**Prototype Design and development of LizCapApp Mobile Application in Xamarin**

PANKAJ SINGH**,** *Software Engineering MS student, Arizona State University, Mesa AZ*

**Abstract:** Nowadays, Field Biologists use a mobile application LizCapApp available for both Android and iOS devices to survey data. The app is developed on Capture–Mark–Recapture Data Collection. Though the app serves its initial purpose it has inherent technical limitations. One of them was the static layouts of the screen and input controls are focused on Lizards and similar taxonomies. To increase the user base of the app among the biologist it needs to be generalized which require dynamic input controls. Along with its software support the app needs to be enhanced to accommodate new requirements over the time. As the app program code is developed natively for Android and iOS platforms, which are quite different in their design and architecture, require two different sets of experts to maintain and enhance current apps features. It is financially difficult to hire experts from two different platforms. So we proposed a solution to develop new version of LizCapApp in one of the popular cross platform tools like Xamarin, Titanium and PhoneGap. After careful evaluation of each platform we found Xamarin to be the most suitable for our app requirement. Once the platform was decided, features of the new version of app has been elicited. A prototype of the app and supporting proof of concepts were developed in Xamarin to find solutions targeting implementation level issues of the elicited requirements to pave the way for a new version of LizCapApp in Xamarin.

**INTRODUCTION**

Earlier mobile phones were used to call other people carrying mobile or landline devices. With the advent of smart phones, the uses of mobile phones have gone beyond calling. Enhanced software platforms have provided mobile applications to perform different functionalities. Most of the traditional methods/devices of performing activities have been replaced by mobile applications like alarm clocks, maps etc. Since the emergence of the mobile devices, variety of mobile applications have been developed. One such mobile application was LizCapApp, developed for biological field survey. It is very helpful in collecting survey data and synchronizing it with the database for further analysis. As happens with most of the software applications it also require enhancements and upgradation meeting the current and future requirements. This project targets requirements and technical feasibility for the transition of the current LizCapApp to a newer version to meet its business and technical goals.

**BACKGROUND :** background ofcurrent Lizcapapp from Bateman’s paper.

Capture–mark–recapture (CMR) methods are often used in the management and conservation of wildlife species to estimate populations and survival [1]. Earlier biologists used paper based datasheets to collect and record wildlife data. Then they come to the lab and update these values to the computer for storage and analysis. This method was easier but error prone. So in 2012 Dr. Bateman with the help of Dr. Tim Lindquist, Mr. Richard Whitehouse and Maria M. Gonzalez proposed to use mobile application to collect data and then synchronize it with lab database. They created a mobile application named LizCapApp and experimentally collected data for 2 months [2]. They found that using mobile application reduces errors and was significantly faster at transforming field collected data to the lab database. Pre-populated drop down values helped to select correct data. Initial validations of input fields to correct data at the field itself rather than correcting on memory/ assumption about that data in the lab. App has the capability to suggest unique codes to mark every animal, this helped to identify and count new and revisited animals. Mobile app also provides history of a captured animal and can be visited while at field thus enabling to record further analysis of a recaptured animal. The project was able to prove the advantages of using a mobile application over traditional datasheets for collecting field data in terms of higher data accuracy, less time consumption and better analysis in the field. It is currently used by Dr. Bateman and her students for collecting actual data on Lizards and Arthropods for several research purposes.

**LIMITATIONS OF CURRENT APP:-**

Though the current app was capable of capturing data and reducing errors in collecting field data it has underlying technical limitations. We have found some of the major limitations which prompted for the need of developing a new mobile application. The requirements of the new app have been elicited and documented in a separate file named LizCapApp\_SRS1.2.

1. Natively developed in iOS and Android:

There are two versions of LizCapApp one natively developed for Android and another for iOS. Both the apps are in production and can be downloaded from iTunes (Apple, Inc.) or Google Play (Google, Inc., Mountain View, CA). Both Android and iOS are the market leaders of the Mobile devices. In today’s market Android and iOS together have a reach to around 99.6% of the smartphone mobile devices [3]. Both the platforms are different from each other in their architecture and design. So while developing an app in either platform, software developers require different skill sets and knowledge to utilize the best from each platform. In Android the app is developed by programming in Java (programming language), while for iOS the developer should know programming in swift/objective C (programming language). This requires hiring different teams for each platform which is very costly and requires a huge amount of time and resources. This increases the maintenance cost of the app over its life cycle. As even a small change or enhancement in app require two experts to implement in software code for each platform. It would have been a cost effective solution to code app in one common programming language targeting both the platforms, thus reducing the size of software developer teams and ultimately cost.

1. Generalization of the app

The screens and layout controls of current LizCapApp is highly focused on one Taxonomy of animals i.e. Lizards. Though there are multiple taxonomies available but there input data fields are subset of input fields of Lizards as shown in Table 1. Now suppose a biologist has to collect color of mammal also or change the input field name from mass(g) to weight(kg); as the input fields are statically implemented within the apps program, so it requires changes in the apps programming code. As the mobile application has to be used by Biologists, who rarely have deep understanding of programming or app development, always need support of software developers to modify and enhance apps input fields and their names. To make it generic and usable by large community of Biologist for data collection purpose screens and data fields should be configurable by Biologists without any support of software developers. If the name of the input field has to be changed or an additional input field needs to be added it should be done by the Biologists themselves.

1. Locations has to be entered by the recorder manually.

In current app Site of the location where the data has been collected has to be selected manually. Though prepopulated values in the dropdown help to get the accurate location value but it with current technology of GPS (Global Positioning System)and maps we can eliminate the recording of incorrect manual value. Even the Array field which contains the trap array number at a particular site can be automatically adjusted.

1. No bug tracker has been implemented with the app.

No software is absolutely bug free, there are always less or more bugs.

If the app does not perform naturally, it shuts down or gets hanged while recording any data value it needs to be reported by the user to software maintenance team. Most of the time user forget to report or if reports this unusual behavior are unable to describe the bug they faced while using the app, even if they describe, the description is not technical in nature. So the technicians are unable to understand the cause of the actual bug or unable to reproduce the bug in the app. There should be an automatic bug reporting tool implemented within the app which can store the state of the app at the occurrence of the bug, thus helping the

maintenance team to reproduce and fix it quicker.

1. UnSynced History Data cannot be edited

Once the field data has been collected it is stored in the local database of the mobile app and can be seen in the “UnSynced History” module of the app. If there are any errors or inconsistencies in this data which needs to be edited, there are currently no provision to do in the app. The only way is to update it to the Lab database and then correct it there. This correction causes extra effort and time to be spent in finding the inconsistent collected data and do correction as the Lab Data is much larger than recent collected UnSynced data on the app. The effort becomes many fold when multiple mobile devices have such issues.

**PROPOSED SOLUTION**

After due deliberation and discussions, we proposed feasible technical solutions for each of the problems/limitations stated above. Even while proposing a

newer version of the app we documented its requirement specification in a separate document named Software Requirement Specification. Solution for each of the limitations are discussed below:

1. *Natively developed in iOS and Android*

This was one of the major limitation to be taken care while developing newer version. Many companies have launched their platform/tools to develop mobile apps, which provide special feature to develop once and run anywhere capabilities. These tools/ framework/platforms uses a single technology to develop application/s for multiple environments and are generally referred as “Cross Platforms”. We have explored three most popular

cross platforms available in the market i.e. PhoneGap, Titanium and Xamarin on the technical and business needs of the newer version of LizCapApp. We tried to develop sample applications containing most essential user interface components

like Input Text fields, Spinners, Buttons, MySQL (SQLite) File upload/download, Images, Connection to Wi-Fi for each platform and uploaded sample applications [Github repository](https://github.com/PankajSingh-ASU/HybridPlatformDemos). Each platform has been evaluated on the following features with their decreasing priority:

1. *Pricing:* We evaluated free version and trial version of the platform only.
2. *Native Mobile Features*: Does the platform provide device hardware support? Can they make use of device hardware and optimize applications using native Android or iOS features? After developing an application how close do they resemble to the same app developed in native platforms. Platform has been evaluated for their ability to use native features of the mobile (Android & iOS) devices.
3. Learning Curve: Availability of the tutorials, documentation of APIs and prerequisite learnings.
4. Community Support: Do platforms provide developer forum to discuss bugs and errors if yes, how efficient those forums are.
5. Core Architecture: Flexibility of the directory structure of app in that platform.
6. Developer Community Reviews: How many and how strongly developers recommend this platform.
7. Licensing: Is the platform available to student/educational institution for free to publish their applications to the marketplaces like iTunes for iOS and PlayStore of Android.

We have used priority table to evaluate each platform on different required features and summarized in table 1. Each feature is given a priority value with pricing with max point of 10 and licensing with least as 4. Each platform was given a score out of 10 for each feature and this point was multiplied with the priority value and summed up the total values to get final score. Detailed report can be seen in the appendix A. We found that Xamarin is the most suitable platform for needs with the maximum score of 360. Since the Titanium platform is not free for development it has not been considered.

Once Xamarin has been selected, we have developed a prototype application in Xamarin, named LizApp, replicating few screens of current LizCapApp. This application has been uploaded in github along with other codes.

|  |  |  |
| --- | --- | --- |
|  | | |
| Data Type | UI control | Keyboard type |
| int , integer | EditText  Box | Android.Text.InputTypes.  ClassNumber |
| Decimal, real | EditText  Box | Android.Text.InputTypes.  NumberFlagDecimal |
| Text, string | EditText  Box | Android.Text.InputTypes.  ClassText |
| Boolean | Check box | Not required |
| Date, datetime | DatePickerDialog | Not required |

1. Generalization (1 am)

Since we are not going to modify lab database or web framework accessing lab data base, we have to find a way to generalize the mobile app with mobile app code itself. In current architecture of the app the lab database is synchronized with the mobile app which means in mobile app we get a copy of lab database in the form of SQLite database (local database of mobile app). So we proposed using SQLite database we should dynamically generate input controls and screen layouts. After discussion with biologists and our online research we found that most of the input data collected belongs to the following data types:

1. Integer or Number
2. Float or Decimal or real numbers
3. String or Text
4. Date
5. Boolean or true/false
6. List (dropdown list)

These data types correspond to the data type of columns in a table in SQLite database. When programming with Xamarin we have the privilege to write common logic code or database access code as a common code and design or screen layout code for Android and iOS separately. Using the SQLite query

*<pragma table\_info(‘tableName’) >*

we get the column data types of a given table along with cid (unique for every column) in a table. Next we can simply return the C# object containing column data type details to the presentation layer of respective environment (iOS or Android). We have developed a sample application named *SQliteTest* with Android screens in Xamarin to test feasibility of our idea. Next we matched column data types with the corresponding UI controls as represented in the table 2. While saving the input values collected from the user on the mobile screen takes the reverse route. Depending on the data type user is presented a keypad/UI control to enter values (class DynamicUICreator

). Once he clicks on save the control reads the values from UI controls depending on the UI control type like for numbers/int value would be read by using text property of EditText while for Boolean values selected property of checkbox has to be used( *getInputValues() method of class InputStore* ). A dictionary is created with the column cid as key and UI input as value. This dictionary would be passed to the Data Access layer of the app then an insert query is created with column names and column values is set from dictionary values *(* method *saveTableValues() of* class *DatabaseAccess* ).

Creating a drop down list for selecting a prepopulated list require more deliberation as there is no data type called list for column types in SQLite. There were two logical solutions proposed :

1. to read the distinct values of the column if there were limited count say 5, create a dropdown list as UI control, corresponding to this column. But this solution has many limitations like if this is a new table where less than 5 rows are entered all the columns would be considered a drop down list. Or if the drop down list values contain more than 5 values like days in a week.
2. Create a table containing the possible values of the dropdown list and reference this list column as a foreign key in the original table. For example, while entering Amphibian data user needs to select fence trap, this fence trap column of Amphibian table needs to be a foreign key referencing a column *trap* in *FenceTraps* table.

We still have to figure out and finalize the best way to create dropdown lists for an input field without modifying database.

1. Location Logic

With the help of GPS technology present in mobile devices we can find the current location of the user. It will return latitude and longitude of the users’ location. Since all the sites are stored with their latitude and longitude value, we can calculate the distance between the site and the users’ location. Least distant site from the user will be the site selection. In this way we will be able to automatically record site of the data collection. Determination of location of a mobile device depends on GPS, Cell-ID, and Wi-Fi but they have their tradeoffs of accuracy and power consumption [4]. Google suggests using ACCESS\_COARSE\_LOCATION or ACCESS\_FINE\_LOCATION permissions. We will recommend using ACCESS\_COARSE\_LOCATION which though did not provide less accurate location but it uses less battery consumption and being the sites far away from each other nearest site can be accurately calculated.

1. Bug Tracker (1:50 am)

Xamarin provides Xamarin Insights to track and report app crashing. HockeyApp is another tool which can be signed up with students account or work or school account. They provide a free plan offering unlimited distribution, crash reporting, and feedback for 2 apps [5]. This will be sufficient for newer version of LizCapApp.

1. UnSynced data( 1para)(2:10 am)

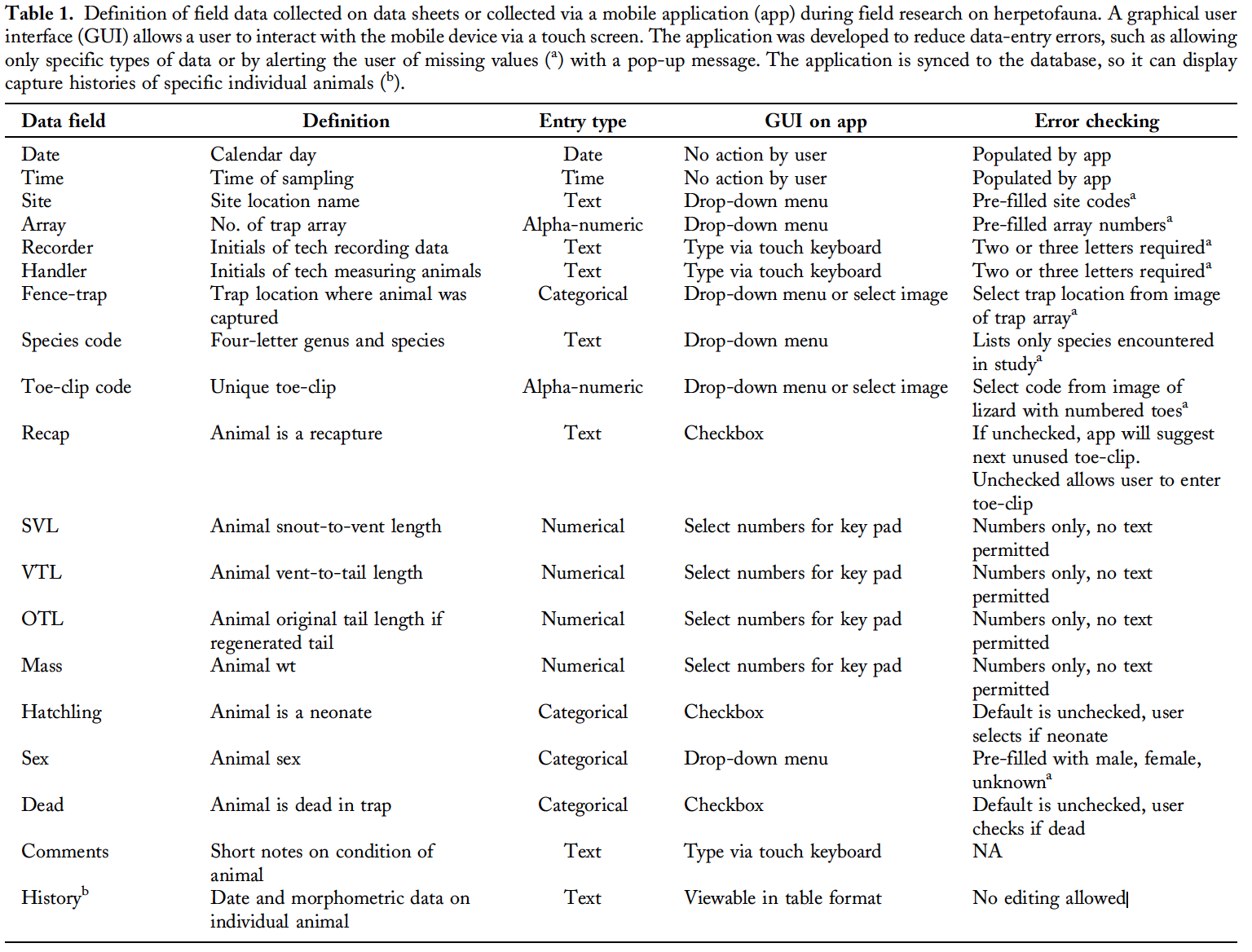
Currently UnSynced history can be seen in a linear tabular format in a screen in LizCapApp. On click of any cell value in the table a screen navigation should be created to navigate to the data collection screen with prepopulating input field values by reading values already stored in SQLite database. Major advantage of this will be to correct the data when the animal is “in the hand” instead of correcting the data back at the lab.

Future Work (2:30 pm)

This project provides the technical feasibility and implementation level details for the newer version of LizCapApp. Development of a fully-fledged new LizCapApp in Xamarin platform, accommodating the features elicited in Software Requirement Specification1.2, will be undertaken next. Dynamic UI control generation from the SQLite database will help in generalizing the LizCapApp for wide variety of field surveys. It will expand the user base including biologists surveying different taxonomies. Even it will help to generate future mobile applications where the static layouts are not preferred and controls needs to be customized by the end users and not the developers. Best way to implement drop down lists needs to be figured out. There will be many other technical challenges to be addressed and solutions have to be figured out during the actual development of LizCapApp in Xamarin. Evaluation of cross platforms can be used to evaluate suitability of the best platforms for future development of mobile applications.

Conclusions

The major goal of the project was to develop a prototype mobile application for field surveys whose development and maintenance should be mobile platform independent. We have developed several and prototypes for the evaluation of the platforms. We have identified and evaluated Xamarin to be the most suitable platform for the development of a new field survey mobile application. We developed a prototype to replicate screens of current app in Xamarin to showcase its usability. Another prototype for dynamic generation of UI controls have been developed to find out the implementation details required. Most of the known requirements of the new app have been analyzed and their solutions have been proposed. This project would smoothen and accelerate the programming of the new mobile application for field survey.



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Table 2. Priority table containing features and priority value for each feature 10 being highest and 4 being lowest. Each platform has been scored out of 10 for each feature. Priority points have been multiplied with Platform score for each feature to get Platform value for each feature. Platform value for each feature have been summed up to get total platform score. | | | | | | | |
| Features | Priority Points | PhoneGap | PhoneGap  Value | Titanium | Titanium  Value | Xamarin | Xamarin Value |
| Pricing | 10 | 10 | 100 | X | X | 8 | 80 |
| Native Mobile Features | 9 | 4 | 36 | 7 | 28 | 9 | 81 |
| Learning Curve | 8 | 8 | 64 | 8 | 64 | 5 | 40 |
| Community Support | 7 | 9 | 63 | 4 | 36 | 8 | 56 |
| Core Architecture | 6 | 4 | 24 | 6 | 24 | 8 | 48 |
| Developer Reviews | 5 | 5 | 25 | 6 | 30 | 7 | 35 |
| Licensing | 4 | 8 | 32 | 6 | 48 | 5 | 20 |
|  | Total | 13 | 344 | X | X | 12 | 360 |

**Reference:**

[1] Sears, A., and Y. Zha. 2003. Data entry for mobile devices using soft keyboards: understanding the effects of keyboard size and user tasks. International Journal of Human–Computer Interaction 16:163–184.

[2] Heather L. Bateman, Timothy E. Lindquist, Richard Whitehouse and Maria M. Gonzalez 2012. Mobile Application for Wildlife Capture–Mark–Recapture Data Collection and Query. Wildlife Society Bulletin; DOI: 10.1002/wsb.322

##### [3] Egham, U.K., February 15, 2017, press release. <http://www.gartner.com/newsroom/id/3609817>

##### [4] Location Strategies <https://developer.android.com/guide/topics/location/strategies.html>

### [5] Getting Started with HockeyApp

##### https://www.hockeyapp.net/blog/2016/03/31/welcome-xamarin-insights-users.html