A Framework for Socialisable and Expirable Mobile Apps Built by Non-programmers

Hin-Wah Yip, Karl Kwan, and Ying-Ming Ng

***Abstract*—The framework proposed and implemented in this paper includes a web-based app builder which allows non-programmers to build personalized mobile applications and mechanisms to allow these applications to be shared easily, hence socialisable. Another key feature of the framework allows the created applications to be tagged with an expiry date and then removed from the mobile device once that date is reached, hence expirable. There are many similar mobile applications builder [1] but none offered both the socialisable and expirable features. Key to the framework that facilitate the support of these features are the Datasource (DS) concept, the Application Engine (AE) that provides the run time environment in the mobile device and a server with cloud database connectivity (SC) that housed sharable data relevant to the applications running in the mobile device.**

***Index Terms*—Applications, expiry, media, mobile, personalized, sharable, share, social.**

I. Introduction

Applications are the key driving force behind the proliferation and pervasiveness of mobile smartphones. With more than 1 million applications on both the Google Play [2] Store and Apple App Store [3], [4], and the ever increasing trend of downloads [5], [6], smartphones can be flooded with applications. This is compounded by the effect in social media where information about good applications spreads speedily, further congesting a smartphone's limited memory. The socialisable and expirable framework proposed and explained by this paper not only allows non-programmers to build their own personalised mobile applications easily, it also simplifies the sharing of applications and allows these applications to be expired automatically when users do not need them anymore.

With funding from the Ministry of Education (MOE), Singapore, the authors have started this project in April 2013 to design and implement a sustainable solution for non-programmers to create personalized mobile apps based on their own unique requirements and be able to share them easily [7], [8].

II. Overall Operational Architecture

Fig. 1 shows the overall architecture of the framework where the user using the web-based app builder to build a mobile application changes the status of his project to "publish" once he has completed building the application. The web-based app builder uses a drag-and-drop paradigm via HTML5 components and is accessible from any web browser. Both non-published (i.e. still being built) applications and published applications are stored at the SC under the user's account. The SC provides connectivity to a cloud database to store data relevant to the applications being built.

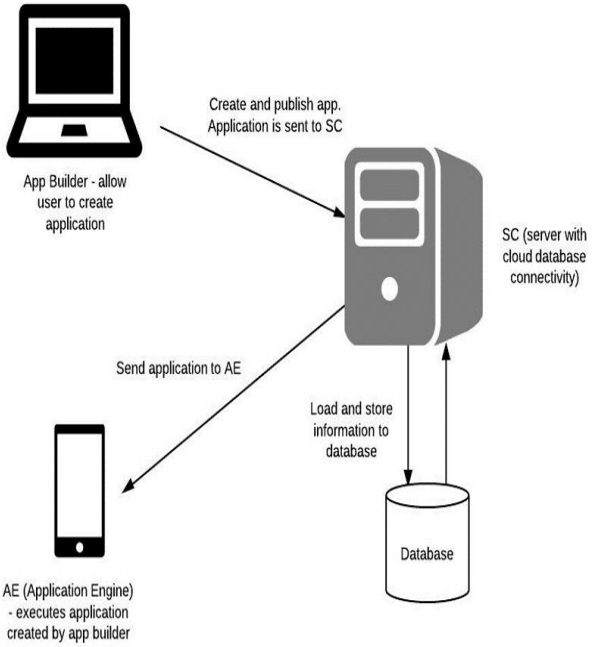

Fig. 1. Operational architecture of framework.

III. Key Framework Mechanisms

The key mechanisms of the framework consists of:

- The DS Concept
- The SC: housing the applications and their data
- The AE: downloadable and installed in the mobile device

A. The DS Concept

While the web-based app builder provides the user interface to build an application, the DS provides the mechanism within the application to enable data being used and shared. Each user interface (UI) component within a particular screen of an application is called a widget; for example, a textfield for the user to key in some text, or a button for the user to tap on to activate an event. While the use of the term datasource is not new [9], [10], a datasource (DS) is defined here as a handler to where the data for a widget is stored. Every widget on the screen can be assigned

Manuscript received January 10, 2015; revised April 28, 2016. This paper is supported by Ministry of Education Singapore.

The authors are with School of Digital Media and Information Technology, Singapore Polytechnic, Singapore (e-mail: yiphw@sp.edu.sg).

DOI: 10.7763/IJET.2017.V9.981

a DS. The DS in turn will point to either a local storage memory location or a cloud memory location as shown in Fig. 2.

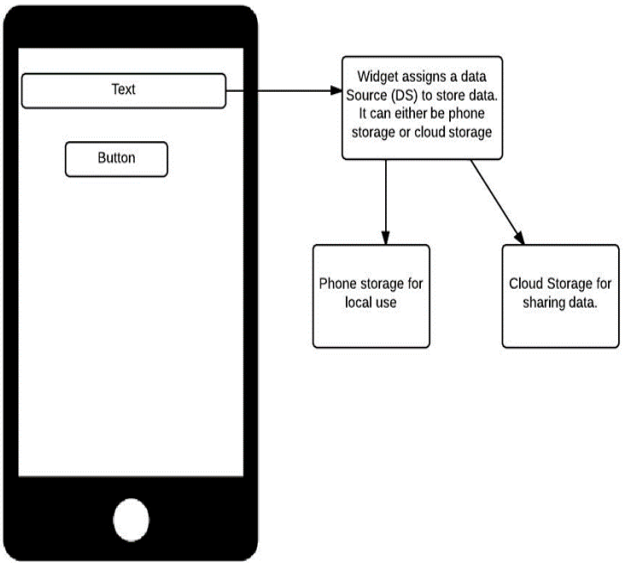

Fig. 2. Assigning a DS to a widget.

Local storage refers to the fact that the data for a widget is being stored in the memory of the smartphone. Data stored in local storage are used privately within an application or it can be shared among applications built by the same user.

Cloud storage refers to the data being stored in the cloud database. Data stored in the cloud database is not only used by the application running in the user's smartphone, it also provides the mechanism for the same data being shared with other users whom the owner (the user who builds the application) has shared his application with.

B. The Server with Cloud Database Connectivity (SC) and Application Expiry

The SC provides a user a repository to store the applications and related data he is developing as well as to manage his sharing options. An important feature in the web-based app builder is to allow the owner to set an expiry date to the application he is developing. This, along with the application logic and data, is saved at the SC as a JSON object.

Before the SC downloads the applications to the AE running in the smartphone, it checks the expiry date of each of the published applications and only downloads those which are not expired. This expiry feature allows users to seamlessly remove applications from their smartphones without the need to do any uninstallation. Although the expired applications no longer appear in the list of published applications in the AE, they still exist at the SC. The user is able to re-publish the expired applications any time when he needs the application again.

C. The AE (Application Engine)

This customised mobile application that is to be installed in the smartphone provides the run-time environment for the application built using the web-based app builder. Upon the user tapping on one of the published applications in the list downloaded from the SC, the AE confirms that the application is not expired and retrieves the DOM-like objects [11] from the SC and create the application dynamically using the Titanium Software Development Kit (SDK) [12] as the implementation platform.

Equally important, it is the ability of the AE to match which widget within a screen of an application to which datasource the widget is tied to. This in turn will allow the AE to know the kind of storage the datasource is tied to; i.e. whether it is local storage or cloud storage. In this way, the application run-time environment knows which widget data is sharable and which is not.

While the proposed framework works on Android, the cross-platform capability of Titanium makes it easier to support other smartphone solutions (e.g. IOS and Windows).

IV. Socialisable Feature

Although sharing application objects is not new [13], [14] does not include the mobile application domain. The framework in this paper acknowledges that often, users require a mobile application to support their personal ad-hoc tasks, or to support collaborative tasks like project work and events; hence the need of an app builder to develop personalised apps. Within this framework, users are able to share the app they create with other users. Data within the shared application is updated dynamically as the application executes among the group of users. Other use cases are further elaborated in Section 5. The app builder, SC and AE collaborate to realise this socialisable feature as shown in Fig. 3.

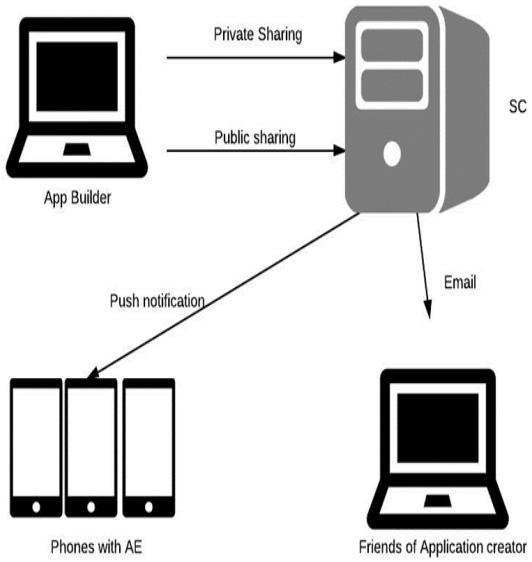

Fig. 3. Realisation of Socialisable feature.

A. How the App Builder Supports Sharing

There are 2 categories of sharing: private sharing and public sharing. For private sharing, a key feature of the web-based app builder allows the owner to indicate via email addresses or login id's the people he wants to share the application with. Both a push notification and an email will be sent to the intended recipient. The push notification is sent

to the AE executing in the recipient smartphone so that he can either accept or reject. In the event of the AE not being downloaded to the intended recipient's smartphone yet, the actual email provides the instructions.

For public sharing, a QR code with an URL pointing to the application will be generated by the app builder. Through the app builder, the owner can share this QR via Facebook or use it the way he deems fit. See Use Cases in Section 5.

B. How the SC Supports Sharing

The SC keeps track of the invites that are sent out, the acceptances and rejections in various database tables. Users who have accepted invites will have the applications downloaded by to them by the SC once the AE is launched on their smartphones.

An equally important role the SC plays is in communicating data to all users using the shared application via the DS concept. An update by one user on a particular widget in a shared application will see the data related to that widget being updated at the SC cloud storage and subsequently refreshed for all users using the shared application.

C. How the AE Supports Sharing

In private sharing, the AE serves as the destination of a push notification issued by the owner who wants to share an application. Through the AE, the user is able to either accept or reject the invite. Upon accepting, the application will appear in the list of applications within the AE that he can launch.

V. Use Cases

Although Sharable applications are ideal for collaborative and monitoring scenarios. An example will be a group project work where there is a project leader and members. Work to be done and progress can be assigned and updated dynamically via a shared application and progress can be monitored by the project leader through the use of shared DS's. On a less serious note, scenarios may include a family picnic, a class reunion party, or a garage sale. Such applications can include chat features and can be set to expire once the event is over.

Personalized applications for ad-hoc tasks like to-do-lists, one-off events which include sharing images and locations are also readily implementable within this framework.

Dissemination of information to masses is implementable via public sharing using a generated QR code. Examples include program for a dinner and dance or a graduation show, and agenda for a conference.

VI. Conclusion

A framework for building personalized mobile applications, with key features of sharing these applications and making them expire has been implemented. A common, customized application downloadable from Google Play provides the run time environment for applications built using this framework. Key advantages of this framework include enabling smartphone applications to be built with minimal programming effort and facilitates collaboration and monitoring through the use of shared DS's.

The following URL provides the web-based application builder for this framework: http:// dmit.bulletplus.com/moedappsweb2.

References

[1] C. Y. Hui, and R. C. Yi. (2014). Build your own smartphone app with these web-based mobile app builders. [Online]. Available: http://e27.co/build-smartphone-app-web-based-mobile-app-builders-20140829/

[2] Wikipedia. Google Play. [Online]. Available: http://en.wikipedia.org/wiki/Google_Play

[3] C. Warren. (25 July 2013). Google play hits 1 million apps. [Online]. Available: http://mashable.com/2013/07/24/google-play-1-million/

[4] Sarah Perez. (15 April 2014). "Google Play Still Tops iOS App Store Downloads, And Now Narrowing Revenue Gap, Too," Techcrunch.com. [Online]. Available: http://techcrunch.com/2014/04/15/google-play-still-tops-ios-app-store-downloads-and-now-narrowing-revenue-gap-too/

[5] Nathan Ingraham. (2013). "Apple announces 1 million apps in the App Store, more than 1 billion songs played on iTunes radio," TheVerge.com tech. [Online]. Available: http://www.theverge.com/2013/10/22/4866302/apple-announces-1-million-apps-in-the-app-store

[6] Portio Research, Mobile Factbook – Feb 2013, pp. 49.

[7] K. Karl, H. W. Yip, and Y. M. Ng, *in Proc. Computer Science and Information Technology*, pp. 168-172, vol. 59, Shanghai, China.

[8] A. Omojola. (2013). The shortage of developer talent is crushing mobile. [Online]. Available: http://www.forbes.com

[9] A. Jamshidi, F. Farahod, and H. Singh. Binding data from data source to cells in a spreadsheet. [Online]. Available: http://www.google.com/patents/US6631497

[10] R. Williamson, L. Upson, J. Greenfield, and D. Willhite. Method and apparatus for mapping objects to a data source. [Online]. Available: http://google.com/patents/US5873093

[11] Document Object Model (DOM) Level 1 Specification, version 1. W3C Recommendation, 1998.

[12] Appcelerator. The Titanium SDK. [Online]. Available: http://www.appcelerator.com/titanium/titanium-sdk/

[13] K. Alex, B. Linas, C. Karen, and B. Mat, "Climbing the app wall: enabling mobile app discovery through context-aware recommendations," in *Proc. 21st ACM intl. conf. on Information and knowledge management*, pp. 2527-2530, 2012.

[14] J. Harvey, A. Fegly, M. Hulan, and R. Dekelbaum. System and method for information and application distribution. [Online]. Available: http://www.google.com/patents/US6487583

Hin Wah Yip received his bachelor of engineering degree (2nd class lower) in 1992 and his master of science (electrical engineering) degree majoring in wireless communications in 1999 both from the National University of Singapore (NUS).

He is currently a senior lecturer in the school of Digital Media and Infocomm Technology (DMIT), Singapore Polytechnic and has been teaching for over 13 years. He has more than 4 years of Android apps development experience and has worked in the wireless internet, mobile handset and software development domain since 1992. His past mobile experience includes being a Defense Science Engineer at DSO National Labs (Singapore), a Motorola handset Software Engineer, a Nokia Certified Instructor and as well as being a certified java programmer and a certified Java mobile application developer.

His current research interests are in mobile computing and networking.

Karl Kwan obtained his BSc (computer science) from Madison, University of Winconsin, USA and MPhil (computer science) from Chinese University of Hong Kong. He is currently the head of R&D at School of Digital Media and Infocomm Technology, Singapore Polytechnic. His research interests are in cloud computing and mobile computing.

Ng Ying Ming received his bachelor of computer engineering from Nanyang Technological University (NTU), Singapore in 2007.

He joined SP in 2013 and is currently a research engineer in the School of Digital Media and Infocomm Technology (DMIT), Singapore Polytechnic. His research focuses on application builders for mobile devices. Prior to joining SP, he worked in Pendulab Pte. Ltd. as a software engineer. His main focus was flash games and application development as well as web portal development with Ruby.

He likes to learn new technologies in area of programming as well as game design and development.