**International Journal of Research Studies in Computer Science and Engineering (IJRSCSE)**
*Volume 8, Issue 1, 2021, PP 1-11*
*ISSN 2349-4840 (Print) & ISSN 2349-4859 (Online)*
*DOI: http://dx.doi.org/10.20431/2349-4859.0801001*
*www.arcjournals.org*

# Examining the Usage of Flutter to Design  a Yatch in 3D

**Mohammed Kpannah Fahnbulleh*, Xu SHUOBO**

*School of Information Science and Electric Engineering Shandong Jiaotong University, Jinan, China*

***Corresponding Authors:** **Mohammed Kpannah Fahnbulleh,** School of Information Science and Electric Engineering Shandong Jiaotong University, Jinan, China*

**Abstract:** *A mobile application has to be able to keep up with heavy demands to compete with all the new applications that are developed each day. Good performance and nice visuals are base requirements for the development of mobile applications. There are many options for tools when developing and one of these choices is a native application, which is said to have better performance and suitability to the mobile environment. Another choice is a tool which requires only one code base for multiple platforms and is therefore easier to maintain. Flutter is an open-source User Interface (UI) toolkit created by Google that can create cross-platform applications with one code base while said to maintain the aspects of looking native.*

*This paper examines how Flutter compares to native applications, which are currently seen as superior in mobile behaviour and performance. An experiment was conducted to test how Flutter as a cross-compiler compared to two native applications made of kotlin and Android studio and swift and XCode, in terms of CPU performance. A survey was created to see if there was a difference in the perception of users with regards to appearance and animations. A literature study was conducted to strengthen the results from the experiment and survey and to give a background to the subject. Flutter is a new tool and it continues to grow incredibly fast. Conclusions are drawn that a Flutter application can compete with a native application when it comes to CPU performance, but is not as developed in the animation area. Flutter does not require complex code for creating a simple application and uses significantly less lines of code in development compared to native. The final conclusion is that Flutter is best to use when building smaller to medium-sized applications, but has a potential to grow to overcome its current drawbacks in the animation department. Further examination of the areas examined in this paper is needed in order to ensure and strengthen the results.*

**Keywords:** *Flutter, Swift, Dart, Kotlin, Cross-platform, Native, Performance, Mobile Applications.*

## 1. INTRODUCTION

The purpose of this paper is to evaluate Flutter as a UI tool for creating mobile applications. A method that is used for creating mobile applications is by creating them for a specific platform from the beginning. Another method is to write a code base that can be compiled to several types of platforms. It is a method called cross-platform and is a popular method because of how flexible and fast a mobile application can be created. Choosing between native and cross-platforms is often a question about money and the proper way of developing. Flutter is an open-source UI toolkit that uses the language Dart [15] that can create mobile applications with a single code base and compile the code into both Android and iOS [11]. It was created by Google year 2018 and can according to themselves, create applications that inherits the same type of look, feel and performance as if they would have been developed as a native mobile application. This paper contains a study where Flutter is compared to native applications in different aspects such as performance of the CPU, visuals, code complexity and code needed to perform its designated task.

The study in this paper was done in collaboration with the Company Consider. They are a company with multiple ones throughout China and work with consulting in the software engineering field, developing products such as web applications, mobile applications and other software. Their customers normally choose native applications and often have strong opinions of which technique and solution they want from the beginning. By researching Flutter, consider will gain more knowledge of Flutter to be able to give a better motivation to if it could be a good candidate for their customers or not. To be confident in their recommendations, consider requested to see the performance results from the comparison between native and Flutter. They also requested the look and feel to be compared together with the code structure and amount of code that it takes to create a Flutter application. There will be a creation of 4 applications with the help of 3 programming languages (Kotlin, Dart and Swift). Flutter with Dart as the programming language, will produce two of these applications. The other two applications will be made with Kotlin and Android Studio as well as Swift and XCode.

### 1.1 Background

Despite its recent release, Flutter has a lot of talk around it and many praises the tool for how good it is even though no one seems to actually use it in action. Even though many praises the tools qualities, there are still little knowledge on where it fits in between the native development and the cross-platform tools. A motive for this paper is to understand if Flutter is a good candidate as a cross-compiler, how close it comes to competing with native in CPU performance and if it performs better, worse or the same as a native application. Where can a developer set the preference whether to use Flutter over native [6]? This is going to be measured by a survey, literature study and an experiment where the results will be weighted together when creating an opinion about how close Flutter comes to competing with native

This study will develop deeper into how Flutter works as a tool for creating mobile applications. Furthermore, the study will answer the research questions to bring value for Consider and for their customers. This is for them to be able to choose new alternatives to native applications. By researching the performance, it gives Consider an understanding of how or if Flutter [1] can be useful to their customers. Studying the look and feel will in addition to performance metrics, give an insight of how well Flutter can perform and be presented when discussing alternatives to native applications. The structure and amount of code together with the look and feel result, will reveal if Flutter can keep up its small code base while maintaining a closeness to the same visuals as a native application as well as perform equally or better to a native application.

## 2. RELATED WORK

Related work that has been found for the subject revolving Flutter and comparisons with Native are close to none except for an article by Coninck that was based on a published paper which could not be found. There are however, many academically papers on performance differences between cross-compilers. One of these papers [3] discusses the differences between cross-platforms. The papers results showed that cross-platforms were better for shorter time and budget. Native were preferred when interacting with the phone’s integrated system.

[3] writes about a study made on performance for Phone Gap versus native. The result showed that the cross-platform applications were slower than native in 7/8 performance tests. However, this performance study was specifically targeted on android devices. A paper written by [3], brings up the differences in cross-platforms and native by researching the platform, development environment and code base. Amatyas conclusion was that native is a good fit for heavier applications but is not always the most suitable choice for all applications because of it being cost heavy and more time consuming than cross-compilers.

As seen in earlier mentioned related work papers, many uses React Native or Phone Gap for comparing with native. Guerra carried out a comparative study on cross-platform frameworks and native. In the study, he measured the execution time amongst other metrics and the results showed that Flutter had a significantly faster execution time than the other cross-platforms which were React Native and Ionic. This can show to be useful for this paper’s study since the earlier performance results are based upon React Native [2], but is not an official publication and the original paper could not be found.

The study will look at the aspects of Flutter and how it performs, structures code and manages looks of the application. This involves the CPU usage and the grade of confirmability to native behavior. This report will not look at other metrics regarding performance. Run time CPU usage were the metric that Consider specified for the study. They regarded the metric as the one that would be the most interesting when using and developing a mobile application. Build Process CPU usage was not researched because Consider did not mention this as an area that they wanted to explore further.

## 3. Focus

### 3.1 Research Questions

• RQ1: How does Flutter differentiate in code size and how complex code is needed in comparison to native built applications?

• RQ2: How well does Flutter perform in run time CPU usage compared to native applications on Android and iOS?

• RQ3: How much does Flutter and android native differentiate in terms of look and feel for the application users?

### 3.2 Research questions explained

RQ1: How does Flutter differentiate in code size and how complex code is needed in comparison to native built applications? Code size and code complexity is two things that is important for the development of a mobile application. This question will help to understand the development differences that there are in code size and complexity. This is done to create an understanding of how easy the code base languages are to learn and how much code is needed to obtain the wanted results.

RQ2: How well does Flutter perform in run time CPU usage compared to native applications on Android and iOS? This question will investigate if Flutter can perform at the same level of CPU performance as native applications. CPU performance is currently one of the reasons why native is considered a better option to other solutions.

RQ3: How much does Flutter and android native differentiate in terms of look and feel for the application users? The look and feel of an application is the first thing a user experiences when using an application. This question will delve deeper into how much difference there is between a Flutter and a native application in terms of looks and feel. Since both Flutter [9] and Android is developed by Google, it would have been a better option to compare iOS to Flutter [1] [4] [12] [8][14]. Instead, Android was chosen because of unpredictable events regarding the iOS technology that was to be used.

## 4. Method

To find the results for the three research questions and the under-laying support for this paper, three different methods were carried out. A literature study was conducted to give an understanding to the author and the reader how Flutter works. It contains information on how the three tools and languages that was used in the experiment, works in development. The research that was found in the literature study acted as a support for methods used in the experiment.

To answer RQ2, an experiment was used. The experiment included four applications that were made in Dart [15] with Flutter, Kotlin with Android [11] Studio and Swift with XCode. Two of them are made of the single Flutter and Dart code base. It was carried out to find out how Flutter compares in the areas of CPU usage compared to native applications. RQ1 was answered by studying the authors' code review for the development of the experiment applications. It was done after the development to ensure that the code bases were complete before examining. Development time was measured by taking the time that each of the applications took from that of generating the starting layout to compiling and running the final version. Measurements of lines, files, size and application size were taken from the finished projects and applications. A survey was created and sent out to answer RQ3. It had questions revolving look and feel between the two earlier created android applications. This was done to see if the survey takers could notice any difference between them and to see how much difference it would make for them as users.

### 4.1 Literature Study

The literature study for this paper was executed to give the reader a better understanding of the cross-platform and native applications. Studying how Swift/iOS, Kotlin/Android and Dart/Flutter handles development and compiling. It contains earlier studies on cross-platform and native comparisons. The peer reviewed publications that were used for this paper comes from the databases: Google Scholar, Diva, IEEE and BTH Summon. The following keywords and sentences were used:

• Cross-platform vs. native

- Google Flutter
- Cross-platform performance
- Native
- Cross-platform tools
- Mobile application performance

Another search word that was used in the search for material on performance was" Phone Gap performance". By looking at the earlier related work, the majority of studies found on cross platform comparisons uses Phone Gap which makes it a relevant search word for finding comparative studies. The tools official pages were used for primary information about documentation and code standards. Online technology articles were used to find general information on differences between the ways of creating applications. Multiple articles were used to ensure a strengthening of the information that came from online articles. With Flutter being fairly new, it was hard to find a good amount of publications. This is why most of the studies and articles specifically about Flutter are taken from non-peer reviewed publications. However, the priority was still put on finding peer reviewed publications in the first place. Information on cross-platforms versus native and books about Flutter was taken from the earlier

mentioned databases. Snowball sampling was used on the first scholarly database results that showed up from searching with the search words. All of the paper types that was found in the scholarly databases were used to conduct the literature study.

### 4.2 Experiment

In this report an experiment was carried out to answer RQ1 and RQ2. This section presents the experiment and how it was prepared, its goals and the process itself. It is done to give the reader a better understanding of how the experiment was executed and planned. Due to outside influence, the method had to be changed significantly from the prior method of this thesis work. In summary, promised software that were to be used in the project was not delivered. Therefore, adjustments had to be made that changed the method and execution significantly. This will be further analysis in the analysis section of the report. Battery consumption is an important metric for mobile applications [11] CPU usage. The measurement of battery consumption was not possible due to the measuring tools needing a connection through USB cord which made the mobile phones charge automatically while performing the tests. Four applications were made with Flutter (which creates two applications), Kotlin and Swift [10][13]. The applications were created to look like each other and with code that followed according to each documentation. A relatively simple layout was chosen because of the time constraint of the project and to be able to handle complex code. The application layout consisted of a navigation bar and two-page views" Dashboard" and" Notification". Dashboard were filled with a picture of a salad and had the headline" Spring Salad Recipe" while notification had a simple list of labels with the word" Notification". Each view had an app bar at the top with the view name.

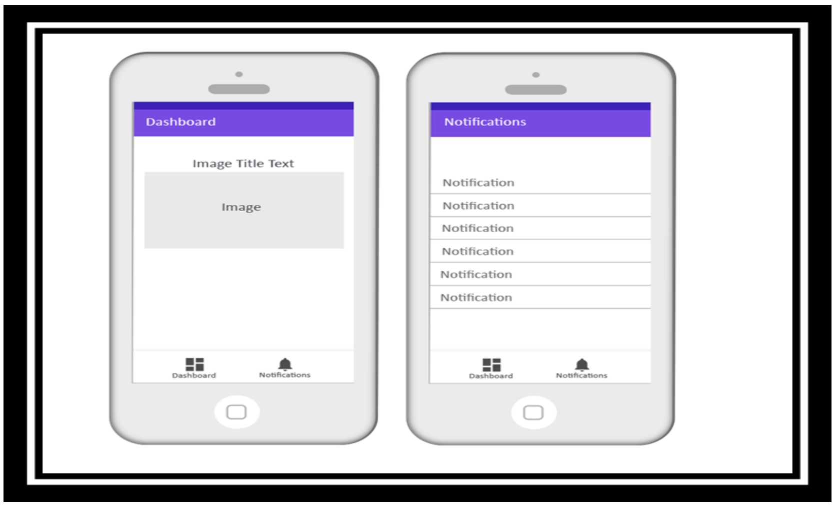


**Figure1.** *Example application layout appearance*

For this system to be able to represent a real-life system, the UI and flow of the application were inspired by applications such as: Tasty application and the Notification history application. Navigation bars exist in most of the applications today that has content and is not dependent on the amount of items that exist in the bar itself, but the actual function and look of the navigation bar. The example application was inspired from the" Tasty" application where the same look of the navigation bar exists as well as the recipe part of the example application. A similar list view to the one that is used in the example application design, can be found in the" Notification history" application.

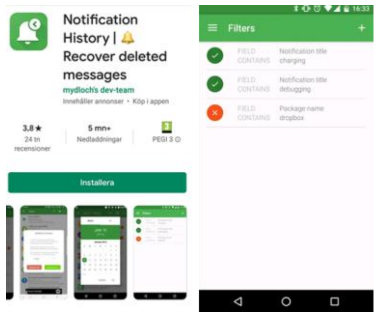


**Figure2.** *Notification History application*

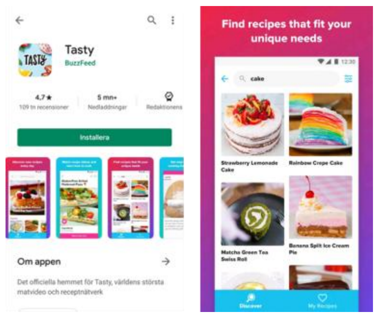

Tasty
BuzzFeed
Installera
Om appen
Find recipes that fit your unique needs
Strawberry Lemonade Cake
Rainbow Crepe Cake
Matcha Green Tea Swiss Roll
Banana Split Ice Cream Pie
Discover
My Recipes

**Figure3.** *Tasty application*

*4.2.1 Experiment Goal*

The goal with this experiment was to find out the difference between the Flutter and the native applications in terms of run time CPU usage. Another area that was studied, was the amount of code that each development environment required to create the desired looks and functionality of the applications. This was a side-to-side measurement and gave an insight of how the native code bases compared to the Flutter code base.

*4.2.2 Execution*

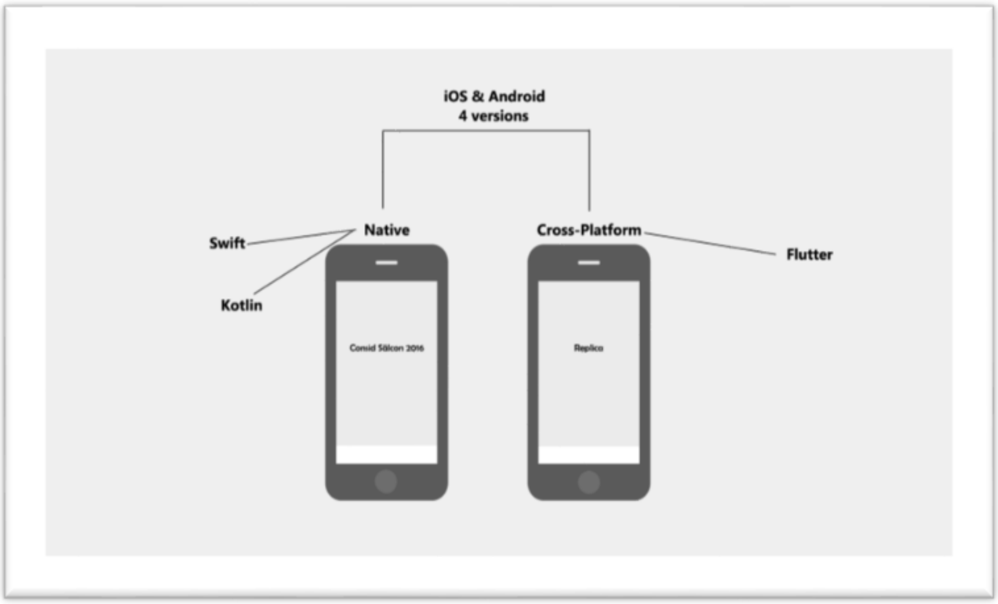


**Figure4.** *Graphic representation of the applications and their technology four applications in Flutter, kotlin and swift were created and built. Each development environment was setup accordingly to their documentation:*

- Flutter
- Android - Android Studio
- iOS - XCode

The CPU usage was measured three times manually per application build and the highest, lowest and mean values were written down. Values that were higher than was chosen for the lowest, as the CPU showed 0 when the applications were passive. The tools that were used to measure were: Android studio profiler (Android) and XCode instruments (iOS). Because the output from the measurements instruments only showed as graphs, sampling was used to determine mean value and ease measurement. Standard deviation was then calculated for each set of values. The collected values and mean of each test case were gathered in a summarized table. The author created the applications and measured development time without prior experience with application development. The first prototypes were used for the experiment. Code complexity was measured by the author through the code review.

Devices that were used for the experiment were one iPhone 7 and one Android Samsung S7. These mobiles were chosen to match each other's performance for more precise results and because of easy accessibility. Exact device specifications that were used for this experiment can be found in Appendix A under” Device specifications”. To create a base that would align to each language’s standards, terminal commands and templates were used to generate a code base that contained a navigation and two connected views. This was done to follow the documentation as much as possible. Package and tools can be found in Appendix A under” Tools and packages”. Default margin was used when positioning parts such as images and lists. This means that the recommended margin was used for each development environment. The navigation route below was followed manually when measuring the CPU usage. It was done to generate a good amount of data of the applications functionality and to ensure all the test measurements to follow the same guidelines.

1. Start the application
2. Wait for Dashboard view to load
3. Navigate to Notifications view
4. Navigate back to Dashboard view
5. Navigate to Notifications view
6. Scroll up and down in list 4 times
7. Return to the Dashboard view

#### *4.2.3 Preparation of the Devices for testing*

The devices were prepared by doing the following before the measures:

- Fully charging the battery
- Activating Flight mode
- Clearing the processes by restarting the phone
- Restarting the application before each measurement

#### *4.2.4 Hypothesis*

The hypothesis is that Flutter performs equal to or better than a native based application in this particular study. This theory is based upon the earlier study made by Coninck where an experiment where Flutter was compared to native android, Xamarin forms, native iOS and react native which showed that Flutter had a lower CPU usage. Taking a look at studies made between native and cross-platforming without specifying Flutter, the result shows that the cross-platforms are to be preferred because of their many advantages in development speed and that the difference in performance are small.

### 4.3 Survey

To answer RQ3, a survey was created and sent out to people who worked or had been educated in the software industry. Two of the applications android native and android flutter were compared without the participants knowing which one was android native or android Flutter. The look and feel of the application was measured by the survey takers answering questions with a 1 to 5 scale where 1 could be “I don’t use a phone” and 5 would then be”I use a phone everyday” together with yes/no questions as well as input fields for explanations in which the user had to explain the choices based on the looks, behavior and animations of the applications. This type of answering was chosen because it produces easier measured answers than if the answers would have been in free form.

### 5.1 Kotlin

Kotlin is a programming language that is made by Jet brains [10]. It is open source software statically typed language. The result when doing so would be a compiling error. Kotlin is fully compatible with Java and supported on the Android platform for creating Android applications.

#### *5.1.1 Kotlin compiling*

According to the documentation, Kotlin compiles into compatible by the code for Java when targeted on the JVM [10]. Kotlin is a versatile programming language that aims to be able to function on multiple platforms. If targeted to native, Kotlin will go through the LLVM to produce specific code for the platform in question.

#### *5.1.2 Kotlin/Android UI management*

When designing an UI in Android together with Kotlin, the developer is inclined to use the Android view group system [10]. The layout and UI elements can either be declared in XML or in code at run time. Drag and drop is available for the non-programmable layout creation. Both of these options need to have the UI parts connect to the code and will need to be initiated upon creation in the on Create method for the class layout. The objects in the layout is referred to as view Groups and views. The views are called widgets which are can be a text object or an image. These are encapsulated by the view Groups that are the layouts.

### 5.2 Swift

Swift is a programming language created by Apple Inc 2014 [12]. It was made to be able to work on multiple of platforms and aims to be a replacement to the C-languages. Swift is managed as a group of projects where it is split into: swift compiler, standard library, core libraries, LLDB library, swift package manager and XCode playground support.

#### *5.2.1 Swift compiling*

Swift code is compiled down to machine code with the help of seven level steps in the compiler [13]. The parser is ran at the beginning to check if there is any grammatical errors and show warnings. After the parser, a semantic analysis is carried out to see if it's safe to compile the code without errors. Clang importer then imports clang modules that can be referred from the analyzers. This is followed by something called a SIL generation and SIL guaranteed transformations. The first SIL runs another analysis on the code to improve optimization. The second SIL do data flow diagnostics to check so there are no uninitialized variables and results in a canonical SIL, meaning that it is a SIL that exists after the optimization and analyses has been done. There is an additional SIL step that is called the SIL Optimizations where extra high-level swift optimizations in additions to the ones before, are carried out. At the end there is an LLVM IR Generation which means that SIL is transformed into machine level code with the help of LLVM, which are compiler tools.

#### **5.2.2 XCode UI management**

When developing a UI in XCode, the Interface Builder is used to create story-boards. Scenes are used in storyboard to represent what is seen and what is happening on the device screen. Segues connects the scenes and holds upholds their relation. Scene objects can be dragged and dropped to create a new item to view. These items and controllers can be connected to code manually or with the help of XCode assistant that can create or generate code automatically.

## 5. RESULTS

This section displays the results that were discovered through the experiment and literature review. Pictures of the final applications are found below.

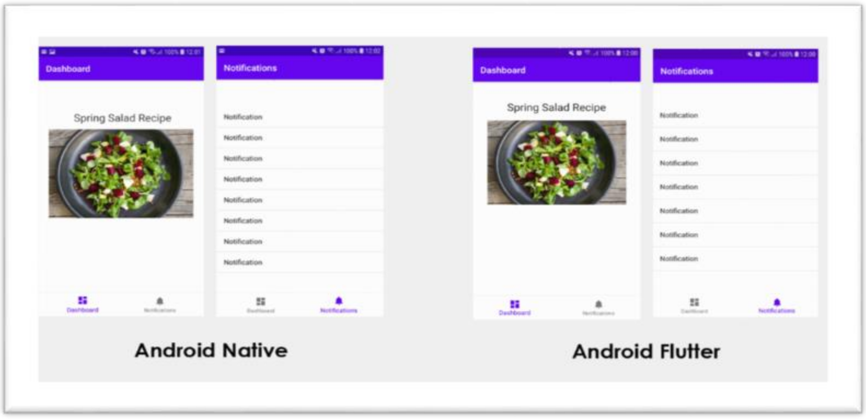
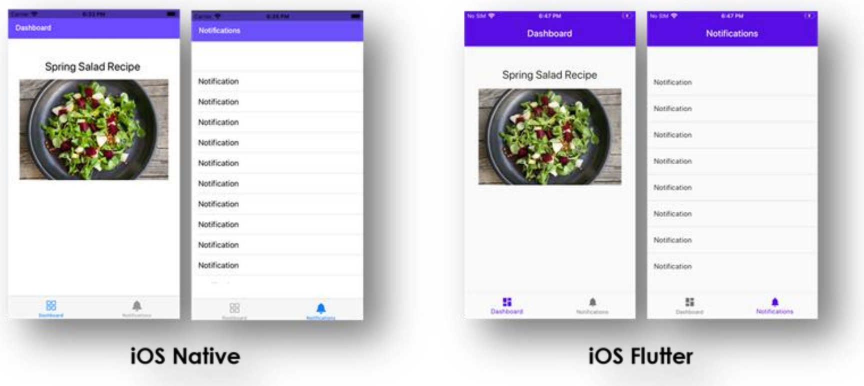
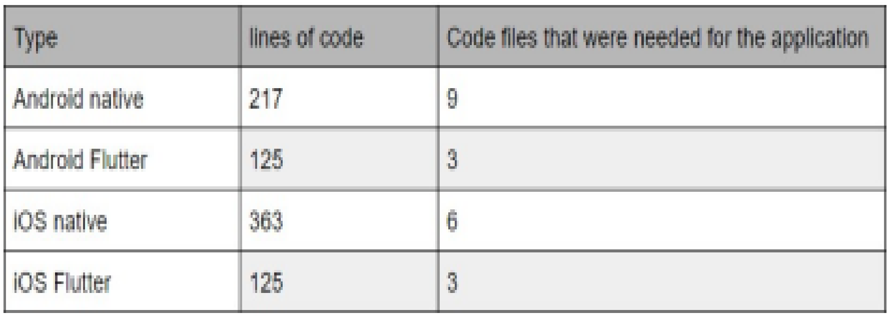


**Figure6.** *Images of final android native and android Flutter applications*



## 6. Flutter Differentiate in Code Size and How Complex Code Is Needed in Comparison to Native Built Applications

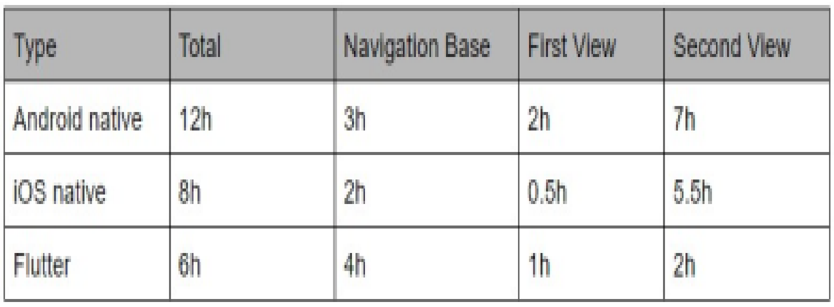
Looking at the results as a whole, Flutter wins the majority of most categories in the development area. There are however some differences that are interesting to take note of when comparing Flutter to native builds



| Type | lines of code | Code files that were needed for the application |
|---|---|---|
| Android native | 217 | 9 |
| Android Flutter | 125 | 3 |
| iOS native | 363 | 6 |
| iOS Flutter | 125 | 3 |

.

**Figure6.1.** Application lines of code and file count



| Type | Total | Navigation Base | First View | Second View |
|---|---|---|---|---|
| Android native | 12h | 3h | 2h | 7h |
| iOS native | 8h | 2h | 0.5h | 5.5h |
| Flutter | 6h | 4h | 1h | 2h |

Figure 6.1: Development time of each code base

### 6.1.1 Code size

As seen in figure 8, Flutter had the lowest amount of code lines and files that were needed in order to create the application. Native iOS had the lowest size of project files and app size but a significantly larger amount of code lines than the other builds. The native android had the most amount of files created and required lower amount of code lines than the iOS native.

### 6.1.2 Code Complexity

A part of answering Q3 is comparing the code complexity of the development code of each of the applications. The part of the application that was chosen for this was the creating of the notification list. It was chosen in particular because it contained the most code that was written and because the other view only featured an image and a title. The code that is shown in this section is only a part of the application code bases.

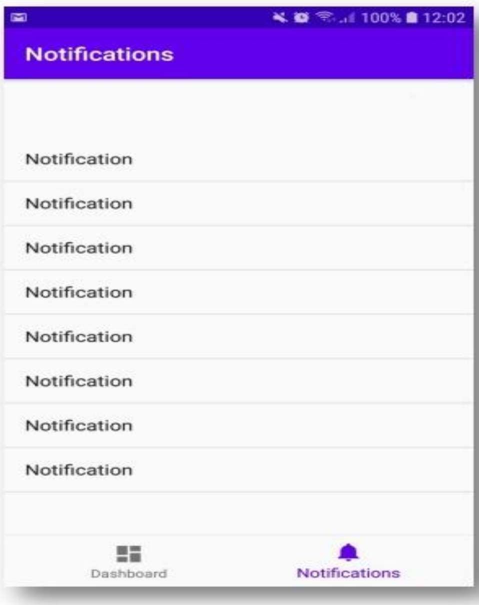
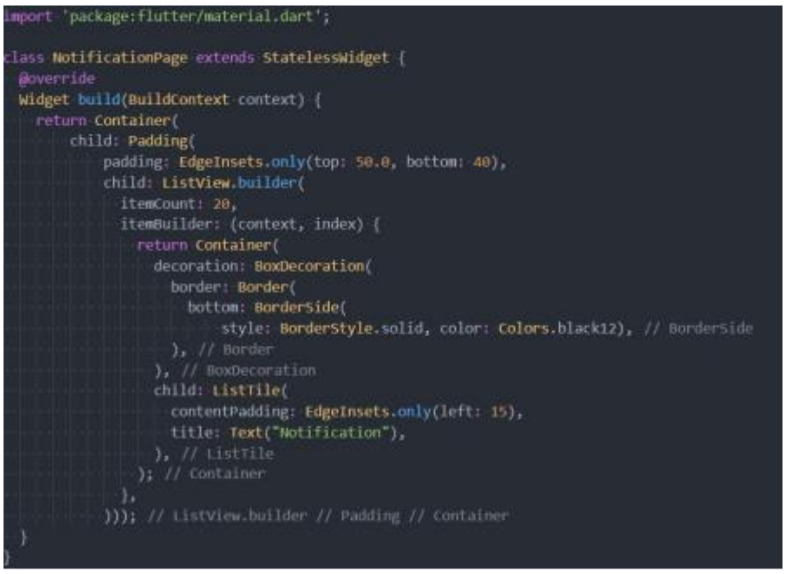


Figure 6.2: Picture showing notification list that was chosen for the code complexity part

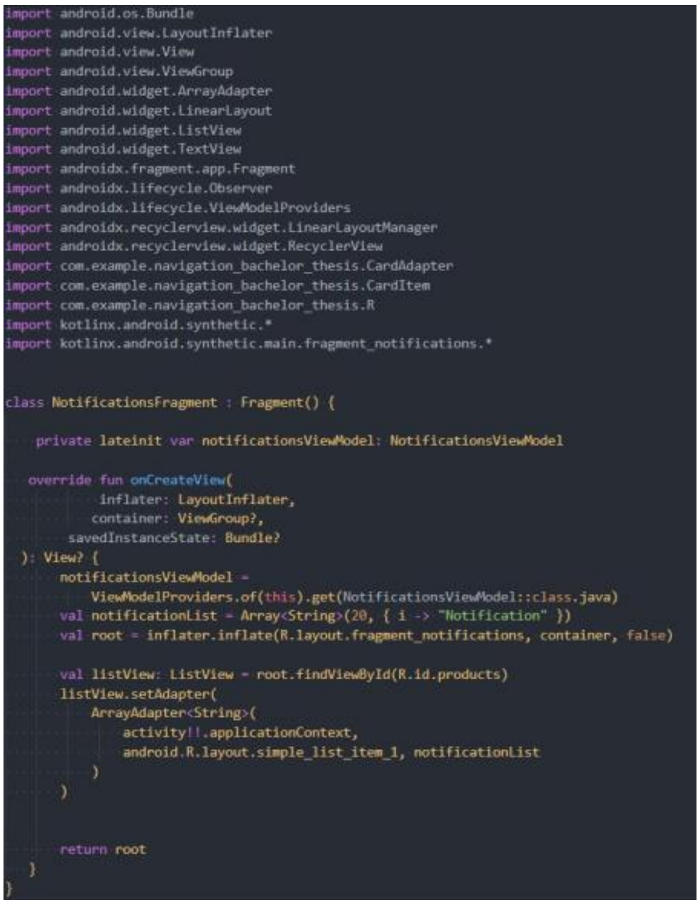


```
import 'package:flutter/material.dart';

class NotificationPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Container(
        child: Padding(
            padding: EdgeInsets.only(top: 50.0, bottom: 40),
            child: ListView.builder(
              itemCount: 20,
              itemBuilder: (context, index) {
                return Container(
                  decoration: BoxDecoration(
                    border: Border(
                      bottom: BorderSide(
                          style: BorderStyle.solid, color: Colors.black12), // BorderSide
                    ), // Border
                  ), // BoxDecoration
                  child: ListTile(
                    contentPadding: EdgeInsets.only(left: 15),
                    title: Text("Notification"),
                  ), // ListTile
                ); // Container
              },
            ))); // ListView.builder // Padding // Container
  }
}
```

Figure 6.2: Flutter code snippet showing the creation of notification list

Figure 6.2 shows how the Flutter code creates the visual layout and functional code in the same code. In the code image, there is a child parent relationship for the widget elements. The code shows that there is a widget that is built to return nested widgets. The child to the main containers padding, shows the creation of a List View Flutter widget. This view returns a List

View with 20 items and 20 container items.

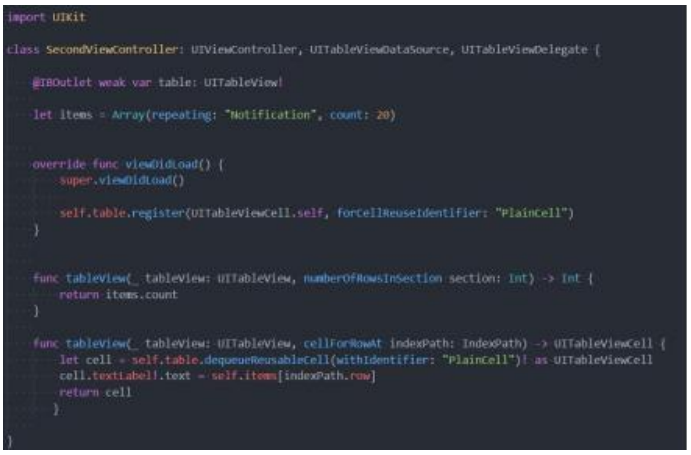


```kotlin
import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.ArrayAdapter
import android.widget.LinearLayout
import android.widget.ListView
import android.widget.TextView
import androidx.fragment.app.Fragment
import androidx.lifecycle.Observer
import androidx.lifecycle.ViewModelProviders
import androidx.recyclerview.widget.LinearLayoutManager
import androidx.recyclerview.widget.RecyclerView
import com.example.navigation_bachelor_thesis.CardAdapter
import com.example.navigation_bachelor_thesis.CardItem
import com.example.navigation_bachelor_thesis.R
import kotlinx.android.synthetic.*
import kotlinx.android.synthetic.main.fragment_notifications.*


class NotificationsFragment : Fragment() {

    private lateinit var notificationsViewModel: NotificationsViewModel

   override fun onCreateView(
            inflater: LayoutInflater,
           container: ViewGroup?,
        savedInstanceState: Bundle?
  ): View? {
        notificationsViewModel =
            ViewModelProviders.of(this).get(NotificationsViewModel::class.java)
        val notificationList = Array<String>(20, { i -> "Notification" })
        val root = inflater.inflate(R.layout.fragment_notifications, container, false)

        val listView: ListView = root.findViewById(R.id.products)
        listView.setAdapter(
            ArrayAdapter<String>(
                activity!!.applicationContext,
                android.R.layout.simple_list_item_1, notificationList
            )
        )


        return root
    }
}
```

Figure 6.3: Android native code snippet showing the creation of notification list

In figure 12, the android Kotlin application code declares the view model and import necessary items for the code. Everything happens in the on Create View faction, which in flates the XML layout for the notification list and creates an array with the same strings ”Notification” that it injects as data to the already existing List View XML target with the id ”products”.

The android native code has a lot of environment specific variables that a developer has to take in for consideration.



```swift
import UIKit

class SecondViewController: UIViewController, UITableViewDataSource, UITableViewDelegate {

    @IBOutlet weak var table: UITableView!

    let items = Array(repeating: "Notification", count: 20)


    override func viewDidLoad() {
        super.viewDidLoad()

        self.table.register(UITableViewCell.self, forCellReuseIdentifier: "PlainCell")
    }


    func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
        return items.count
    }

    func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {
        let cell = self.table.dequeueReusableCell(withIdentifier: "PlainCell")! as UITableViewCell
        cell.textLabel!.text = self.items[indexPath.row]
        return cell
       }

}
```

Figure 6.4: Image of iOS code

As shown in figure 6.4, the iOS swift fetches the table from the corresponding story file and creates an array within, which it registers a table cell(item) with the name” Plain Cell”. The Table View function checks the number of items in the created array” items” and returns a cell to the Table View with the text from the array. All this happens in the class for the specific controller in which the notification view exists. This code is relatively short for its purpose and it is easily structured for a beginner in mobile development. There are language specific parts of the code but otherwise it could probably be understood by someone who have knowledge in other languages.

## 7. Conclusion

Flutter is a useful toolkit that enables easy ways of creating new applications. It has gotten more and more popular recently and is talked about in the application development industry as a possible replacement of React Native and how it can be compared with native applications. The experiment of this report revealed that there is a small difference between the CPU performance of Flutter iOS and native Swift iOS respective Flutter android and flutter native. There seem to be a difference between the performance of iOS and android but when it comes to how well Flutter can perform in CPU usage compared to native applications, there is barely a difference. The summery for this is that Flutter can perform up to par with a native application for the type of application size that was tested. To verify these results and determine that Flutter can keep up with native, further testing needs to be carried out.

## References

[1] Google. (2019). Google flutter, [Online]. Available: https://flutter. dev/ (visited on 12/03/2019).

[2] B. D. Coninck. (2019). Flutter versus other development frameworks: A ui and performance experiment, [Online]. Available: https : / / blog .codemagic . io / flutter - vs - ios - android - reactnative - xamarin/ (visited on 02/06/2020).

[3] M. Palmieri, I. Singh, and A. Cicchetti, "Comparison of cross-platform mobile development tools," 2012 16th International Conference on Intelligence in Next Generation Networks, Researchgate.net, 2012, p. 8.

[4] G. Flutter. (2020). Flutter setup, [Online]. Available: https://flutter.dev/docs/get-started/install (visited on 04/30/2020).

[5] Android. (2020). Android studio setup, [Online]. Available: https: / /developer.android.com/studio (visited on 04/30/2020).

[6] M. Bellinaso. (2018). Flutter vs react native for iOS development, [Online].Available: https://medium.com/asos-techblog/flutter-vs-react-

[8] G. Flutter. (2020). Faq flutter, [Online]. Available: https://flutter.dev/docs/resources/faq (visited on 02/13/2020).

[9] C. Software. (2019). What is flutter? here is everything you should know, [Online]. Available: https://medium.com/@concisesoftware/what- is-flutter-here-is-everything-you-should-know-faed3836253f

(visited on 03/05/2020).

[10] Jetbrains. (2020). Kotlin-faq, [Online]. Available: https://kotlinlang. org/docs/reference/faq.html (visited on 03/11/2020).

[11] G. Developers. (2020). Android layout, [Online]. Available: https : / /developer . android . com / guide / topics / ui / declaring - layout # kotlin (visited on 05/07/2020).

[12] A. Inc. (2020). Swift, [Online]. Available: https://swift.org/about/ (visited on 05/07/2020).

[13] A. Inc. (2020). Swift compiling, [Online]. Available: https://swift.org/ swift-compiler/#compiler-architecture (visited on 05/06/2020).

[14] G. Flutter. (2020). Google flutter animations, [Online]. Available: https: //flutter.dev/docs/development/ui/animations (visited on 03/12/2020).

[15] Google. (2020). Dart, [Online]. Available: https://dart.dev/ (visited on 06/05/2020).

**Citation:** *Mohammed Kpannah Fahnbulleh & Xu SHUOBO " Examining the Usage of Flutter to Design a Yatch in 3D ", International Journal of Research Studies in Computer Science and Engineering (IJRSCSE), vol. 8, no. 1, pp. 1-1, 2021. Available: DOI: http://dx.doi.org/ 10.20431/ 2454-9428.0801001*