Deanship of Graduate Studies

Al-Quds University

Comparison of Native and Cross-Platform Mobile Development Tools for Android Mobile Application

Khalid Waleed Izzat Zohud

M.Sc. Thesis

Jerusalem -Palestine

2024/1446

Comparison of Native and Cross-Platform Mobile Development Tools for Android Mobile Application

Prepared by :

Khalid Waleed Izzat Zohud

Computer Science Department – Al-Quds University, Palestine

Supervisor: Dr. Rashid Jayousi

A Thesis Submitted in Partial Fulfillment of Requirements for the Degree of Master in Computer Science / Department of Computer Science / Faculty of Graduate Students / Deanship of Graduate Studies / Al-Quds University.

2024/1446

Al-Quds University

Deanship of Graduate Studies

Department of Computer Science

Masters in Computer Science

Thesis Approval

Comparison of Native and Cross-Platform Mobile Development Tools for Android Mobile Application

Prepared by: Khalid Waleed Izzat Zohud

Registration No.: 21812285

Supervisor: Dr. Rashid Jayousi

Master thesis submitted and accepted, Date: **21/8/2024**

1- Head of Committee: Dr. Rashid Jayousi Signature ……R. Jayousi……

2- Internal Examiner: Dr. Radwan Qasrawi Signature ……

3- External Examiner: Dr. Muath Sabha Signature ……Muath Sabha……

Jerusalem - Palestine

2024/1446

Declaration

I certify that this thesis submitted for the degree of Master in Computer Science is the result of my own research, except where otherwise acknowledged, and that this thesis (or any part of the same) has not been submitted for a higher degree to any other university or institution.

Khalid Waleed Izzat Zohud

Date : 21/8/2024

Signature : 

Dedication

قال تعالى (يَرْفَعِ اللَّهُ الَّذِينَ آمَنُوا مِنكُمْ وَالَّذِينَ أُوتُوا الْعِلْمَ دَرَجَاتٍ) (المجادلة: 11).

To my father Waleed Zohud, whose unwavering support, wisdom, and belief in me have been my guiding light throughout this journey. And to my dear mother, Lubna , you have always offered emotional, spiritual, and moral support at every step of this academic journey, for her boundless love, patience, and encouragement, which have been my greatest source of strength, for my brothers, sisters and my wife who always believe in me.

Thank you for always being there for me.

Acknowledgment

At the beginning, all thanks and praise be to ALLAH, with his generosity, I finished this thesis . A deep appreciation and gratitude to my project supervisor Dr. Rashid Jayousi who dedicated himself to the successful completion of this dissertation.

I would like to thank everyone who helped and guided me throughout my research. Who, Without his guidance, patience and support, this thesis would not have been possible. I also owe a lot of thanks to my family .

Thanks to friends and colleagues.

I extend my sincere thanks to the Deanship of the College of Science and Technology at my dear university for their help and cooperation

Khalid Waleed Izzat Zohud

Abstract

The main objective of this study is to compare the different programming languages used in creating smartphone applications for Android phones in a native or cross-platform format.

This study examines the performance of five different programming languages by creating an application that runs the same tasks in each programming language. The comparison was based on the amount of time spent on each executed task.

This study uses an experimental approach to explore and discuss the approaches and applications of cross-platform application development. Therefore, a sample project (Test Bench) was implemented with a native framework android studio with Kotlin programming language, followed by four cross-platform frameworks visual code with React Native and Cordova framework, android studio with Flutter farmwork and visual studio with Xamarin framework. The collection and recording of information and analysis were for efficiency, workload, software procedures, and performance by recording the time of tasking on the program that was created for this purpose, where the researcher recorded the advantages and disadvantages of each framework, and eventually compare the five frameworks (native and four cross-platform languages).

The results of this study shows that every cross-platform, such as Xamarin, has its own advantages and disadvantages. It has the best result for image capturing and compressed files, but it came to the end regarding the time in read/write storage. Xamarin is the best choice for c#.NET and desktop developers who want to start developing mobile applications. The result showed that React Native, Cordova is the best choice for web developers, Native frameworks like Kotlin and Flutter for desktop developer.

Table of Contents

List of Tables

List of Figures

List of Appendices

Definitions

List of Symbols and Terminology

Symbol:	**Definition**
OS:	Operating System
GUI:	Graphical User Interface
UI:	User Interface
API:	Application Programming Interface
SQL:	Structured Query Language
IDE:	Integrated Development Environment
PL:	Programming Language
VS:	Visual Studio
WYSIWYG:	What You See Is What You Get

Chapter 1
Introduction

1.1 Introduction

With today's cross-platform mobile applications (software applications that can be developed and run on multiple platforms such as Android, iOS, and Windows through a single codebase), the development process is made easier, making it ideal for developers of mid-sized and small companies (Sreekanth Dekkati, 2019). Creating an app once and deploying it on multiple platforms requires much less effort and costs even less than developing native apps for each platform separately.

The concept of cross-platform mobile application development has gained popularity in recent years because of the growing demand for mobile applications and the need for businesses to reach their target audiences on different platforms. With cross-platform development, developers can create apps that offer consistent user experience across multiple devices and platforms without the need to write separate codes for each platform.

Various tools and frameworks are available for cross-platform mobile app development, such as React Native uses React and JavaScript as programing languages, Xamarin uses C# as programing language, Cordova uses web languages like JavaScript, and Flutter uses dart programming languages. These tools make it easier for developers to create high-quality, native-like mobile applications that can operate seamlessly on different platforms.

Overall, cross-platform mobile app development offers an excellent solution for businesses and developers who want to save time and resources while still delivering high-quality mobile applications that meet the needs of their users.

This research will answer the following questions:

A. What are the key differences between the mobile application development methods?

B. What is the correct technology stack for the project? Cross-platform or native?

C. Which is the best or fastest programming framework for repetitive tasks?

1.2 Problem Statement and Research Questions

Developing the same app on multiple mobile platforms requires considerable time and resources to get done. This thesis has two main purposes. The main purpose is to provide an overview of the various technologies available to support the development of mobile applications for various platforms. The second aim is to compare the various methods that are available to develop these applications, especially comparison between Native and Cross-Platform (Xinogalos, 2013).

The Research Questions were as follows.

What are the key differences between the mobile application development methods?

What is the correct technology stack for the YOUR project? Cross-platform or native platform?

1.3 Hypothesis

Hypothesis: Compared to cross-platform development frameworks such as React Native (Sreekanth Dekkati, 2019), Flutter, Xamarin and Cordova (Top 25 Apache Cordova Interview Questions and Answers, 2023) native programming typically offers greater

performance and access to device-specific capabilities, but what comes after native programming in performance, line of code and application ABB size.

Rationale: Native development involves writing code only for each platform (iOS or Android) using tools and languages native to that platform (Java/Kotlin for Android, Swift/Objective-C for iOS), which may improve performance and provide direct access to platform APIs. Although cross-platform frameworks facilitate code reuse between platforms, they may impose performance overhead and restrict access to platform-specific features.

1.4 Research Motivation

The field of smartphone programming is considered the most widespread and in demand in the programming market in general, and the demand for the number of developers is expected to increase from 17% to 24% by 2026. (Kaushik, 2024), as the demand for programmers in this field is increasing.

A problem for individuals and companies appears to be that programming for phones is divided into three sections: Android, iOS, and cross-platform, which build programs for these two mobile operation systems (Anitha Periyasamy, 2019).

If the programmer chooses to program in one of the two operation systems, he/she will need full time to build for each application, but if he/she decides to build applications in the cross-platform, he/she will take half the time it takes to build an application on the two mobile operation systems, but the method of programming has advantages and disadvantages.

During this research, the researchers will find the difference between programming native and cross-platform programming in terms of the performance of applications with the phone in the field of hardware (camera, memory, storage, create file, compression, and

decompression), as it will show the strength of each programming language in dealing with the phone's hardware and which will have the best performance among the native and cross-programming languages to be compared (Desai, 2023).

1.5 Research Objective

This study aims to help developers to choose the best programming language that suits them and their field of work, as each test we conduct this study will be a specific field in the programming fields, where there will be the field of photography, and it will appear through the results that programming languages are faster in the field of photography, which will be faster in the field of compression and decompression, and faster in the field of memory storage.

If the application focuses on using the camera, photography, and the speed of opening and closing, then there will be a programming language that has the highest performance in this field. In the case of file storage and management, a programming language will have the fastest performance. (Bhoyate, 2024).

1.6 Research Questions

As mentioned previously, mobile application development has a huge demand and is evolving every day in each field. The main research questions that this thesis research work tried to answer:

What are the key differences between the mobile application development methods?

What is the correct technology stack for the YOUR project? Cross-platform or native?

Which is the best or fastest programming framework for repetitive tasks?

1.7 Methodology

To achieve these goals, we used an experimental methodology. For this purpose, we created a test beach application that was used to compare multiple cross-platform programming languages to perform different functions. Comparisons were made between the average times for each task. After beginning the tests and recording the data results, we ensured that no other application ran in the background. After starting the task and finishing it, a wait of 3 s was made before moving to the next task to ensure that there were no processes running in the background that could affect the results, such as sending the data to the cloud database.

1.8 List of Contribution

The main contribution of this thesis is the development of a test bench application for these native and cross-platform applications to test features on Android smartphones.

These are the sub contributions related to the main one:

- Develop a function to test read/write on storage with a microsecond counter.

- Develop a function to capture automatic image speed in a microsecond counter.

- Develop a function to test read/write files in storage at a microsecond counter.

- Develop a function to compress and decompress files at a speed.

1.9 Research Limitations

Throughout the execution of this research, several limitations could be highlighted. One of the major issues is many frameworks have regular updates and many things will change from release to another, the frameworks have different code structure causing difficulty to write same code in various frameworks, this will lead to writing different codes, which

will make us to write similar programming codes as possible, when it's not possible, we will build functions that do the same job as possible.

1.10 Theses Outline

The remainder of this thesis is structured as follows.

In addition to the introduction (Chapter 1), this thesis is divided into four chapters. They are listed below with a brief description about the content of each one of them.

Chapter 2: This chapter discusses and analyzes the relevant research on native and cross-platform programming (Background and Literature Review).

Chapter 3: This chapter details the Research Methodology, starting with data collection and preprocessing steps to discuss the developing code (research methodology).

Chapter 4: This chapter discusses the experimental testbed and experimental results with discussions and analysis (Experiment Methodology).

Chapter 5: This chapter concludes the main findings of this thesis and sheds light on future work for further improvements.

Chapter 2

Background and Literature Review

This chapter presents some previous work and research relevant to our work, which introduces work on design, programming methods, and their performance. It presents the statistics, classification algorithms, and the compression of these algorithms.

2.1 Background

Smartphones have become essential tools for modern living and play a critical role in education, healthcare, business, and social interactions. As smartphones continue to evolve, their importance in daily life is expected to increase.

Most of the users are confused to choose between the main two operation systems (Android, iOS) according to the Statista website (Sherif, 2024) 70.1% are Android users over the world, 29.2% for IOS, and the rest 0.6% for other mobile OS such as Samsung Tizen, Blackberry, Symbian windows phone they build with different architectures, frameworks, and programming languages.

If one wants to build an application that works on the two platforms (OS), one must build the applications in parallel on the two systems (using native programming language) it will cost us additional time, effort, and costs, and when the cross-platform has relieved it was efficient and productive (Xinogalos, 2013) (Keith Vassallo, Cross-Platform Development Frameworks, 2019).

After many years of improving and developing cross-platform frameworks (Khanna, 2024), dozens of platforms have specialized in cross application.

Cross platform mobile app development is useful for both starting entrepreneurs and big corporations. This is the case as this tool helps expand your clientele. This is suitable for a Minimum Viable Product (MVP) when you have prospects however you are short on finances and time. Due to the structures utilized in cross-platform application development, they can run on iOS as well as Android systems and all other operating system platforms such as Linux, windows, and Blackberry among others. Additionally, cross-platform frameworks are the most appropriate choice when looking for a straightforward mobile application without intricate functions and animations (Khanna, 2024).

Combining mobile consulting with cross-platform app development frameworks for enterprise solutions adds a level of proficiency to the cross-platform deployment and thus improves the performance of the applications across different platforms and enhances the satisfaction of the users. Additionally, mobile developers and designers assist with design, involvement into functionality, and optimization, further enhancing the benefits of cross-platform development and easing the deployment of unique business solutions (Khanna, 2024).

- Lower Cost and Resources

Cross-platform mobile app development is centered on the use of pre-existing codes and follows the agile mobile app development processes, which allows you to control the budget, quality, and development timelines nicely (Khanna, 2024).

- Enhance Exposure to Target Customers

Mobile applications developed employing cross-platform app development frameworks cannot be restricted to a few applications since the developed single app targets both Android and iOS platforms, thus widening your market base (Khanna, 2024).

- Easy Deployment & Maintenance

Rather than writing and maintaining multiple versions of an app, mobile app developers using cross-platform development techniques only have to write and maintain one source-

code base. This results in quick and simple development and easier app maintenance. Also, all upgrades are easily done since they are done once for all the platforms and this saves time and effort (Khanna, 2024).

- Consistent Design

Adopting a consistent user interface and user experience across multiple platforms enables all design quirks to remain true to that particular platform. Thus if users are accustomed to working with your application, they will not have difficulty using it on any of the platforms (Khanna, 2024).

- Integrative Sync to Cloud

Integrating cross-platform apps in the cloud is an easy task. The cross-platform application developers unite several extensions and plugins into the common source code, improving the capabilities and extensibility of your app (Khanna, 2024).

Research has suggested that React native, Xamarin, Cordova, and flutter are becoming widely popular because of their better performance and stability (Dongliang You, 2021) .

Before we get into the details for each framework, we provide a quick summary of the primary differences between these popular cross-platform app development frameworks:

Framework	Kotlin	Flutter	React native	Apache Cordova	Xamarin
Owned by	JetBrains	Google	Meta	Apache software foundation	Microsoft
Year	2011	2017	2015	2011	2011
Performance	Native-like	Native-like	Average due to the use of bridge	Average due to the use of web technology	Cross to Native-like
Dev-language	Kotlin	Dart	JavaScript	Web language (Html,css,js)	.net
Hardware API-support	Extensive	Extensive	Limited	Limited	Extensive
Community	Small but rapidly growing	Fair-sized but rapidly growing	Huge and growing	Huge and growing	Fair-sized but shrinking

Table 2.1: development Frameworks comparison

2.1.1 Kotlin

Kotlin is a modern, statically typed programming language that runs on the Java Virtual Machine (JVM) and compiles to JavaScript (Siddhi Sanjay Shinde, 2021) allowing it to be used for both server-side and client-side development. It was developed by JetBrains, the makers of IntelliJ IDEA, and released in 2011 as an open-source project under the license of Apache 2.0. Kotlin was designed to be concise, expressive, and safe, with a focus on interoperability with existing Java codes and libraries.

Kotlin offers several features that make it easier to write clean, concise, and maintainable codes. Some of these features include null safety, type inference, lambdas, extension functions, coroutines and data classes. Kotlin also strongly focuses on functional programming concepts, such as immutability and higher-order functions.

Kotlin has gained popularity in recent years and is now used by several companies and organizations, including Google, Netflix, Uber, and Pinterest. It is also the official language for Android app development, replacing Java in this role and becoming a new native language.

2.1.2 React native

The Act Native is an open-source framework for building mobile applications using JavaScript and React (Ancheta, 2020) It was developed by Facebook and first released in 2015 (Danielsson, 2016) (Khanna, 2024) React Native allows developers to create mobile applications that run natively on both Android and IOS platforms using a single codebase. This means that developers can write code once and deploy it on multiple platforms, thereby saving time and effort.

React Native uses a combination of JavaScript and a subset of CSS, called Flexbox, to create mobile user interfaces. It also provides access to platform-specific APIs such as cameras, contacts, and locations, which can be used in apps. React Native also supports hot reloading, which allows developers to see changes in an app in real time without having to rebuild the entire application.

The React Native has gained popularity among developers because of its ability to create high-quality, cross-platform mobile apps with a single codebase. It is used by several companies, including Facebook, Instagram, Airbnb, and Uber.

2.1.3 Cordova

Apache Cordova (formerly known as PhoneGap) is an open-source mobile application development framework. It allows developers to build mobile applications using web technologies such as HTML, CSS, and JavaScript, and then deploy them to various mobile platforms, including iOS, Android, and Windows Phone. (Malavolta, may-2016)

Cordova provides a set of APIs that allow developers to access device-specific features, such as the camera, accelerometer, contacts, and geolocation. These APIs are exposed through a set of JavaScript interfaces that can be accessed from within a web-based application.

Cordova works by wrapping the web-based application in a native shell, which allows it to run as a native app on the target platform. The native shell provides access to device-specific features and handles other native functionalities, such as notifications and storage (Merenych, 2023).

Cordova is often used in conjunction with other web technologies, such as AngularJS, React, and Ionic, to create hybrid mobile applications. Hybrid apps offer a number of benefits, such as cross-platform compatibility and a familiar development environment, and are often used by developers who want to create mobile apps quickly and efficiently.

Cordova is maintained by the Apache Software Foundation and has a large community of contributors. It is used by several companies, including Adobe, IBM, and Microsoft, and is a popular choice for developing mobile applications.

2.1.4 Flutter

Flutter is an open-source mobile application development framework created by Google (Shivanandhan, 2023). It helps developers build natively compiled applications for web, desktop, and mobile application platforms from a single codebase. Flutter was first introduced in 2017 and has gained popularity among developers owing to its fast development cycle, high-quality user interfaces, and ability to create cross-platform apps with a single codebase. (Arif Md. Sattar, 2023).

Flutter uses the Dart programming language, which was also created by Google, and provides a rich set of pre-built widgets that allow developers to create beautiful and responsive user interfaces (Khanna, 2024). Flutter also includes a hot reload feature that allows developers to see changes in the app in real time without having to rebuild the entire application.

Flutter also provides access to native APIs and platform-specific features such as cameras, geolocation, and storage through a set of plugins. This allows developers to create cross-platform applications that take full advantage of the capabilities of the target platform.

Flutter has gained popularity among developers because of its fast development cycle, high-quality user interfaces, and ability to create cross-platform apps with a single codebase. It is used by several companies, including Google, Alibaba, and Capital One, and is quickly becoming a popular choice for mobile application development.

2.1.5 Xamarin

Xamarin is a cross-platform mobile application development framework that allows developers to create native mobile applications by using C# and .NET framework. It was founded in 2011 and acquired by Microsoft in 2016 (Patel, 2024). Xamarin allows developers to create applications for the iOS, Android, and Windows platforms with a single codebase, which can save time and resources.

Xamarin provides access to platform-specific APIs and features such as cameras, geolocation, and contacts through a set of bindings (Khanna, 2024). These bindings provide

a way to call native APIs from within the C# code, which makes it possible to create high-quality native mobile applications that can take advantage of the unique features of each platform.

Xamarin also includes a set of prebuilt UI components and layouts that can be used to create native user interfaces for each platform. Developers can also use Xamarin. Forms: a UI toolkit that allows for the creation of cross-platform user interfaces using XAML, a markup language used by Microsoft's WPF, and Silverlight technologies.

Xamarin has gained popularity among developers owing to its ability to create high-quality cross-platform mobile applications using C# and .NET framework. It is used by several companies, including Microsoft, Kelloggs, and Alaska Airlines.

2.2 Literature Review

Many researchers make the right decision to approach this type of research, focusing on the comparison between native and cross-platform frameworks. In Chapter 2, we review some related works, present some approaches and methods provided in this literature, and introduce and explore how to create UI for mobile applications, build algorithms, and export ABB/APK applications.

Ahmad (2023) compared and analyzed the frameworks used for cross-platform mobile application development. This thesis compares six frameworks, namely Ionic, React Native, Native Script, Flutter, MAUI, and ReactJS, based on objective criteria. Experimental data were gathered by developing similar Android applications in each framework and conducting tests for CPU usage, memory usage, and application size. The data for the other criteria were obtained from official documentation. The results of this thesis provide a set of guidelines for developers looking to make the choice of framework, offering educated guidance within the defined problem space.

You and Hu (2021) explored and discussed the approaches and applications of cross-platform application development through an experimental methodology. Therefore, a

sample project was implemented with the native framework and then with three cross-platform frameworks: React Native, Flutter, and Xamarin. The data from each project were collected and analyzed in terms of functionality, workload, development procedure, and performance to determine their advantages and disadvantages. Finally, the study compared these three cross-platforms and also compared them to the native framework, and then drew a conclusion on which cross-platform framework is the best for mobile application development.

Vassallo and Garg (2019) discussed the contemporary background within which cross-platform technologies are being developed, full-stack web development using a MEAN stack, cross-platform mobile development methodologies, and web-based desktop application development.

Chapter 3

Experiment Design

This chapter contains three sections that address the methodology used in this study. It presents Data Collection, and Project Design, such as the architecture and UI design.

In Data collection, the researcher shows the tasks that were made and how the data were collected, in project design, shows application designs, and what it will look like.

3.1 Data Collection

The research developed a practical project (five applications), called the test bench, five mobile applications (Kotlin, Cordova, react native, Xamarin, and Flutter), and they performed the same task as read/write in the storage memory, took an automatic image from the camera and compressed/decompress file (60 MB) and stored task time to the cloud database; the time is the main measurement for this test, before starting the task timer will start and after the task is completed, it will stop, then the application calculates time in nanoseconds and then stored in the cloud database as microseconds (nanosecond *1000).

Type	Description
UI Design	How user friendly ,efficient is the design tools?
Functionality	Can the framework and do the required function ?
Performance	The performance of the application including size of the release package ?
workload	How many line of code where written in the project ?

Table 3.1 Plan of data collection

3.2 Project Design

The test Bench is a test application running on an Android mobile device that allows the user to perform tests (tasks). Features such as storage speed, camera speed, file test (read and write), compress file, and decompress speed. Where developed in the same application in five frameworks (see Figure 1), and the server side is the MYSQL cloud database to collect data from these applications. To ensure that the project can be tested under the same environment (software, hardware) on the user side, which was developed and tested under the same mobile phone and internet network.

Figure 3.1. workflow of mobile application architecture

Each application performed the same task storage test, photo, file test, and zip file test. The UI shows the result of the task time in microseconds, and the result is transferred to the cloud database via the PHP API.

An example of the workflow of the tasks using these mobile apps in Figure2. First, all of these applications require promotions to use mobile hardware, such as storage and cameras. Each of these applications must click on task buttons to work and collect data. The application then shows the resulting time in application Ui (labels) and sorts it into the cloud database.

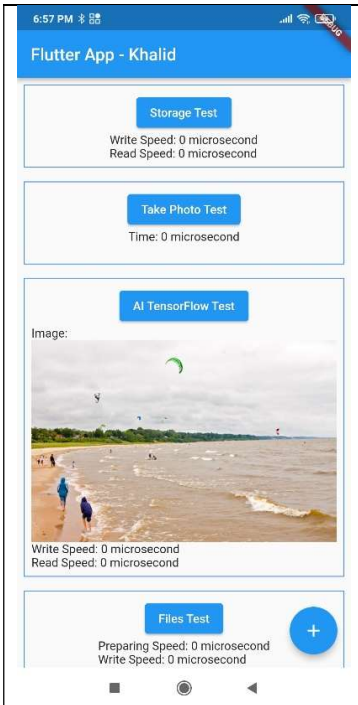
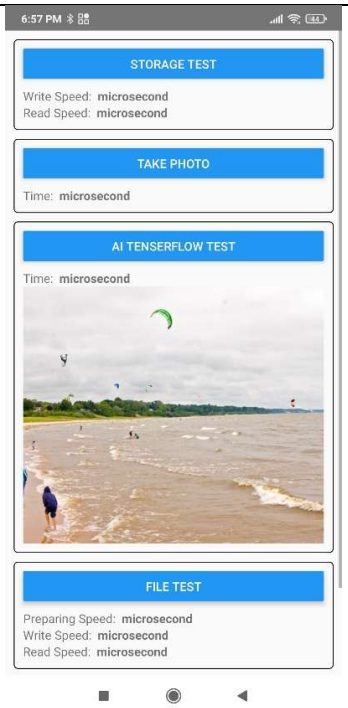
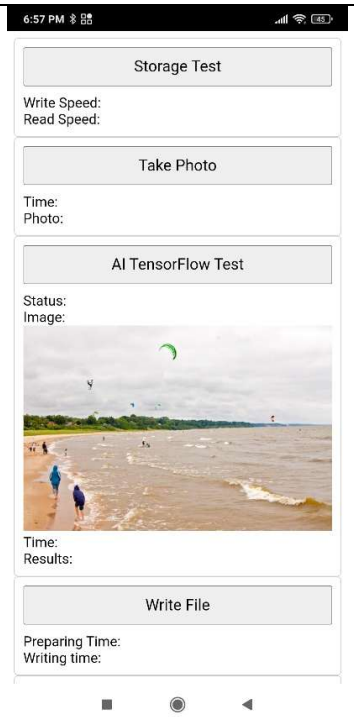
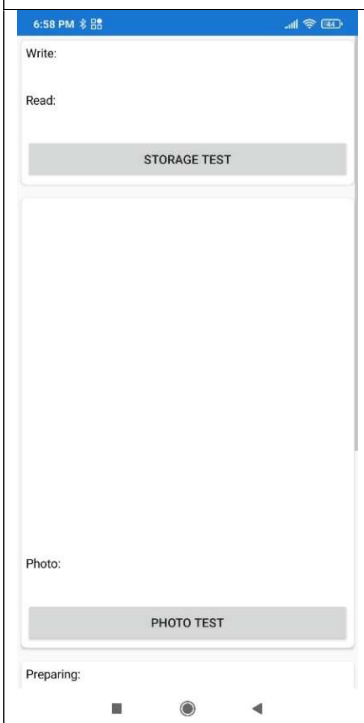
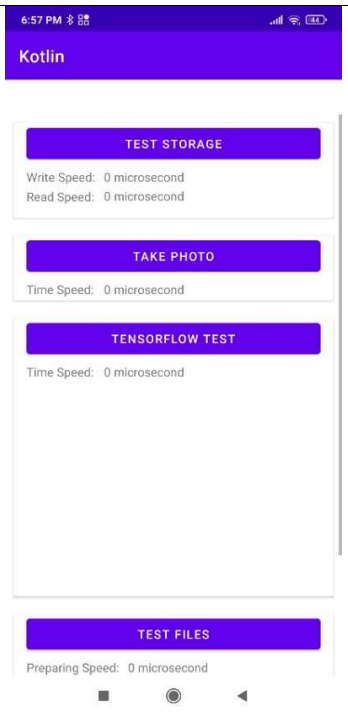
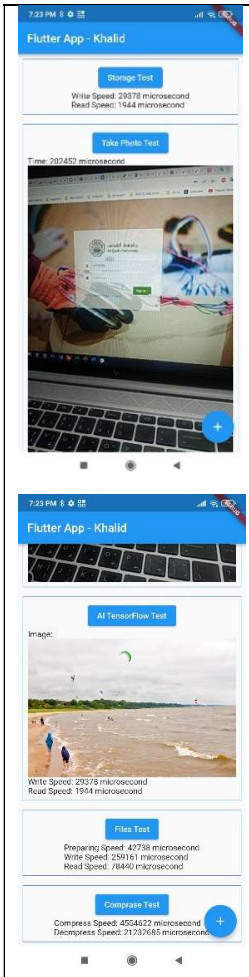
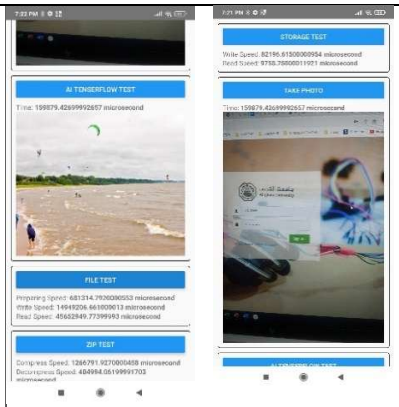
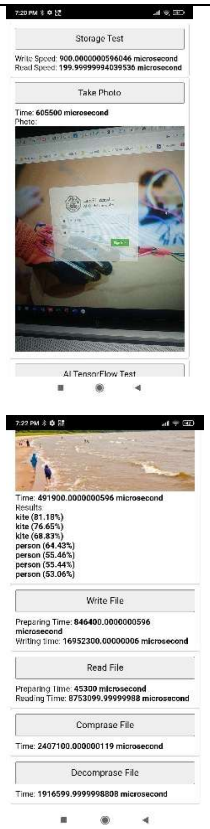
		
Flutter	React native	Cordova
		
Xamarin	Kotlin	

Table 3.2.Before run the Tests

		
flutter	React native	cordova

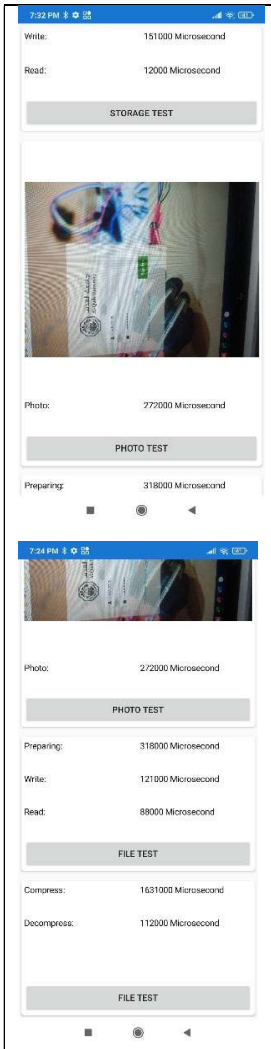
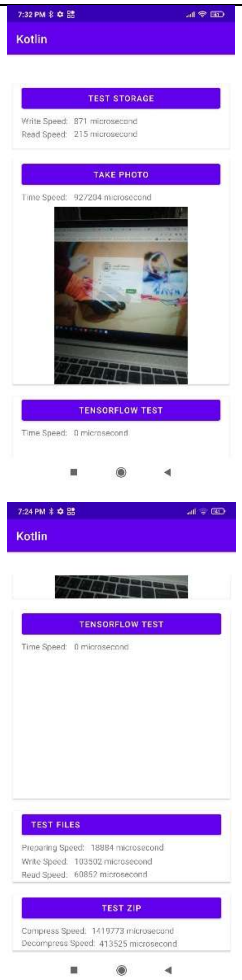
		
Xamarin	Kotlin	

Table 3.3.After run the Tests

Chapter 4

Experiment Results

This chapter contains three sections that address the study's results. It presents the workload and performance, Code Comparison, UI Development, research instruments, data-collection process, and data analysis.

4.1 Workload

After completing the five applications, we used statistical tools called VS Code Counter for IDE extinction, which is used to show statistics for every project count blank lines, comment lines, and physical lines of source code in many programming languages.

Workload	Android native	React native	Flutter	Cordova	Xamarin
Java/Kotlin	374	292	4334		
JavaScript		333	1524	172	
C#					320
XML	349	59	18212	12	2805173
Json	20	22067	44512	10299	1589
Dart			2499158		
Html			1566	91	
Css			474	14	
Total (line)	743	22751	2569780	10588	2807082

Table 4.1. Workload statistics

As the Table 5 shows Xamarin (C#) and Flutter (Dart) takes the most workload in the code line than Kotlin , Java Script and React Native according to VS Code Counter .

Xamarin and flatter have the largest line of code because the code is generated automatically on a mobile (iOS, Android) desktop (windows) and web pages at the same time.

4.2 performance

The performance shown in Table 3 and chart number 3 shows that AAB (extension for An Android App Bundle or AAB) is a publishing file format that includes all the application's compiled code and resources. It defers APK generation and signing to Google Play. (Felice, 2023) Cordova take the smallest memory usage and the most compact binary package because it's Build as web view and use web development language like HTML, CSS front end and java script as backend.

React native has the largest ABB size because it contains all the necessary libraries to work, but flutter has the largest memory usage. To complete the size test, we removed unused code lines, libraries, images, and data.

For the smallest app application for Cordova 1.4MB it's expiated and overuse because the WebView app and react native are the largest because the file package must include all libraries to work.

Application	AAB Size	Memory Usage
Cordova	1.4MB	8.86MB
Flutter	24.27MB	137.8MB
Kotlin	26.1MB	55.72MB
React native	27.34MB	62.73MB
Xamarin	15.69MB	40.94 MB

Table 4.2. Memory and AAB size comparison

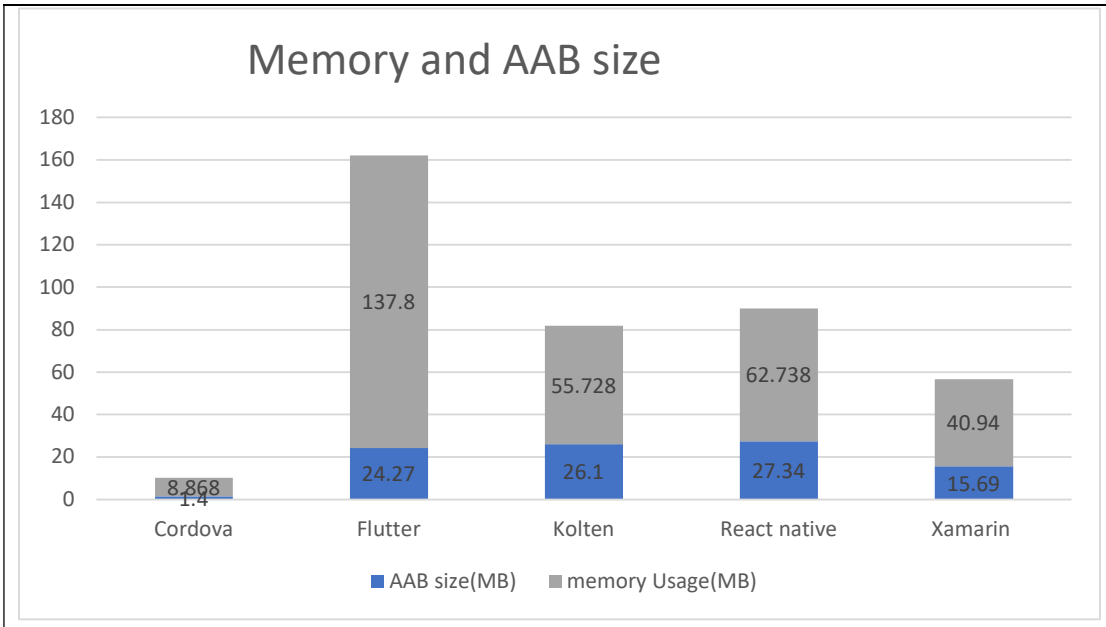


Figure 4.1. Memory and AAB size chart

Moreover, the study explored the different development frameworks used among these cross-platforms, using Android Studio for native applications (Kotlin), React Native, Cordova, Xamarin, and Flutter. Visual studio code was used for the visual studio export Android studio project after translation from written code to Java (compiler).

4.2.1 Code Comparison

Read/write from storage

From appendix1 shown the average code for read/write storage is 19 lines for Kotlin, Cordova, react native, and Xamarin, but Flutter is double the line of code 40, indicating that flutter needs more function to perform regular tasks.

i. Take automatics image from camera

From Appendix 2, it can be seen that the average code for the captured image is approximately 22 lines for flutter, Cordova, React native, and Xamarin; however, Kotlin is

double the size of the line of code 42 , which shows that Kotlin needs more line to do regular images, because it is close to the hardware level, it requires more lines of code to run and all the details in the photography process can be controlled with it line by line, and functions can be created to make short written code.

ii. Compressed and decompress files

From appendix 3 shown that average code for capture image is about 25 lines for Kotlin, react native and Xamarin but flutter and Cordova, is about 35-40 line, it shows that Kotlin React Native and Xamarin has built-in function to compress and decompress files on other hand but flutter and Cordova need external libraries and function to make it works correctly and it take more line of code.

iii. Read/Write files

From appendix 4 shown that average code for Read/Write file is about 19 lines for flutter, React native, Xamarin and Kotlin but Cordova is double size for line of code 50 line, it shows that Cordova need more line to create and read file as cache file because Cordova works as web view but Kotlin, flutter, Xamarin and react native when use hardware (storage) natively, when works with natively with hardware requires less line of code to run.

4.2.2 Testbench Result

After testing more than 17k times for all five applications, data were collected and stored in the cloud database via SQL Queries (Matusiak, 2023) .

Using SQL queries reduces a lot of time and effort using the AVG, MAX, MIN mathematical function with SQL, which gives us the output number for average, maximum, and minimum time in microseconds.

i. First Test is Read Storge Test

	Kotlin	Cordova	React Native	Flutter	Xamarin
MIN	3	100	2412	13	**1000**
MAX	1527	4700	1659286	3295	**171000**
AVG	14.2	65	38759	45	**3375**
These result is time in microsecond (time to complete the task)					

Table 4.3. Result of Read Storge Test

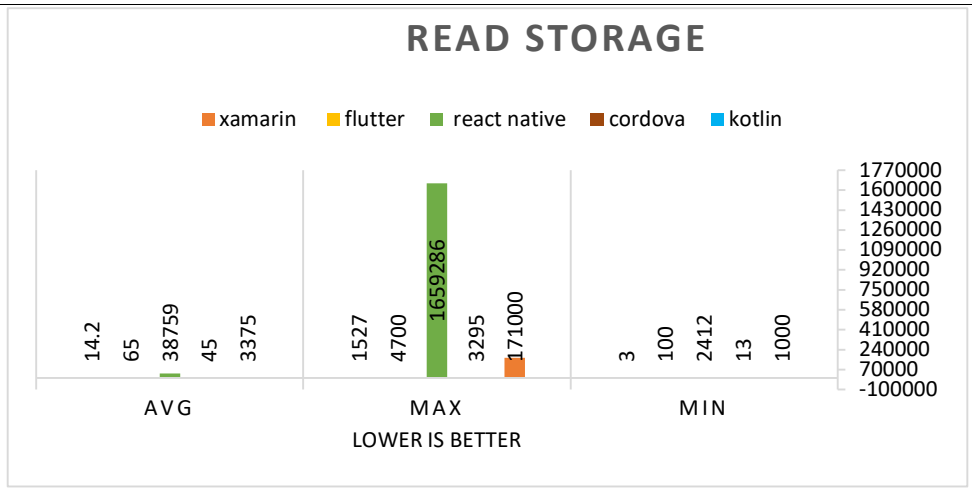


Figure 4.2. Time of Read Storage

The average recorded results show that Kotlin is the fastest programming language to read file in android OS , flutter is slower than Kotlin 316% , Cordova is slower 457% ,Xamarin is slower 23767% and React native is slower 272000 times .

ii. Secund Test is Write Storge Test

	Kotlin	Cordova	React Native	Flutter	Xamarin
MIN	96	100	7692	1361	**1000**
MAX	4640	6100	1576043	61625	**353000**
AVG	319	498	55533	9071	**4533**
These result is time in microsecond (time to complete the task)					

Table 4.4. Result of Write Storge Test

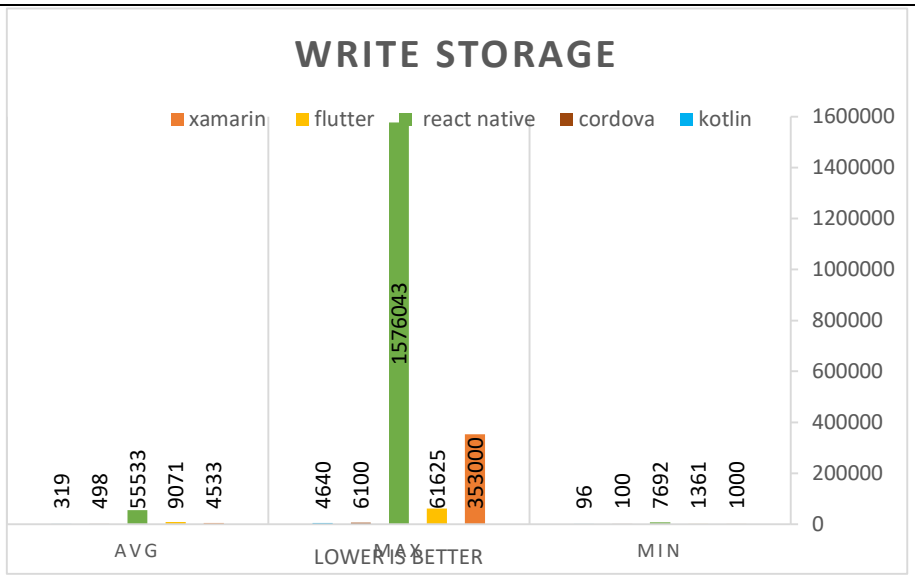


Figure 4.3. Time of Read Storage

The average recorded results show that Kotlin is the fastest programming language to write files in the Android OS, flutter is slower than Kotlin at 2843%, Cordova at 156%, Xamarin at 1421%, and react native at 17408 times.

iii. Third Test is Camera Capture Speed Test

	Kotlin	Cordova	React Native	Flutter	Xamarin
MIN	701087	280400	108965	182390	**2000**
MAX	1817558	16035700	1575230	425681	**418000**
AVG	954181	794560	195537	296281	**167668**
These result is time in microsecond (time to complete the task)					

Table 4.5. Result of Camera Capture Speed Test

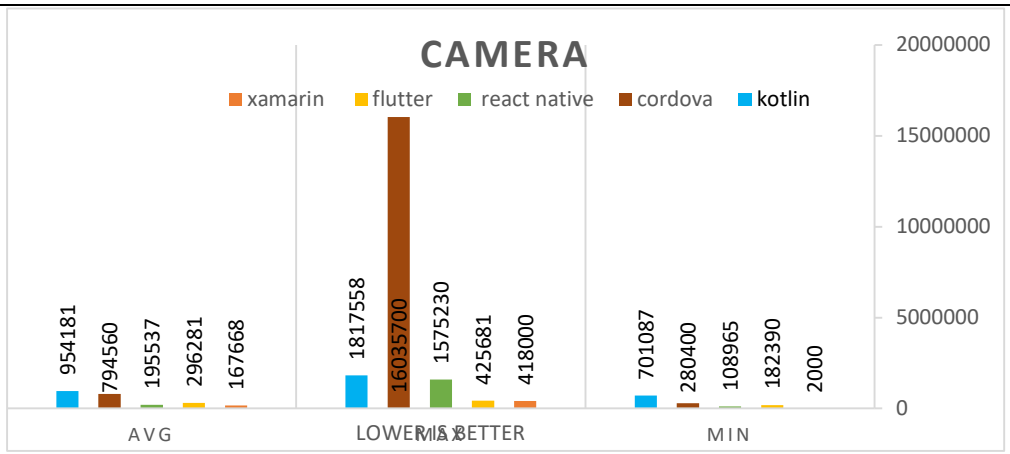


Figure 4.4. Time to capture image

By the average of recorded result show that Xamarin is the fastest programming language to capture images by camera in android OS, flutter is faster than Kotlin 322%, Cordova is faster 83.2%, Xamarin is faster 569% and react native is faster 487.9% time.

iv. Forth Test is Read file Speed Test

To test the reading of the file, the researcher created a 60MB binary file, and the application reads the file and records the time, reading the entire file, and storing the time spent inside the cloud database.

	Kotlin	Cordova	React Native	Flutter	Xamarin
MIN	59429	2180400	45076194	61169	**59000**
MAX	85979	21436500	56274391	240379	**135000**
AVG	63860	9052056	45997910	112534	**82728**
These result is time in microsecond (time to complete the task)					
Table 4.6. Result of Read file Speed Test					

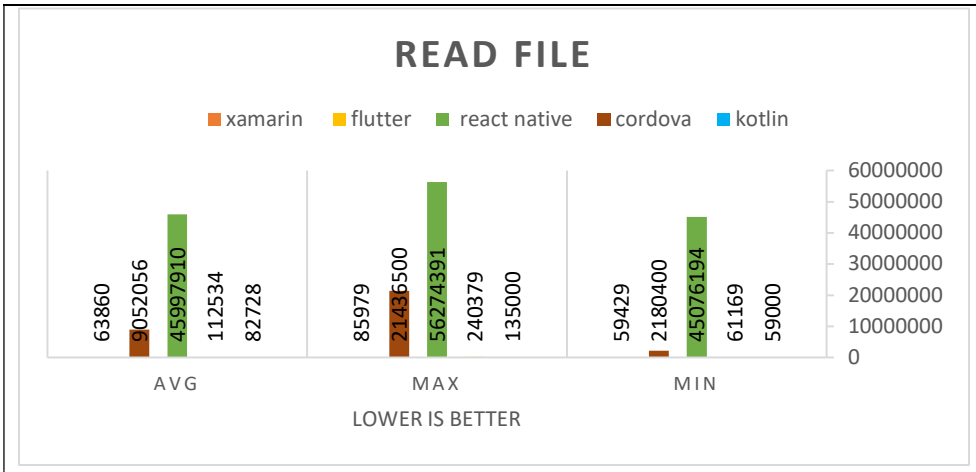


Figure 4.5. Time of Read File

The average recorded results show that Kotlin is the fastest programming language to read files in Android OS, flutter is slower than Kotlin by 322%, Cordova is faster by 83.2%, Xamarin is faster by 569%, and react native is faster by 487.9%.

v. Fifth test is Write file

To test the write file, the researcher write code to create a binary file of 60MB, and the application records the time to create it, then stores the time spent inside the cloud database.

	Kotlin	Cordova	React Native	Flutter	Xamarin
MIN	94217	16188600	12889880	168946	**106000**
MAX	409479	23824500	16776714	681554	**1052000**
AVG	117139	17722131	14625487	284719	**142458**
These result is time in microsecond (time to complete the task)					

Table 4.7. Result of Write file Speed Test

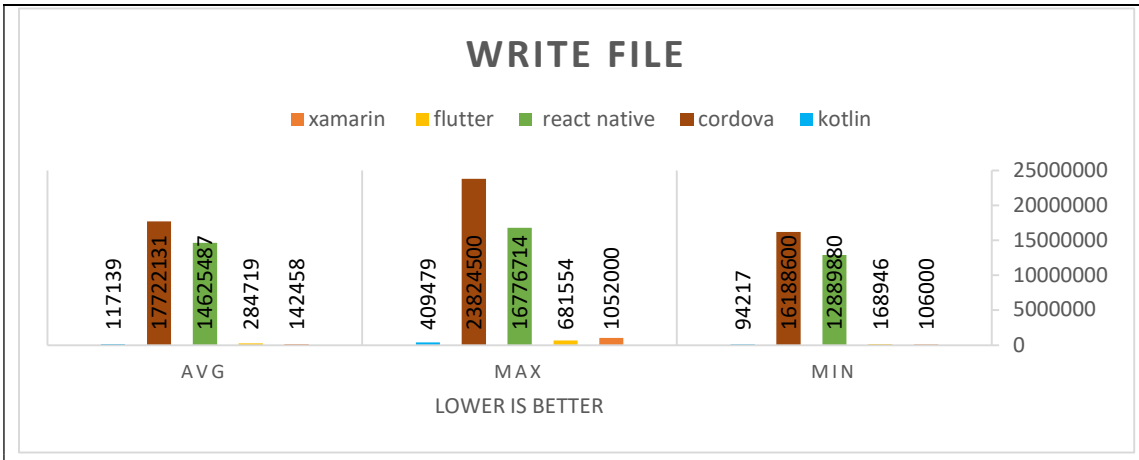


Figure 4.6. Time of Write File

The average recorded results show that Kotlin is the fastest programming language to write files in Android OS, flutter is slower than Kotlin 243%, Cordova is slower by 15129%, Xamarin is slower by 21%, and react native is slower by 12485%.

vi. Sixth test is compressing file (zip)

To compress the file, the researcher created a 60MB binary file, and the application compressed the file, recorded the time, and then stored the time spent inside the cloud database.

	Kotlin	Cordova	React Native	Flutter	Xamarin
MIN	1371807	800400	457802	4234678	**1337000**
MAX	1866770	4415700	1518122	4737229	**2472000**
AVG	1480884	2408090	1169935	4416807	**1743513**
These result is time in microsecond (time to complete the task)					

Table 4.8. Result of compressing file

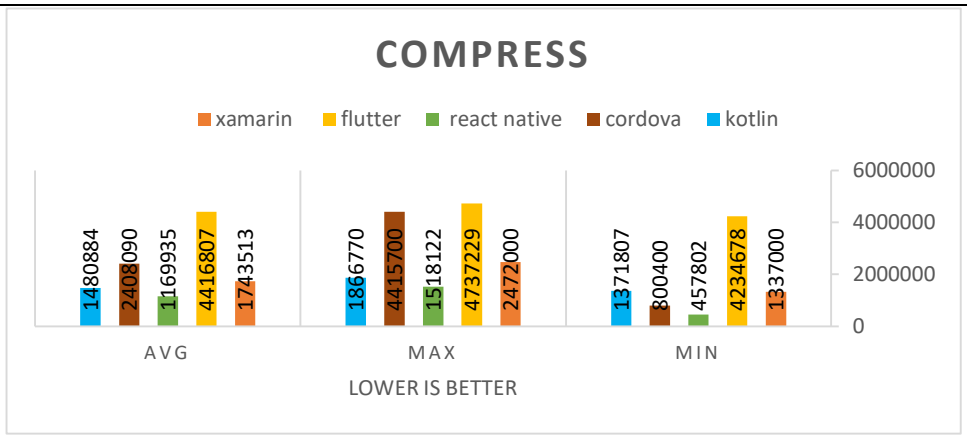


Figure 4.7. Time of compress File

The average recorded results show that native reaction is the fastest programming language to compress files in the Android OS, flutter is slower than native 377%, Cordova is slower by 205%, Xamarin is slower by 49%, and Kotlin is slower by 26 %.

vii. Seventh test is decompressing file (zip)

The result in the sixth test is compressed file, in this test the application decompressed the zip file and store the time spend in cloud database

	Kotlin	Cordova	React Native	Flutter	Xamarin
MIN	471649	1082900	311716	26308	**92000**
MAX	1067933	11140100	1530103	22393314	**555000**
AVG	494619	2460310	422804	20242307	**103184**
These result is time in microsecond (time to complete the task)					
Table 4.9. Result of decompressing file					

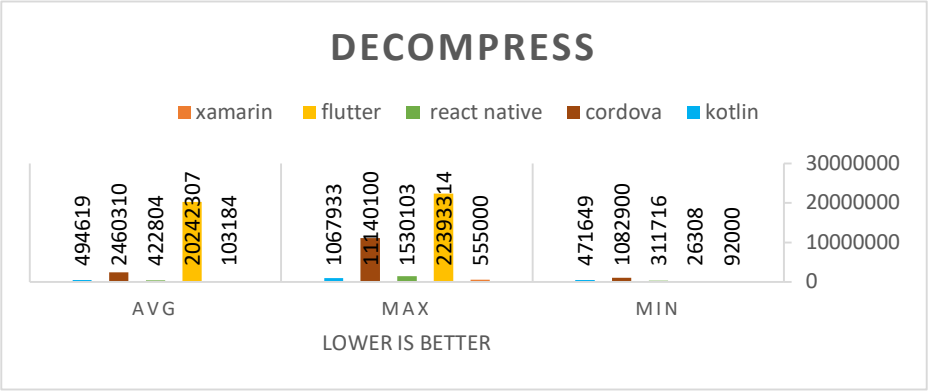


Figure 4.8. Time of Decompress File

The average recorded results show that Xamarin is the fastest programming language to decompress files in Android OS, flutter is slower than native 19617%, Cordova is slower by 2384%, reacts native is slower by 409%, and Kotlin is slower by 479 %.

4.3 UI Development

All of these frameworks can implement the same features of application development except for some minor functions, and users have to use different UI experiences because each programming language has different characteristics and its own design language.

Android studio has an IDE that covers all of the application development life Cycle and provides WYSIWYG experience in code wizards for deployment, debugging, and designing, the most important of which is the design without using an emulator or another way.

Visual studios need to install plugins and tools like Rich Text Editor to make it work (because it is universally applicable to several platforms in several programming languages). Visual studios can deploy, debug, and design, but they leak features of view design, such as Android studios and it's not good choice for graphic-heavy app development. (Khanna, 2024) .

In the old version of the visual studio, there was no way to build an AAB application without using a command line, and now you can build an AAB application in command line and using GUI in both Android studio and visual studio applications.

Chapter 5

Conclusions and Future Works

This section discusses the results of the research questions and hypotheses. In addition, the conclusions and recommendations are presented. It should be noted that the experiments followed the methodology mentioned in Section 3 to achieve the goal of this research. Recall that the goal was to compare the performance of the five mobile applications. One experimented in the Native Framework with four experimented in cross-platforms and how that affects the performance to get the best one, then the best task result for each application will be the best solution it.

As indicated in Table 7,8,9,10,11,12 and 13 in Chapter 4, the averages of each task were compared with the others. The figure 4,5,6,7,8 and 9 also show the variation in performance between different languages with different tasks.

5.1 Comparison of Cross-platform Frameworks with native Framework

Kotlin is a modern programming language developed and built for Android to be an alternative for Java or to enhance Java and use Java VM to work. React native use reacts language that needs interpreter during build and run, flutter uses dart language and implements web view and native control for Android OS, which intrudes viewing and

navigation high performance; Cordova uses web language and Java script; Xamarin uses C# and interrupts it to Java eventually and performs very well.

Next, there are two kinds of programmers to use the five programming languages Web developers and desktop developers: React Native and Cordova are the closest to web developers and require less effort to learn any of them, but flutter requires a bit more effort to learn dart language because it is less popular than JavaScript and reacts.

Xamarin (C#) and Kotlin are easy for desktop developers to learn because they are the most popular programming languages.

Flutter and Kotlin support and work Android Studio IDE, which can use all of its plugins and features, Xamarin support in visual studio IDE, react native Cordova uses many IDE such as Visual code and then imports and builds on Android Studio.

Kotlin (Android studio) is only used to create an Android application, but Xamarin, flutter, react native, and Cordova support android, iOS, windows, and mac platform, which give them an advantage over Kotlin for general purposes.

On the other hand, using a benchmark to choose which of them is the best language for some features used in mobile applications.

Overall, Xamarin and Kotlin preferred when the performance was necessary, and most developers were good at the last one (Java, net), flutter is suitable for Java or C++ developers who wish to create an app for a different platform, Cordova, and react native is suitable for web developers.

To perform the best despite the language to read/write storage and Read/Write file is Kotlin, the best camera action is Xamarin because it uses build from scratch camera app function to perform speed (need to be installed in visual studio as add-ons), React Native has the best performance in compressing and decompressing the zip file; Cordova comes in the average of these PL.

5.2 Framework limitation

After using these five programming languages and their frameworks were run as expected and data were collected in the cloud database, a limitation was found during and beyond the development process, which included the following:

i. During data collection, all results were collected via the old Android version (Android 10), and the result may change if the older version is used.
ii. During data collection, all applications were tested on old smartphones (Xiaomi MI 8 Lite "6years old"), which lid to improved results (better storage speed like UFS 4 , better processor.
iii. Most of the features implemented and tested in the project are natively included by the framework, except for the compress/decompress test, and the Xamarin camera was used as an external library, perhaps because there were better extensions and the library makes it work better and gives different times.
iv. The update and iteration of all cross-platform frameworks are very fast, and mismatch is allotted by library (native, external); after every update, the programmer should test the application and check the changes, and the analyses result in changes from one version to another.
v. Many libraries that are not supported on cross platforms such as Google image processing are supported in Cordova, partially supported and a bit hard to do, not supported in flutter, react native, and Xamarin.
vi. Many frameworks need to export the project to Android studio to be built like Cordova and react native, which leads to many extra steps to test and export AAB applications.

5.3 Conclusion

This study used five programming languages for comparison using this sample application. All programming languages are able to do what is required, but they differ in performance as some languages perform better tasks than others. The result indicated that every programing language has strong point to make it more usable and reduce workload. Every cross-platform framework and its programming language attempt to achieve the best performance and usability to compare it with the native. Some of them has achieve to do better performance like Xamarin in camera capture, react native in compress and decompress, some of them where easer in designing like flutter and Xamarin, Cordova it was the easiest to work with if your web development, Xamarin were better for C++, java developer.

Most programming languages used in cross-platform are Xamarin (C#) and flutter (Dart), and there is no better language or platform because every language and framework is suitable for developers with different development backgrounds. These results match those reported by You and Hu (2021). They agreed that every framework has a category of programmers that can be handled easily, and it has a better field of programming than the others, depending on the nature of the use.

For further improvement, more features should be tested in different devices at different Android OS and IOS, may perform better in IOS, and the data should be updated and analyzed after several platform updates because each update may provide more optimization or degradation.

In summary, all cross platforms can significantly improve their development. However, each framework is suitable for different developers, applications, costs and scenarios.

Reference

- *The Holy Qur'an.* (القران الكريم).

- Abid Bin Syeed, S. H. (2021). Study of Mobile App Development Industry. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, pp. 154-170.
- Ancheta, W. (2020). *Getting Started with React Native*. Retrieved from sitepoint: https://www.sitepoint.com/getting-started-with-react-native/
- Anitha Periyasamy, D. M. (2019). PERFORMANCE ANALYSIS OF FRAMEWORKS TO DEVELOP PLATFORM INDEPENDENT MOBILE APPLICATIONS. *Computational Intelligence and Big Data Analytics*, pp. 85-94.
- Arif Md. Sattar, P. S. (2023). Accelerating Cross-platform Development with Flutter. *Journal of Open Source Developments*, pp. Volume 10, Issue 2, 2023.
- Bhoyate, S. (2024). Analysis Of Android Programming Languages Based On Several Factors. *Journal of Advanced Zoology*, pp. Volume 45 Issue S-4 Year 2024 Page 73-76.
- Danielsson, W. (2016). *React Native application development.* Linköping: Linköpings universitet.
- Desai, A. (2023). Comparative Study of Android Native vs Cross-Platform Mobile Application Development. *International Journal for Research in Applied Science and Engineering Technology*.
- Dongliang You, M. H. (2021). A Comparative Study of Cross-platform Mobile. Whitireia Polytechnic, New Zealand.
- Felice, S. (2023). *How to test .aab file on Android device*. Retrieved from browserstack: https://www.browserstack.com/guide/test-aab-file-on-android-device#:~:text=to%20APK%20file%3F-,What%20is%20Android%20App%20Bundle%20(aab)%3F,and%20signing%20to%20Google%20Play.
- Kaushik, S. (2024). *The Rising Demand to Hire Mobile App Developers in 2024*. Retrieved from mobileappdaily: https://www.mobileappdaily.com/knowledge-hub/hiring-mobile-application-developers#:~:text=The%20demand%20for%20the%20number,24%25%20by%20the%20year%202026.&text=The%20world%20is%20fast%2Dforwarding,the%20internet%20and%20vice%20versa.

- Keith Vassallo, L. G. (2019). Contemporary Technologies and Methods for Cross-Platform Application Development. *Journal of Computational and Theoretical Nanoscience*.
- Keith Vassallo, L. G. (2019, 9). Cross-Platform Development Frameworks. *Journal of Computational and Theoretical Nanoscience*, pp. 3854-3859.
- Khanna, G. (2024). *10 Best Cross Platform App Development Frameworks 2024*. Retrieved from appwrk.com: https://appwrk.com/best-cross-platform-and-no-code-app-development-frameworks
- Malavolta, I. (2016). Web-based hybrid mobile apps: state of the practice and research opportunities. *the International Workshop.* the International Workshop.
- Marghoob, A. (2023). ANALYSIS OF CROSS PLATFORM MOBILE APPLICATION DEVELOPMENT FRAMEWORKS. RIGA: RIGA TECHNICAL UNIVERSITY.
- Matusiak, D. (2023). Basics of SQL: IT tables. In D. Matusiak, *Basics of SQL: IT tables* (p. 1). Zgierz: Eclipse Publishing House. Retrieved from amazon: https://aws.amazon.com/what-is/sql/#:~:text=Structured%20query%20language%20(SQL)%20is,relationships%20between%20the%20data%20values.
- Merenych, S. (2023). *Cordova vs React Native App Development: Which One to Pick?* Retrieved from https://clockwise.software/: https://clockwise.software/blog/cordova-vs-react-native-comparison/
- Patel, B. (2024). *9 Top Mobile App Development Frameworks in 2024*. Retrieved from spaceotechnologies: https://www.spaceotechnologies.com/blog/top-mobile-app-development-frameworks/
- Sherif, A. (2024). *Market share of mobile operating systems worldwide from 2009 to 2023, by quarter*. Retrieved from statista: https://www.statista.com/statistics/272698/global-market-share-held-by-mobile-operating-systems-since-2009/
- Shivanandhan, M. (2023). *How to Build Mobile Apps with Flutter*. Retrieved from freecodecamp: https://www.freecodecamp.org/news/how-to-build-mobile-apps-with-flutter/
- Siddhi Sanjay Shinde, P. P. (2021). A Review Paper on Kotlin Programming Language. *International Journal of Trend in Scientific Research and Development (IJTSRD)*, pp. 1182-1185.

- Sreekanth Dekkati, K. L. (2019). React Native for Android: Cross-Platform Mobile. *Global Disclosure of Economics and Business, Volume 8, No 2/2019*, pp. 153-164.
- *Top 25 Apache Cordova Interview Questions and Answers*. (2023). Retrieved from interviewprep: https://interviewprep.org/apache-cordova-interview-questions/
- Xinogalos, S. X. (2013). A Comparative Analysis of Cross-platform Development Approaches for Mobile Applications., (pp. 213-220).

Appendix

JavaScript (Cordova)	Dart (Flutter)

function runTestStorage() {
localStorage.clear();
$("#storageButton").prop('disabled', true);
var a1 = getNanoSecond();
localStorage.setItem("text", text);
var a2 = getNanoSecond();
var savedText = localStorage.getItem("text");
var a3 = getNanoSecond();
console.log(a1, a2, a3);
document.getElementById("writeSped").innerHTML = "" + (a2 - a1) + " microsecond";
document.getElementById("readSped").innerHTML = "" + (a3 - a2) + " microsecond";
console.log(savedText);
$("#storageButton").prop('disabled', false);
sendResults('read_storage', (a3 - a2));
sendResults('write_storage', (a2 - a1));}
function getNanoSecond(){
return window.performance.now() * 1000;
}
	class TestStorage {
	int getTime() {
	return DateTime.now().microsecondsSinceEpoch;
	}
	Future<bool> _requestPermission() async {
	if (await Permission.storage.isGranted) {
	return true; } else {
	var result = await Permission.storage.request();
	if (result == PermissionStatus.granted) {
	return true; } }
	return false; }
	Future<String> get _localPath async {
	final directory = await getExternalStorageDirectory();
	print('path' + directory!.path);
	return directory!.path; }
	Future<File> get _localFile async {
	final path = await _localPath;
	return File('$path/testfile.json'); }
	Future<int> read() async {
	try {
	final file = await _localFile;
	
	int a = getTime();
	var s = await file.readAsBytes();
	int b = getTime();
	return b - a;
	} catch (e) {
	return -1; } }
	Future<int> write(var data, int count) async {
	final file = await _localFile;
	await file.writeAsBytes(data, mode: FileMode.write, flush: true);
	int a = getTime();
	for (int i = 0; i < count; i++) {
	await file.writeAsBytes(data, mode: FileMode.append); }
	int b = getTime();
	return b - a;}}

React Native (React)	**C# (Xamarin)**

const storageTest = async () => {
let a = getTime();
await AsyncStorage.setItem('text', text);
let a1 = getTime();
let data = await AsyncStorage.getItem('text');
let a2 = getTime();
setStorage([a1 - a, a2 - a1]);
sendResults('read_storage', a2 - a1);
sendResults('write_storage', a1 - a);
};
function getTime() {
	public async Task StorageTest() {
	timer.Start();
	Preferences.Set("TestStorage", text);
	timer.Stop();
	long writeTime = timer.ElapsedMilliseconds * 1000; timer.Reset();
	storageWrite.Text = writeTime + " Microsecond";
	timer.Start();

```
  return performance.now() * 1000;}
```

```
  string data =
Preferences.Get("TestStorage", "");
timer.Stop();
   long readTime =
timer.ElapsedMilliseconds * 1000;
storageRead.Text = readTime + "
Microsecond";
await sendData("write_storage", writeTime);
 await sendData("read_storage", readTime);
}
```

Kotlin (android studio native)

```
fun testStorage() {
      var a = getTime()
      with (sharedPref.edit()) {
         putString("text", text)
         apply()
      }
      var b = getTime()
      findViewById<TextView>(R.id.storageWrite).text = (b - a).toString() + " microsecond"
      var a1 = getTime()
      var text1 = sharedPref.getString("text", "")
      var b1 = getTime()
      println(text1)
      findViewById<TextView>(R.id.storageRead).text = (b1 - a1).toString() + " microsecond"
      sendResults("write_storage", b-a)
      sendResults("read_storage", b1-a1)    } }
```

Appendix 1: Code of Read/write from storage

JavaScript (Cordova)	Dart (Flutter)

function takePhoto() {
  let options = {
  x: 0,
  y: 0,
  width: window.screen.width,
  height: window.screen.height,
  camera: CameraPreview.CAMERA_DIRECTION.BACK,};
  CameraPreview.startCamera(options);
 	
CameraPreview.setFocusMode(CameraPreview.FOCUS_MODE.FIXED);
  $("#photoButton").prop('disabled', true);
  setTimeout(function () {
  var a1 = getNanoSecond();
  CameraPreview.takePicture({ quality: 80 }, function (base64PictureData) {
  var a2 = getNanoSecond();
  // One simple example is if you are going to use it inside an HTML img src attribute then you would do the following:
  document.getElementById("photoSped").innerHTML = "" + (a2 - a1) + " microsecond";
  imageSrcData = '' + base64PictureData;
  document.getElementById("photo").innerHTML = '';
  CameraPreview.stopCamera();
  $("#photoButton").prop('disabled', false);
  sendResults('photo', (a2 - a1));
  });
  }
	void _takePhoto() async {
	  print("Test 1111"); {
	  if (!cameraSetuped) {
	  final cameras = await availableCameras();
	  final firstCamera = cameras.first;
	  _controller = CameraController(
	  firstCamera,
	  ResolutionPreset.medium,
	  );
	  _initializeControllerFuture = _controller.initialize();
	  } else {
	  _controller.resumePreview(); }
	  cameraSetuped = true;
	  } else {
	  _controller.resumePreview(); }
	  Timer _timer1 = Timer(Duration(seconds: 1), () async {
	  showCamera = true;
	  setState(() {
	  showCamera;

```
}, 2000);}
```

```
});  });
```

React Native (React)

```
async function checkCameraStatus() {
  return await Camera.getCameraPermissionStatus();}
async function askCameraPermission() {
  let status = await checkCameraStatus();
  if (status == 'authorized') {
    return true;}
  let details = await Camera.requestCameraPermission();
  if (details == 'authorized') {
    return true;}
  return false;}
  const getCamera = async () => {
  let devices = await Camera.getAvailableCameraDevices();
  for (let c of devices) {
    if (c.position == 'back') {
      setDevice(c);}}};
 const takePhoto = async () => {
  let hasPermission = await askCameraPermission();
  if (hasPermission) {
    await getCamera();
    setShowCamera(true);
    setTimeout(async () => {
      let a = getTime();
      const p = await camera.current.takePhoto();
      let a2 = getTime();
      setPhoto([a2 - a, p]);
      sendResults('photo', a2 - a);
      console.log(photo[1]);
      setShowCamera(false);
    }, 1000);}};
```

C# (Xamarin)

```
public void Photo_Clicked(System.Object
sender, System.EventArgs e)      {
          timer.Reset();
          cameraView.IsVisible = true;
          //cameraView.IsEnabled = true;
          Console.WriteLine("Start Camera");
          if (cameraView.IsAvailable)    {
             Console.WriteLine("Start Shutter");
             timer.Start();
             cameraView.Shutter();       }    }
  void cameraView_OnAvailable(System.Object
sender, System.Boolean e)  {
          Console.WriteLine("Start Shutter");
          timer.Start();
          cameraView.Shutter();    }
void CameraView_MediaCaptured(object
sender, MediaCapturedEventArgs e)    {
          timer.Stop();
  long photoTime = timer.ElapsedMilliseconds *
1000;
          photoTest.Text = photoTime + "
Microsecond";
          cameraView.IsVisible = false;
          photoPreview.IsVisible = true;
          photoPreview.Source = e.Image;
          sendData("photo", photoTime);    }
```

Kotlin (android studio native)

```
private fun takePhoto() {
       val imageCapture = imageCapture ?: return
        val name = SimpleDateFormat(FILENAME_FORMAT,
Locale.US).format(System.currentTimeMillis())
    val contentValues = ContentValues().apply {  put(MediaStore.MediaColumns.DISPLAY_NAME,
name)  put(MediaStore.MediaColumns.MIME_TYPE, "image/jpeg")
   if(Build.VERSION.SDK_INT > Build.VERSION_CODES.P) {
 put(MediaStore.Images.Media.RELATIVE_PATH, "Pictures/CameraX-Image")          }       }
val outputOptions = ImageCapture.OutputFileOptions  .Builder(contentResolver,
  MediaStore.Images.Media.EXTERNAL_CONTENT_URI,         contentValues)    .build()
    var a = this.getTime();
 var self = this;
imageCapture.takePicture(
  outputOptions,
ContextCompat.getMainExecutor(this),
object : ImageCapture.OnImageSavedCallback {override fun onError(exc: ImageCaptureException) {
Log.e(TAG, "Photo capture failed: ${exc.message}", exc)          }
            override fun
  onImageSaved(output: ImageCapture.OutputFileResults){
  var b = self.getTime();
    self.setPhotoTime(b - a);
   findViewById<PreviewView>(R.id.viewFinder).setVisibility(View.GONE);
```

```
    findViewById<ImageView>(R.id.imageView).setImageURI(output.savedUri);
    findViewById<ImageView>(R.id.imageView).setVisibility(View.VISIBLE);
     val msg = "Photo capture succeeded: ${output.savedUri}"
      Toast.makeText(baseContext, msg, Toast.LENGTH_SHORT).show()
  Log.d(TAG, msg)    }       }     )    }
    fun setPhotoTime(a: Long){
       findViewById<TextView>(R.id.photoTime).text = a.toString() + " microsecond"
       sendResults("photo", a)    }
    @SuppressLint("MissingSuperCall")
    override fun onRequestPermissionsResult(
       requestCode: Int, permissions: Array<String>, grantResults:
       IntArray) {
       if (requestCode == REQUEST_CODE_PERMISSIONS) {
          if (allPermissionsGranted()) {
             startCamera()
          } else {
             Toast.makeText(this,
                "Permissions not granted by the user.",
                Toast.LENGTH_SHORT).show()
             finish()          }       } }
```

Appendix 2 : Code of Take automatics image from camera

JavaScript (Cordova)	Dart (Flutter)

function comprase() {
// var PathToFileInString = cordova.file.externalRootDirectory + "ProjectFiles",
// PathToResultZip = cordova.file.externalRootDirectory;
// var a1 = getNanoSecond();
// console.log(PathToFileInString);
// JJzip.zip(PathToFileInString, { target: PathToResultZip, name: "ProjectFiles" }, function (data) {
// var a2 = getNanoSecond();
// console.log(a2 - a1);
// document.getElementById("zTime").innerHTML = "" + (a2 - a1) + " microsecond";
// /* Wow everiting goes good, but just in case verify data.success*/
// },function(error){
// /* Wow something goes wrong, check the error.message */
// console.log(error)
// })
var source = cordova.file.externalDataDirectory,
zip = cordova.file.cacheDirectory + '/ProjectFiles.zip';
var a1 = getNanoSecond();
$("#compraseButton").prop('disabled', true);
Zeep.zip({
from : source,
to : zip
}, function(results) {
var a2 = getNanoSecond();
document.getElementById("zTime").innerHTML = "" + (a2 - a1) + " microsecond";
console.log('zip success!');
console.log(results, zip);
$("#compraseButton").prop('disabled', false);
	void _testZip() async {
	if (!disableZip) {
	disableZip = true;
	setState(() {
	disableZip;});
	var path = (await getExternalStorageDirectory())!.path;
	var encoder = ZipFileEncoder();
	int a = getTime();
	encoder.create(path + '/ProjectFiles.zip');
	encoder.addFile(File(path + '/testfile.json'));
	encoder.close();
	int b = getTime();
	_zipWrite = b - a;
	var decoder = ZipDecoder();
	int a1 = getTime();
	final archive = decoder.decodeBuffer(InputFileStream(path + '/ProjectFiles.zip'));
	Directory(path + '/ext').create(recursive: true);
	for (final file in archive) {
	final filename = file.name;
	if (file.isFile) {
	final data = file.content as List<int>;
	print(filename);
	File(path + '/ext/' + filename)
	..createSync(recursive: true)
	..writeAsBytesSync(data);} }
	int b1 = getTime();
	_zipRead = b1 - a1;
	this.sendResults('read_zip', _zipRead);
	this.sendResults('write_zip', _zipWrite);

```
    sendResults('write_zip', (a2 - a1));
}, function(e) {
    console.log('zip error: ', e);
}); }
```

```
    disableZip = false;
    setState(() {
      _zipRead;
      _zipWrite;
      disableZip;
    });} // } }
```

React Native (React)

```
const zipTest = async () => {
    const targetPath =
`${RNFS.ExternalCachesDirectoryPath}/ProjectFile.zip`;
    const sourcePath = RNFS.ExternalDirectoryPath;
    const targetPath2 = RNFS.ExternalCachesDirectoryPath;
    let a = getTime();
    setDisabledZip(true);
    zip(sourcePath, targetPath)
      .then(path => {
        let a1 = getTime();
        console.log(`zip completed at ${path}`);
        unzip(targetPath, targetPath2)
          .then(p => {
            let a2 = getTime();
            setZipd([a1 - a, a2 - a1]);
            setDisabledZip(false);
            sendResults('read_zip', a2 - a1);
            sendResults('write_zip', a1 - a);        })
          .catch(error => {
            console.error('e', error);
          });   })
      .catch(error => {
        console.error('e2', error);
      });  };
```

C# (Xamarin)

```
public void Zip_Clicked(System.Object sender,
System.EventArgs e){
            timer.Reset();
            string fileName =
Path.Combine(FileSystem.AppDataDirectory, "test");
            string fileNameZip =
Path.Combine(FileSystem.AppDataDirectory,
"testfile.zip");
            string fileNameZipExtract =
Path.Combine(FileSystem.AppDataDirectory,
"extract");
            if (Directory.Exists(fileNameZipExtract)){
               Directory.Delete(fileNameZipExtract);}
            if (File.Exists(fileNameZip)){
               File.Delete(fileNameZip);}
            timer.Start();
   ZipFile.CreateFromDirectory(fileName,
fileNameZip);
            timer.Stop();
  long compress = timer.ElapsedMilliseconds * 1000;
            zipCompress.Text = compress + "
Microsecond";
            timer.Reset();
            timer.Start();
            ZipFile.ExtractToDirectory(fileNameZip,
fileNameZipExtract);
            timer.Stop();
 long decompress = timer.ElapsedMilliseconds * 1000;
            zipDecompress.Text = compress + "
Microsecond";
            sendData("write_zip", compress);
            sendData("read_zip", decompress);}
```

Kotlin (android studio native)

```
private  fun testZip(){
  val path = this.externalCacheDir?.path
var zipOut = ZipOutputStream(BufferedOutputStream(FileOutputStream(path + "/ProjectFile.zip")))
   var f = File(path + "/testfile.json")
val data = ByteArray(2048)
FileInputStream(f).use { fi ->
    var a = getTime()
BufferedInputStream(fi).use { origin ->
 val path = File.separator + f.name
   val entry = ZipEntry(path)
    entry.time = f.lastModified()
   entry.size = f.length()
    entry.isDirectory
   zipOut.putNextEntry(entry)
    while (true) {
        val readBytes = origin.read(data)
   if (readBytes == -1) { break }
    zipOut.write(data, 0, readBytes)    }   }
     var b = getTime()
       findViewById<TextView>(R.id.zipWrite).text = (b - a).toString() + " microsecond"
     sendResults("write_zip", b-a)
      zipOut.closeEntry()
     zipOut.close()       }
        var unzip = ZipUtils()
        var a1 = getTime()
       unzip.unzip(File(path + "/ProjectFile.zip"), path + "/ext/")
```

```
    var c = getTime()
      findViewById<TextView>(R.id.zipRead).text = (c - a1).toString() + " microsecond"       sendResults("read_zip", c-a1)    }
```

Appendix 3 : Code for Compressed and decompress files

JavaScript (Cordova)	Dart (Flutter)

function getFile() {
  var a1 = getNanoSecond();
 	
window.resolveLocalFileSystemURL(cordova.file.externalData
Directory, function (dir) {
  console.log('Reading')
  document.getElementById("r1Time").innerHTML =
"Reading.....";
  $("#readButton").prop('disabled', true);
  dir.getFile('testfile.json', { create: true }, function (file) {
  jsonFile = file;
  file.file(function (file) {
  var a2 = getNanoSecond();
  document.getElementById("rTime").innerHTML =
"" + (a2 - a1) + " microsecond";
  var reader = new FileReader();
  var a3 = getNanoSecond();
  reader.onloadend = function (results) {
  var a4 = getNanoSecond();
  document.getElementById("r1Time").innerHTML
= "" + (a4 - a3) + " microsecond";
  $("#readButton").prop('disabled', false);
  sendResults('read_file', (a4 - a3));
  };
  reader.readAsArrayBuffer(file);
  }, error => {
  console.log(error) }) }); });}
function writeToFile() {
  document.getElementById("wTime").innerHTML =
"Preparing Data.....";
  if (jsonFile == undefined) {
  alert("File is undefined")
  } else {
  $("#writeButton").prop('disabled', true);
  var a1 = getNanoSecond();
  data = [];
  for (let i = 0; i < 2048000; i++){
  data.push(254); }
	Future<int> read() async {
	  try {
	  final file = await _localFile;
	  int a = getTime();
	  var s = await file.readAsBytes();
	  int b = getTime();
	  return b - a;
	  } catch (e) {
	  return -1; } }
	Future<int> write(var data, int count) async {
	  final file = await _localFile;
	  await file.writeAsBytes(data, mode:
	FileMode.write, flush: true);
	  int a = getTime();
	  for (int i = 0; i < count; i++) {
	  await file.writeAsBytes(data, mode:
	FileMode.append);}
	  int b = getTime();
	  return b - a;}}

```
        var blob = new Blob([new Uint8Array(data)], {type:
'text/pain'});
        var a2 = getNanoSecond();
        document.getElementById("wTime").innerHTML = "<b>"
+ (a2 - a1) + " microsecond</b>";
        sendResults('perparing_file', (a2 - a1));
        jsonFile.createWriter(function (fileWriter) {
           document.getElementById("w1Time").innerHTML =
"Writing Data.....";
           var a3 = getNanoSecond();
           let numbers = 0;
           fileWriter.seek(0);
        fileWriter.onwriteend = () => {
           console.log(numbers);
         if (numbers < 30) {
              numbers++;
            fileWriter.seek(fileWriter.length);
      fileWriter.write(blob);
             } else {
             var a4 = getNanoSecond();
      document.getElementById("w1Time").innerHTML = "<b>"
+ (a4 - a3) + " microsecond</b>";
     $("#writeButton").prop('disabled', false);
      sendResults('write_file', (a4 - a3));          }      };
      fileWriter.write(blob);
       });   }}
```

React Native (React)	C# (Xamarin)

```
const fileTest = async () => {
   var path = RNFS.ExternalDirectoryPath + '/testfile.json';
   setDisabledFile(true);
   let a11 = getTime();
   let bytes = '.';
   for (let i = 0; i < 2048000; i++) {
     bytes += '.';   }
   await RNFS.writeFile(path, '', 'ascii');
   let a = getTime();
   setFile([a - a11, 0, 0]);
   for (let i = 0; i < 31; i++) {
     await RNFS.appendFile(path, bytes, 'ascii');  }
   let a1 = getTime();
   setFile([a - a11, a1 - a, 0]);
   for (let i = 0; i < 31; i++) {
     await RNFS.read(path, 2048000, i * 2048000, 'ascii');  }
   let a2 = getTime();
   setFile([a - a11, a1 - a, a2 - a1]);
   setDisabledFile(false);
   sendResults('read_file', a2 - a1);
   sendResults('write_file', a1 - a);
   sendResults('perparing_file', a - a11); };
```

```
public void File_Clicked(System.Object sender,
System.EventArgs e)
      {     timer.Reset();
         timer.Start();
         long size = 2048000 * 31;
         byte[] bytes = new byte[size];
         for (int i = 0; i < size; i++)       {
            bytes[i] = (byte)254;          }
         timer.Stop();
         if
(!Directory.Exists(Path.Combine(FileSystem.AppDataDirec
tory, "test")))       {
Directory.CreateDirectory(Path.Combine(FileSystem.AppD
ataDirectory, "test"));       }
         string fileName =
Path.Combine(FileSystem.AppDataDirectory, "test",
"testfile.json");
         long preparingTime = timer.ElapsedMilliseconds *
1000;
         preparingTest.Text = preparingTime + "
Microsecond";
         timer.Reset();
         timer.Start();
         Console.WriteLine(fileName);
         File.WriteAllBytes(fileName, bytes);
         timer.Stop();
         long writeTime = timer.ElapsedMilliseconds * 1000;
         fileWrite.Text = writeTime + " Microsecond";
         timer.Reset();
         timer.Start();
         byte[] bytesRead = File.ReadAllBytes(fileName);
         timer.Stop();
         long readTime = timer.ElapsedMilliseconds * 1000;
         fileRead.Text = readTime + " Microsecond";
         sendData("perparing_file", preparingTime);
```

```
                                                    sendData("write_file", preparingTime);
                                                    sendData("read_file", preparingTime);      }
```

Kotlin (android studio native)

```
private  fun testFile(){
    val path = this.externalCacheDir?.path
    var p = getTime()
    var byteArray = ByteArray(2048000)
    for (c: Int in 0..2047999) {
        byteArray[c] = (254).toByte() }
    var p1 = getTime()
    File("$path/testfile.json").writeBytes(byteArray)
    var a = getTime()
    for (bt in 0..29){
        File("$path/testfile.json").appendBytes(byteArray) }
    var b = getTime()
    var bytesReaded = File("$path/testfile.json").readBytes()
    var c = getTime()
    findViewById<TextView>(R.id.fileWrite).text = (b - a).toString() + " microsecond"
    findViewById<TextView>(R.id.filePerparing).text = (p1 - p).toString() + " microsecond"
    findViewById<TextView>(R.id.fileRead).text = (c - b).toString() + " microsecond"
    sendResults("write_file", b-a)
    sendResults("read_file", c-b)
    sendResults("perparing_file", p1-p)}
```

Appendix 4: Code of Read/Write files

مقارنة بين لغات البرمجة الاصلية ولغات البرمجة متعددة المنصات وادواتها في انشاء تطبيقات الاندرويد للهواتف الذكية

إعداد: خالد وليد عزات زهد

إشراف: د. رشيد جيوسي

الملخص

الهدف الرئيسي من هذه الدراسة هو مقارنة لغات البرمجة المختلفة المستخدمة في إنشاء تطبيقات الهواتف الذكية لهواتف Android باستعمال البرمجة الاصلية (Native platform) مقابل برمجة متعدد المنصات (cross-platform).

تدرس هذه الدراسة أداء خمس منصات برمجية مختلفة من خلال إنشاء تطبيق يقوم بعمل مهام متشابهة في كل اللغات البرمجية المستخدمة. استندت المقارنة على مقدار الوقت الذي يتم قضائه عند تنفيذ كل مهمة والذي يقاس بالميكروثانية.

تستخدم هذه الدراسة المنهج التجريبي لاستكشاف ومناقشة مناهج وتطبيقات تطوير التطبيقات عبر المنصات البرمجية المتخلفة. لذلك، تم تنفيذ مشروع منصة الاختبار (Test Bench) باستعمال لغة البرمجة الاصلية مثل Kotlin (Patel, 2024) ، يليه أربعة إطارات برمجية (Frameworks)، مثل Flutter , Cordova , React Native و Xamarin .

تم جمع وتسجيل المعلومات ثم تحليلها من أجل إيجاد الكفاءة وحجم المقطع البرمجي للبرامج حيث قمنا من خلال البرامج المنشئة لهذا الهدف بتسجيل وقت المستغرق لتنفيذ كل مهمة وتخزين الزمن المستغرق على قاعدة بيانات موجودة على السحابة، ثم سنقوم بتسجيل مزايا وعيوب كل منها، حيث في النهاية سيتم مقارنة اللغات الخمس الاصلية ومتعددة المنصات من خلال البيانات المسجلة في قاعدة البيانات.

أظهرت نتائج هذه الدراسة أن كل منصة متعددة، مثلXamarin ، لها مزاياها وعيوبها الخاصة. لديها أفضل نتيجة لالتقاط الصور والملفات المضغوطة، لكنها كانت الأخيرة فيما يتعلق بالوقت في تخزين القراءة/الكتابة على الذاكرة والملفات. بينما Xamarin هو الخيار الأفضل لمطوري لغة c # .NET وأسطح المكتب الذين يرغبون في البدء في تطوير تطبيقات الهاتف المحمول، وإن منصة React Native و Cordovaهو الخيار الأفضل لمطوري الويب الذين يرغبون في البدا في تطوير تطبيقات الهاتف المحمول ،اما لغات الاصلية (Native) مثل Kotlin و هي الأفضل لمطوري سطح المكتب الذين يرغبون في البدا في تطوير تطبيقات الهاتف المحمول .