An Analysis of Android Web App Manifest

Yahya M. Tashtoush, Ashraf Alsmadi, Amnah Al-Abdi, Noor Ababneh, Omar Almousa
Jordan University of Science and Technology
Irbid, Jordan.
{yahya-t, osalmousa}@just.edu.jo,{amalsmadi16, amalabdi15, nmalababneh16}@cit.just.edu.jo

Abstract—Nowadays, mobile applications are provided in several types. One type of mobile applications is mobile web application manifest (a standard of World Wide Web Consortium (W3C)). It is based on manifest metadata that enables developers to include metadata information about a web application into JSON (JavaScript Object Notation) datatype file. Such information provides developers with the capability to declare properties in order to control the behavior of web applications. For example, the manifest file may specify a labeled icon for an application when it is added to the home screen of a device. In this paper, we statically analyze a large dataset of JSON metadata files that we collected from mobile web applications. Our analysis includes some of the sub-properties from the collected metadata files such as display, background_color, icons, lang, orientation, permissions, and theme_color. Our results highlight most commonly used features and properties in mobile web applications.

I. INTRODUCTION

On Android platforms, mobile applications (apps) can be delivered in two ways: client-side apps, or web apps. In client-side apps, an app is developed using the Android Software Development Kit (SDK), and installed on user devices using Android Package Kit (APK). On the other hand, web apps are client-server computer programs in which a client, including client-side logic and the user interface, runs using embedded web browser [1].

Common examples of web applications are webmail, online auctions, online retail sales, instant messaging services, wikis and others [2]. The main distinction between a web application and a dynamic web page of any kind is unclear> However, websites are most likely to be indicated as web applications which have similar functionality to a mobile app, or to a desktop software application [2].

In a web application, there are several ways of targeting mobile devices in the mobile web: responsive web design and progressive web apps. Responsive web design is used for making a web application, mobile apps or native apps that directly run on a mobile device. In another hand, a progressive web app is a hybrid of a mobile application and a regular web page (website), in which a mobile website is embedded inside a native app [3], [4].

The web app manifest contains information (metadata) about an application such as name, theme_color, icon, author, permissions, etc.. This information is stored in a JSON file that is a document format used for "lightweight" data exchange [5]. The main goal of a manifest file is to store a web application like regular application in the home screen of a device, i.e., a site bookmark that acts like installable application. It helps in giving users better experience and quicker access [6]. Web app manifest gives developers an ability to control the application appearance to users in regions that they expect, guidance of what users can launch, and determine the appearance at launching. When a site is launched it has a unique name and an icon so users can differentiate it from other sites, it shows users something while resources are restored or downloaded from cache, it gives default display features to the browser to prevent abrupt transition when a site is available, and it does all this over simple techniques of metadata in a JSON file, which called the web app manifest [6].

Web technologies involve web app manifest as progressive web apps that can be downloaded to a device without using the application store or other third-party software. In addition to that, it enables other features like getting push notifications, being available when offline, and controlling the display mode of a device. An example of a manifest web application is shown in Fig.1, which provide a JSON structure of an application called "Start Magazine" and it has three icons with different sizes and "start_url" property which represent the path that the application will use when the application is launched. This application has other properties which we describe later in this research.

```
{
    "short_name": "Start Magazine",
    "name": "Celltick's Start Magazine",
    "icons": [
        {
            "src": "assets/favicon_16_16.png",
            "sizes": "16x16",
            "type": "image/png"
        },
        {
            "src": "assets/favicon_128_128.png",
            "sizes": "128x128",
            "type": "image/png"
        },
        {
            "src": "assets/favicon_196_196.png",
            "sizes": "196x196",
            "type": "image/png"
        }
    ],
    "start_url": "index.html",
    "display": "standalone",
    "orientation": "portrait",
    "background_color": "#fff",
    "theme_color": "#000",
    "add_to_home_enabled": true
}
```

Fig. 1. Manifest JSON file application

The difference between a web bookmark and a mobile web application is that a mobile web application can be restricted by metadata, and thus shown in a device like a native application with an icon and a label. This makes it hard to distinguish between an installed application and a mobile web application [7].

In this paper, our target is to analyze the “manifest.json” files for a large dataset of web apps to extract some useful statistical information for a set of selected properties. The rest of the paper is organized as the follows: Section 2 shows some of the related works. In Section 3 we describe the steps we performed to collect and process our dataset. Section 4 is to show our experiments, results, and a brief discussion of them. The conclusion and proper future work are represented in Section 5.

II. Related work

David et. al [8] had mainly two goals from their paper. Firstly, they gave an explanation to how well-known web deployment methods can be used in developing some map-based apps. Secondly, they explained how the deployment of map-based apps can take benefits from geolocation services. The authors focused on the development of two web app prototypes: CDU Flashback and MyMap. Then they provided a comparasion between the development of the two in contrast to the deployment of native apps. They discussed that the usage of certain technologies for building web apps such as HTML5, CSS3, and JavaScript will make the development process of a web-based app rather than building a native app. especially for non-programmer developers. The authors concluded that web apps will be the trend to developers in the future with the rapidly expand of web technologies.

William et. al [9] made a comparison between native apps and mobile web apps to invistigate the viability of exchanging native application with the mobile web application. For testing, the authors used two mobile web applications; one to track run and the other work for scheduling "slum runs" as booking system. After testing, the result indicated that native application performed much better than mobile web application because of the poor GPS performance in mobile web app. The authors concluded that native applications are the best for hardware intensive applications such as GPU, camera or GPS applications.

Charland et. al [10] discussed the native code vs. web code in term of user interface code, user experience, performance, and design. Henning Heitk¨otter et. al [11] developed eleven criteria to evaluate mobile web frameworks, the need of developing these criteria is due to the difficulty of selecting an appropriate framework among a plenty of them. The authors classified the proposed criteria into two categories; developers’ presepctive and user’s perspective. Seven of the proposed criteria are from a developer perspective, namely: license and costs, long-term feasibility, documentation, support, learning success, development effort, extensibility, and maintainability. The remaining four that are from users’ perspective are: user interface elements, native look and feel, load time, and runtime performance. After defining the criteria, the authors then used it to evaluate four mobile web frameworks: jQuery Mobile, Sencha Touch, The-M-Project, and Google Web Toolkit. The results showed that jQuery Mobile is the best for mobile UI, while Sencha Touch is appropriate for mobile web applications with high complexity.

Serrano et. al, [12] examined different current approaches for developing mobile web apps. They aimed at helping developers to make a correct decision when they need to select an approach. The authors classified these approaches into five types: standard web apps, responsive web apps, mobile web apps, hybrid apps, and native apps. The authors defined criteria so the developers can use it in order to decide which of the above-mentioned approaches will be more suitable to their situation than the other approaches. Their criteria were classified into technical and non-technical parts. Technical criteria are: platforms and version support, device capabilities, user experience, performance, and upgrade. On another hand, the non-technical criteria are: distribution, approval cycle, and monetization.

Ivano Malavolta et. al [13] described the content of a tutorial for web-based hybrid mobile applications. The tutorial discussed the problems occurred when development and maintenance of mobile apps are platform dependent, i.e., different platforms use different programming languages and tools. Web-based hybrid mobile apps are the solution proposed by well- known companies such as IBM and Adobe. The proposed solution enables the developers to use standardized web technologies, then use cross-platform wrappers and other tools to spread them in app stores. Many benefits will be gained using the proposed solution such as the ability to use cross-platform, exploitation of existing knowledge of developers, simplicity, and cost-effectiveness. However, there are some drawbacks such as constrained access to hardware, the differences between user's experiences, and decreasing in performance.

Sheppard et.al [14] provided a perfect tutorial to describe how to add an application into a mobile home screen, and how it can be controlled using the web application manifest file. The authors described the properties that can be controlled using manifest.json file. As a result of following the tutorial, a web app will be very similar to its corresponding native app; it will launch faster, loaded faster, work as offline apps, and will be launch from home screen.

Biørn et. al [15] provided a way to classify and unify native applications and web applications. Authors took the testing of performance and design as parameters to make compensation. Also, they have provided a feature comparison for Progressive Web Apps. The authors find that progressive mobile apps will be the main way to unify the development of web applications, and the end user will not notice any difference between regular applications and Progressive Web Apps.

III. Dataset: accumulating and prepossing

We collected the dataset as domain names from “publicwww” [16]. “publicwww” is a website that has a search engine that can access the source code (HTML) of websites over the internet. To cover a big dataset, we used this query keyword:

"<link rel=\"manifest\" href=\"manifest.json\">"

This query returns the domain name that has “manifest.json” file in the domain root directory to a CSV file (domain, rank). The returned collected dataset has 19222 domains.

After that, we preprocessed the domain names stored in a CSV file into MySQL database using a script that we called “read.php”. This script stores the dataset into a programmer friendly style. The MySQL database has three tuples (ID, URL, Rank). ID is auto-generated for each domain. The URL

contains the domain name. Rank contains the domain rank over the web.

After that, the list of domain names has been crawled using "crawler.php" script. The "manifest.json" files crawled and stored in a dataset directory with their ID from the database as a filename.

Finally, the dataset has been reviewed and bad results were removed such as files do not have normal "manifest.json" file structure. The final total preprocessed dataset has 14784 files.

To perform this experiment, we use PHP version 5.6 of PHP-CLI (PHP command line interface) and PHP-FPM over an Nginx web server that was installed on Ubuntu 14.04 server.

IV. Experiments and Results

After collecting and processing our dataset, we start the decoding stage. The "json.php" file decodes the dataset JSON files into array data type. The code algorithm counts array keys (property of JSON file) and values for each JSON file. As a result of the first iteration that counts the JSON properties of arrays we found that 62810 main properties in all JSON files. This is shown in Fig.2 in which we excluded properties that count less than 200. In Fig.2, the y-axis shows the names of properties, while the x-axis shows the count of each property. On top of each column, we have the percentage of a property in our data set. One can notice that most of the mobile web applications have "name" property and 15.3% don't have icons. The rest of results displayed in the Fig.2.

After counting primary properties, we have selected some of these properties for further analysis:

A. Display

The display property has four main types:

1) Standalone
Makes an application to act like a native application.

2) Browser
Makes an application to act as a bookmark of a webpage.

3) Fullscreen
Makes an application to launch in fullscreen.

4) Minimal-UI
Similar to Fullscreen property, but with less content in the user interface and minimum navigation buttons.

The final results show that the display's distribution as follows: standalone: 96.03%, browser: 2.38%, Fullscreen: 1.05%, and minimal-UI: 0.54%. This result exhibits that most of mobile web applications have standalone display.

B. Orientation:

This property restricts the application to a specific orientation. The results are as follows: portrait: 69%, natural: 12.4%, any: 8.6%, landscape: 6.4%, portrait-primary: 3.4%, and landscape-primary: 0.2%. The results show that most mobile web applications use portrait orientation mode.

C. Lang

This property declares the language of an application in ISO representation. The language with their localization of applications is English with a ratio of 47% of the dataset.

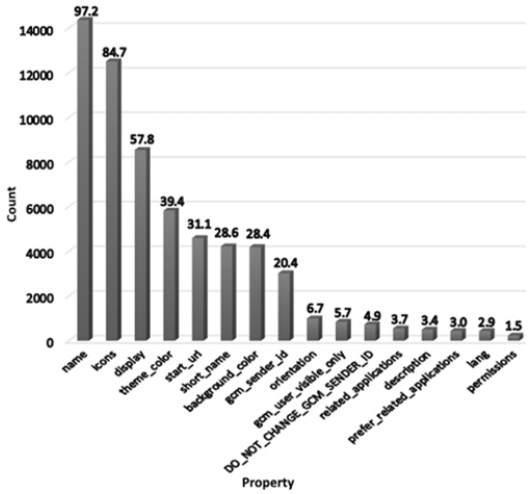

Fig. 2. Main properties

TABLE I. Permission property results

Permission	Percentage
Gcm	58.45%
Notifications	19.20%
Storage	14.61%
pushMessaging	2.29%
webRequest	0.57%
webNavigation	0.29%
background	0.29%
tabs	0.29%
cookies	0.29%
webRequestBlocking	0.29%
activeTab	0.29%

D. Permission

This property identifies the special permissions that applications request from the device. Applications that use the "gcm" (Google Cloud Messaging) are about 58.45% of all that request special permissions, 19.2% use notifications permission, and 14.61% use the storage of the device. The rest permissions appear in Table I.

E. theme_color

This property provides a default color of the theme in the context of an application. The results are: 62.44% for white color, 2.61% for black color, and 1.61% for steel blue color.

F. background_color

This property describes the predictable color of the background in an application. The results are: 64.5% for white color, 4.1% for gray color, 2.2% for light steel blue color, and 1.7% for black color.

G. Icons

This property represents an iconic image of an application in a mobile. The property has three sub-properties: (1) src: to identify the image file location and all images have src, (2) size: to describe the icon size, and (3) type: to classify the icon meta-type.

The results of the sub-property (size) are presented in Table II. The results of the sub-property (type) are as follows: the value "image/png" has 99.2%, the value "image/jpeg" has 0.21%, and all other types have percentage less than 0.2%.

TABLE II. ICON SIZE PROPERTY RESULTS

Size	Percentage
192x192	21.0%
144x144	14.1%
96x96	13.4%
72x72	12.9%
48x48	12.9%
36x36	12.2%
512x512	4.4%
256x256	3.1%
384x384	1.2%

V. CONCLUSION AND FUTURE WORK

In this work, we provide a statistical analysis of mobile web application metadata files for a large dataset. This statistical information is useful for the web application developers to detect which properties are used in most web apps. The results show that a lot of web apps do not use all the properties of metadata. However, the more important result is that a lot of them have missed important properties. For example, the applications that do not have "icon" property are around 15.3% which is an important property.

For future work, our dataset can be used for many other purposes rather than only providing statistical information. For instance, a classification analysis can be considered using different classifiers and compare between them according to some criteria such as accuracy, recall, precision, and f-measure. Moreover, instead of only providing an abstract statistical information about the "manifest.json" file properties, we may investigate the relationships between different properties and their distribution. This can be done in order to detect reasons behind selecting the most used properties by the developers.

REFERENCES

[1] X. Li, and Y. Xue, "A Survey on Web Application Security," *Nashville, TN USA*, Vanderbilt University, 2011.

[2] S. Al-Fedaghi, "Developing web applications," *Int. J. Software Engineering and Its Applications.*, vol. 5, no. 2, pp. 57–68, 2011.

[3] A. I. Wasserman, "Software Engineering Issues for Mobile Application Development," in *Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research*, New York, NY, USA, 2010, pp. 397–400.

[4] A. Connors, Google, B. Sullivan, and AT&T, "Mobile Web Application Best Practices," 2010. [Online]. Available: https://www.w3.org/TR/mwabp/. [Accessed: 16-Dec-2017].

[5] P. Carter, "Understanding JSON". In *SQL Server Advanced Data Types,* Berkeley, CA. Apress, pp. 181-200. 2018.

[6] M. Gaunt and P. Kinlan, "The Web App Manifest | Web Fundamentals," *Google Developers*. [Online]. Available: https://developers.google.com/web/fundamentals/web-app-manifest/. [Accessed: 16-Dec-2017].

[7] M. Caceres, K. R. Christiansen, M. Lamouri, A. Kostiainen, and R. Dolin, "Web App Manifest." [Online]. Available: https://w3c.github.io/manifest/. [Accessed: 16-Dec-2017].

[8] D. Sin, E. Lawson, and K. Kannoorpatti, "Mobile Web Apps - The Non-programmer's Alternative to Native Applications," in *2012 5th International Conference on Human System Interactions*, 2012, pp. 8–15.

[9] W. Jobe, "Native Apps Vs. Mobile Web Apps," *Int. J. Interact. Mob. Technol. IJIM*, vol. 7, no. 4, pp. 27–32, Oct. 2013.

[10] A. Charland and B. Leroux, "Mobile Application Development: Web vs. Native," *Commun ACM*, vol. 54, no. 5, pp. 49–53, May 2011.

[11] H. Heitkötter, T. A. Majchrzak, B. Ruland, and T. Weber, "Evaluating Frameworks for Creating Mobile Web Apps.," in *WEBIST*, 2013, pp. 209–221.

[12] N. Serrano, J. Hernantes, and G. Gallardo, "Mobile Web Apps," *IEEE Softw.*, vol. 30, no. 5, pp. 22–27, Sep. 2013.

[13] I. Malavolta, "Web-Based Hybrid Mobile Apps: State of the Practice and Research Opportunities," in *2016 IEEE/ACM International Conference on Mobile Software Engineering and Systems (MOBILESoft)*, 2016, pp. 241–242.

[14] D. Sheppard, "Adding your App to the Home Screen with Web App Manifest," in *Beginning Progressive Web App Development*, Apress, Berkeley, CA, 2017, pp. 95–107.

[15] Biørn-Hansen, A., Majchrzak, T. A., & Grønli, T. M. Progressive web apps: The possible web-native unifier for mobile development. In Proceedings of the 13th International Conference on Web Information Systems and Technologies(pp. 344-351).

[16] "Search Engine for Source Code - PublicWWW.com." [Online]. Available: https://publicwww.com/. [Accessed: 16-Dec-2017].