ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 8, Issue 3, September 2018

Evaluation of Cross-Platform Frameworks for Mobile Applications

Jozef Goetz, Yan Li
Department of Computer Science and Computer Engineering, University of La Verne, USA

Abstract*— The goal of the paper is to show the comparison of creating an e-commerce application with the same user interface and functionalities using mobile application development frameworks PhoneGap and Xamarin. Next, deploy the e-commerce app from each framework into iOS and android running on Apple iPhone 8, iPhone X, Samsung S8, Samsung Note 8 and Google Pixel2 XL. The application running on different platforms is evaluated according to the user and developer criteria such as execution time, lines of code, application size, CPU usage, memory usage, ease of use IDE such as PhoneGap and Xamarin Studio. Finally, the comparative results are shown from the different perspective.

***Index Terms*— Android and iOS platforms, Mobile App Performance, PhoneGap, Xamarin.**

I. INTRODUCTION

Nowadays, e-commerce takes a more and more important place than before, and the development of mobile applications are growing rapidly. There are many IDEs (Integrated Development Environment) in the market that can be used to develop different kinds of applications on either iOS or Android. They are easy to use and can produce applications on both platforms. However, it is sometimes too difficult for the beginning developer to figure out which tool performs better and is more practical when one decides to write an e-commerce application.

Mobile application development is becoming more challenging with diverse platforms and their IDEs. In order to reduce the cost of development and increase performance, developers are migrating to cross platform application development tools. There are many classification and comparison among the tools. Dalmasso, Datta and Bonnet examine the performance in terms of CPU, memory usage, power consumption for PhoneGap, Titanium and Sencha Touch [3]. Oberg developed a generic evaluation framework for assessing cross-platform mobile development tools [8]. Tunali, Erdgon present an elaborate comparison of several popular cross-platform mobile application development tools [11], namely, PhoneGap, Xamarin, Appcelerator Titanium, and Smartface App Studio from different perspectives like ease and cost of development including programming language and tool support, end-product capability and performance, security, and community support. Prajapati, M., Phadake, D., Poddar A., discuss the most important features of Xamarin frameworks [10].

This paper introduces complementary approach of a comparative performance review of creating mobile applications for different platforms using two mobile application development frameworks PhoneGap [9] and Xamarin [12]. The same applications running on different platforms are evaluated according to the seven different criteria from the user and developer perspectives, among them are the Execution Time and Lines of Code needed to develop an application.

A. Framework Overview

1) PhoneGap

PhoneGap is a development tool published by Adobe [9]. It uses HTML, JavaScript and CSS to develop cross platform mobile applications without using native development languages. Developers can write code once and deploy their app across multiple mobile operating systems such as iOS, Android, Windows Phone, BlackBerry, and Amazon FireOS. PhoneGap provides a JavaScript programming interface that allows developers to access platform-specific features with plain JavaScript. It is very convenient to design the UI (User Interface) and package a website into a mobile application.

2) Xamarin

Xamarin [12] is another development tool owned by Microsoft. Xamarin is a development platform that allows developers to code native, cross-platform iOS, Android, and Windows Phone apps in C# language [10]. The choice for Xamarin development environment is Visual Studio, which works on both Mac and Windows [12]. It also supports iOS and Android and uses C# to develop applications. The UI can be designed by coding only either by C# codes or XAML codes (Extensible Application Markup Language). Xamarin.Forms, allows single code-based apps for a specified set of UI controls.

B. Comparison Criteria

The applications created are evaluated according to the following criteria:

1. Execution Time
2. Launch Time
3. Lines of Code (LOC)
4. Application Size
5. Memory Usage
6. CPU Usage
7. Framework Usefulness.

Criteria 1 and 3 were intensively investigating using different IDEs [4] and criteria 1, 3 to 6 were used for a multithreading craps game simulator developed using Java,

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 8, Issue 3, September 2018

Objective-C in order to run on Android and iOS platforms [6].

II. DESIGN AND IMPLEMENTATION

The application achieves the minimum required features as an e-commerce template. It contains functions such as display categories, display items in each category, add item into cart, display items in the cart, add/subtract/remove items in cart and empty cart. This application simulates the regular e-commerce applications in order to compare the app performance developed on different devices using PhoneGap and Xamarin. There is no server side. The application runs locally only in order to eliminate the computer network traffic. The execution time of each action is measured as well as lines of code, CPU usage and memory impact.

A. Application Design

1) PhoneGap

The PhoneGap built application is written by HTML, CSS and JavaScript. It is a small website and packaged into a mobile application through PhoneGap builder. We use the default XML created by PhoneGap builder. The item displayed on page is dynamic which means all categories use the same page to display their items. The content shows different items due to different category entrance button which the user clicks. The shopping cart function is achieved by using JSON (JavaScript Object Notation) and local Storage statement, which is a new feature in HTML5. There is no global variable between webpages, so we use local Storage and JSON to pass values. The local Storage stores strings of item's information and JSON converts them into objects.

2) Xamarin

The Xamarin built application is written by C#. Since there is global variable available between different pages for C#, there is no need to use JSON here. It only increases the run-time. We've created a cross-platform application using only PCL (Portable Class Libraries), which means the Xamarin built application on different devices and platforms uses the same source code. The shopping cart function is achieved by using Observable Collection statement.

B. Measurement

We simulate the shopping process of a client. Therefore, we test the following seven most important actions by measuring execution time:

1. Load the category page
2. Add an item into shopping cart
3. Show the shopping cart page
4. Increase the quantity of an item in the cart page
5. Decrease the quantity of an item in the cart page
6. Remove an item from shopping cart
7. Empty the shopping cart.

Every execution time of each action is calculated by the program and distribute to corresponding variables for storage, which is displayed in the developer's page. In the Xamarin-built application, we use the built-in Stopwatch object to calculate the elapsed time, while in the PhoneGap built application; we calculate the execution time difference between the beginning and the ending of the code fragment as the elapsed time. Hence, all time is automatically recorded by the program itself. The tester needs to go to developer's page to read the result. The test applications were running exclusively on the brand-new devices with preset system tasks.

We use system built-in tools to measure the memory usage and CPU usage. Due to system limitation, the measurement built-in tools are only available on the Android platform. As a result, we only use the result from Android platform into our consideration.

For the lines of code, we do ***not*** count the auto generated lines of code. Comments and blank lines are ***not*** counted as well.

To compare the launch time, we measure the cold start, warm start and lukewarm start time separately. A cold start refers to "an app's starting from scratch: the system's process has not, until this start, created the app's process" [1]. We test the cold start launch time by rebooting the devices every time before launching the application. A warm start refers the process when all of the application's activities are still resident in memory; the system brings the activity to the foreground [1]. We measure the warm start launch time by closing the application then relaunching it. A lukewarm start example is like "the user backs out of the app, but then re-launches it. The process may have continued to run, but the app must recreate the activity from scratch via a call to on Create ()" [1]. Hence, we test the lukewarm launch time by backing out (the back button) of the application then re-launch it.

The phone devices used for testing are listed in Table I.

C. User Interface

Each application needs only **four** pages: Home Page, Category Page, Cart Page and Developer's Page. Home Page shows 4 different categories. The Category Page displays all items under this category. Though 4 categories exist in the application, they share the same one page. It means that the content of Category Page is dynamic. The Cart Page shows shopping cart content and Developer's page shows the result of elapsed time of each action.

The screenshots of user interface are listed in Table II.

III. COMPARISON

A. Testing Sequence

The following testing sequence is applied:

The first repetition (an elapsed time is automatically recorded for each operation in each step and it is shown on the developer page):

1. Load the e-commerce application.
2. Show Category Page: Click the Fashion image, then go back for 3 times (the elapsed time is automatically recorded each time).

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 8, Issue 3, September 2018

3. Add Item into Cart: Add each item into cart. In total 6 items, so program records elapsed time for 6 times.
4. Show Cart: Go to the cart page
5. Add item Quantity in Cart: add each item's quantity by 1 for the 6 items.
6. Subtract Item Quantity in Cart: Then subtract each item's quantity by 1, so you get one in quantity for each item.
7. Remove Item from Cart: Use the Del button to delete items one by one. Each item disappears from the cart page. Record each operation.
8. Go to the developer's page and press the "Add 6 Items" button. The shopping cart adds one in quantity for each item in the Fashion category.
9. Go to the cart page
10. Empty Cart: Empty the shopping cart, repeat 3 times step 8-10
11. Repeat steps 1-10 for the second repetition
12. All results of the elapsed time for each action are summed up and calculated an average.

Repeat all the above steps for each e-commerce app (PG Build App and XA Build App) for five different phone devices (Samsung S8, Samsung Note8, Google Pixel 2, Apple iPhone 8, Apple iPhone X).

For the launch-time measurement, the Android Studio should be connected via USB port with the particular phone device. All the results are read from logs via Android Studio.

All averages for two functional equivalent e-commerce apps running on the five different phone devices are shown in Table III. Each average was calculated based on 6 instances of the same action.

The following testing sequence is applied:

1. Close all applications and shut down the device.
2. Start the device. After that, open the application and read the launch-time.
3. Close all running applications and reopen the application and read the launch-time.
4. Press back button to exit the application then re-launch it and read the launch-time.
5. Repeat step 1 - 4 for in total 3 times
6. All results of the launch-time are calculated and available from the log are read via Android Studio.
7. Calculate all averages.

The launch-time results are shown in Table IV.

A. Figures and Tables

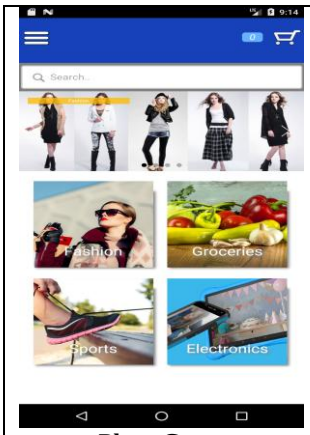
Test results are shown in Table III to V. Fig. 1 to 4 show charts based on the results in Table III and IV.

Table I: Device Specification

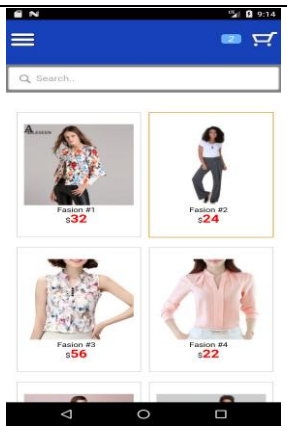
Device Name	Chipset	CPU	Memory	Storage	Operating System
Apple iPhone 8	Apple A11 Bionic	Hexa-core 2.39 GHz (2x Monsoon + 4x Mistral)	2GB RAM	64GB	iOS 11.1.2
Apple iPhone X	Apple A11 Bionic	Hexa-core 2.39 GHz (2x Monsoon + 4x Mistral)	3GB RAM	64GB	iOS 11.2.1
Samsung S8	Qualcomm MSM8998 Snapdragon 835	Octa-core (4x2.35 GHz Kryo & 4x1.9 GHz Kryo)	4GB RAM	64GB	Android 7.0
Samsung Note8	Qualcomm MSM8998 Snapdragon 835	Octa-core (4x2.35 GHz Kryo & 4x1.9 GHz Kryo)	6GB RAM	64GB	Android 7.1.1
Google Pixel2 XL	Qualcomm MSM8998 Snapdragon 835	Octa-core (4x2.35 GHz Kryo & 4x1.9 GHz Kryo)	4GB RAM	64GB	Android 8.1.0

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 8, Issue 3, September 2018

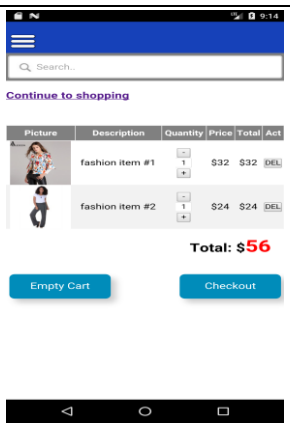
Table II: Screenshots of User Interface



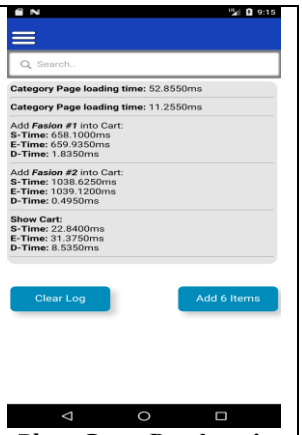
PhoneGap – Home Page



PhoneGap – Category Page

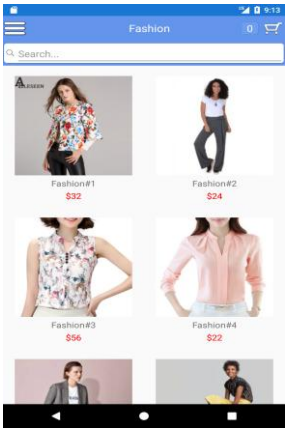


PhoneGap – Cart Page

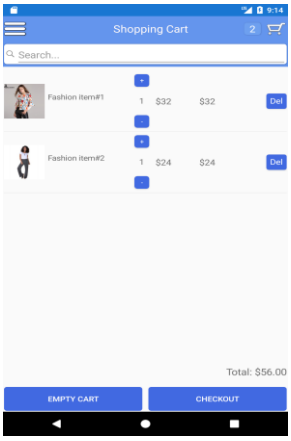


PhoneGap – Developer's Page

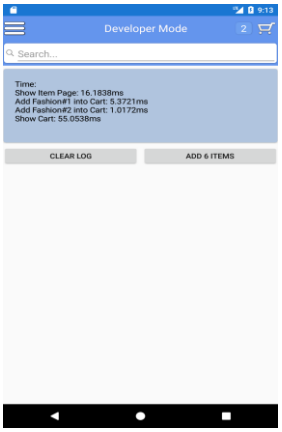
Xamarin – Home Page



Xamarin – Category Page



Xamarin – Cart Page



Xamarin – Developer's Page

Table III: Action Time Results

(Bold items in yellow mean the best in the category/column and underlined items mean the best within the same device – two consecutive rows, for each column)

Actions / Devices	Action [unit: ms] 1 Show Items Page	2 Add Item into Cart	3 Show Cart	4 Add item Qty in Cart	5 Sub item Qty in Cart	6 Remove Item from Cart	7 Empty Cart
Samsung S8/ PhoneGap	42.6900	1.2129	13.6308	1.3879	1.3617	2.8396	4.3233
Samsung S8/ Xamarin	6.4968	0.9682	8.7853	0.6706	0.5057	4.1841	6.7118
Samsung Note8/ PhoneGap	48.0667	0.8500	10.6167	1.2417	1.0250	2.0750	3.0333
Samsung Note8/ Xamarin	6.0320	1.0022	7.3931	0.5179	0.4349	4.8761	7.1302
Google Pixel2 XL/ PhoneGap	24.3917	**0.6871**	9.6558	0.7996	0.7463	**1.4850**	2.2242
Google Pixel2 XL/ Xamarin	**5.0918**	0.8418	7.1148	0.4378	0.3602	3.8857	4.1940
Apple iPhone 8/ PhoneGap	8.8000	1.4167	7.0000	1.6750	0.9667	2.1250	2.9500
Apple iPhone 8/ Xamarin	7.1455	0.8958	8.0489	**0.2927**	0.2645	6.1263	**1.0146**
Apple iPhone X/ PhoneGap	7.2000	1.0833	**6.3167**	1.8833	1.3667	1.8333	2.6833
Apple iPhone X/ Xamarin	8.6243	0.7203	7.6835	0.3212	**0.2626**	6.3384	1.0887

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 8, Issue 3, September 2018

Table IV: Launch-Time Results

Launch-Time [unit: ms]				
Launch-Time / Devices	Cold Start Avg.	Warm Start Avg.	Lukewarm Start Avg.	Total Launch-Time Avg.
Samsung S8/ Xamarin	1958.00	1696.33	253.00	1302.44
Samsung S8/ PhoneGap	786.33	351.00	114.67	417.33
Samsung Note8/ Xamarin	1854.33	1710.67	239.33	1268.11
Samsung Note8/ PhoneGap	**422.00**	**299.33**	125.00	**282.11**
Google Pixel2 XL/ Xamarin	1926.00	1818.67	260.00	1334.89
Google Pixel2 XL /PhoneGap	518.67	322.00	**101.33**	314.00

Table III: Launch-Time Results

Criteria / Devices	Lines of Code (LOC)	Application Size [MB]	Average Memory Usage [MB]	Size/Memory Ratio	Average CPU Usage [%]	Framework Usefulness
Samsung S8/ PhoneGap	1730	**5.64**	98.42	0.057	11%	**PhoneGap -** better compatibility; uses third-party IDE; good for small app development
Samsung Note8/ PhoneGap			154.92	0.036	8%	
Google Pixel2XL/ PhoneGap			175.44	**0.032**	4%	
Samsung S8/ Xamarin	**1338**	29.7	**83.7**	0.355	7%	**Xamarin** – powerful first-party IDE; provides various configurations; good for medium and big size app development
Samsung Note8/ Xamarin			125.34	0.237	**3%**	
Google Pixel2XL/ Xamarin			164.32	0.181	**3%**	

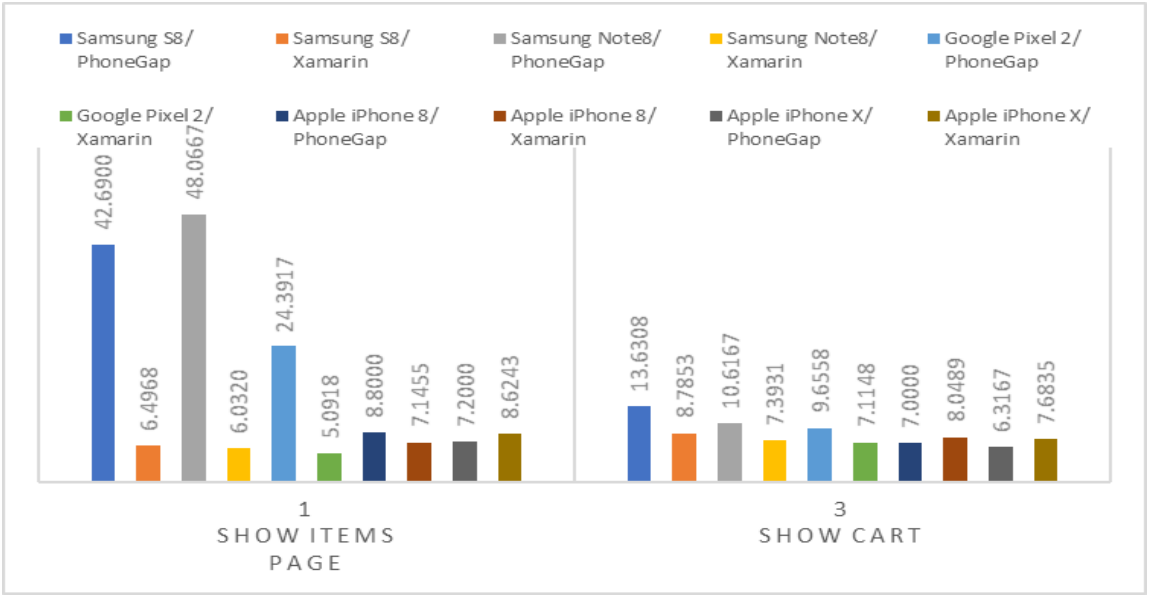


Fig. 1: Comparison of Phone Devices (a) (unit: ms)

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 8, Issue 3, September 2018

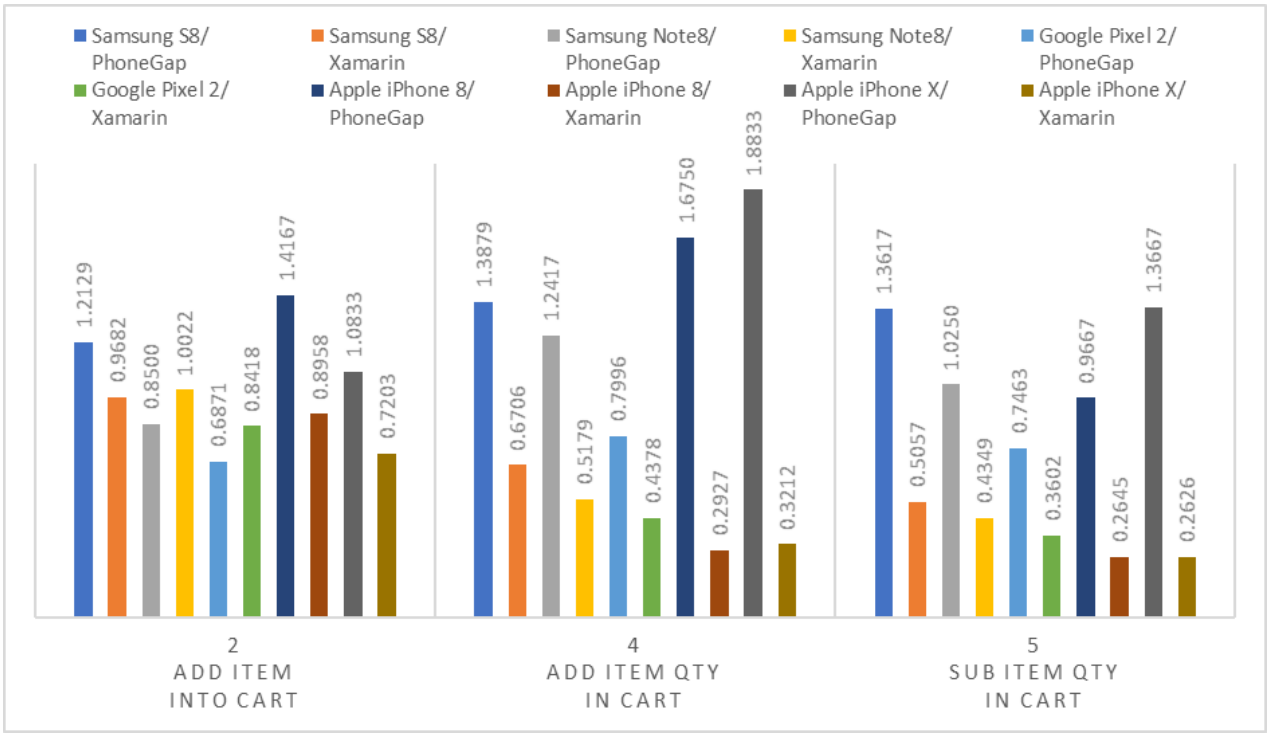


Fig. 2: Comparison of Phone Devices (b) (unit: ms)

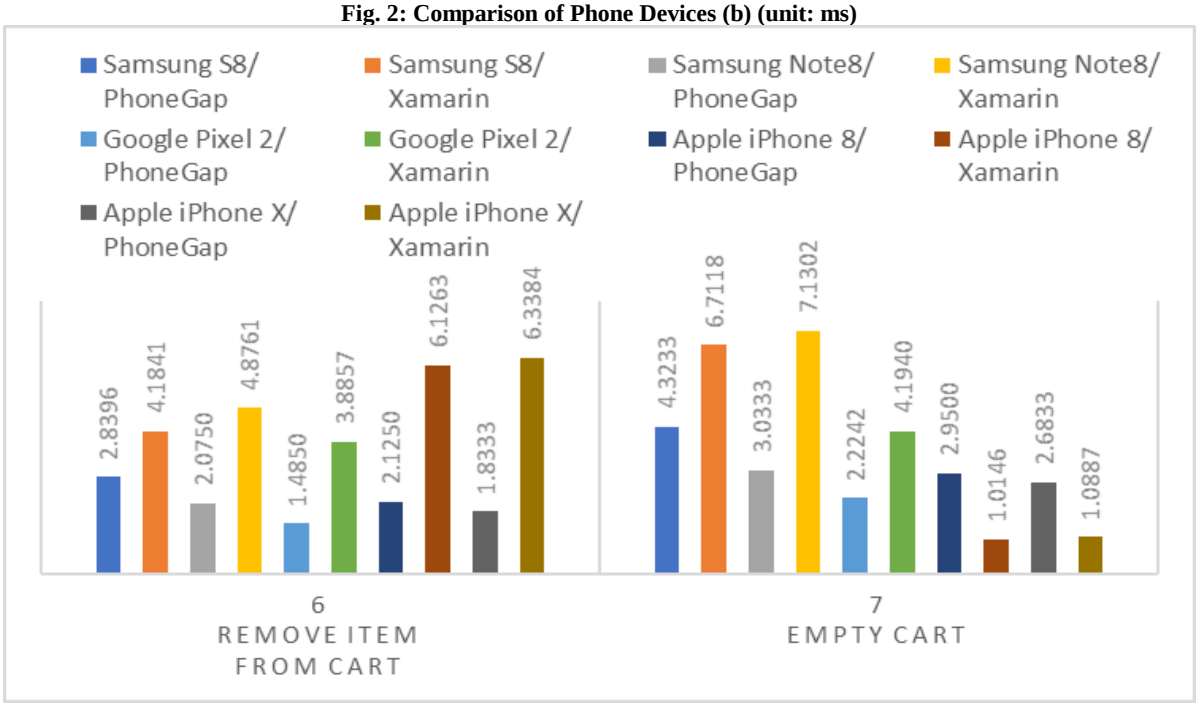


Fig. 3: Comparison of Phone Devices (c) (unit: ms)

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 8, Issue 3, September 2018

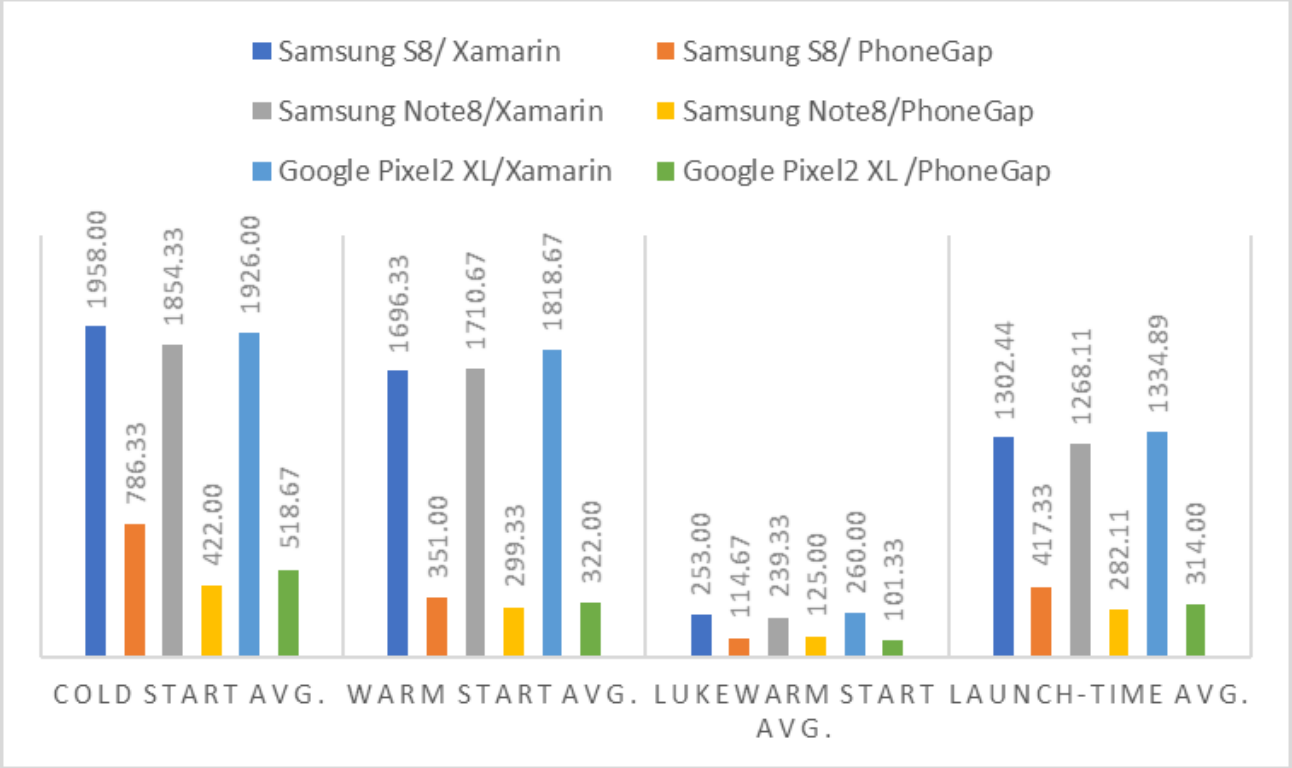


Fig. 4: The Launch-Time Chart (unit: ms)

A. Analysis

From the action **results**, we can conclude that **Xamarin** built app performs generally better than PhoneGap one in Android platform (Table III, 22 Xamarin cells underlined vs 13 PhoneGap cells. Fig. 1 and 2). The differences in page loading time are determined that the Xamarin.Forms is a cross-platform that can be shared across Android and iOS and it is used to create the optimized app interface in the Xamarin framework. The hybrid HTML is used to create the general app interface in the PhoneGap framework.

The **PhoneGap** built app performs better in the **launch-time** with default setting and no special optimization (Table IV, Fig. 4) and the **application size** of **PhoneGap** built app is almost 6 times less than the Xamarin one (Table V). The Xamarin built application's package must include the application, the associated libraries, the content, the Mono runtime, and the required Base Class Library (BCL) assemblies [12]. PhoneGap is smaller in application size, but it takes relatively more system resource when runs (Table V). For the developers, it is still **more convenient** to use **Xamarin** to develop applications. When developers use data binding, they write much less codes than using PhoneGap. Developers also can set a series of configurations by choosing the options provided by Visual Studio. There are no such options in PhoneGap, and developers have to create their own XML files to set the configurations. It is much more complicated (Table V - column Framework Usefulness). However, the **compatibility** of **PhoneGap** built application is better.

B. Improvement in the Future

In the future, we plan to use the Monkey testing to test the stability of the application. The Monkey testing is a technique where a program stimulates the users and generates series of random input to test whether an application will crash [2]. With this technique, we can determine which framework will generate more reliable applications.

IV. CONCLUSION

Considering the **execution time**, the **Xamarin** built application in comparison to the PhoneGap built application performs better in almost all cases running on the same phone device - at least in four categories (actions) for each device (Table III – compare all underline items), even though it loses at the Remove Item from Cart action (Table III column 2). With regard of the **execution time for the phone devices**, **Pixel2 XL** wins in three categories (actions) and each **iPhone 8** and **X** wins in two categories (underlined in yellow items in Table III). The **PhoneGap** built application has the advantage of small application **size** but users will not care too much about it because today smart phones generally come with large internal storage (Table V). The **Xamarin** and **PhoneGap** built applications running on the same phone device are relatively close for the **memory** and **CPU usage** (Table V) even though PhoneGap or Xamarin built application running on Samsung 8 requires the lowest memory usage in comparison to all test devices.

However, users do care about the **launch-time.** The **PhoneGap** built application wins with big lead. When we compare the total average **launch-time** for each phone

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 8, Issue 3, September 2018

device, the PhoneGap built application running on Note8 has the lowest one (Table V, Fig. 4). It seems that the reason for that is that the Note8 has the biggest memory capacity (6GB, Table I), second, it uses Android 7.1.1 which is more stable than the most updated one in Pixel2 XL (Android 8.1.0).

The process to build/debug application is way more stable in PhoneGap. However, it is very convenient that we could use IDE which is Visual Studio in this case for Xamarin. Visual Studio is powerful but still not stable enough when using Xamarin to develop cross-platform applications. However, it is more convenient to use Xamarin to develop applications. Developers can write **less codes** than using **PhoneGap** (Table V, Lines of Code column). Besides that, the **maintenance** is easier by using **Xamarin**. Developers can easily have individual settings for different platforms and devices. We can use IDE for PhoneGap too (for example WebStorm by JetBrains), but it is already a third-party software, not as good as Visual Studio.

The both two development tools look very good. For simple small mobile application or small companies, PhoneGap is a good choice. However, for the large complex mobile applications, the Xamarin will always be a better option with Microsoft's Visual Studio on its back.

ACKNOWLEDGMENT

This research was supported by the Department of Computer Science and Computer Engineering at the University of La Verne.

REFERENCES

[1] Android Developers, "Launch-Time Performance", https://developer.android.com/topic/performance/launch-time.html, 2018.

[2] Android Developers, "UI/Application Exerciser Monkey", https://developer.android.com/studio/test/monkey, 2018.

[3] I. Dalmasso, S. K. Datta, C. Bonnet, "Survey, comparison and evaluation of cross platform mobile application development tools", The 9th International Wireless Communications and Mobile Computing Conference (IWCMC), 2013.

[4] J. Goetz, D. Michalopoulos, and J. Pangilinan, "E-Commerce Technology Comparison: ASP.NET vs. Ajax", Current Computing Developments in E-Commerce. Security, HCI, DB, Collaborative and Cooperative System, published by ATINER, ISBN: 960-6672-07-7, pp. 471-488, 2006.

[5] J. Goetz, D. Michalopoulos, and E. Reao, "Validating E-Commerce Solutions, Computer Science and Information Systems", published by ATINER, ISBN: 960-88672-3-1, 221-236, 2006.

[6] J. Goetz, M. Ruvacaba, "Mobile Application Performance for Different Platforms", Proceedings of International Research Conference on Engineering and Technology, Kitakyushu, Japan, June 6-8, 2017, 62-71, 2016.

[7] Y. Li, "Mobile Application Performance Comparison", unpublished, Department of Computer Science, University of La Verne, USA.

[8] L. Oberg, "Evaluation of Cross-Platform Mobile. Development Tools. Development of an Evaluation Framework". Master Thesis. Umea University, Department of Computing Science, 2015.

[9] PhoneGap, https://phonegap.com/, 2018.

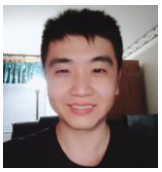
[10] M. Prajapati, D. Phadake, A. Poddar, "Study on Xamarin Cross-Platform Framework", International Journal of Technical Research and Applications e-ISSN: 2320-8163, Volume 4, Issue 4 (July-Aug, 2016), pp. 13-18, 2016.

[11] V. Tunali, S. Erdogan, "Comparison of Popular Cross-Platform Mobile Application Development Tools", https://www.researchgate.net/profile/Volkan_Tunali2, 2015.

[12] Xamarin, Inc.. All Guides. https://developer.xamarin.com/guides/, 2018.

AUTHOR BIOGRAPHY

Jozef Goetz received his M.S. degree in Computer Science, minored in Electronics from Wroclaw University of Technology, Poland. He received his Ph.D. in Computer Science with Honors from Cybernetics Institute of Wroclaw University of Technology. He is currently a professor in Department of Computer Science, University of La Verne, US. Jozef Goetz has a balanced academic experience as well as technical, hands-on, 12-year experience in industry as a senior software developer engineer at Fujitsu. His broad academic experience consists of teaching and research first at the Wroclaw Polytechnic University, Poland, then the California State University, Fullerton, US, finally the University of La Verne. His interests include WEB and mobile app development, Website design, design best practices, e-commerce, Web analytics, programming and modeling systems using Petri Nets.



Yan Li, Computer Science major, concentration in Internet Programming, University of La Verne, USA. His interests include WEB and mobile app development, mobile app performance, C++ and C# programming.