International Journal of Computer (IJC)
ISSN 2307-4523 (Print & Online)
https://ijcjournal.org/index.php/InternationalJournalOfComputer/index

Cross-Platform Android App Gateway Payment System using Xamarin

Osaremwinda Omorogiuwa[a*], Ejaita Abugor Okpako[b], Juliana Ndunagu[c]

[a]*Department of Computer Science & Information Technology, Igbinedion University Okada, Edo State, Nigeria*
[b]*Department of Information and Communication Technology, University of Delta, Agbor, Delta State, Nigeria*
[c]*Department of Computer Science, National Open University, Nigeria*
[a]*Email: ask4osas@iuokada.edu.ng,* [b]*Email: ejaita.okpako@unidel.edu.ng*
[c]*Email: jndunagu@noun.edu.ng*

Abstract

Most mobile applications lacks cross platforms portability and capabilities. As such, developers tend to use specific code base that runs only on a native android application built using Java or a native iOS application built using Swift. In developing mobile application, same application is therefore required to be developed using the appropriate native app required software development. This leads to duplication of efforts, more cost, time consumption and maintenance. Although, the applications are the same, mobile application has to be developed separately because of platform independence. This paper proposes the use of Xamarin in developing mobile apps due to its cross-platform capabilities. Using Xamarin save cost, create a single code base for faster development and less maintenance while still maintaining native app performance and user experience. To substantiate Xamarin suitability, a gateway payment system was development and tested, the results showed actual cross platform functionalities in a seamless manner.

Keywords: android apps; cross-platforms; hybrid apps; native apps; xamarin; web apps.

1. Introduction

Mobile applications are becoming an integral part of our lives. Not long ago, many people used to feel uncomfortable without a cell phone, but nowadays most people feel uncomfortable without a smart phone. A great revolution in the world of mobile industry [1], that has evolved very rapidly since its beginning, from simple tiny pieces of calendars and calculators to a huge growing number of modern applications that fulfill our everyday tasks such as shopping, business, banking, diary planning and social networking, Generally, software evolution is the dynamic behaviour of programming systems as they are maintained and enhanced over time.

Received: 3/24/2023
Accepted: 4/21/2023
Published: 5/13/2023

---

* Corresponding author.

Reference [2] defined the phenomenon, but what is the specificity of mobile application evolution during its life cycle? When can we say that this application has evolved? Reference [3] earlier proposed laws for software evolution but are they applicable on mobile apps? For the best of our knowledge we say that mobile application evolution is a software evolution but with more challenges in improving quality. Since mobile app evolution takes into consideration during its life cycle many constraints such as ecosystem conditions, general context, and diversity of devices. A mobile application (app) is a software application designed to run on mobile devices such as smart phones, tablet computers and other mobile devices [4]. In mobile computing an application is considered to be mobile if it runs on an electronic device that may move (e.g., mp3 readers, digital camera, mobile phones) [5], synthetically captures the uniqueness and conceptual difference of mobile computing through four constraints namely limited resources, security and vulnerability, performance and reliability variability, and finite energy source [6], defines mobile applications from a testing perspective an app for mobile application, driven by user inputs, runs on a mobile device with limited resources. In the mobile technology, we often hear terms like native apps or Web apps, or even hybrid apps. In the following subsection we try to present similarities and differences between them.

1.1 Native apps

Native apps are installed through an application store (such as Google Play or Apple's app Store) on the device and are accessed through icons on the device home screen. They are developed specifically for one platform, and can take full advantage of all the device features, they can use the camera, the GPS, the accelerometer, the compass, the list of contacts, and so on. They can also incorporate gestures (either standard operating system gestures or new, app-defined gestures). Native apps can use the device's notification system and can work offline [7].

1.2 Mobile Web apps

Web apps are not real applications; they are really websites that, in many ways, look and feel like native applications, but are not implemented as such. We can run them by a browser and they are typically written in HTML5. Users first access them as they would access any web page: they navigate to a special URL and then have the option of installing them on their home screen by creating a bookmark to that page. Web apps became really popular when HTML5 came around and people realized that they can obtain native-like functionality in the browser. Today, as more and more sites use HTML5, the distinction between web apps and regular web pages has become blurry [7].

Mobile Web applications take advantage of the standardized Web technologies and good browser support of mobile platforms. The application is implemented as a single Web site optimized for mobile devices. The site can be accessed on any mobile device, and behaves largely the same way regardless of platform. The application is not installed on the device, but instead it is retrieved via a Uniform Resource Locator (URL) using a mobile browser on the device, and most of the functionality is stored on the web server. This makes maintenance of the application simple, as an update only requires the user to reload the page. It does, however, require constant network connection in most cases, and disconnections may cause serious issues in some applications [8].

Developing a mobile web application is very similar to developing a website [9]. It uses the same technologies and languages with mobile-specific modifications such as accessing the platform API and adapting to different screen sizes. Web applications often follow the standard three-tier architecture, where the application is divided into presentation, logic, and data tiers. Client side of the applications includes the presentation tier, and server side includes the logic and data tiers. Each tier can have its own selection of programming languages and concepts and therefore require a different set of skills. Recently new unified solutions have also been introduced, that allow developing all three tiers with a single tier's technologies [10].

Web applications by default have the look and feel of a web site, and are mostly consistent across different platforms. Different browsers can still have small differences in how they render the sites, and some platform-specific programming is usually required due to differences in hardware and platform conventions, especially when targeting both tablet and smartphone markets [11].

1.3 Hybrid apps

Hybrid apps are partly native apps and partly web apps, as a result (because of that, many people incorrectly call them web apps). Like native apps, they live in an app store and can take advantage of the many device features available. Like web apps, they rely on HTML being rendered in a browser, with the caveat that the browser is embedded within the app. Often, companies build hybrid apps as wrappers for an existing web page; in that way, they hope to get a presence in the app store, without spending significant effort for developing a different app. Hybrid apps are also popular because they allow cross-platform development and thus significantly reduce development costs: that is, the same HTML code components can be reused on different mobile operating systems. Tools such as PhoneGap and Sencha Touch allow people to design and code across platforms, using the power of HTML. Native and hybrid apps are installed from an app store, whereas web apps are mobile-optimized web pages that look like an app. Both hybrid and web apps render HTML web pages, but hybrid apps use app-embedded browsers [7].

The objective of this paper is to discuss the ways of building cross-platform mobile applications (apps) using the Xamarin framework. In recent times, developers have used different approaches in building cross-platform applications. Technologies like HTML5 and JavaScript has been popularly used due to the fact that the applications ran on a native browser, but this did not turnout as expected, as the user experience was not as that of a native mobile application.

Mobile developers nowadays want their apps to be available on all the platforms, and since each platform works using different user interfaces and is developed using varying Software Development Kits (SDKs), it is more difficult to build a native app for multiple platforms as this requires the knowledge of the various development language to build for the respective platforms. Xamarin is a unique cross-platform mobile application development platform that provides up to 75% shared code across iOS, Android and Window. It offers Xamarin.Forms and Xamarin Native (Xamarin Android and Xamarin iOS) to quickly build native mobile apps across platforms. It enables the use of shared C# and Net libraries for building a common sharable backend.

2. Statement of the Problem

When we think of mobile development, the first programming language or framework that comes to mind for android development is using Java programming for native development and Swift for native iOS development.

Due to the recent and continuously growing rise and dependence in the use of mobile applications, developers as well as companies are tasked with creating solutions available for both the Android mobile Operating System (OS) and the iOS for Apple mobile devices. This would mean two sets of code base is needed to build native applications for both operating systems, as well as two teams or units needed to manage the code base and implement required updates to the applications. This scenario is more expensive to execute and a wide range of skilled personnel is needed. It will also require personnel that have an understanding of both platforms. Also, this problem can lead to inconsistency in the business logic of the application as consistent interaction between the teams developing for each platform must be present so as to integrate the same features on both platforms.

In Xamarin, we have a solution, where in with a single codebase, a single team would be needed to build applications that are native to the mobile platforms (i.e. the Android OS and the iOS for Apple mobile). All that is needed are mild adjustment based on the User Interface (UI) of each platform, but the general codebase remains relatively the same.

3. Aim and Objectives of the Study

The aim of this study is to show that using the Xamarin .Net framework, a cross-platform mobile application, native to the various platforms can be developed and the application runs as smooth as a native android application built with Java, or a native iOS application built using Swift.

To this end, the following objectives were actualized:

i. The design and implementation of a native Android application using the Xamarin framework
ii. The integration of a payment gateway financial transaction system in the Android application.

4. Review of Related Literatures

Research have shown that performance of RAM and CPU for Xamarin based applications were much slower than native Android applications, with the best Xamarin result being 10.4% slower and at worst a whole 77.9% slower than a Native Android application [12, 13] did a comparison between Xamarin and native development, where he compares Xamarin.Forms to native applications. Xamarin were surprisingly efficient to work with and in the example of buttons in Android vs Xamarin the method in Xamarin is much more efficient to use and were quicker to build applications. Reference [14] comparing performance and looks of Flutter and Native applications, concluded that the performance of Flutter is comparable to Native applications when looking at CPU usage and lines of code. Flutter is another cross platform tool that can be used to build mobile apps. While xamarin uses C#, flutter uses dart programming language [15], compared different kinds of cross-platform methods, among others compilation-based which is the method used by Flutter and the runtime-based and

Interpreted type such as xamarin, though they use native script which also is a runtime based and interpreted framework, their findings showed some interesting inherent differences between the different approach for the frameworks. They mentioned among "other cross-platform frameworks were found to possibly be more in performance on certain metrics, such as NativeScript's time to completion and CPU usage, and flutter's minimal increase in memory usage during task benchmarking. They concluded in their findings that when using a cross-platform framework you might get lower performance than an native application[16], but depending on the selected metric compared and framework you might get close or equal performance as well which shows the importance of selecting framework after what performance needs a project might have [17].

5. Methodology

The design methodology adopted for this research work is object oriented methodology (OOP). All entities in the world can be described as objects. These objects exist in nature, in man-made entities, in business, and in the products that we use. They can be categorized, described, organized, combined, manipulated and created. Therefore, an object oriented view has come into picture for creation of computer software. An object oriented approach to the development of software was proposed in late 1960s. Object-Oriented development requires that object-oriented techniques be used during the analysis, and implementation of the system. This methodology asks the analyst to determine what the objects of the system are, how they behave over time or in response to events, and what responsibilities and relationships an object has to other objects. Object-oriented analysis has the analyst look at all the objects in a system, their commonalties, difference, and how the system needs to manipulate the objects. Object oriented modeling of building systems takes the objects as the basis. Firstly, the system to be developed is observed and analysed and the requirements are defined. Secondly, the objects in the required system are identified e.g. students, admin, computer systems, online allocation system etc. in simple terms, object oriented modeling (OOM) is based on identifying the objects in a system and their interrelationships, once this is done, the implementation of the system is done. The basic steps of system designing using Object Modeling may be listed as:

5.1 System Analysis

Requirements analysis is one of the major tasks in software engineering, which is vital to the success of a software development project. It involves the determination of the requirements or functions of a software project. The main task to be performed before analysing requirements is the listing of the requirements.

The requirements for this application were gathered based on Quality Function Deployment (QFD). This is because QFD prioritizes both explicit and implicit requirements for the application. Also, it focuses on client satisfaction all through the development process.

QFD is a requirements elicitation technique used to convert client's requirements and expectations into technical requirements for the software product. It aims at building a software system that fulfils client satisfaction by focusing on what is relevant to the client. More so, it utilizes different methods, such as interviews, surveys, and review of historical data to achieve its objectives.

According to user needs and expectations, QFD prioritizes requirements into three types:

i. *Normal requirements*: These are must have requirements with priority level 1. They are requirements that fulfil client satisfaction if present.
ii. *Expected requirements*: These are should have requirements with priority level 2. They are requirements that are not explicitly declared by the client but could be a reason for customer dissatisfaction if not accomplished.
iii. *Exciting requirements*: These are nice to have requirements with priority level 3. They are needs that are beyond the scope of the project but could result in client satisfaction when present.

5.2 System Requirements

Whether you are developing on a Mac or Windows workstation, you will need to download and run the Xamarin unified installer at http://xamarin.com/download. This will allow you to install and configure the Xamarin platform including the Xamarin Android SDK, Xamarin iOS SDK, Xamarin Studio, and Xamarin plug-in for Visual Studio, as appropriate.

Xamarin products rely upon the platform SDKs from Apple and Google to target iOS or Android, so Xamarin system requirements match theirs.

6. System Design

A Unified Modelling Language (UML) based tool was used to model this application. UML diagrams give both static and dynamic views of an application and it is well suited for object-oriented languages like Java and C#. The following sub-sections present the UML diagrams used to model this application.

- ***Use Case Diagram***

The use case diagrams for this application illustrate the interactions that exist between users (actors) and use cases (actions) within the application. There are two actors identified for this application; administrator (admin) and customer actors. As a result, there are two use case diagrams for the software application; admin use case diagram and customer use case diagram. The admin is the owner of the e-commerce store who performs various administrative tasks such as add products, view orders, and update order status while the customer is any individual who buys a product or products from the online store.

The admin use case diagram in fig [1] depicts how the admin communicates with the application. More so, it shows all the actions that the admin can perform on the application. As can be seen in fig [1], before any of these actions could be executed the admin will have to login in order to be authenticated.

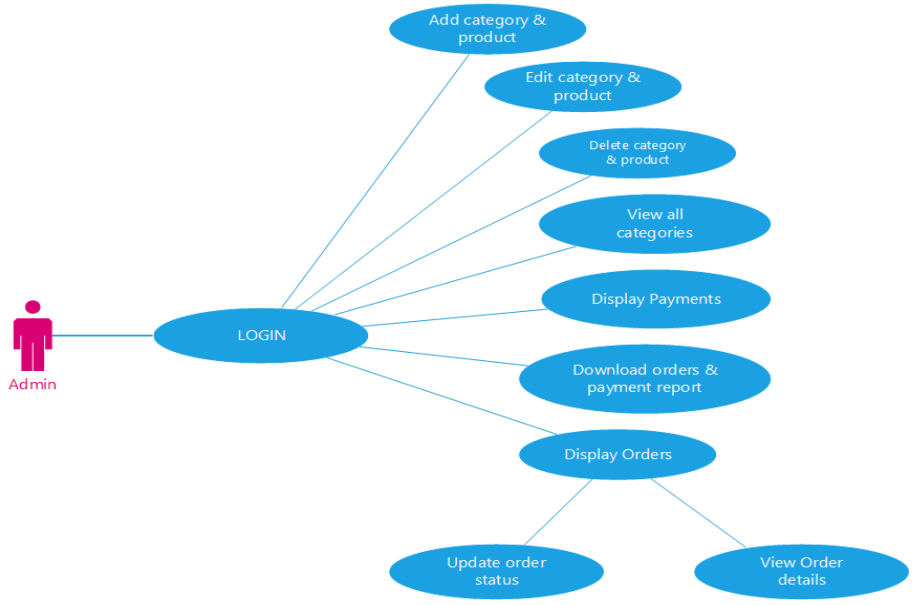

Figure 1: Admin Use Case Diagram.

Figure [2] shows the customer use case diagram. It describes the different use cases that can be executed by the customer on the e-commerce application.

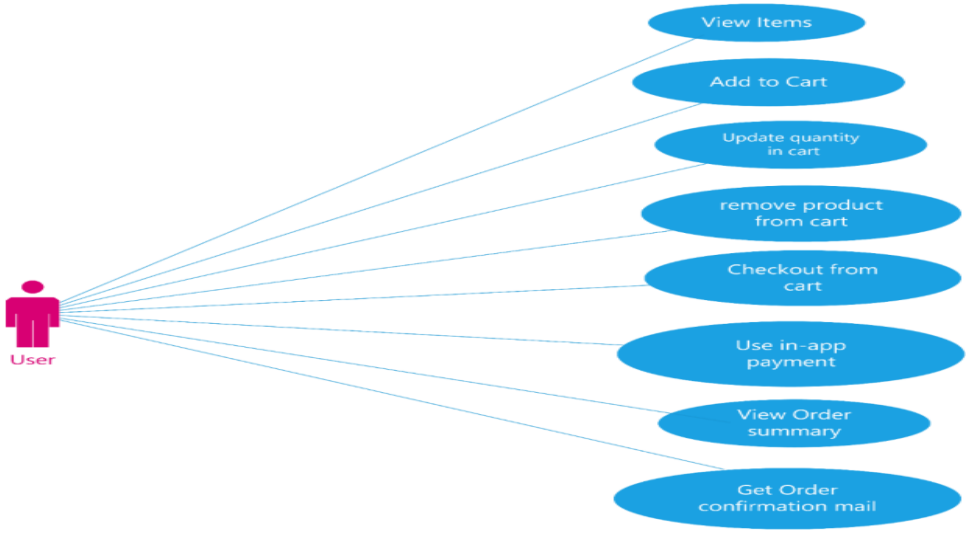

Figure 2: Customer Use Case Diagram.

7. Results and Discussion

Xamarin has been found to be a great way for development as it saves time and resources. An aspiring developer looking to offer an app on different platforms should basically build using the Xamarin framework. Here are some of the biggest advantages to choosing Xamarin:

i. *A single code base for faster development:* Xamarin, created with Visual Studio, Xamarin-based apps are developed using a single language. C#. Xamarin apps utilize C# and share code bases that covers a

lot of each platform's particular language. By developing in C# and allowing Xamarin to handle cross-platform implementations, development teams will be able to accomplish much more with less.

ii. *Native Performance and User Experience:* In Xamarin, it is possible to access each and every native API, so it is possible to use completely native UI, SDKs, etc. Since Xamarin can take full advantage of system and hardware-specific APIs, apps built using the software will run as well as apps compiled in each platform's preferred language. Users won't be able to tell the difference between your app and a native app, because there isn't one.

iii. *Reduced Time to Market:* Building applications with shared codebases eliminates time that would typically be spent translating, rewriting or recompiling codes to work on different platforms. This takes off weeks or even months off the development cycle, allowing applications for all three major platforms to be developed simultaneously.

iv. *Less Maintenance:* Maintaining and updating apps built using the Xamarin framework requires less work. Once changes has been made to the source file, they can be applied directly to the applications, eliminating the need to update the source code of your apps individually should any updates, bug fixes or new features become necessary.

v. *Apps for all Platforms:* With Xamarin, developers can create applications for mobile and even desktop experiences simultaneously. This also helps development teams cut back on having to decide whether to develop for just one single platform, as Android, iOS and Windows can be handled simultaneously.

The user interface of the app was designed using Microsoft Visual Studio 2017. The next section shows screenshots of some of the activities are shown as follows:

i. Create Account Activity: This activity in fig 3 is instantiated when the user wants to create an account. This comes after the splash screen. Details of the user is collected and entered into the database.

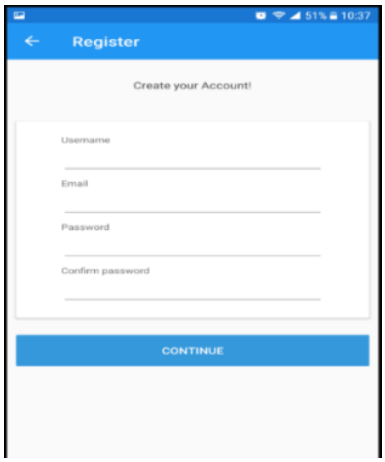

Figure 3: Screenshot of the Create Account Activity.

ii. Cart Activity: This activity in fig 4 is where all the products that has been added to cart is displayed. From this activity, a product can be paid for and also removed from the cart.

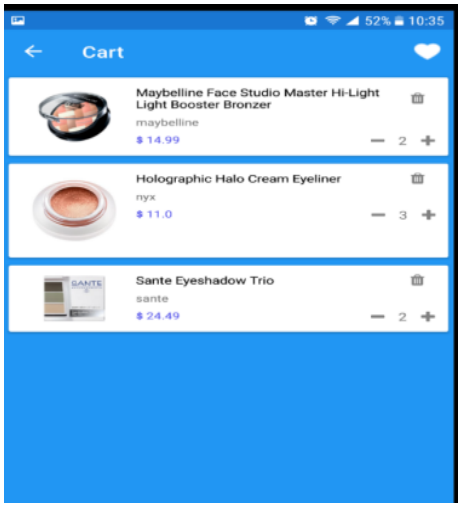

Figure 4: Screenshot of the Cart Activity.

iii. Login Activity: This activity in fig 5 is where the user logs into the app after completing the signup process. This activity collects the login credentials of the user and runs a check in the database for a corresponding credential. If it exists, the user is logged into the app, if it doesn't, a prompt is displayed to check input details, recover credentials or create a new account.

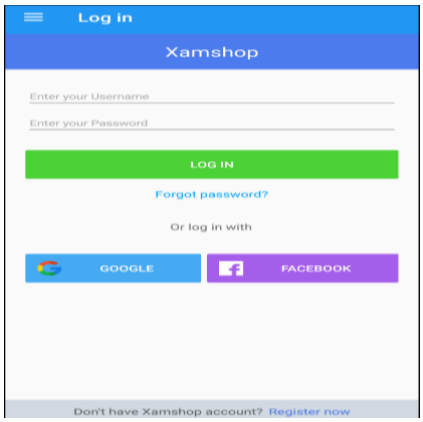

Figure 5: Screenshot of the Login Activity.

8. Conclusion

Nowadays people are using their mobile phones for most of their personal computing. The expected user behaviour is that the application they are using will work in the same way on any mobile platform. This brings a great challenge to developers who are trying to build apps with a great user experience, look and feel on all three platforms namely iOS, Android and Windows Phone. To tackle this problem Xamarin forms was used to develop the application. Xamarin.Forms is a framework from Xamarin that allows developers to reduce the amount of platform specific UI code required when creating cross platform mobile applications. This provided the advantage of increasing the amount of code-reuse between platforms. The use of Xamarin.Forms showed an advantageous way to build usable, performance native apps for Android. An an e-commerce mobile application

using Xamarin for a small retail store where the store owner manages products, customers, and orders, while the customers make orders and pay for products was developed. This e-commerce application is run conveniently on iOS, Android and Windows phone. The use of Xamarin in the development of cross-platform applications tends to be much more beneficial because, it saves time, resources and highly platform independent.

References

[1] Revinr Software Development. The evolution of Mobile Application Development. Revinr, Technology Made Simpler, published March 2023. Linkedin.com/pulse/evolution-mobile-application-development-revinr.

[2] Lehman, M., and Ramil J. F. Software evolution in the age of component-based software engineering. Annals of Software Engineering (14) pp. 275–309, 2002.

[3] Lehman M. M., Ramil J. F., Wernick P. D., Perry D. E., and Turski W. M. "Metrics and laws of software evolution-the nineties view in Software Metrics Symposium, 1997 Proceedings." Fourth International, IEEE; pp. 20–32. 1997.

[4] G. Chen and D. Kotz, A survey of context-aware mobile computing research (Tech. Rep. TR2000-381) Dept. of Computer Science, Dartmouth College, Hanover, N.H., Dartmouth, 2000.

[5] Satyanarayanan, M. (1996). "Fundamental challenges in mobile computing, in Proceedings of the fifteenth annual ACM symposium on Principles of distributed computing, ACM" pp. 1–7.

[6] Muccini, H., A. Francesco, D., and Esposito P. Software testing of mobile applications: Challenges and future research directions, in Automation of Software Test (AST). 7th International Workshop on, IEEE, 2012, pp. 29–35. 2012.

[7] Budiu Raluca. Mobile: Native apps, web apps, and hybrid apps. September, 2013. http://goo.gl/ZEsfGJ.

[8] Heitkotter, H., Hanschke, S., and Majchrzak, T. A. Comparing Cross-Platform Development Approaches for Mobile Applications. In Web Information Systems and Technologies. pp 299-311, 2012. doi: 10.1007/978-3-642-36608-6xx.

[9] Occhino, T. React Native: Bringing Modern Web Techniques to Mobile, 2020. Retrieved from: https://engineering.fb.com/android/react-native-bringing-modern-web-techniques-tomobile/ Access date: 05/05/2023.

[10] Laine, M., Shestakov, D., Litvinova, E. and Vuorimaa, P. "Toward United Web Application Development. IT Professional", September, 2011. 13(5), pp30-36.

[11] Charland A. and LeRoux B. Mobile application development: web vs. native. Communications of the ACM pp 1-4, April 2011. doi: 10.1145/1941487.

[12] Karlsson M. and Andersson Vestman F. Experimentell studie av prestandaskillnader mellan native Android och Xamarin f¨or mobilapplikationer. Lula University of Technology Publications, June 2018, pg35.

[13] Borop A. Xamarin Forms vs Native Platform Development. In: *Student Scholarship Computer Science 5*. February, 2018. https://digitalcommons.olivet. edu/csis_stsc/5.

[14] Olsson M. A Comparison of Performance and Looks Between Flutter and Native applications: When to prefer Flutter over native in mobile application development, December, 2020. http://urn.kb.se/resolve?urn=urn:nbn: se:bth-19712.

[15] Grønli et al TM, Biørn-Hansen A., and Rieger C. An empirical investigation of performance overhead in cross-platform mobile development frameworks. In: *Empir Software Eng 2*. *June* 2020. https://doi.org/10.1007/s10664-020-09827-6.

[16] Stander S. and Akesson H. Cross-platform Framework Comparison: Flutter React Native. In: (2020). doi: http://urn.kb.se/resolve?urn=urn:nbn:se:bth-19749. 2020. Accessed 06/05/2023.

[17] Ekrem G., Ahmet B. U., Naset S. Comparison of flutter and React Native Platform, December, 2021. Vol,12, issue2, pg129 -143. https://doi.org/10.34231/iuyd.888243