Communications of the IIMA

Volume 15 | Issue 2 Article 2

2017

Code Generation on Mobile Devices for Mobile Apps

Nathan Amanquah
Ashesi University College, Ghana

Susana Ndede
Ashesi University College, Ghana

Code Generation on Mobile Devices for Mobile Apps

Nathan Amanquah
Ashesi University College
namanquah@ashesi.edu.gh

Susana Ndede
Ashesi University College
susana.ndede@gmail.com

ABSTRACT

There is both a demand and a need for rapidly developing mobile apps for data management as well as apps for conducting surveys. Much of the code for data management operations – create, retrieve, update and delete (CRUD) is the same, except for the parameters passed. Such boilerplate type code is well suited for code generation. Many tools are available for creating apps for conducting surveys, but are ill suited for data management scenarios since CRUD operations are not supported. The tools available generally require a PC to access the web-based build tool, and do not provide source code for the application created. This paper extends that, and proposes a mobile app for creating a full data management app to run on Android devices, as well as the generation of all the code for the native app, the supporting PHP pages for a web application, and the SQL scripts for the associated normalized database. An example app created by this mobile app code generator and app builder is also given.

Keywords: mobile app, code generation, database application, tools

INTRODUCTION

Developing countries and emerging economies have a dire need for data management and information systems. They lag significantly behind advanced countries in the collection and use of data in business processes, and in decision-making. Data is required for meaningful analysis to be made in the decision making process. There is also a lack of supporting infrastructure in terms of computer hardware for most applications. Recent years have seen mobile phones and tables outsell PCs. It is known that whereas many individuals (and small businesses) have access to, or own mobile terminals and phones, they do not always have PCs. There is nevertheless a growing need and desire to automate business processes.

Applications for more traditional activities like organizing and tracking customers, inventory, and employee management can all be done from a mobile terminal. There are numerous stories of additional ways in which mobiles are being used innovatively to address new and emerging challenges. These range from money transfer schemes, improving agricultural outcomes by disseminating information on the weather, improving farmer profits by monitoring market prices, improving community health by automating health management records (Dafla, Amanquah, &

Osafo-Maafo, 2015), and the list is endless. Many of these applications mostly have one thing in common: they essentially are database management systems made to run on mobiles and backed by cloud based back-end systems. They may also work offline in disconnected environments. It is believed that adoption of ICT including the use of mobile apps by Small and Medium Enterprises (SMEs) will dramatically boost economic growth and productivity (Boston Consulting Group, 2013).

The mobile device[1] is a natural choice for many small organizations for a number of reasons: They are much cheaper than a PC, and can thus be afforded even in difficult economic circumstances. For some applications, mobile terminals represent the best use case. Mobiles can be used to collect a wider range of data types more easily. For example, location information (GPS), multimedia data (audio and video recordings and photos), and the use of sensors and instruments on a mobile present a compelling use case. A mobile can easily be used to scan QR codes in the field. Mobiles are portable and can be carried by field staff. For example, a debt collector out in the fields collecting money from clients can record their transactions in real time with little or no setup time required, compared to using a PC or perhaps using a paper and transcribing that information subsequently onto a PC. Mobile terminals can be always connected, and do not require an additional accessory to get online, compared to using a bulky laptop and a modem for the cellular network. Length of life on a battery is a crucial factor. Many environments where these applications will be used in a developing country either have no reliable power or may have very frequent long lasting power outages - for example, 12 hours of power followed by 24 hours without. A desktop will be of no use without a generator when power goes out, and a laptop will run out in 3 to 8 hours. A mobile phone could work for 24 hours or more (up to 14 days on a popular feature phone (Quarter, 2015). Thus the mobile device wins on many fronts.

As more organizations see the benefit of adopting ICTs, particularly the use of mobiles in their operations, they seek mobile apps that meet their needs. The apps may be complemented by a supporting backend, a web based application. It may be argued that users could make use of web hosted applications from within the browser. However, cellular coverage is patchy and quality is not uniform even where there is coverage. While GSM coverage for voice calls may be present, there may be poor data coverage. (EDGE, GPRS and 3G technologies may have spotty provisioning). There is thus often the need to be able work offline and synchronize with online database, if such are used.

To drive the adoption of mobile app use for data management, it is imperative that rapid app development options be available to developers and novices alike to be able to meet the needs of the growing mobile data management app market. Organizations seeking data management apps for their operations are often unwilling to pay more that the customary $0.99. There is a perception based on pricing in app stores that mobile apps should be cheap or free, even if custom built. Either way, there is pressure on developers' time to churn out apps in reasonable time but at minimal cost. This is where rapid application development (RAD) tools and code generation technologies enter the discussion.

This work proposes the development of a mobile tool to create mobile apps (data collection apps) with or without knowledge of programming. It additionally provides code for the generated app

[1] 'Mobile device' is also alternately referred to simply as 'mobile.'

so that a programmer can customize the app further if desired. It is an end-to-end tool that creates code for native android apps, the database required, as well as PHP scripts for the web backend.

The rest of this paper as organized as follows: Section II discusses related work done in this space and points out the gap that this work fills. Section III goes on to describe how the proposed application has been built. Section IV describes how it works while section V demonstrates how it has been used practically. Section VI concludes with a discussion of the results and points out limitations and work for the future.

The objective is to make the development of mobile data collection tools on mobile devices not only possible but easy. This paper describes a cross platform application which can be used by the designer or developer to specify the data descriptions and logic to develop an android data management app. The user will only specify the variety of input types to be collected. The result will be more than a data collection tool –one that allows viewing, editing, and deletion of data on mobile devices.

RELATED WORK

There is a long history of using code generation tools for rapid application development. The Symbian platform used by Nokia smart phones in the early days was a particularly challenging development environment. There were no easy to use IDEs with the requisite tools integrated, and required developers to have specialist knowledge. The work done by Forstner et al (2005) highlights the complexity of writing code for Symbian, and demonstrates the use of a visual control flow language to aid in the generation of code for that platform. Their approach is to employ an easier to use class library and a modeling tool to make the GUI easier to build. The developer must still hand-code much of the event handler. Another proposed approach focuses on building the structure of the visual interaction with the user only (Carboni, Sanna, Giroux, & Paddeu, 2002), but the developer has to implement the concrete tasks.

Mobile cloud computing offers many advantages (Shamim, Sarker, Bahar, & Rahman, 2015), (Warhekar & Gaikwad, 2013). These include extending battery lifetime since heavy processing is offloaded to the cloud servers, storage capacity is significantly enhanced – the mobile device need not store all the data required or captured, but can upload data captured to a cloud hosted database. Other advantages include scalability of the backend, and ease of integration with other services. A variety of models have been proposed to take advantage of cloud computing power (Kahoro, Kahoro and Kanyi, 2015). The availability of mobile apps, broadband and cloud hosted services make this a significant part of the future of computing in emerging markets.

The data collection process can be expensive. Tools have been created for building data collection apps. A good overview of over 40 of them is provided by Humanitarian Nomad (Humanitarian Normad, 2016), and includes an online tool for selecting the most appropriate app generating tool for a task. However, many limitations can be identified.

The apps created by the existing tools generally allow a user to do survey type data collection where each visit represents one data point. The most popular of the lot is Open Data Kit tool (ODK) (Open Data Kit , n.d.) and there are several others based on it such as Kobo ToolBox (Kobo

Toolbox, n.d.) and Enketo Smart Paper (Enketo LLC, n.d.). The ODK toolkit is made up of a number of tools including

i. *ODK build* which a user uses to specify what data will be collected. The data collection is done online in a web browser,
ii. *ODK collect* which is a mobile app that is used in the filed for collected the data. It runs an application created by the ODK build. The ODK collect app fetches the previously created "survey" from the cloud to the mobile. These are made up of the questions, possible answer types, and skip logic (i.e. navigation between the questions). Data can often be saved locally on the device.
iii. Data can be saved offline and modified while still offline.
iv. *ODK aggregate* holds the data uploaded from the ODK collect app. Once data is submitted to ODK aggregate, it can no longer be edited from the mobile easily.

There is however a lack of master-detail among relationships created. Thus if data must be collected on the same subject, each record added is entirely new, and unrelated to any previous data (e.g. it is not possible to record patient visits and attach it to a previous patient record). There has been effort recently to create ODK Tables (Open Data Kit , n.d.) to enable the retrieval of data, editing of submitted data, and the creation of master-detail relationships. However, these still fall short of a true data management system. Two or three different applications are needed to simply modify data previously saved, and the user must switch between them to accomplish basic CRUD functions. It is not straightforward to search data collected by the app. Certainly, not by using ODK collect.

The process of creating the data collection apps is also challenging. The existing tools require desktops or laptops, and generally use web interfaces to build the mobile data collection tools. However, reliable, cheap, internet connectivity and a PC are not always easily accessible to a small scale or micro enterprise business user. Developers may own a laptop but can often run out of power because of recurrent, persistent power outages. Such a potential user will nevertheless most likely own a mobile device. Developers may also be on the move, seeking to address the needs of potential clients. A mobile device presents an interesting platform on which to do mobile app development on the go, or with little additional resources.

Besides using the web based ODK build tool, a user may also specify the data collection requirements for building the app using XForms (The World Wide Web Consortium (W3C), 2017), a spreadsheet-like specification, or XLSForm, an XML format. While it is human readable, it is not quite friendly to use, and certainly not quick to use for rapid application development.

In the process of building a mobile data collection tool, the user must specify the details (meta data) of the data to be collected. Data collection tools are essentially database applications. From a developer perspective, the process of development can be very repetitive since such apps have a lot of boilerplate code, but the context in which it is to be used may be different.

Code generation is a mechanism to produce computer programs in some automatic manner. Code generation is the act of writing or using existing programs that build applications and system code. Code generation can occur at two levels of detail. The lower level involves generation of code by compilers. In compilers, code generation is the translation of high-level code that is written by a human or some other means into machine code.

This work focuses on the higher layer of code generation. Higher level code generation also termed as Generative Programming involves the generation of actual source code that is used by a programmer to build an application. As defined by Stephen Marr (2006), “Generative programming is a software engineering paradigm based on modelling software families such that, given a particular requirements specification, a highly customized and optimized intermediate or end-product can be automatically manufactured on demand from elementary, reusable implementation components by means of configuration knowledge.” In general terms, generative programming involves the generation of actual source code that is used by a Programmer to build an application. This technique can enhance software development in different ways by improving developer productivity, producing complete quality code and save time spent on application development.

Generative programming is the method adopted for creating apps in this project. It is possible to generate code for data management apps because there is a lot of similar boiler plate code for all kinds of data management apps, irrespective of the particular use case. The aim is thus to reduce the drudgery of writing such code by providing high quality code that works correctly always, given a description of the data to be managed.

Work in (Amanquah & Eporwei, 2009) describes a code generator that accepts the data specification and proceeds to create a J2ME app, along with supporting PHP pages and database script. That was a desktop based java application and not suited for the current use case. Besides, creating a J2ME app is fairly simple as all the code for the mobile app can be made to reside in one file. It is a simpler development case compared to an Android application with its variety of files needed to build a one app.

Unlike ODK other apps built on the same framework which use HTML+CSS+Javacript in an attempt to build cross platform apps, but which mostly run on android, this project seeks to build native Android apps since they are significantly more responsive.

DESIGN

The work consists of four main modules as shown in **Figure 1**. A cross platform mobile app is used to collect meta data for the mobile app to be built. The meta data is kept in a database. A code generator reads this meta data and proceeds to output a usable Android app. The modules are described below:

<u>The App:</u> The Mobile Data Management App Builder (mDMAB) is a cross-platform application that will aid users specify the desired fields in the mobile app to be created. The objective is to create a “Field Data Collection and Management” application (FDCAM app). Users must specify input fields/questions for the forms of the FDCAM app they intend to build. The specification may be kept on the mobile and modified at will. The mDMAB is built with Phonegap (Phonegap, n.d.), a cross platform tool.

The Specification Database: The specification obtained from the user by the mDMAB app is uploaded to a cloud based database. It is from this database that the target FDCAM app will be built and code will be generated. This "specification database" is designed to hold metadata, a blueprint for any application.

The Code Generator: This module is responsible for reading specifications from the database and generating scripts that the user can download and re- use for future purposes. This generator creates code that android Development SDKs can build and run. The code includes a number of different file types.

- The XML files are responsible for management of layouts and resources in android. These files include, the manifest files, XML files for the layout for each activity and xml files for String resources.
- The Java files determine the behavior of the User Interface (activities). These files include the class files for each activity and may contain multiple classes that are related.
- SQL scripts for the application database that will hold data to be managed by the FDCAM app, and it is kept online. However, there is provision for offline storage on the mobile (by the FDCAM app) when internet not available. The code generated in the second setep makes provision for offline storage.
- The PHP scripts to handle requests from the FDCAM app are also generated. These scripts mostly interact with the online database, receive requests for CRUD operations.
- PHP scripts for a web application. These allow a human to review data on the site that hosts the database.

The output app: The Android Field Data Collection and Management application (FDCAM app): This is the mobile data management app (the output app, the result of code generation) that will be used to collect and manage data. This app can retrieve and list data stored on the server for the user to view, edit and delete. The output app (FDCAM) allows offline storage of data where there is no internet connectivity, and can capture a variety of data types including GPS coordinates, and has a barcode/QR code scanner.

Figure 1 Parts of the code generation system.

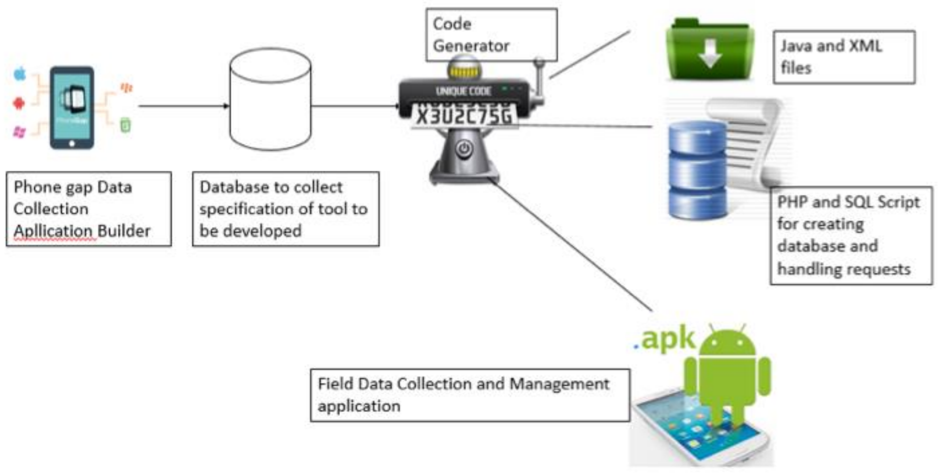
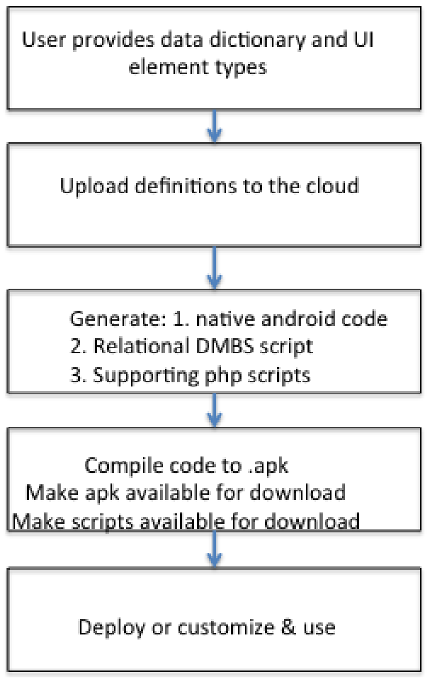

Code Generator
Java and XML files
Phone gap Data Collection Apllication Builder
Database to collect specification of tool to be developed
PHP and SQL Script for creating database and handling requests
.apk
Field Data Collection and Management application

Figure 2 illustrates what a user goes through from specification of app to its generation.

Figure 2 Steps in code generation process.


User provides data dictionary and UI element types
Upload definitions to the cloud
Generate: 1. native android code
2. Relational DMBS script
3. Supporting php scripts
Compile code to .apk
Make apk available for download
Make scripts available for download
Deploy or customize & use

The application has a wizard type navigation that guides the user from screen to screen and prompts them for what data to enter. Key functional requirements of the mDMAB are as follows:

a. User Login: This allows the user to login to the phone gap application to view forms he has previously developed. The user can choose to make new forms or edit old ones to save as new form. Login is important in identifying records belonging to a user, as data is held on a shared database.
b. Create new data management app and data entry forms: User creates new form in the new app by specifying form name as well as form description. User can create many forms with this application.
c. Edit old forms and save as new forms: If a user needs to make changes to a form that has already been developed into an APK, the user is given the option to retrieve this form, make changes and save this form as a new form. This will also allow users wish to clone an existing app to do so without having to specify all form details again. It is also important that changes are saved as a new application (unless the app has not yet been built). This is to safeguard against the possibility that an app that has been used to collect some data already has a subsequent version with a different database schema, and therefore incompatible.
d. Specify Number of data items per form: This allows the user to add on as many fields as possible to a form (or number of questions per screen in the case of a survey type app). The user can select the number of fields to appear on a screen, otherwise the default of one data item/question per page is applied.

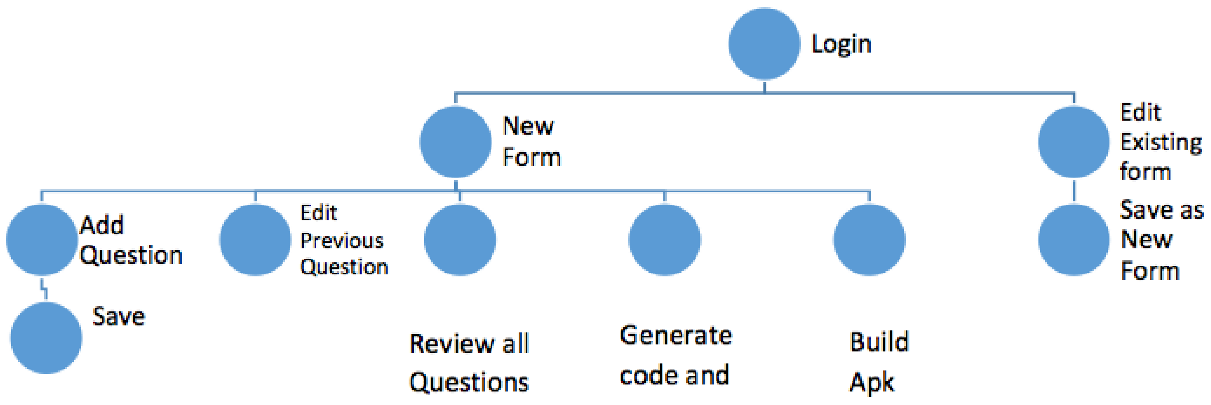
e. Specify Input Types for field types/questions: When drafting input prompts, the user specifies the input type the particular data type accepted. The allowed input types are text, Long Texts, numbers, emails, decimals, GPS, Locations.
f. Specify Widget Types: The user can choose the widget for data entry. This is useful in designing the User Interface. The allowed widgets are date and time pickers, textboxes for different input types, radio buttons, Spinner, QR and barcode reader
g. Review fields: The user can review fields before the final submission when the information supplied are used to develop the android application.
h. Build APK for android data collection tool: The user can obtain an APK that can run the android data collection tool developed.
i. Obtain generated code for the application: The user can obtain a set of files that represent the code used to build the android application. The files include java files, xml files and SQL and PHP files.

IMPLEMENTATION

The mDMAB app was built with JQuery Mobile and Javascript, and built as a cross platform application. The backend on the cloud that does the code generation of the app was written in PHP. A MYSQL database was used to hold the data. An apache ANT script (a build.xml file) is generated which can be executed on the command line to generate an android APK, or automatically by the PHP script.

The mDMAB app makes use of an interactive user interface and a number of form widgets to fetch the specification from the user. Unlike the XForms or XLS forms which requires a strict format, which a user may provide incorrectly, in this case, the user makes selections mostly from options provided in drop down boxes and is unlikely to provide an incompatible specification. A high level navigation chart for the mDMAB is shown in **Figure 3**.

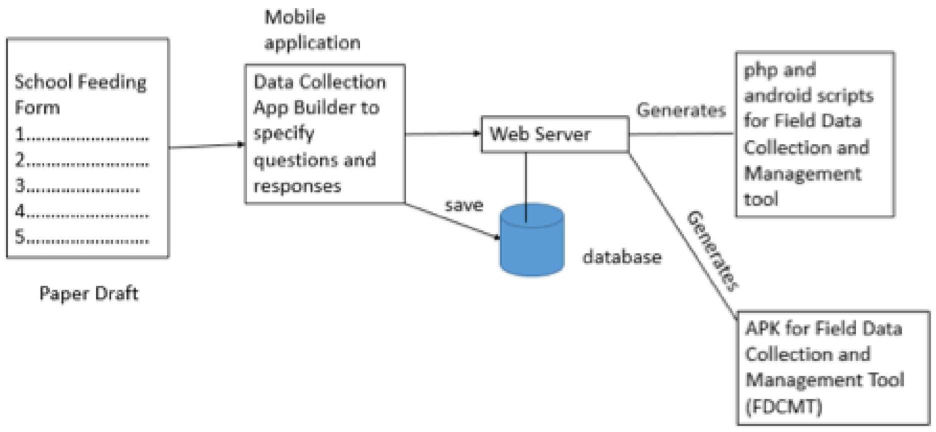
Figure 3 Navigation chart for mDMAB.



RESULTS

mDMAB was tested with a scenario where a user would like to create a questionnaire to determine how helpful School Feeding Program has been. This is not unlike the survey type applications, but was also to illustrate the capabilities and features of the mDMAB. The general overview of steps involved in this process are illustrated in **Figure 4**.

Figure 4 Overview of steps for creating an app with mDMAB.



The specification to be provided to the app for testing purposes are illustrated in **Figure 5**. **Figure 6** to **Figure 10** illustrates some of the screen shots on mDMAB in the app creation process.

Figure 5 Specification for sample database app to be create.

Question Number	Question	Widget Type	Options if needed	Input Type
1	What is your name?	Edit Text	N/A	Text
2	How old are you?	Edit Text	N/A	Number
3	Select Your School	Spinner	Akai JHS Kola JHS Bui JHS	Text
4	Enter your headmaster's email address	Edit Text	N/A	email
5	Which option feeds you better?	Dropdown	School Feeding, Pocket Money	Text

Figure 6 Form creation or editing page.

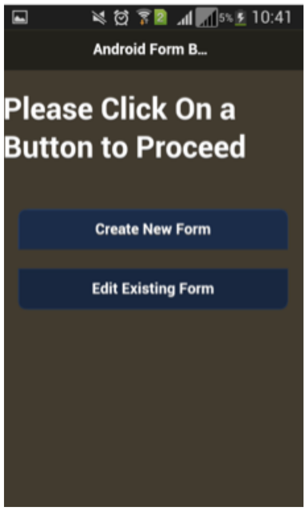


Figure 8 Options to create or modify a field on a form.

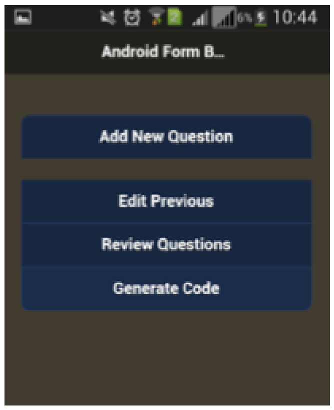
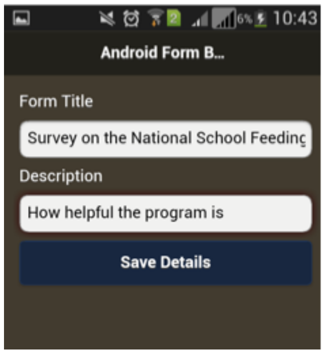
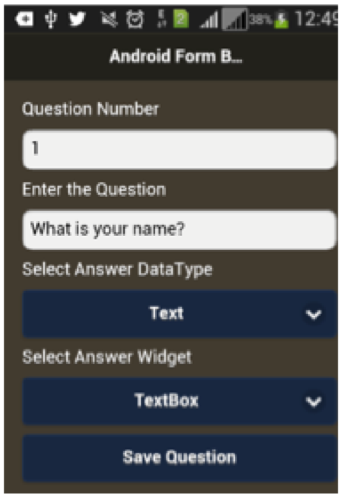


Figure 7 Form description.



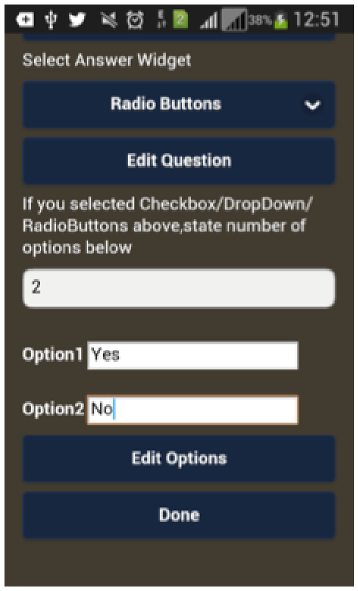
Form Title: Survey on the National School Feedinç
Description: How helpful the program is

Figure 9 Specification for a field –a textfield entry type.



Question Number: 1
Enter the Question: What is your name?
Select Answer DataType: Text
Select Answer Widget: TextBox

Figure 10 Specification for a field –a radio button type.



Select Answer Widget

Radio Buttons

Edit Question

If you selected Checkbox/DropDown/RadioButtons above,state number of options below

2

Option1 Yes

Option2 No

Edit Options

Done

The high level set of activities of the resulting FDCAM app is illustrated in **Figure 11**. The resulting normalized database schema for the simple setup is shown in **Figure 12**.

Figure 11 Activities in the resulting android FDCAM app.

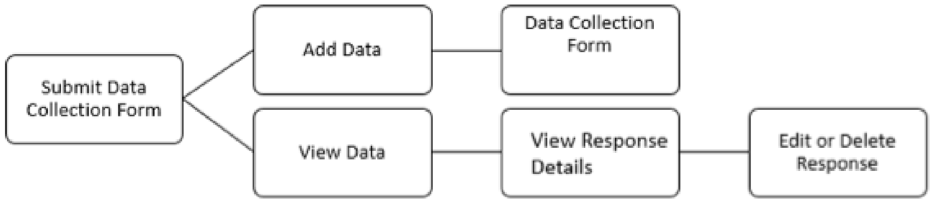
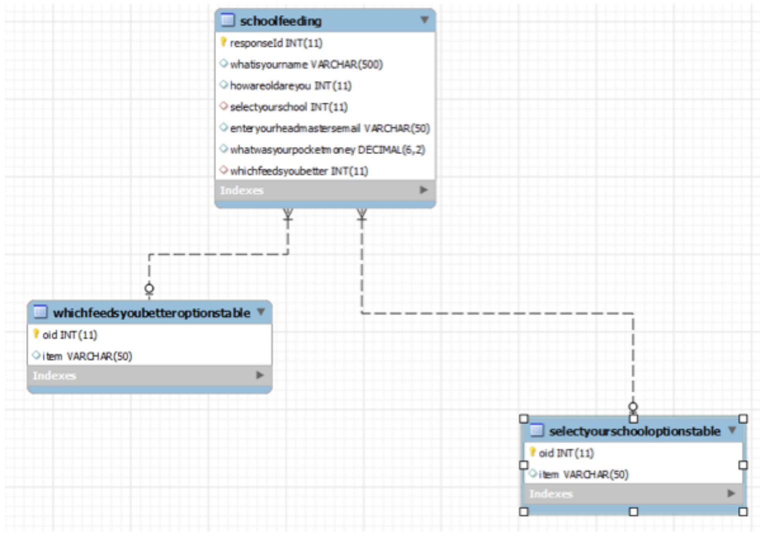


Figure 12 Database schema created for the simple scenario.


schoolfeeding
responseId INT(11)
whatisyourname VARCHAR(500)
howareoldareyou INT(11)
selectyourschool INT(11)
enteryourheadmastersemail VARCHAR(50)
whatwasyourpocketmoney DECIMAL(6,2)
whichfeedsyoubetter INT(11)
Indexes
whichfeedsyoubetteroptionstable
oid INT(11)
item VARCHAR(50)
Indexes
selectyourschooloptionstable
oid INT(11)
item VARCHAR(50)
Indexes

The output of the resulting app is shown in **Figure 13**. It shows multiple questions on a single form, and is a native Android app. There is even a splash of color, although the theming is currently fixed, but can be modified by any developer who compiles the code themselves.

Figure 13 Add page in the generated application

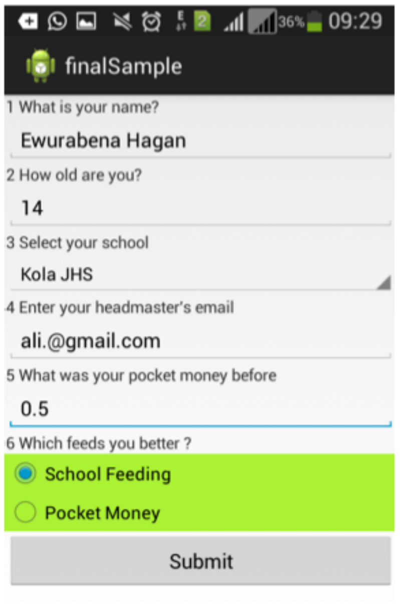
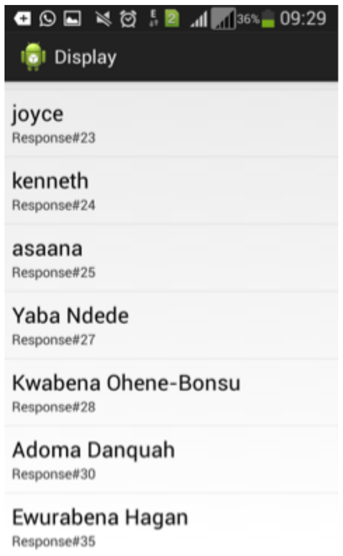
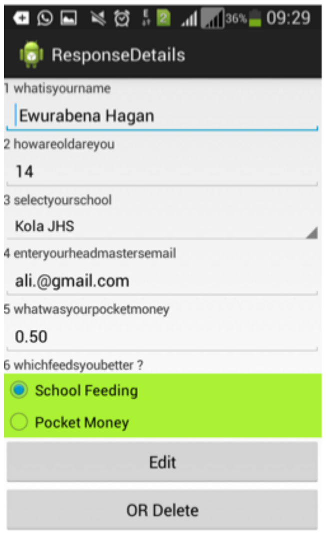

09:29
finalSample
1 What is your name?
Ewurabena Hagan
2 How old are you?
14
3 Select your school
Kola JHS
4 Enter your headmaster's email
ali.@gmail.com
5 What was your pocket money before
0.5
6 Which feeds you better ?
[✓ School Feeding] Pocket Money

Figure 14 List of previously collected data


09:29
Display
joyce
Response#23
kenneth
Response#24
asaana
Response#25
Yaba Ndede
Response#27
Kwabena Ohene-Bonsu
Response#28
Adoma Danquah
Response#30
Ewurabena Hagan
Response#35

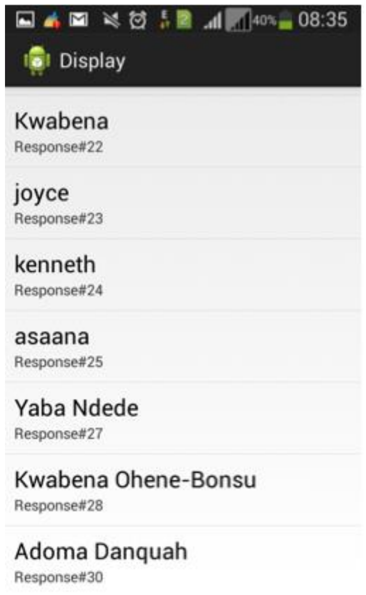
The following shows an edit screen, and the resulting changed list of entries after deleting one entry (last response, response 35 is deleted).

Figure 15 Editing of previously collected data



1 whatisyourname
Ewurabena Hagan
2 howareoldareyou
14
3 selectyourschool
Kola JHS
4 enteryourheadmastersemail
ali.@gmail.com
5 whatwasyourpocketmoney
0.50
6 whichfeedsyoubetter ?
[✓ School Feeding]
Pocket Money

Figure 16 Updated list after deletion



Kwabena
Response#22
joyce
Response#23
kenneth
Response#24
asaana
Response#25
Yaba Ndede
Response#27
Kwabena Ohene-Bonsu
Response#28
Adoma Danquah
Response#30

DISCUSSION AND FUTURE WORK

This paper has proposed and demonstrated an app that offers significant advantages over what is presently available.

- Unlike the output of other tools that create mobile data apps, this one allows users to actually mange the data- doing not only add operations (data *collection* operations), but also edit, delete and search operations, even if the data has already been uploaded to a server.
- The ability to develop mobile data management apps on the go using a mobile app will enable the proliferation of data management tools. Many will to build their own database to manage their small operations. Developers will use this a basis for their work. They will quickly generate all the boilerplate code and have fully functioning code, which they can then proceed to apply theme it, to make it unique. It will be a great time saver, and they will thus be able to churn out apps faster and more cheaply.
- Apps can be specified on the go as no internet access is required until it is time to build the app.
- App specifications can be done with significant accuracy. The mDMAB app makes use of prompts and drop down boxes that make it difficult if not impossible to specify data types and responses types incorrectly, compared to making out a properly formed xml file for the XLSForm, or using a spread sheet to prepare the specification. Also, no drag and drop is required.
- The output app (FDCAM) supports not just surveys type applications, but full data management
- FDCAM allows a page to have more than one question or data entry field, compared to the other apps which typically have one question per page.
- A normalized relational database is created out of the specification for storing the data. Data that shows up a combo boxes in the generated app come from a separate lookup table unlike other apps that keep all that data a fixed data. First, this will allow the options in the combo box to be extended without having to recreate the app. Second, the normalized nature of the data collected is easier to analyze and use for further processing, compared to a single spreadsheet of data that is not normalized.

There are a few limitations however. mDMAB is a mobile app that can run on small terminals. Drag and drop is not enabled in this application for the specification of widgets because of screen size. Perhaps it may be useful to have an alternative that has drag and drop enabled, for use on bigger screen tablets.

Although the specification phase requires no Internet access, the build process requires internet access since the meta data collected must be uploaded to the build server in the cloud.

Currently, when the specifications in the mDMAB app are changed, it must be saved as a new app because it is likely that data previously collected by the app will use a different schema from the new one. It will be useful to provide an upgrade path for the old data if there is some measure of compatibility that can be attained.

It will be useful to completely automate the process of APK creation, and perhaps explore possibility of signing and deploying directly into the app store, or to push directly on to a target mobile device. This will make it particularly easy for the novice or casual user to build and deploy data management apps. The apps built were often compiled in an IDE using the source code generated, but the command line option could be extended to cater for different versions of android API, made more robust.

Other minor updates needed include allowing the user to specify the server on which to deploy the php pages and automatically uploading the pages to the intended server. It would be desirable to extend the generator to produce source code and also build native apps for other platforms include iOS and Windows mobile platforms.

REFERENCES

Amanquah, N., & Eporwei, O. T. (2009). Rapid Application Development for Mobile Terminals. 2nd IEEE *International Conference on Adaptive Science and Technology (ICAST'09),* December 14-16, 2009, Accra.

Boston Consulting Group. (2013). *Lessons on Technology and Growth from Small-Business Leaders.* Retrieved January 29, 2017 from https://www.bcg.com/publications/2013/technology-software-globalization-ahead-curve-lessons-technology-growth-small-business-leaders.aspx

Carboni, D., Sanna, S., Giroux, S., & Paddeu, G. (2002). Interactionss Model and Code Generation for J2ME Applications . *In Proc Proceedings of the 4th International Symposium on Mobile Human-Computer Interaction*, (pp. 286-290).

Dafla, A., Amanquah, N., & Osafo-Maafo, K. G. (2015). A Mobile Devices Health Information Application for Community Based Health Services. *2015 Conference on Raising Awareness for the Societal and Environmental Role of Engineering and (Re)Training Engineers for Participatory Design (Engineering4Society),* June 18-19, Leuven, Belgium.

Enketo LLC. (n.d.). *Overview for Developers*. Retrieved January 15, 2017, from Enketo Smart Paper: https://enketo.org/develop/

Forstner, B., & et al. (Nov 2005). Supporting Rapid Application Development on Symbian Platform. *in Proc of The International Conference on Computer as a tool (EUROCON 2005),* Belgrade.

Humanitarian Normad. (2016). *Find the Right Mobile Solution*. Retrieved 06 15, 2016, from Humanitarian Operations Mobile Acquisition of Data: https://humanitarian-nomad.org

Kahoro, P. K., Kahoro, P., & Kanyi, G. (2015). Mobile Cloud Computing Models, Infrastructures, & Approaches. *Conference on Cloud Computing, Vol 4.0, January, 2015.*

Retrieved February 6, 2018 from https://www.researchgate.net/publication/270574846_Mobile_Cloud_Computing_Models_Infrastructures_Approaches

Kobo Toolbox. (n.d.). *Data collection Tools for Challenging Environments*. Retrieved from www.kobotoolbox.org

Marr, S. (2006). *Feature Charts and Variability*, Grin Verlag Publishing, 2006. Retrieved from https://www.grin.com/document/110351.

Open Data Kit . (n.d.). *Open Data Kit*. Retrieved 10 15, 2014, from https://opendatakit.org/

Phonegap. (n.d.). *Phonegap*. Retrieved from www.phonegap.com

Phonegap. (n.d.). *Phonegap*. Retrieved January 15, 2016, from www.phonegap.com

Quartey, E. (2015). The Mystery of the Power Bank Phone Taking Over Accra.*Quartz Africa, May 26, 2015.* Retrieved February 6, 2018 from https://qz.com/411330/the-mystery-of-the-power-bank-phone-taking-over-ghana/

Shamim, S. M., Sarker, A., Bahar, A. N., & Rahman, M. A. (2015). A Review on Mobile Cloud Computing, *International Journal of Computer Applications, Vol 113, No. 16.* Retrievd February 6, 2018 from http://research.ijcaonline.org/volume113/number16/pxc3901883.pdf

The World Wide Web Consortium (W3C). (2017). *XForms 1.1 W3C Recommendation 20 October 2009*. Retrieved from The World Wide Web Consortium (W3C): https://www.w3.org/TR/xforms/

Warhekar, S. P., & Gaikwad, V. T. (2013). Mobile Cloud Computing: Approaches and Issues. *International Journal of emerging trends & Technology in computer science, Vol 2* (2). Retrieved February 6, 2018 from https://pdfs.semanticscholar.org/034b/d7b1cf79af4ece336b51a74a8aced90bce0d.pdf

Wikipedia. (n.d.). *Dumsor.* Retrieved December 2, 2016, from https://en.wikipedia.org/wiki/Dumsor