency will be based on personal liking of maximizing resources and reducing time wasted.

When the necessary tools have been chosen for the implementation approach. The full approaches chosen and considerate for the thesis mentioned in chapter 4 section 3, the actual implementation process of the development phase will begin.

3.2.4. Demonstration

When the process of developing the two applications is near or fully completed, the project transitions into the phase of demonstration. During this phase the DSR methodology aims to validate that the scientific goals have been fulfilled and the research questions have the potential to be answered in the best way possible. The results will be demonstrated in terms of the performance factors and application features during the study with the supervisors to monitor the progress being made.

To have the ability to present the final results in the best manner for clarity and understanding of the study, a full showcase of the apps with key functions, measured performance will be made alongside comparison to the scientific goals in chapter 1.

3.2.5. Evaluation

The evaluation phase of the study can be seen as the most influential on the subject due to its aims to highlight the differences and get an answer of which application is the most superior based on the settings for this research. The evaluation method will investigate the given results and measurements that can be seen in chapter 6 provided by performance test and features being able to be accessed for the two apps. Specifically the total CPU load and the FPS will be measured for the PWA and the native mobile app, crafting the discussion seen in chapter 7 and conclusion of the study in chapter 8.

3.3. Project evaluation method

When measuring the success of the thesis work, it will be done in an general perspective, not only focusing on the results of the work but also the working process throughout the study. The success will be evaluated based on the clarity of the scientific goals, the effectiveness in terms of the project structure and the quality regarding the given results.

4. Approach

This chapter covers the chosen and considerations of approaches for the thesis. This part of the study will arguably be one of the most crucial there will be, since it paves the way of how the study will be degenerated in terms of results and final products. To be able to set up an analysis possible for comparing the PWA and the native mobile at an acceptable level and later performing it, the decision of approaches must be made thoughtfully.

4.1. Approaches for the Progressive Web Application (PWA)

This chapter focuses on approaches used in the development processes of the PWA.

4.1.1. Frameworks and library approaches

Choosing the right framework/library for the Progressive Web Application is a crucial step in the working process for the implementation that can be seen in chapter 5.1 and contribution to the built product. The framework/libraries could potentially make the process more effective and reduce errors with organization of programming code and reuse of components, for the development of the application.

The consideration of possible approaches is based on the needs and limitations in terms of achieving the scientific goals that can be seen in chapter 1.3.

By doing vast research into the status of the current web development, the three framework/libraries Vue.js, React and Angular can be furthermore researched in detail based on the theory information in chapter 2 to find the best possible way for approaching the implementation.

4.2. Approaches for the native mobile application

This chapter focuses on approaches for implementation of the native mobile application.

4.2.1. Programming language approach

To be able to get the best possible approach of building a native mobile application, the choice of programming language serves as a foundation for the implementation.

With research that was done alongside chapter 4.1.1 but with a perspective of the programming languages suitable for the native mobile application instead, the languages Swift and Objective-C can be used as considerations for possible approaches.

The goal for this chapter is to find the most optimal approach for being able to build a functioning native mobile application, from personal experience, knowledge and information about each language that can be found in chapter 2.

4.3. Comparison of approaches

To compare all the considered approaches that have been named in chapter 4 for the project, for analysis. The analysis is made of information created with the Pugh-matrix, that is used to compare each column based on a set of criterions. The Pugh matrix compares the approaches objectively with ratings on each criterion, for creating the PWA and the native application.

The Pugh matrix be based on a rating system with –1, 0 and +1. The –1 rating can be used when the criteria is below average fulfilled, the 0 rating can be placed when the criteria is fulfilled on an standard level and the +1 rating can be used when the criteria is fulfilled above the average.

4.3.1. Approaches for the Progressive Web Application development

Criteria	Vue.js	React	Angular
Learning Curve	+1	+1	-1
Performance	+1	+1	+1
Development speed	+1	+1	0
Documentation	0	+1	+1
Suitable	-1	+1	+1
Integration	+1	-1	0
Total	3	4	2

Figure 6: Overview for the Pugh matrix used for frameworks/libraries approaches.

The Pugh matrix is based on research from the sources [15], [16], [17] and [18] can be seen in figure 6.

4.3.2. Approaches for the native mobile application development

Criteria	Swift	Objective-C
Learning Curve	0	-1
Performance	+1	0
Development speed	+1	-1
Documentation	0	+1
Suitable	+1	0
Total	3	1

Figure 7: Overview of the Pugh matrix used for comparing programming language approaches.

The Pugh matrix seen in Figure 7 has been crafted by research from the sources [19] and [20].

4.4. Chosen approach

This chapter handles the chosen approaches for the PWA and the native mobile application.

4.4.1. Chosen approach for Progressive Web Application development

Based on the criteria's fulfilled of the Pugh matrix seen in chapter 4.3.1 and Figure 6, it can be seen vividly that the best choice for implementing the Progressive Web Application can be done with the library React.

4.4.2. Chosen approach for native mobile application

Based on the Pugh matrix seen in chapter 4.3.2, in figure 7. The best possible approach for the native mobile application development would be Swift.

5. Implementation

This chapter handles the implementation process of the project, with technologies and specifications used to create two functional applications that fulfill the scientific goals stated in chapter 1. The chapter will distinguish the methods used for development of the PWA and the native mobile application in separate subchapters, to get a precise and comprehensive understanding of each aspect. By clarifying the details of the process, the reader can gain understanding of the technical complexities and together with the theory chapter 2 be able to recreate these two applications.

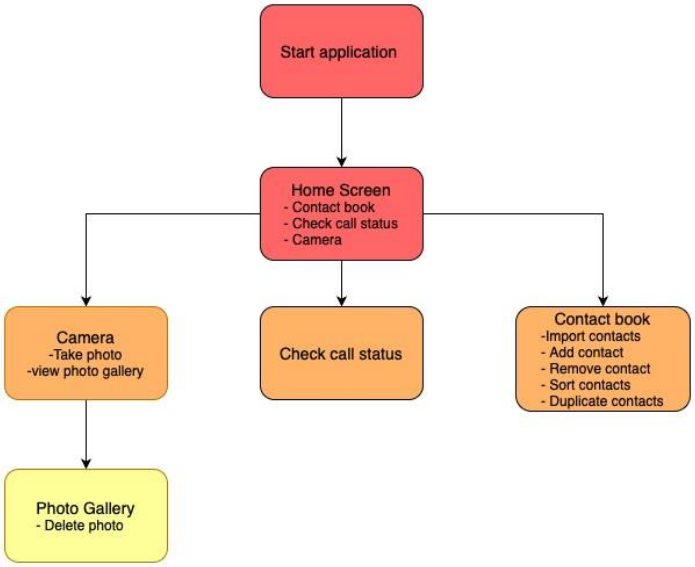

Figure 8: Flowchart of use cases for the applications

In Figure 8, an overview for the use cases for the applications can be seen. One Flowchart describing both the Progressive Web Application and the native mobile application is used, since they will strive to have the exact same functionalities and design. As seen in Figure 7, when the user enters the application, it will be redirected to a homepage containing three icons representing each application feature, the camera, contact book containing a long list of contacts and a function to check the call status of the user's device.

5.1. Implementation for the native mobile application

This section handles the implementation process of the native mobile application. The app develops with the best possible choice of programming language that can be seen in the Approach chapter 4, Swift and with Xcode as a development environment

5.1.1. Constructing the User Interface

The main foundation of the application is the component “NativeApp”. The component defines the overall behavior and life cycle of the application. The NativeApp uses its contentview component for setting up the UI with a navigationview for directing the program to be able to navigate to each of the features.

When the user launches the application, it is introduced to a vertical grid of specific icons that symbolizes each of the features it includes. Icons for the camera, call status and the contact book function are implemented.

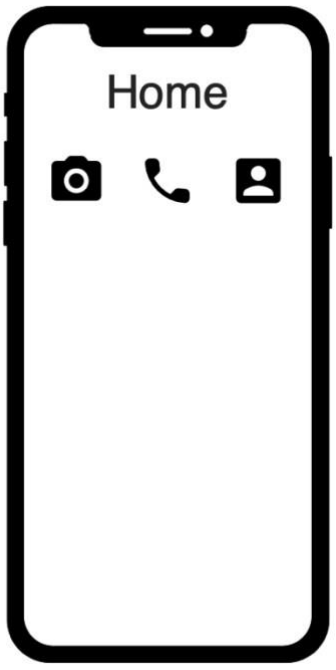

Figure 9: Mockup for the native mobile applications user interface

As seen in figure 9, this is how the application is intended to have its user interface. When the user presses one of the visual icons it gets redirected to the page associated with its feature. This is possible due to the built in component called “navigationlink” in Swift, that processes the transition between each feature view containing its functions.

5.1.2. Feature construction

The implementation of the features is the most crucial part of constructing the application and consists of the camera, contact book and “in call” function. These features have been chosen to use for the best possible comparison between the PWA and the native application in terms of which version of application currently has the most options of usable features. By implementing these features that in the app development scene serves as essential parts for building an application, this intends to serve as a analysis of the flexibility aspect and feature potential of the apps.

The first feature implemented for the native mobile application was the camera function. The obvious choice of using the SwiftUI with the UI kit frameworks based on the relation to the programming language Swift were made. These frameworks were used to build the UI functionality for the feature. In more detail, the most crucial component for building the camera function UI was the “UIImagePickerController”. By using this component, the camera feature uses the standard iPhone camera layout, without having to manually construct it, which reduces development time. To be able to use it with the terms of SwiftUI, the protocol UIViewControllerRepresentable were used. The protocol functions as a connector between the UIkit and the swiftUi framework, which makes every class implemented from UIkit usable with SwiftUI.

Next the class coordinator was implemented to serve as a delegate of the UIImagePickerControllerDelegate and UINavigationControllerDelegate, more specific to handle the lifecycle from the moment a picture has been captured all the way of storing it for use at a later stage. The delegate uses communication from user action and from that information decides how the program would respond. In the coordinator class this works so that when the user engages with the delegator UIImagePickerController by

taking a photo, it will inform its delegate (the coordinator class) to handle the operation.

```
picker.delegate = context.coordinator
```

Figure 10: Setting the delegate for the instance of the coordinator class.

In figure 10 it can be seen that the coordinator is set up to be notified with its methods if the delegators registers an event, more in detail the line sets “context.coordinator” as an delegate for the picker object.

For a better understanding of the delegates' functionalities works refer to chapter 2.

The feature for determining if the user's mobile device is engaged in a call or not is built principally using the CallKit framework. The feature functions in a straightforward way, if the user receives and enters an incoming phone call during the usage of the device, the application is built to sense it and update an visualized text at the user screen indicating if the device currently is in call or not. Due to CallKit being built for VOIP (Voice over internet protocols) calls, the class named CxObserver is used to in detail sense what call state the device obtains in real time if its incoming, started or ended.

The contact book feature is the most comprehensive and hypothetically the most advanced, due to the feature gathering multiple information used for setting the results in chapter 6. The feature principally has the goal of accessing the mobile devices built in contact book and displaying the contact object in the application. The sub function of the feature has the purpose of multiplying the contacts into 10000 elements. This is done to achieve a long list of scrollable elements for various performance testing. Thus, one of the scientific goals was stated as to create a long list of elements, the natural choice of combining an implemented feature build with it was made.

The feature was built utilizing the Contacts framework provided by Apple. To be able to fetch the user device contacts, the component “CNContactstore” was used. All the imported contacts will be stored in an array called “Contacts”. When the user grants the permission for the

application to use the devices contacts, the “fetchContacts()” function is called which imports and creates a contact instance for each fetched object. In figure 11, the pseudocode for the fetchContacts() function can be seen.

```
Step 1
      Start by initializing contactStore as a new instance of the
      CNContactStore class.

      Create an empty Array "Contacts" to hold objects for each contact.

Step 2
      Define keys as the required information to be fetched from the contact
      store, which is the full name in this case.

      Construct a request object of type CNContactFetchRequest, which asks
      for the keys.

Step 3
      Start a try-catch block to handle potential errors:

      Within this block, use the enumerateContacts method of contactStore to
      iterate through each contact.

      For each contact, construct a fullName string by combining the
      givenName and familyName.

      Then, form a new Contact instance with this fullName and append it to
      the contacts list.

      In case an error occurs during the process, print out the error
      message.

Step 4

      Next, verify if the count of contacts is fewer than 10. While it is:
      duplicate the contacts list to contain 10 contacts.

      If the fetched contacts is 10 or more:

      Shuffle the contacts, select the first 10 and store them as
      "selectedContacts".

      Clear the "Contacts" array and duplicate the 10 elements 1000 times(to
      achieve 10000 contacts).
```

Figure 11: Pseudocode for the fetchContacts() function.

To fulfill the long list of contacts, test contacts has been created on the mobile device for security reasons and will be fetched from the device and multiplied by 1000 to get 10000 contacts for the final array. If the number of contacts being fetched are smaller than 10, a while statement will multiply the array to reach 10 contacts. The user will be greeted with the list of contacts followed by two functions for deletion and adding of the objects. The buttons called “addbutton” and the function “delete(at)” has been implemented.

5.2. Implementation for Progressive Web Application (PWA)

This section of the project explains the implementation of the PWA. The choice of technique used to craft the PWA can be seen as a big step of the working process, and based on the approach mentioned in chapter 4, the library react will be used with JavaScript and Visual code.

5.2.1. Constructing the User Interface

More in detail the UI is designed with the React Router DOM structure that can be seen in appendix A. The necessary components that form the application features and modules are imported to be able to craft the UI based on Figure 7, in the App.js file.

The React Router DOM imports the "Routes" component, which consist of ingrained "Route" components for each feature. The Camera, the photo gallery, contact book and the "check call status" feature have their own route, that consists of a Uniform Resource Locator (URL). When the URL path is being visited in the web browser, the application user will be displayed the respective feature connected to the element URL.

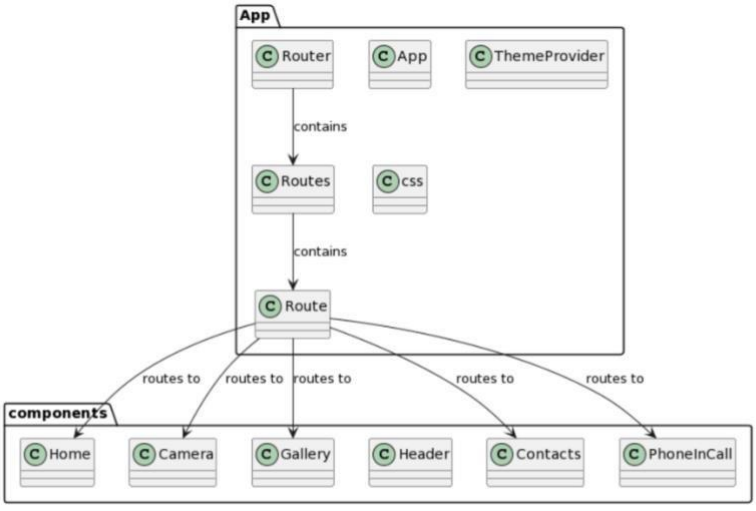

Figure 12: Component diagram overview for the User Interface (UI), for the PWA.

In Figure 12, the router process for each feature component can be seen followed with the themeprovider, the application function and the css used for designing the home screen.

5.2.2. Feature construction

The PWA was constructed to imitate the native mobile application to the fullest extent possible with the same features and design for av even comparison for the research study.

The Camera feature could be implemented with the API called “Mediadevices”, which is supported in modern browsers including Safari, that was used for the PWA. By using React, the component “Camera” is created for handling the functionalities of the Camera feature including capturing and storing images. By using the component called “navigator.mediaDevices.getUserMedia” the program gets access to the Realtime video stream of the mobile device and portraits it as a live camera view. Underneath the live camera view there are two buttons in a vertical alignment, “Capture Image” and “View Gallery” buttons. When pressing the Capture image button, the function called “captureimage” is executed. The function crafts an image canvas based on the live camera view and transforms it into a data URL. By transforming it into a data URL, the image will get stored in the browser's local storage, making it easily accessible. The second button called View Gallery will redirect the user to the gallery of taken photos. All the photos appear with an “delete” button underneath it and when pressed the specific photo will be deleted from the local storage and the user interface.

The “Phoneincall” feature for the PWA is built with the Mediadevice API and gets access to the device's active audio streams. Note that this is a work around method for checking if the mobile is in call or not due to react and PWA:s not having direct access to voice over internet protocol components like the native mobile application does. The feature could not be implemented with a success and reliability level as the native mobile application due to this, but the work around methods would try to optimize it. The functions that would try to serve as work around

methods were to sense if the device currently used the audio input at the same time not using the audio input generated using video recording.

The contact book feature shows a slight difference in the PWA compared to the native app. As for the native app the PWA contains a functional interface for adding, deleting and sorting the contacts but with the downside of not importing the user devices contacts, due to the PWA not having functional API:s for the feature on IOS supported devices. Even though the feature access limitations, the overall feature was built with the aspiration to imitate the contact book with the same implemented functions for the native app, this allows for a steadier performance comparison.

5.2.3. Web worker

To optimize performance for the PWA, an web worker had been implemented. The web worker is set up to handle the heaviest resource demanding tasks, the function of duplicating the contacts to reach a total of 10 000 objects and the function of sorting the contacts in alphabetical order. Theoretically these tasks could potentially demand a high CPU load from the measuring device if running directly on the main UI-thread, with consequences of blocking it, especially when handling this large number of objects at once. By implementing web worker for the PWA, later measured tests could be done to see if performance could be faster with optimization compared to the native app.

The code for the web worker is written as a string containing a switch case statement for communication with the main thread, and the actual functions for duplicating and sorting the contacts. The string converts to a Blob that stores the raw data of the web worker and an URL is created referencing to the Blob. The actual web worker is then started with the given URL and runs the script in a different thread instead of the main thread. In figure 13, the implemented code for starting the web worker can be seen.

```
const workerBlob = new Blob([workerCode], { type: 'application/javascript' });
const workerUrl = URL.createObjectURL(workerBlob);
const worker = new Worker(workerUrl);
```

Figure 13: Code for starting the web worker.

The main thread is communicating with the web worker by sending specific messages, ordering it what action to perform. Based on what feature function the user decides to use, functions placed in the main thread is sending a message containing the operation type and the contact data. The web worker receives the messages, performs the chosen function, and sends back the new data to the main thread.

5.2.4. Publishing

Uploading the PWA to the web was the last step in the process of building the application. There are various ways of doing this task but by doing detailed research, the most prominent way of solving the problem was to use AWS. Thanks to the company Knightec [2] this method could be used to upload the application effectively and satisfactorily to the web.

The first step in transforming the application to an actual PWA was to implement an S3-bucket. For this purpose, the S3-bucket was used to store all the information and programming code that's being used for constructing the app.

The next step was to upload the material to the bucket, to achieve this AWS CLI were configured and used. The CLI was configured with personal credentials provided by Knightec. The AWS CLI was later used to synchronize the program code to the S3 bucket, this was done with the following command "aws s3 sync build s3://wibe-pwa-website" were the "wibe-pwa-website" represents the S3 bucket used for the project.

To be able to ensure safe communication within the PWA, the PWA was set to be operated over HTTPS. To accomplish this the Amazon CloudFront feature was used for distributing the web material. A CloudFront distribution was made through the AWS console and was assigned the "wibe-pwa-website" S3 bucket as its source origin, for the bucket content to be ready for web delivery. The S3-bucket settings was set to enable static web hosting.

The CloudFront distribution was set to be operated over HTTPS by the service itself assigning a generated SSL certificate, encrypting the information transferred between the PWA and the user. When the service had been implemented in the project, CloudFront yielded a domain used

for accessing the PWA. In figure 14, all of the components used in AWS can be seen.

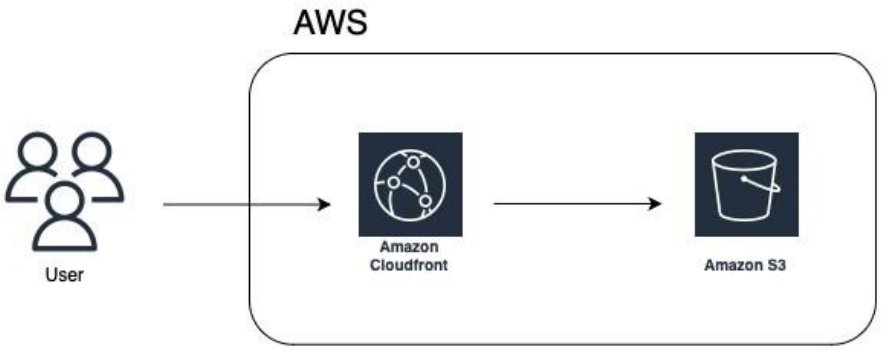

Figure 14: Component diagram overview for hosting the PWA.

5.3. Measurement

The measurements will be handled with focus on the CPU (Central Processing Unit), FPS (Frames Per Second) use and the speed for integration with the implemented features in the PWA and the native mobile application. The CPU works as a way of gaining detailed information about the power of processors for each application, and how much of the total CPU processing power being used. The FPS will be measured to understand how fast the screen-based content for each application is and how it reacts to being put under pressure for a series of feature tests.

For testing CPU and the FPS for each app application a built-in tool within Xcode called “Instruments” is utilized. Instruments lets you choose the specific measurements for the needs and record a sequence when the tests are performed for getting the wanted information. See chapter 2 for more information about the tool “Instruments”.

The following are the tests that will be performed for the PWA and the native mobile application, with the intention of fulfilling the objectives seen in chapter 1.

- Scroll through the contact list containing 10 000 elements from top to bottom, at a fast pace.
- Use the multiply function for the contacts in total 8 times then sort the long list of contacts, in fast pace.
- Navigate quickly between the home and the contact list in total 10 times.
- Taking eight photos, then entering the gallery, deleting all of the 8 photos, navigating to the home and then re navigating to the camera feature.
- Randomly deleting 30 contacts from the contact list at a fast pace.
- Adding eight contacts, then deleting the same contacts at a fast pace.

6. Results

This chapter handles the results given in the thesis, with measured performance features that could be accessed for respective app.

6.1. PWA feature UI

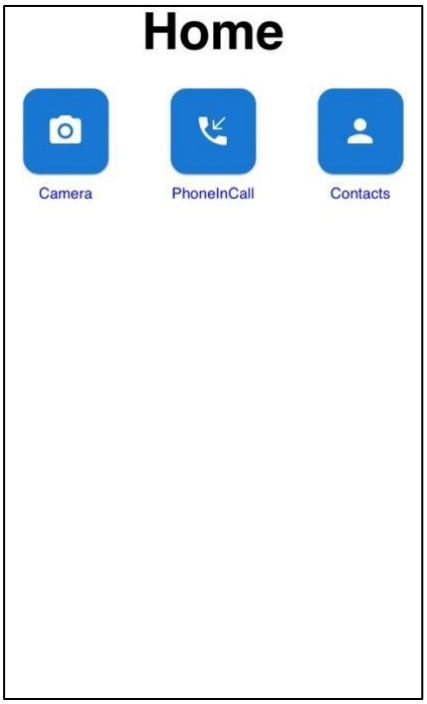

Figure 15: Screenshot of the home page for the PWA.

Figure 16: Screenshot of the "in-call" function for the PWA, with the mobile device not being in a call at the specific moment.

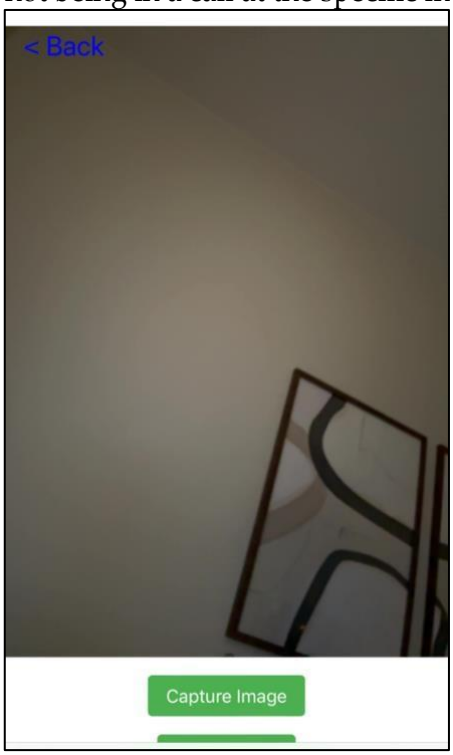

Figure 17: Screenshot of the camera function start page for the PWA.

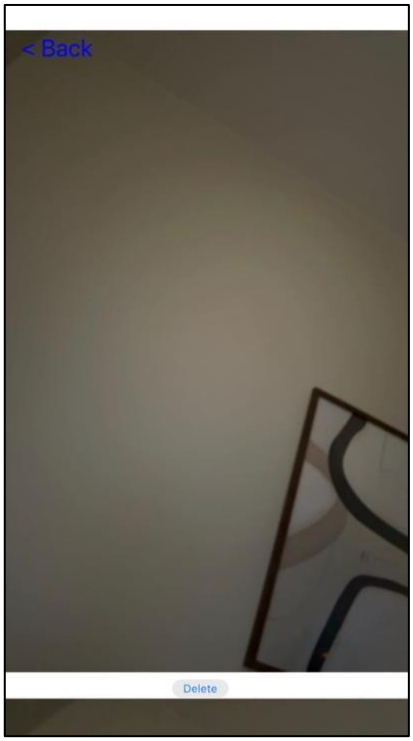

Figure 18: Screenshot of the image gallery for the PWA, showcasing photos being taken.

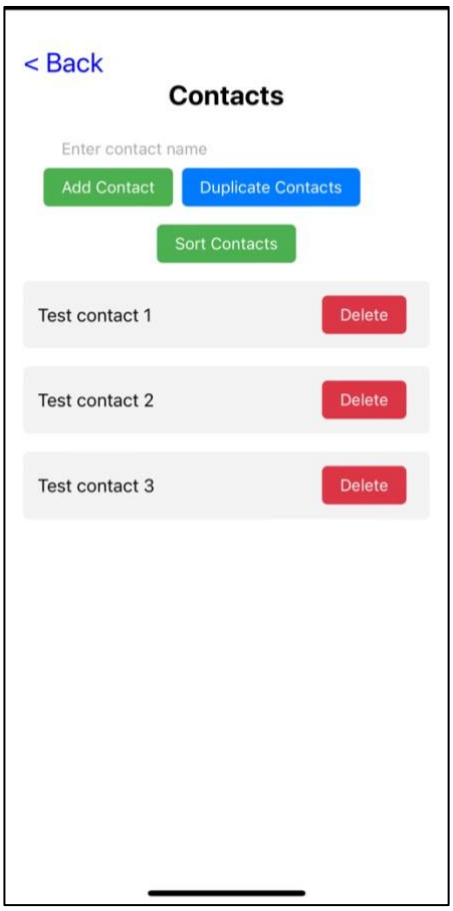

Figure 19: Screenshot showcasing the contact feature for the PWA, with three contacts being created with the “Add Contact” function.

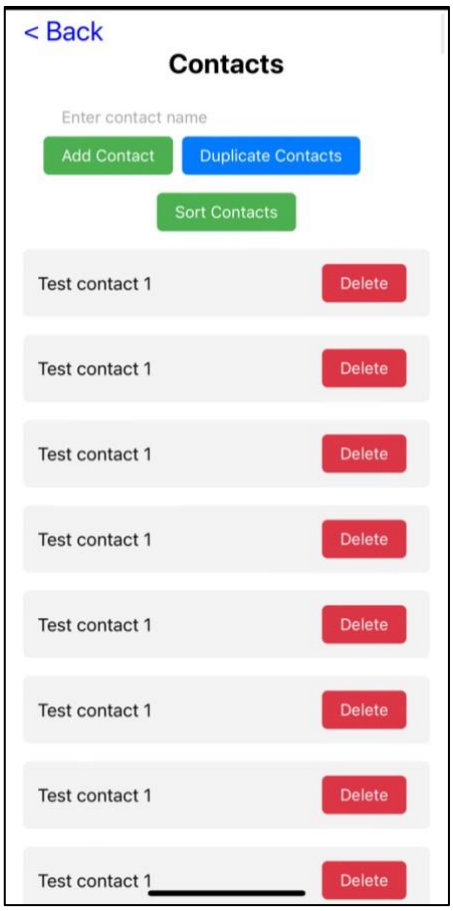

Figure 20: Screenshot visualizing the contacts being sorted and duplicated to reach in total 10 000 contacts.

6.2. Native mobile application feature UI

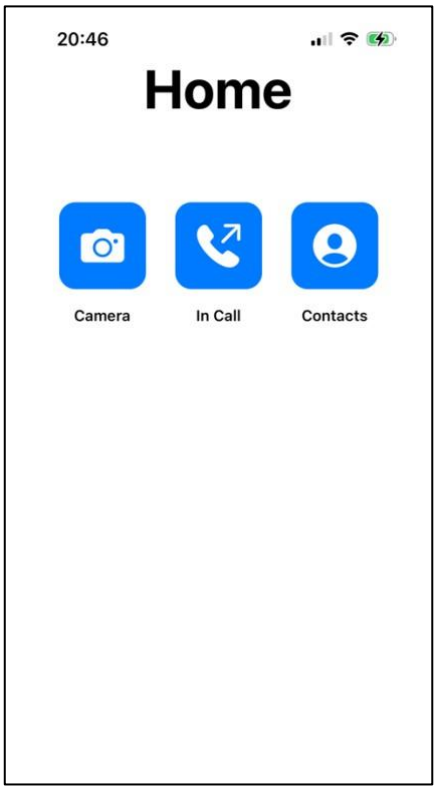

Figure 21: Screenshot showcasing the home screen for the native mobile app.

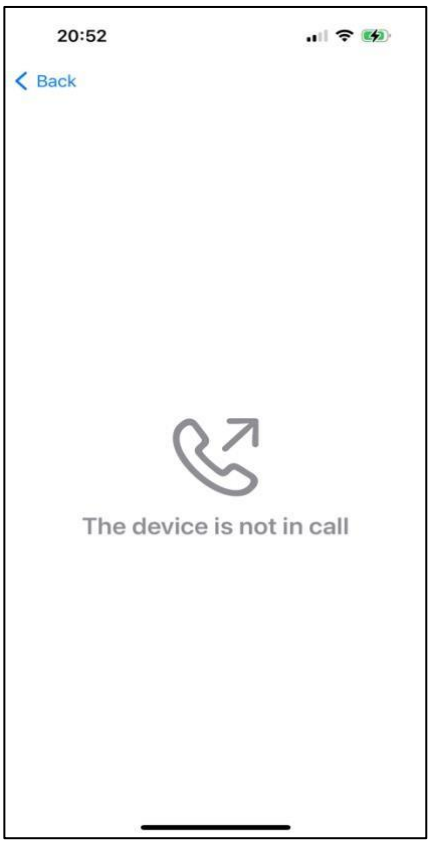

Figure 22: Screenshot showcasing the In-call function for the native mobile app, where the mobile user device is not in a call.

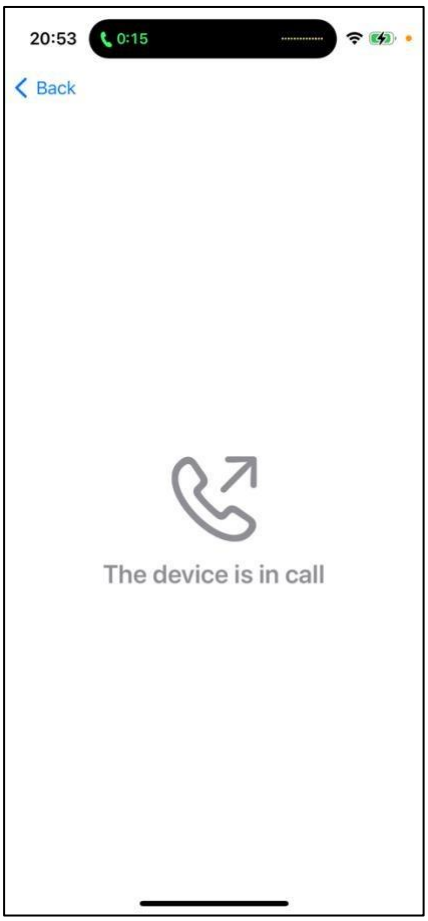

Figure 23: Screenshot showcasing the In-call function for the native mobile app, where the mobile user device is in a call.

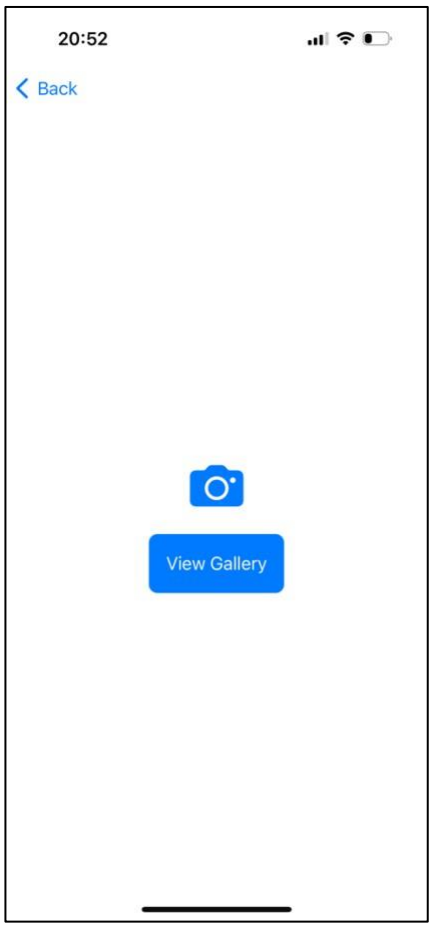

Figure 24: Screenshot visualizing the camera feature for the native mobile app.

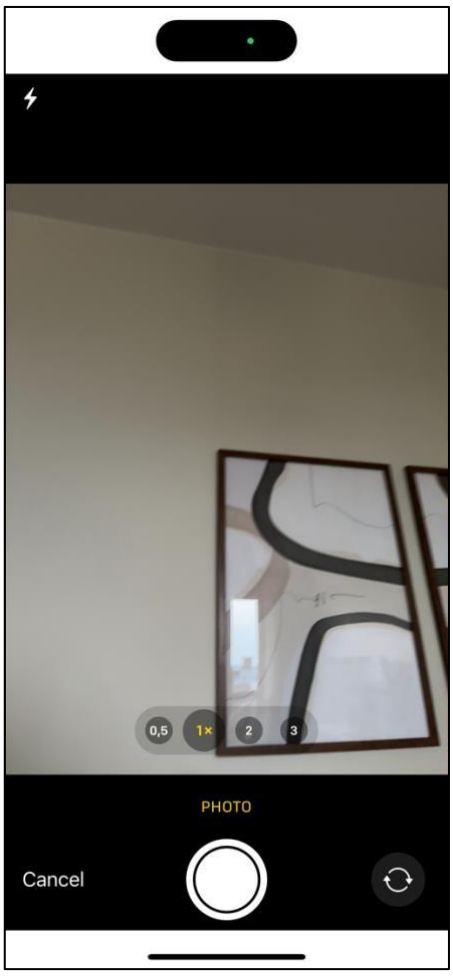

Figure 25: Screenshot showcasing the photo preview of the camera feature.

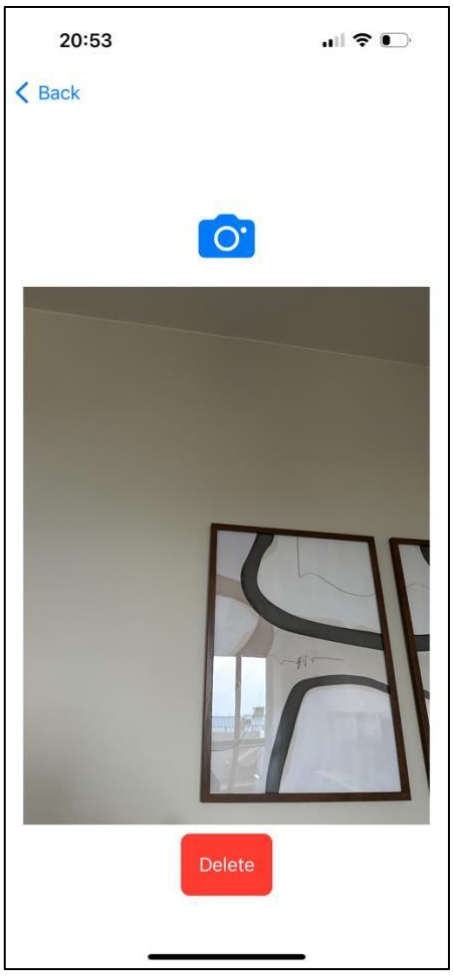

Figure 26: Screenshot showcasing the Image gallery of the camera featuree.

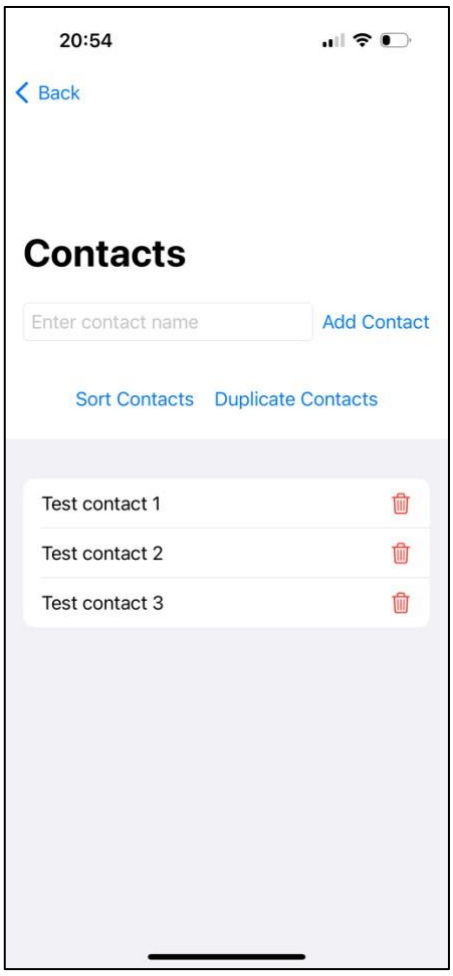

Figure 27: Screenshot showcasing three contacts being imported from the mobile device called “Test contact 1”, “Test contact 2” and “Test contact 3”.

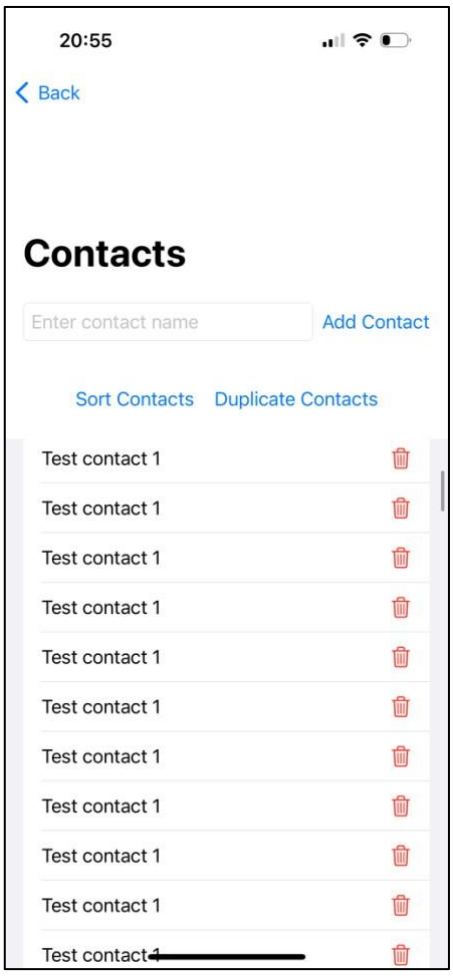

Figure 28: Screenshot showcasing the three contacts being multiplied and sorted.

6.3. Feature access

Features accessed	Progressive Web Application (PWA)	Native mobile application
Contacts	NO	YES
Camera	YES	YES
Photo Gallery	YES	YES
Delete Photos	YES	YES
Check call status	YES*	YES

Figure 29: Results of features being accessed

- YES: The feature can be accessed directly.
- YES*: The feature cannot be accessed directly, but could be imitated with workaround methods, steps and tools.
- NO: The feature cannot be accessed, there are no ways of accessing this feature directly, since the application doesn't support this feature.

6.3.1. Measured performance results

This section will handle the performance results for the applications, with total CPU load and FPS.

The figures seen in chapter 6.3.1 showcases the total utilization of CPU load where the percentage indicating how much of processor cores being used. For clarification, a 100% CPU load represents one fully utilized core, while a 200% CPU load represents two fully utilized cores for the testing device.

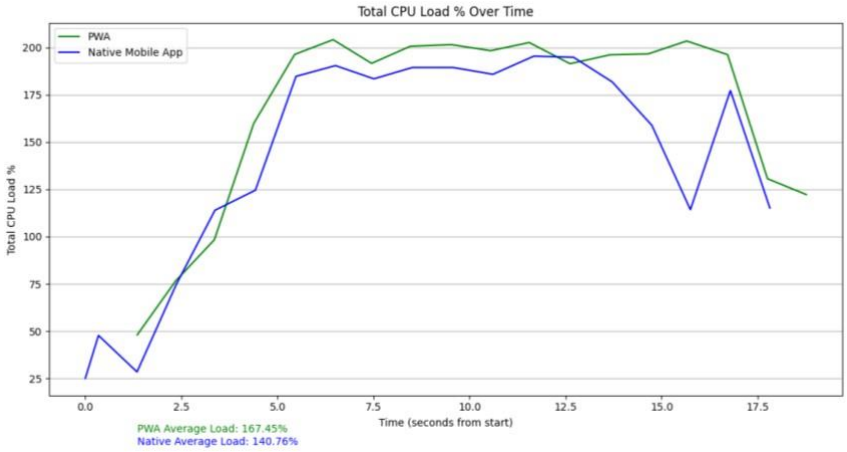

Figure 30: The CPU load when scrolling top to bottom in the contact list.

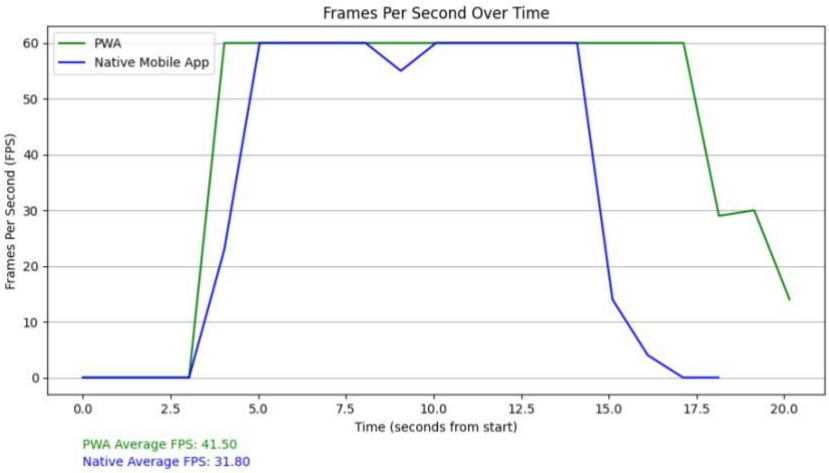

Figure 31: FPS when scrolling top to bottom in the contact list.

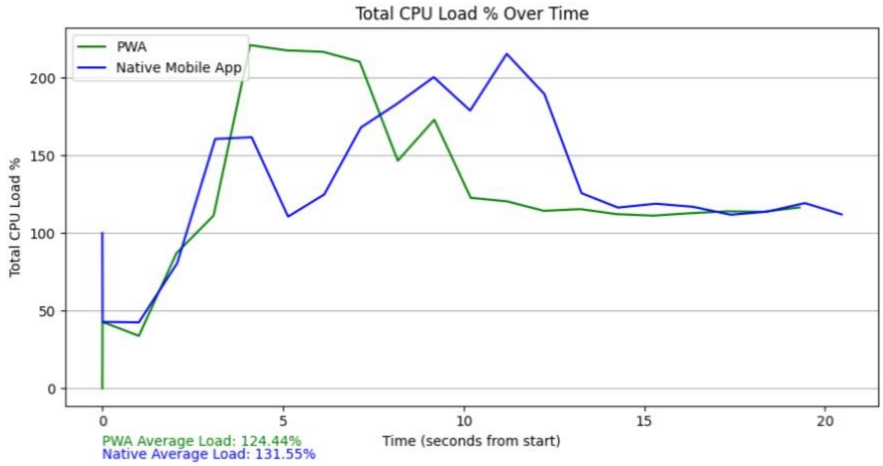

Figure 32: The total CPU load when using the multiplying function for the contact list in total 8 times, then sorting the list.

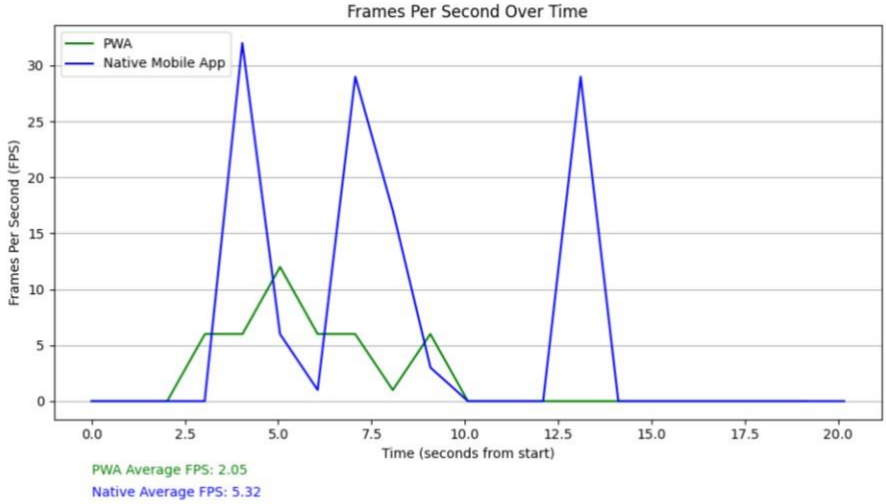

Figure 33: The FPS when using the multiplying function for the contact list in total 8 times, then sorting the list.

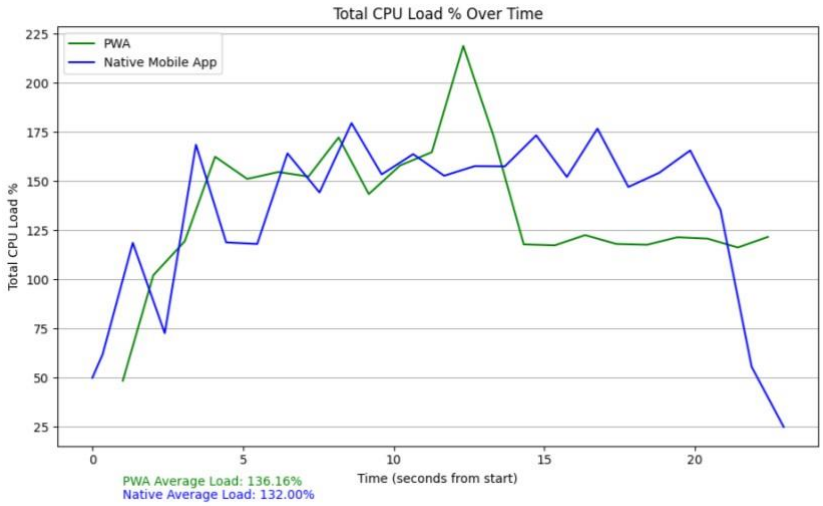

Figure 34: The CPU load when navigating quickly between the home page and the contact list in total 10 times.

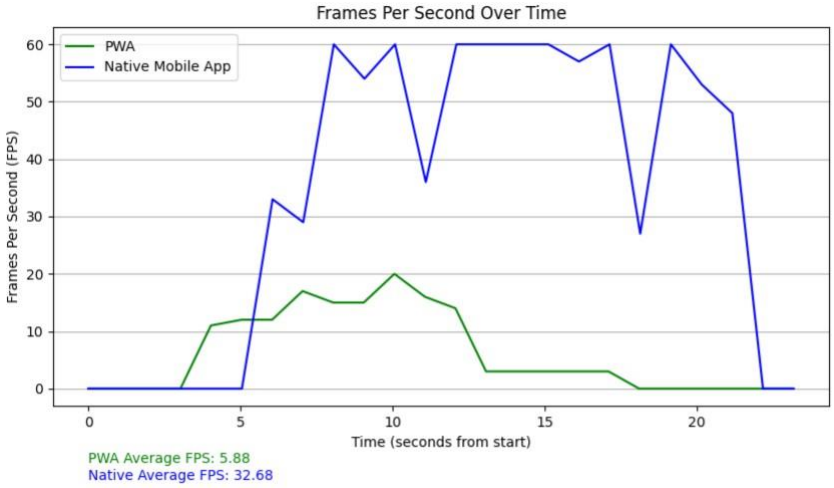

Figure 35: The FPS when navigating quickly between the home page and the contact list in total 10 times.

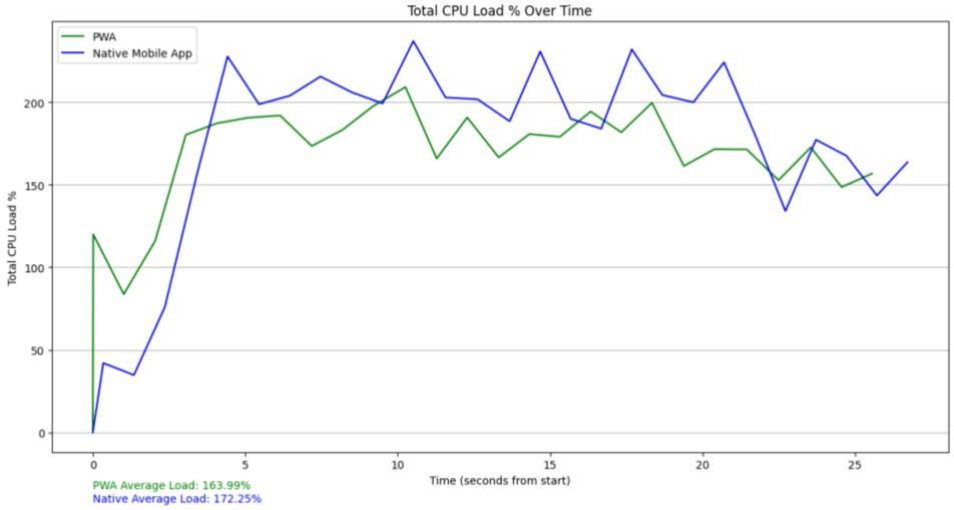

Figure 36: The CPU load when taking eight photos, entering the gallery, deleting the 8 images, navigate to the home then navigate to the camera.

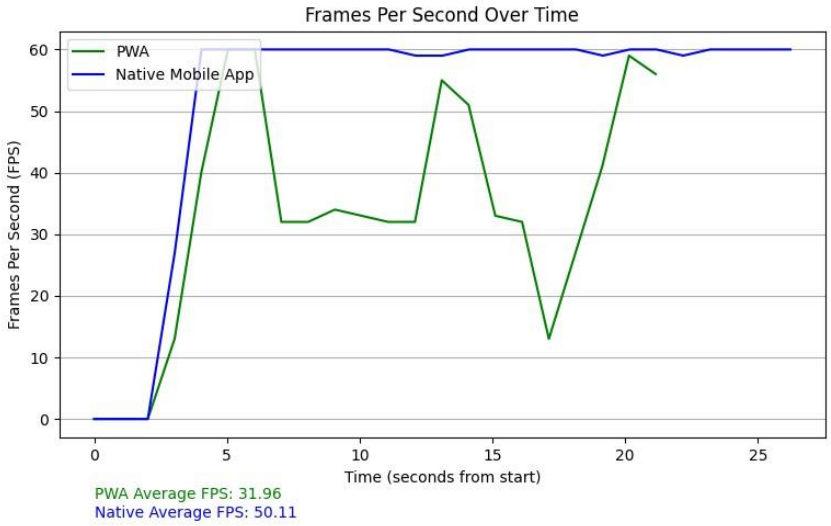

Figure 37: The FPS when taking eight photos, entering the gallery, deleting the 8 images, navigate to the home then navigate to the camera.

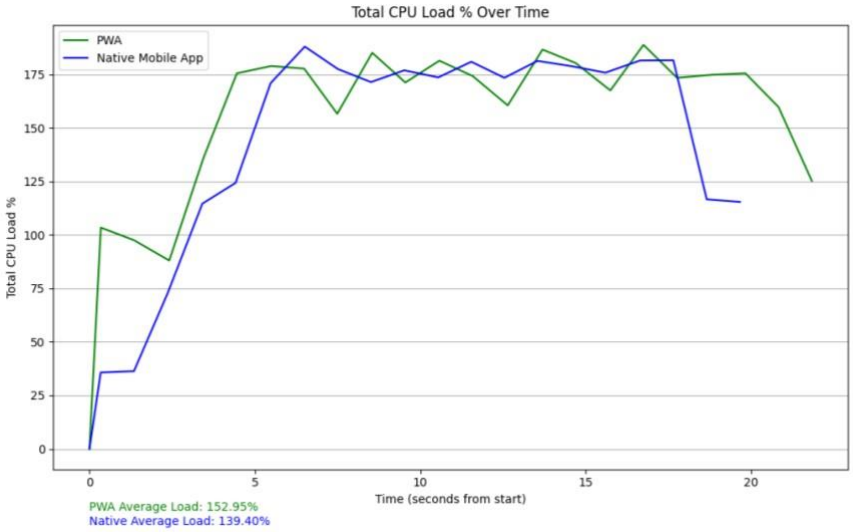

Figure 38: The CPU load when deleting 30 contacts from the contact list at a fast pace.

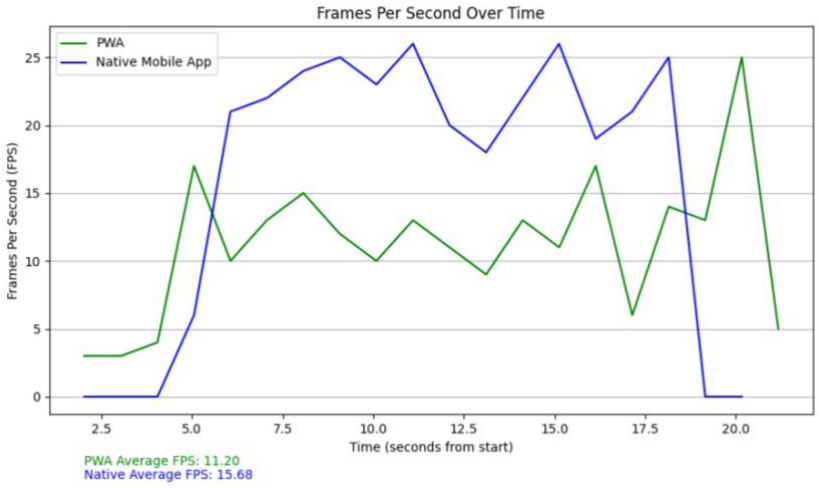

Figure 39: The FPS when deleting 30 contacts from the contact list at a fast pace.

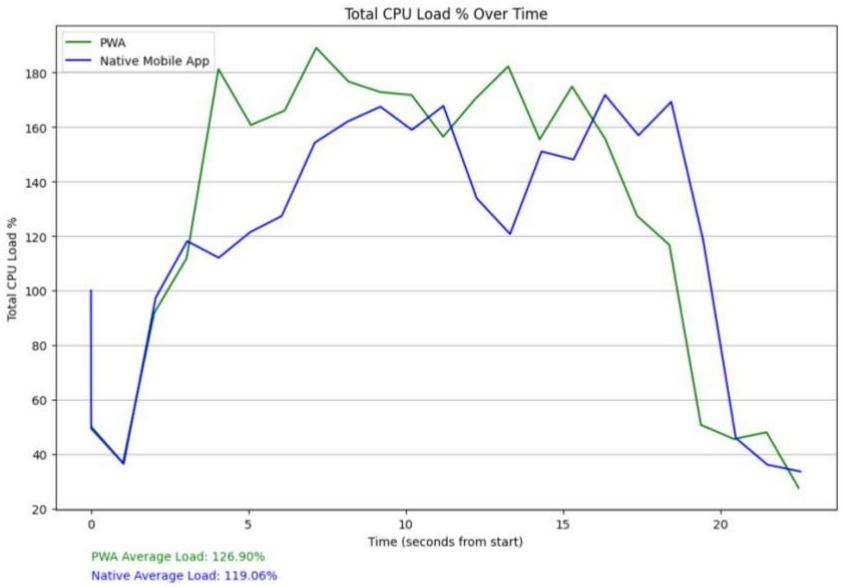

Figure 40: The CPU load for adding 8 contacts, then deleting them at a fast pace.

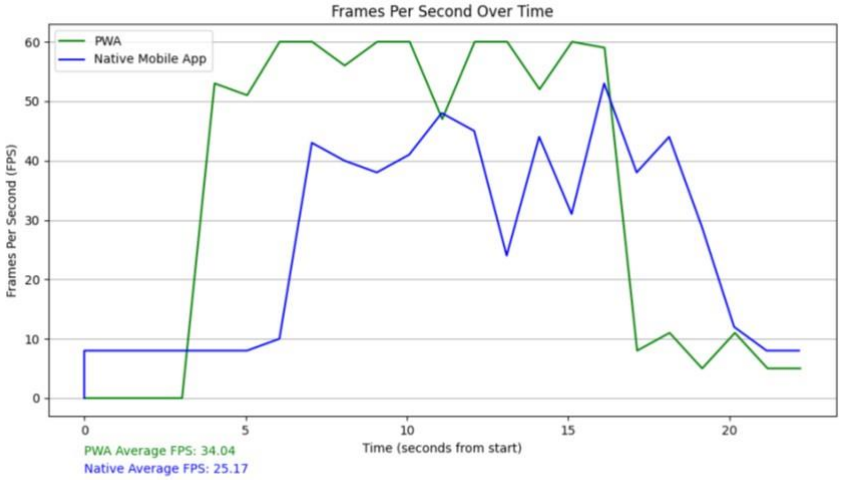

Figure 41: The FPS for adding 8 contacts, then deleting them at a fast pace.

7. Discussion

This chapter will address the analysis and discussion regarding the results obtained, method used, scientific discourse, recommendations, and considerations of ethical aspects.

7.1. Analysis and discussion of results

Analyzing the results in chapter 6, the PWA performed overall worse than expected and the native mobile application had functional but mediocre results.

Given my personal experience interacting with mobile devices, I have used native applications nearly the full time owning a mobile device. The habit of only using native apps containing all necessary features over the years has led me to have expectations in terms of functionality and usability for applications, to at a minimum of being able to provide basic features. As expected, when developing the native mobile application, it could be understood quickly by contributing research that all the wanted features could be implemented without a hitch. On the other hand, when tried to imitate the native app for the PWA, there occurred problems. The first feature implemented for the PWA was the camera with gallery feature, it could be fully accessed. When implementing the second phone in call feature, it could be noted fast that it didn't exist any API:s for direct access. By research it was learnt that the "feature" could be implemented in a non direct way by using work around methods. Even tough work around methods were used, the feature didn't perform good enough for being trustworthy, due to it giving incorrect statements for call status. Disappointing, the contact book feature could not be accessed either, not even with workaround methods. Due to the last feature couldn't be accessed, the contacts for the PWA can't be imported from the user device. The only way to access contacts where to manually add new contacts, in such a way the long list of elements could still be implemented. The feature access result for the PWA was worse than expected and it proves that native mobile applications is way ahead of the PWA, in terms of that metric.

By analyzing the measured performance results, the applications were very close in terms of total CPU load and FPS when conducting the tests.

Overall, the native mobile application performed better with a slightly lower total CPU load and a higher FPS indicating that it can update the display faster with more times per seconds and use the device processing power more efficiently with smaller number of resources. Even though the native app performed better overall, it should be noted that the PWA performed nearly as good with only minor differences. Comparing the results for the feature access with performance results, the performance results could be seen as with a better potential. It could be seen in some tests that the PWA was the better candidate, as seen in figures 38 and 28 with a higher FPS. Also, it can be noticed that when performing the test that included use of implemented web worker for PWA, the total CPU load was lower compared to the native app, as seen in figure 29. An interesting observation, this specific test included the heaviest and most resource demanding tasks for the applications. This proves that with performance optimization the PWA could handle computationally tasks at a stable consent when put under stress test, with even better performance than the native app.

7.2. Project method discussion

The method used to conduct this research study provided a clear pathway from start to finish, especially using the DSR methodology. By using the DSR methods it became evident how truly important well-structured project plans are for facilitating and providing potential for deeper research.

By setting up clear objectives for the scientific perspective of the study as the first phase, it made the process of knowing how to proceed with the thesis much simpler with already set up goals to work towards. The problem statements for this study were somewhat broad with multiple different approaches that could be used for being able to answer the questions. By setting up these goals it helped me to choose a stable route and to be able to focus on the development of the applications as fast as possible, due to me hypothetically thinking that the development phase would be the most time consuming as it also turned out to be. By doing deep research before beginning the actual development, it gave me a large knowledge for what type of approaches currently exists for app development and what approaches would suit me the most for this

study, in terms of making the process efficient with the best results possible. The most important step for basing the development on best grounds, were without a doubt the Pugh matrixes seen in chapter 4. With the use of the Pugh matrixes, it directed me to the most fitting tools for building the apps since I had zero pre-knowledge for the listed approaches.

By demonstrating the ideas and the progress being made during the implementation for the supervisors at school and the company Knightec, it contributed to meaningful discussions about how the program structure could evolve for a more comprehensive comparison of the applications.

With the comparison of the apps in terms of measured performance and feature accessibility, the evaluation phase could be crafted successfully with the given information. By setting up a large number of performance tests instead of a few together with implementation of different features, the evaluation for the performance and feature access status could be done with a steadier ground to work with.

7.3. Scientific discussion

By completing the study, knowledge has been gained in terms of programming languages, libraries together with how the full process of building two versions of functional mobile applications from step zero to being published.

Comparing my study with the related works, it can be seen that the native app performed badly with being “size heavy” in the related work. That is a noticeable difference from the scientific knowledge gained from my study, as the native app performed better mostly regarding the application speed but also in general.

7.4. Recommendation

Based on the information gained from this study, recommendations can be formulated for practical aspects and future business strategies. The understanding of the project allows development for company it

solutions to attain more advanced applications based on the specific situation.

The recommendations being made would depend on the objectives and requirements set by the company. If the company seeks an application with the factors of being stable with a fast development process, rapid performance and in general low-priced. The recommended application version would be the PWA. Thus the PWA is already performing at a stable state, gathering the insights of this study different optimizations is possible also for the application, with extensions such as web workers and service workers, more information about service workers and web workers can be seen in chapter 2. If the purpose of the application were to reach an as large target audience as possible with less advanced features, the PWA would be the right choice. The PWA would only require one base of code due to its cross-platform ability. Every single device with a feature of browsing the internet would be able to reach the application, including operating systems like IOS, Windows and Android. Due to its cross-platform ability and only one program build, it will decrease the cost significantly if you compare it to the native mobile application which requires one developed program for each operating system.

On the other hand, if the company/developer seeks to attain an application with more advanced implemented functionality that requires a direct access to API:s/components adapted for mobile devices, a consideration of developing a native mobile application instead of an PWA will indeed be made. As mentioned in chapter 6, the results submitted that the PWA had even worse than expected access to the mobile devices built in features. Due to the features being basic need of building a applications, the recommendation for a developing team would be to either study if the desired features is available for the PWA or ideally the best case would be to invest in building an native application that has direct access to all the mobile features.

7.5. Ethical and societal discussion

Based on this study subject there are numerous ethical and societal aspects to keep in mind based on the technologies used. First and foremost, argumentatively the most important case could be to grant an safe user experience for potential customers/users, following their integrity. Since the native mobile application has an larger scale of access to the devices built in features than the PWA, there could exist larger problems and points to consider regarding the user integrity during the development for each feature.

8. Conclusions

This chapter provides an overview of the research carried out research, the accomplishments in relation to the stated objectives, proposed solutions, and possible future work.

The purpose of this study was to highlight the differences in terms of performance between PWA:s and native mobile applications. Both applications were created with the same design and functionality, which makes it easier to point out the differences from a more balanced and unbiased evaluation. The results proves that there are differences in various areas which make the future of the mobile application development scene critical.

8.1. Objectives achieved

The goal of this research work was to develop one PWA and one native mobile application with documentation and analysis of the development process with performance tests. By doing this the following problem statements seen in chapter 1 were able to be answered.

- **How much time does it take to develop a PWA versus a native mobile application?**

The time taken to develop both application versions is influenced by different aspects, how complex the application is, what type of functions are used and the developer's prior knowledge and skillset. In this research study it was shown that the applications were quite even in terms of the time it took to develop it, but the PWA were slightly easier to implement. The primary reason for this result could be since me as the developer had much larger prior knowledge of the programming languages HTML, CSS, JavaScript and web development in general. On the other hand, the development process for the native mobile application were slightly more time consuming since I had close to zero previous knowledge of the programming language Swift. During the process of the study, I had to learn and adapt to develop with the language but despite that the development times for each application were still slightly even.

On the other hand, if the app is required to be compatible across multiple mobile operation systems, the PWA has superior development time. This is because of the PWA has accessibility for every device and various mobile operating system, therefore the need of only creating a singular app exists. Compared to the native mobile version that requires one built app for every operating system.

- **Can a PWA be faster and more secure than a native mobile application?**

As seen in the results in chapter 6, the PWA could have a better performance than the native app if optimized with web workers. The PWA could potentially be more secure than a native app, due to the PWA using HTTPS communication for transferring data being encrypted between the user and app. The native app could also use encrypted communication, but it is not a requirement for a functional app.

- **What can a Native Web App do that a PWA cannot?**

As of today, the native mobile application is leading significantly in terms of accessing features compared to the PWA. Basing on what the native app could do from this study, it could fully access the user devices Contacts and sense if the device currently is in a phone call or not, which the PWA couldn't achieve.

- **Will we still have the need for an app store in ten years?**

Currently as of today the native version for mobile applications is the better overall performing app. But if the PWA technology keeps evolving into acquiring the same feature access or even better than the native app, app stores could potentially be meaningless for the human being. But as of today, the app store together with native apps is crucial for use of everyday features.

8.2. Future Work

By doing this research study, information about the current state for the PWA versus the native mobile application as of today has been provided. By using this information, a new way of operating with the question at issue, which application is the most superior has emerged. The opportunities of furthermore investigating this question at hand with future work exists, based on the finding of this research study. Future work conducted in different manners could potentially strengthen the research, providing a more comprehensive understanding of the application statuses.

This chapter delves deeper into the future aspects and possibilities for this research, inspired by the observations that have been made throughout the study.

8.2.1. Cross-Platform access

One possible work that could expand the thesis study is to develop the same applications within different environments, such as with direct support for Android (Android studio) to provide a more comprehensive comparison across different platforms to see if the PWA and the native app would possible behave differently in terms of functionality, performance and security. The goal of this work would mainly be to investigate if the PWA has a larger access of features compared to the PWA created in this study.

If someone would continue this study with future work, they should research the overall subject with this thesis as starting information, find the best possible platforms to use for fitting the project best at hand, develop one PWA and one native mobile application for each environment and compare.

8.2.2. Cost analysis

An better overall perspective in terms of strategical decisions for choosing the best suited application version to work with, an future work could be to analyze the total cost. By getting full disclosure of the cost of developing and maintaining the two applications with ROI, businesses and developers could potentially make better economic decisions.

If someone were to continue the study with this future work, they should investigate the cost for software, possible licenses, publishing, and Maintenance.

References

[1] Magestore, “PWA vs Native app and how to choose between them”, https://www.magestore.com/blog/pwa-vs-native-app-and-how-tochoose-between-them/ Published: 2022-08-03. Accessed: 2023-04-18.

[2] Knightec, “Will PWAs kill the native app market?”, https://knightec.se/Student/willpwas-kill-the-native-app-market/ Accessed: 2023-04-19.

[3] Wikipedia, “Xcode”, https://en.wikipedia.org/wiki/Xcode Published: 2003-11-04. Accessed: 2023-05-13.

[4] Apple, https://www.apple.com/ Accessed: 2023-05-13.

[5] Visual Studio Code, “Getting Started”, https://code.visualstudio.com/docs Accessed: 2023-05-16.

[6] Angular, “Introduction to Angular concepts”, https://angular.io/guide/architecture Accessed: 2023-05-16.

[7] Ericsson, “Ericsson Mobility Report” [Picture], https://www.ericsson.com/49dd9d/assets/local/reportspapers/mobility-report/documents/2023/ericsson-mobility-reportjune-2023.pdf Published: 2023-06-xx. Accessed: 2023-08-28.

[8] AWS, “What's the difference between Web Apps”, Native apps and Hybrid apps, https://aws.amazon.com/compare/the-difference-between-web-appsnative-apps-and-hybrid-apps/ Accessed: 2023-05-21.

[9] International Journal of Innovative Research in Science, Engineering and Technology, “Impact of Progressive Web Apps on Web App Development”, https://www.researchgate.net/profile/Sayali-Tandel-2/publication/330834334_Impact_of_Progressive_Web_Apps_on_Web_App_Development/links/5c5605d3a6fdccd6b5dde018/Impact-ofProgressive-Web-Apps-on-Web-App-Development.pdf Published 2018-09-09. Accessed 2023-05-21.

[10] Jstor, “Design Science in Information Systems Research", https://www-jstororg.proxybib.miun.se/stable/pdf/25148625.pdf?refreqid=excelsior%3Ae25ecf20647d5cc39b4b50a47821abdc&ab_segments=&origin=&initiator=. Published 2023-03-xx. Accessed: 2023-05-23.

[11] SMART goals in education, https://helpfulprofessor.com/smart-goals-ineducation/ Published 2022-02-19. Accessed: 2023-05-23.

[12] HelpfulProfessor, “SMART goals in education: importance, Benefits, Limitations” [Picture], https://helpfulprofessor.com/smartgoals-in-education/ Published 2022-02-19. Accessed: 2023-05-23.

[13] Wikipedia,”Vue.js”, https://en.wikipedia.org/wiki/Vue.js Published: 2016-06-02. Accessed: 2023-05-24.

[14] Microsoft, “Overview of Progressive Web Apps (PWAs)”, https://learn.microsoft.com/en-us/microsoft-edge/progressive-web-appschromium/ Published: 2023-01-24. Accessed: 2023-05-24.

[15] BrowserStack, “Angular vs React vs Vue: Core Differences”, https://www.browserstack.com/guide/angular-vs-react-vs-vue Published: 2023-05-11. Accessed: 2023-05-24.

[16] Codeinwp, “Angular vs React vs Vue: Which framework to choose”, https://www.codeinwp.com/blog/angular-vs-vue-vs-react/#gref Published 2023-03-24. Accessed: 2023-05-24.

[17] Getdevdone, “Vue vs. React vs. Angular – Pros, Cons, Use Cases and Comparison”, https://getdevdone.com/blog/vue-vs-react-vs-angular-pros-cons-usecases-and-comparison.html Published: 2022-01-04. Accessed: 2023-05-24.

[18] Incora, “Angular vs React vs Vue: The Main differences and Use Cases”, https://incora.software/insights/react-vs-angular-vs-vue Published: 2022-02-04. Accessed: 2023-05-24.

[19] Knowledgehut, “Swift vs Objective C – Differences Explained in Detail”, https://www.knowledgehut.com/blog/programming/swift-vs-objectivec Accessed: 2023-05-24.

[20] Educative, “Swift vs. Objective-C”, https://www.educative.io/answers/swift-vsobjective-c Accessed: 2023-05-24.

[21] Careerfoundry, “What is a Programming Library? A beginners guide”, https://careerfoundry.com/en/blog/web-development/programminglibrary-guide/ Accessed: 2023-05-24.

[22] React-UI, “ThemeProvider”, https://react-ui.dev/components/ThemeProvider Accessed: 2023-05-25.

[23] Matplotlib, “Sample Plots in Matplotlib” [picture], https://matplotlib.org/3.4.3/tutorials/introductory/sample_plots.html Accessed: 2023-05-25.

[24] Tutorialspoint, “Python Data Science – Matplotlib”, https://www.tutorialspoint.com/python_data_science/python_matplotlib.htm Accessed: 2023-05-25.

[25] Infinum, “Using Xcode Instruments”, https://infinum.com/handbook/qa/tools/using-xcode-instruments Accessed: 2023-05-25.

[26] Apple, ”Delegations”, https://developer.apple.com/library/archive/documentation/General/Conceptual/DevPedia-CocoaCore/Delegation.html Accessed: 2023-07-18.

[27] AWS, “Overview of Amazon Web Services”, https://docs.aws.amazon.com/whitepapers/latest/awsoverview/introduction.html?wpg_intro1 Published: 2023-09-28. Accessed: 2023-07-23

[28] AWS, ”What is Amazon S3?”, https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html#CoreConcepts Accessed: 2023-07-23.

[29] AWS, “What is the AWS Command Line Interface?”, https://docs.aws.amazon.com/cli/latest/userguide/cli-chapwelcome.html Accessed: 2023-07-23.

[30] AWS, “What is Amazon CloudFront?”, https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/Introduction.html Accessed: 2023-07-24.

[31] Mdn web docs, ”Web Workers API”, https://developer.mozilla.org/en-US/docs/Web/API/Web_Workers_API Accessed: 2023-08-07.

Appendix A: Source Code

Source Code for the PWA, https://github.com/test22131/An-analysis-and-comparison-of-the-Native-mobile-application-versus-the-Progressive-web-application-.

Source code for the native mobile application, https://github.com/test22131/Native-mobile-app.