A Mobile Application for Collecting Plant Observation Data

Jingbo Chu, Yu Qiu, Mao Zheng, Tom Gendreau
Department of Computer Science
University of Wisconsin-La Crosse
La Crosse WI, 54601
mzheng@uwlax.edu

Abstract

The aim of this project is to develop an Android application for collecting plant observation data in the field.

Plant breeders, agronomists and scientists need to record a lot of information in the field. Recording data by pen and paper has many problems. For example, handwriting is often difficult to read and can easily be misinterpreted. In order to do data analysis, handwritten data must be entered into a computer or the analysis must be done by hand. In either case, the process is error prone. A mobile application can increase the efficiency and the productivity of collecting plant observation data. A mobile application also makes it easier to add additional information such as an image of the plant, the time and weather conditions when the data was collected and the location where the data was collected.

The architecture of this application is client-server structure. The client side is the Android mobile application. The server is an Apache web server with a MySQL database. PHP is used as the programming language on the server side. The client communicates with the server through the http protocol. The application uses the MAMP platform.

When the user is collecting the data in the field, the mobile device does not require an Internet connection. When an Internet connection is available, the application will upload data automatically to a server database. The application also allows users to add, modify or delete the observation fields, to export recorded data into an Excel file, and to do some simple data analysis.

There are two major platforms in the mobile device community: iOS and Android. This project chose Android development mainly for the reason of its openness. In addition, all the tools in Android development are free and no special hardware is required.

1 Introduction

Plant breeders, agronomists and scientists need to record a lot of information in the field. Recording data by pen and paper has many problems. For example, handwriting is often difficult to read and can easily be misinterpreted. In order to do data analysis, handwritten data must be entered into a computer or the analysis must be done by hand. In either case, the process is error prone.

The aim of this project is to develop an Android application for collecting plant observation data in the field. A mobile application can increase the efficiency and the productivity of collecting plant observation data. A mobile application also makes it easier to add additional information such as an image of the plant, the time and weather conditions when the data was collected and the location where the data was collected. Figure 1 below shows that a user can create an observation by choosing Trait List, recording the corresponding values for the traits in the selected trait list, setting a modification deadline, adding comments, and taking pictures for the observation or drawing his/her own observation. A trait represents some observable characteristic such as color, height or leaf type. A trait list is a collection of traits.

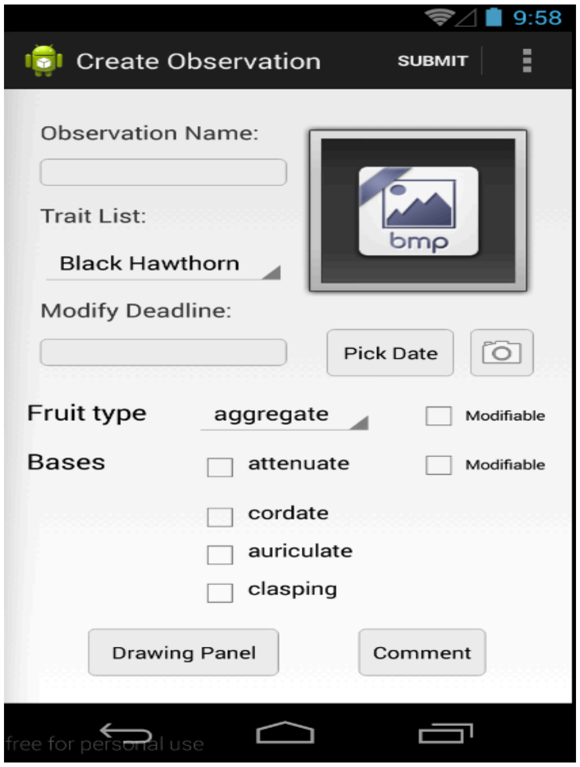


Figure 1 Create Data Observation

When the user is collecting the data in the field, the mobile device does not require an Internet connection. When an Internet connection is available, the application will upload data automatically to a server database. The application also allows users to add, modify or delete the observation fields, to export recorded data into an Excel file, and to do some simple data analysis.

There are two major platforms in the mobile device community: iOS and Android. This project chose Android development [2] mainly for the reason of its openness. In addition, all the tools in the Android development are free and no special hardware is required.

2 Architecture Design

The architecture of this application is client-server structure. The client side is an Android mobile application. The server is an Apache web server with a MySQL database. PHP is used as the programming language on the server side. The client communicates with the server through the http protocol. The application uses the MAMP platform that is a software bundle to run dynamic web sites on Apple Macintosh computers. MAMP is an acronym for Mac OS X operating system, Apache web server, MySQL database and PHP language.

If the Internet is available during the data collection, the observation data will be stored in both local and server databases. Otherwise, the data will be stored in the local SQLite database first, then it will be synchronized to the server MySQL database when the Internet is available.

Figure 2 shows the data synchronization procedure used in this project.

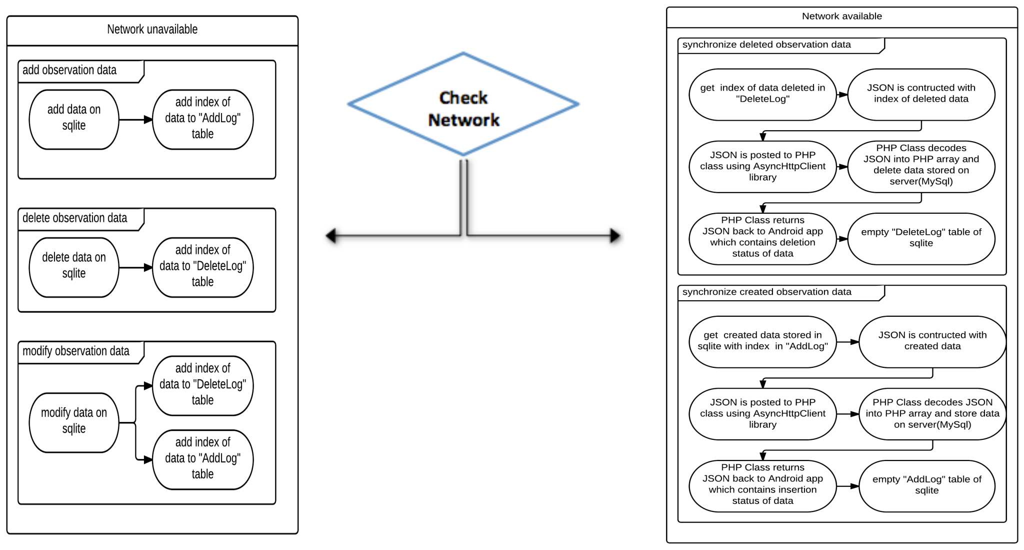

Network unavailable
add observation data
add data on sqlite
add index of data to "AddLog" table
delete observation data
delete data on sqlite
add index of data to "DeleteLog" table
modify observation data
modify data on sqlite
add index of data to "DeleteLog" table
add index of data to "AddLog" table
Check Network
Network available
synchronize deleted observation data
get index of data deleted in "DeleteLog"
JSON is contructed with index of deleted data
JSON is posted to PHP class using AsyncHttpClient library
PHP Class decodes JSON into PHP array and delete data stored on server(MySql)
PHP Class returns JSON back to Android app which contains deletion status of data
empty "DeleteLog" table of sqlite
synchronize created observation data
get created data stored in sqlite with index in "AddLog"
JSON is contructed with created data
JSON is posted to PHP class using AsyncHttpClient library
PHP Class decodes JSON into PHP array and store data on server(MySql)
PHP Class returns JSON back to Android app which contains insertion status of data
empty "AddLog" table of sqlite

Figure 2 Data Synchronization

There are three primary operations for making changes in the database: add, delete and modify. In the local database, there are two additional tables: AddLog and DeleteLog tables to keep track of all additions and deletions to the database when an Internet connection is not available. For the modification operation, we simply delete the previous data and then add the new data. Hence we add two entries in the log tables to keep a record of the modification in the local database when the Internet is not available. Later on when the network is available, the server will make changes to the MySQL database based on the contents in the two log tables.

The source code for checking if the network is available is listed below:

```
public boolean isNetworkConnected(Context context) {
                if (context != null) {
                        ConnectivityManager mConnectivityManager =
(ConnectivityManager) context

        .getSystemService(Context.CONNECTIVITY_SERVICE);
                        NetworkInfo mNetworkInfo = mConnectivityManager
                                .getActiveNetworkInfo();
                        if (mNetworkInfo != null) {
                                return mNetworkInfo.isAvailable();
                        }
                }
                return false;
        }
```

3 Database Design

The E-R diagram for the local SQLite database is shown below in Figure

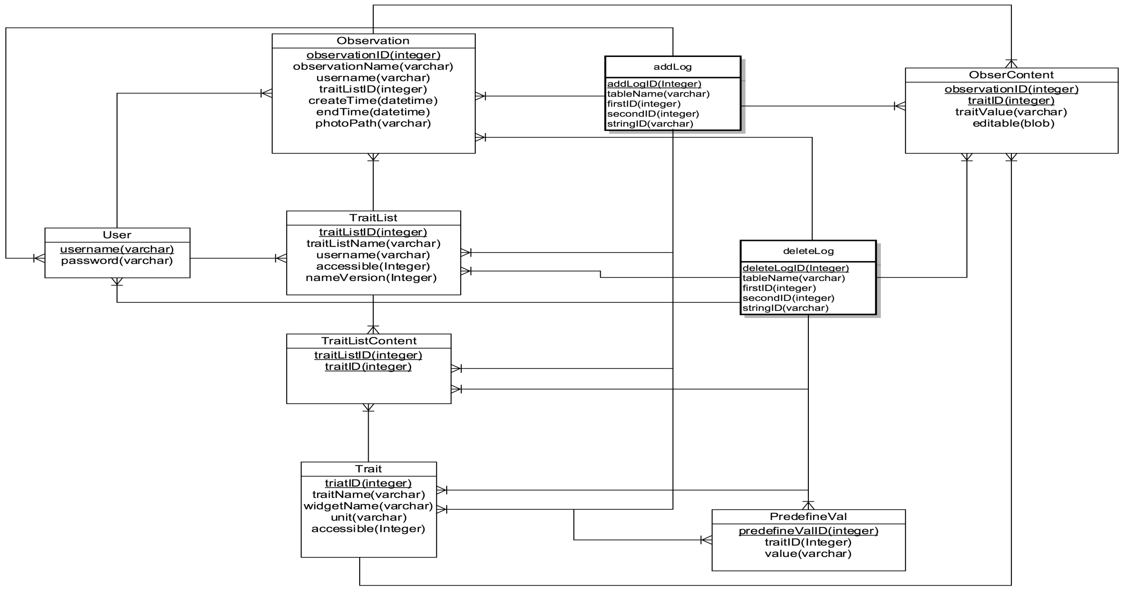


Figure 3 E-R Diagram for the Local Database

As we discussed in the previous section, addLog and deleteLog tables keep track of all the changes made to the local database tables when the network is not available. Hence they are connected with all the tables. Once the network is available, the server will make changes to the server database based on the contents of the two log tables, then delete all the rows in the two log tables.

Figure 4 is the E-R diagram for the server database. It basically has identical tables with the server database, except for a few differences:

a) The server database does not have two log tables.
b) The server has a File table that is used to store the Excel files for data analysis later on.
c) Every table in the server has one more data field: deviceID. It is a unique ID that is used to record the device that was used by the user. In the real world environment, it is possible that a user has multiple devices. He/she can login using any device. It is also possible that multiple users can login and use the same device. Adding the deviceID along with the data when the local data is synchronized to the server is used for data consistency.

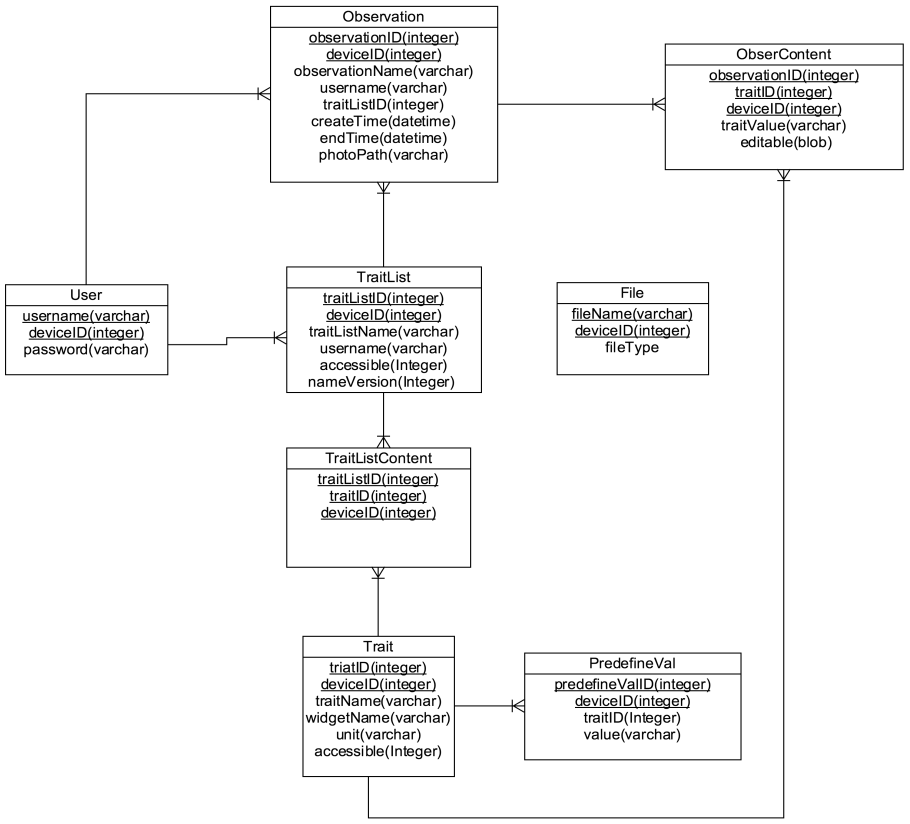


Figure 4 E-R Diagram for Server Database

4 Export to Excel file and Data Analysis

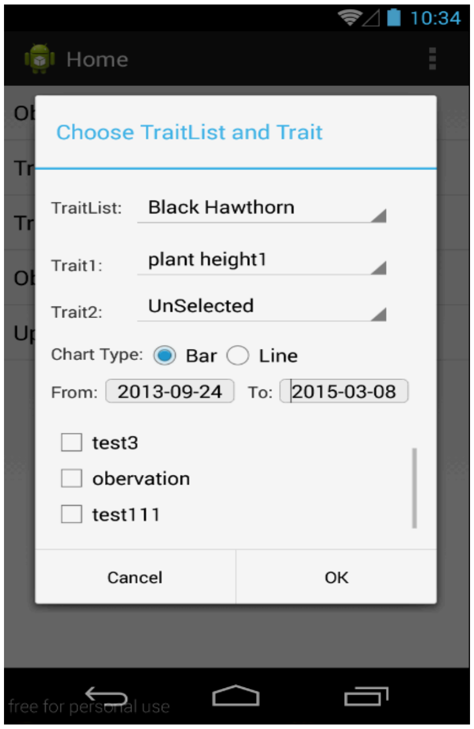
Once the user records his/her observations, the mobile application has the functionality of exporting the observation data into an Excel file. The application will first read all the observation data into a two-dimensional array, then use a Java Excel API called "JML.jar" [3] to generate the Excel file.

The source code of generating the excel file from the two-dimensional array is listed below:

```
try {
                    // use workbook to create itself to write data into the file
                            wwb = Workbook.createWorkbook(new File(fileName));
                    } catch (IOException e) {
                            e.printStackTrace();
                    }
                    if (wwb != null) {
                            // create a writable spreadsheet
                            // @param1-name of spreadsheet,   @param2-location of
spreadsheet
                            WritableSheet ws = wwb.createSheet("sheet1", 0);
                            // create unit
                            for (int i = 0; i < numrows + 1; i++) {
                                    for (int j = 0; j < numcols; j++) {
                                            // @para1-column @para2-row
                                            Label labelC = new Label(j, i, records[i][j]);
                                            try {
                                                    // insert unit into spreadsheet
                                                    ws.addCell(labelC);
                                            } catch (RowsExceededException e) {
                                                    e.printStackTrace();
                                            } catch (WriteException e) {
                                                    e.printStackTrace();
                                            }
                                    }
                            }
                            try {
                                    // write it into file
                                    wwb.write();
                                    // close rescourse and release memory
                                    wwb.close();
                            } catch (IOException e) {
                                    e.printStackTrace();
                            } catch (WriteException e) {
                                    e.printStackTrace();
                            }
```

Once the Excel file is generated, the user can use Excel to view the file assuming the Microsoft Office app is available in the device.

The user can also do some simple data analysis using the Excel file. For example, the user can choose the traitList, then select one or two traits from the traitList to generate either bar or line charts. The screen shot to invoke data analysis is shown in Figure 5 below. It is required that the user needs to choose the time period in order to do observation data analysis. The generated bar chart is shown in Figure 6. The application used AChartEngine [4] for this functionality. AChartEngine is a charting library for Android applications.



Choose TraitList and Trait

TraitList: Black Hawthorn

Trait1: plant height1

Trait2: UnSelected

Chart Type: [✓ Bar] Line

From: 2013-09-24 To: 2015-03-08

- ☐ test3
- ☐ obervation
- ☐ test111

Figure 5 Observation Data Analysis

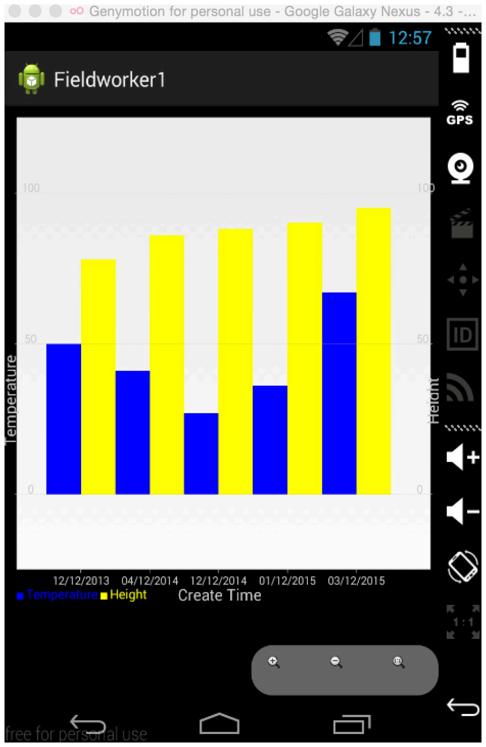


Figure 6 Bar Chart

5 Conclusions

This mobile application assumes that each user will use one device at any given time even if he/she has multiple devices. It is critical to keep data consistency between local data and server data in this application.

When the user registers him/herself in order to use the mobile application, it is required that the registration process must be done when the Internet is available. This is to be sure the username is unique in the entire system. The local data will be appended with the username when it is synchronized into the server database for the same reason of data consistency.

This project aims to provide an electronic-journal for recording science fieldwork data and for bookkeeping. The problem domain we choose in this application is plant observation. It can be easily modified to work in other scientific fields for recording purposes.

References

[1] Computer Science 2008, An Interim Revision of CS 2001(http://www.acm.org/education/curricula/ComputerScience2008.pdf)

[2] Android Developer's Guide. http://developer.android.com/guide/index.html

[3] Java Excel API. http://jexcelapi.sourceforge.net/
[4] AChartEngine, the charting library for Android Application https://code.google.com/p/achartengine/

[5] Abelson, W.F., Collins, C., Sen, R. *Unlocking Android – A Developer's Guide*. Manning Pub. April 2009.

[6] Victor Matos, Rebecca Grasser, Building Applications for the Android OS Mobile Platform: A Primer and Course Materials, *Journal of Computing Sciences in Colleges*, Volume 26 Issue 1, pp: 23-29, October 2010

[7] Derek Riley, Using Mobile Phone Programming to teach Java and Advanced Programming to Computer Scientist, ACM Special Interest Group on Computer Science Education SIGSCE 2012, pp:541-546. Feb. 29-March 3, 2012.

[8] B. N. Schilit, N. Adams, and R. Want. Context-aware Computing Applications. In Proceedings of IEEE Workshop on Mobile Computing Systems and Applications, pp. 85-90, Santa Cruz, CA, Dec. 1994. IEEE Computer Society Press.

[9] A. K. Dey and G. D. Abowd, "Towards a Better Understanding of Context and Context-awareness," in CHI 2000 Workshop on the What, Who, Where, When and How of Context-awareness, 2000.

[10] R. Lowe, P. Mandl, M. Weber. "Context Director: A Context-aware Service for Mobile Context-aware Computing Applications by the Example of Google Android", Tenth Annual IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops 2012), Lugano, Switzerland, March 2012.

[11] Y. Qiu, J. Chu, M. Zheng, T. Gendreau. "The Architecture Design of A Mobile Application for Collecting Plant Observation Data", poster presentation, The 3rd Regional Celebration of Women in Computing in the Upper MidWest (MinneWIC2015), Minneapolis, Feb. 20-21, 2015.