**ANDROID APPLICATION FRAMEWORK FOR QUICKLY DEVELOPING DB BACKED APPLICATIONS**

***Submitted by***

**SHRIRAM D**

**Roll No.: 1201FOSS0019**

**Reg. No.: 75813150002**

**A PROJECT REPORT**

***Submitted to the***

**FACULTY OF SCIENCE AND HUMANITIES**

***in partial fulfilment for the requirement of award of the degree***

***of***

**MASTER OF SCIENCE**

**IN**

**FREE / OPEN SOURCE SOFTWARE (CS-FOSS)**



**CENTRE FOR DISTANCE EDUCATION**

**ANNA UNIVERSITY**

**CHENNAI 600 025**

**AUGUST 2015**

**CENTRE FOR DISTANCE EDUCATION**

**ANNA UNIVERSITY**

**CHENNAI 600 025**

**BONA FIDE CERTIFICATE**

Certified that this Project report titled “**ANDROID APPLICATION FRAMEWORK FOR QUICKLY DEVELOPING DB BACKED APPLICATIONS**” is the bona fide work of Mr. SHRIRAM D. who carried out the research under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other Project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

D.Shriram Dr. Sanjeev Kumar Saini

STUDENT GUIDE

PROJECT IN-CHARGE

**CERTIFICATE OF VIVA-VOCE-EXAMINATION**

This is to certify that Thiru/Ms./Tmt................................................................. (Roll No. ........................; Register No. ........................................) has been subjected to Viva-voce-Examination on ................................... (Date) at ................ (Time) at the Study centre ............................................................... ……….(Name and Address of the Study centre).

**Internal Examiner External Examiner**

Name : Name :

(in capital letters) (in capital letters)

Designation : Designation :

Address : Address :

**Coordinator centre**

Name :

(in capital letters)

Designation :

Address :

Date :

**ACKNOWLEDGEMENT**

I take this opportunity to express my profound gratitude and deep regards to my guide **Dr. Sanjeev Kumar Saini** for his exemplary guidance, monitoring and constant encouragement throughout the course of this project.

I also take this opportunity to express a deep sense of gratitude to **AUKBC Staffs** involved in M.Sc (CS-FOSS) Course for their cordial support, valuable information and guidance, which helped me in completing this task through various stages.

Lastly, I thank Anna University’s Centre for Distance Education (CDE) for providing M.Sc (CS-FOSS) course, without which this assignment would not be possible.

**SHRIRAM D**

**ABSTRACT**

The main objective of the thesis was to provide a framework to generate android applications easily and quickly. The desired screen related requirements will be captured through PHP web page and it generates dynamic\_fields\_admin.xml. Also the PHP web page is provided with a feature for capturing style related information (styles.xml) with an admin user and password for providing enhanced UI features and data abstraction to the Android Application. The Android Application Framework project will read the dynamic\_fields\_admin.xml and styles.xml and render the screen dynamically (with Sqlite database binding).

The following controls are covered in the current version of framework:

1. Normal Edit Text Box
2. Number Text Box
3. Radio Button
4. Date Picker
5. Drop Down
6. Checkbox
7. Phone Text Box
8. Multiline Text Box [Similar to rich text box]
9. Password Text Box
10. Email Text Box
11. Web URL Text Box

The design and implementation of android application framework is described in this document. Android application framework is developed by using Eclipse IDE, Android SDK, and ADT plugin for Eclipse.

**திட்ட பணிசுருக்கம்**

ஆய்வறிக்கை முக்கிய நோக்கம், எளிதாக மற்றும் விரைவில் அண்ட்ராய்டு பயன்பாடுகள் உருவாக்க ஒரு கட்டமைப்பை உருவாக்குதல்.விரும்பிய திரை தொடர்பான தேவைகளை PHP வெப் பக்கம் மூலம் தகவல்களை கைப்பற்றப்பட்டு, அதை dynamic\_fields\_admin.xml மற்றும் styles.xml ஆக உருவாக்குகிறது. அண்ட்ராய்டு கட்டமைப்பு திட்டம். dynamic\_fields\_admin.xml மற்றும் styles.xml படித்து, பின்னர் தானாக விரும்பிய திரை உருவாக்குகிறது.

தற்போதைய பதிப்பு, பின்வரும் உள்ளீடு வகைகளை விவாதிக்கப்படுகின்றன:

1. இயல்பான உரை வகை   
2. எண்கள் வகை  
3. ரேடியோ பட்டன் வகை  
4. தேதி தேர்வாளர் வகை  
5. கீழே இறக்கி வகை  
6. டிக் வகை   
7. தொலைபேசி எண் வகை  
8. பலவரி உரை வகை  
9. கடவுச்சொல் உரை வகை   
10. மின்னஞ்சல் உரை வகை   
11. மின்வலை URL உரை வகை

அண்ட்ராய்டு பயன்பாடு கட்டமைப்பை வடிவமைப்பு மற்றும் செயல்படுத்தல் இந்த ஆவணத்தில் விவரிக்கப்பட்டுள்ளது. அண்ட்ராய்டு பயன்பாடு கட்டமைப்பை எக்லிப்ஸ், அண்ட்ராய்டு SDK, எக்லிப்ஸ் ஏடிடீகளை பயன்படுத்தி உருவாக்கப்பட்டது.

**TABLE OF CONTENTS**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **CHAPTER NO.** | **TITLE** | | **PAGE NO.** | |  |
|  | | **ABSTRACT (ENGLISH)** | | **V** | |
|  | | **ABSTRACT (TAMIL)** | | **VI** | |
|  | | **LIST OF FIGURES** | | **IX** | |
|  | | **LIST OF ABBREVATIONS** | | **X** | |
| **1** | | **INTRODUCTION** | | **1** | |
|  | | 1.1 Objective | | 1 | |
|  | | 1.2 Sub Tasks | | 1 | |
|  | | 1.2.1 PHP Web Page | | 2 | |
|  | | 1.2.2 Android Application Framework to  generate dynamic UI | | 2 | |
|  | | 1.2.3 Generate and deploy the APK file | | 3 | |
|  | | 1.3 Current scope | | 3 | |
|  | | 1.4 Future Visions | | 3 | |
|  | |  | |  | |
| **2** | | **REQUIREMENT ANALYSIS & DESIGN** | | 4 | |
|  | | 2.1 Requirements | | 4 | |
|  | | 2.2 Use case diagram | | 5 | |
|  | | 2.3 Architecture | | 6 | |
|  | | 2.4 Class Diagram | | 7 | |
|  | | 2.4.1 Main Activity Class | | 8 | |
|  | | 2.4.2 Fields Class | | 8 | |
|  | | 2.4.3 MySQLiteOpenHelper Class | | 9 | |
|  | | 2.4.4 LoginScreen Class | | 9 | |
| **CHAPTER NO.** | | **TITLE** | | **PAGE NO.** | |
| **3** | | **PHP** | | 10 | |
|  | | 3.1 PHP Web Page | | 10 | |
|  | | 3.2 Generate XML | | 13 | |
|  | | 3.3 Apache Server | | 13 | |
|  | |  | |  | |
| **4** | | **ANDROID APPLICATION FRAMEWORK** | | 14 | |
|  | | 4.1 Android | | 14 | |
|  | | 4.2 Android Application Framework | | 15 | |
|  | | 4.2.1 Read & Parse XML | | 15 | |
|  | | 4.2.2 Generate Dynamic Data Entry Screen | | 16 | |
|  | | 4.2.3 Sqlite Database Creation | | 20 | |
|  | | 4.2.4 Listing the stored data | | 21 | |
|  | | 4.2.5 Reset the stored data | | 22 | |
|  | | 4.2.6 Dynamic Fields for Normal User | | 22 | |
|  | |  | |  | |
| **5** | | **GENERATE & DEPLOY APK FILE** | | 23 | |
|  | | 5.1 Generate APK file | | 23 | |
|  | | 5.2 Deploy APK file in Android devices | | 23 | |
|  | |  | |  | |
| **6** | | **TESTING** | | 24 | |
|  | | 6.1 Issues & Status | | 25 | |
|  | |  | |  | |
| **7** | | **CONCLUSIONS AND FUTURE WORKS** | | 25 | |
|  | |  | |  | |
|  | | **APPENDIX I – SOURCE CODE** | | 26 | |
|  | |  | |  | |
|  | | **REFERENCES** | |  | |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **FIGURE NO.** | **TITLE** | **PAGE NO.** |
| 2.1 | Usecase Diagram | 5 |
| 2.2 | Architecture Diagram | 6 |
| 2.3 | Class Diagram | 7 |
| 3.1 | PHP Web Page to capture the field parameters | 10 |
| 3.2 | Check box – Field type | 12 |
| 3.3 | Dynamic\_fields\_Admin.xml for android application Framework | 13 |
| 3.4 | Styles.xml for customer information | 13 |
| 4.1 | Android Architecture | 14 |
| 4.2 | Dynamic controls creation – overview | 17 |
| 4.3 | Login Screen | 17 |
| 4.4 | Dynamic data entry screen | 18 |
| 4.5 | Dynamic list screen | 22 |
| 4.6 | Normal user | 22 |
| 5.1 | APK file generation | 23 |

**LIST OF ABBREVIATIONS**

|  |  |
| --- | --- |
| ADT | Android Development Tools |
| APK | Android application package file |
| CSV | Comma Separated Values |
| DB | Database |
| IDE | Integrated Development Environment |
| OS | Operating System |
| PDF | Portable Document Format |
| PDT | PHP Development Tools |
| PHP | Personal Home Page (Hypertext Preprocessor) |
| UML | Unified Modelling Language |
| XML | Extensible Markup Language |

**CHAPTER 1**

**INTRODUCTION**

At present, Mobile Technology is an essential one to fulfil the customer’s day today needs. So a lot of research has been initiated in the field of mobile technology to address these various need s. Today almost every developers aim is to do project in mobile technology especially on the Android platform. This thought force initiated me to do a project in Android. The advantage of this framework is it enables Android users to create android application easy and quickly without any programming knowledge of Android or Java.

* 1. **Objective**

The main objective of the thesis is to design and implement a prototype of Android Application Framework. Here the Eclipse – ADT, Android Studio will be used for android application development. The Java programming language, Eclipse and the android Software Development Kit (SDK) will be used as the development tools and environment.

Integrated development with the Eclipse IDE is selected for the development as it offers direct invoking of tools that are used for developing applications through the eclipse Android Development Tools (ADT) plug-in. Use case, architecture and class diagrams will be used to model activities and processes of the application in the requirements specification phase.

* 1. **Sub Tasks**

This project contain the following three sub tasks

### PHP Web Page

PHP Web page is designed to get from users the following parameters:

* Application Name (mandatory)
* Admin User Name (mandatory)
* Admin Password (mandatory)
* Screen Name
* Field Name (mandatory)
* Field Type (mandatory)
* TypeFace
* TypeSize
* Color
* Field Values

After providing all the parameters, “Generate” button has been provided in the PHP Web page. Clicking this button will generate the dynamic\_fields\_admin.xml and styles.xml file used for generating the UI dynamically. The styles.xml file is generated using the values provided in Type face, Type size and color fields.

### Android Application Framework to generate dynamic UI

Android Application framework will parse the dynamic\_fields\_admin.xml file and styles.xml and generate the UI screen based upon the field name, field type, field values, type face, type size and color values. The following are the sub tasks that need to be done:

* Parsing the XML.
* Generate the Screen for data entry for user specified fields.
* Create database, tables for the user specified fields & store the data into database.
* Providing the screen to list the user entered data as list view style.
* Rest option to delete the database, tables.

### Generate and deploy the .APK file

Eclipse IDE provides facility to generate the APK file. After creating the APK file, it will be easy to deploy it in any of the Android Devices through android installer.

* 1. **Current scope**

The following controls are covered in the current version of the framework:

* Normal Edit Text Box
* Number Text Box
* Radio Button
* Date Picker
* Drop Down
* Checkbox
* Phone Text Box
* Multiline Text Box [Similar to rich text box]
* Password Text Box
* Email Text Box
* Web URL Text Box
  1. **Future Visions**

To enhance the UI, the Android Application Framework can be included with voice recognition feature. This way it could be still more appealing to the users for creating easy mobile applications. Currently around 11 types of controls have been covered. In future, all the possible controls will be included in the android application framework.

**CHAPTER 2**

**REQUIREMENT ANALYSIS & DESIGN**

## Requirements

## Capture the parameters

To generate the Android application framework, the following points have been captured.

* 1. What is the title of application and screen name?
  2. What are the fields have to create?
  3. Each field data type?
  4. How to get the admin User and password?
  5. What are the Type Face, Type Size and colour of each field?
  6. How to enter the values for radio group controls, check boxes captions and drop down items.

If a user interface is provided, it will help in capturing all the user specific fields. So, the PHP web page is designed to address these user requirements. PHP web page will also generate the dynamic\_fields\_admin.xml as well as styles.xml

## Generating Dynamic Screen for data entry

Android Application Framework will read the dynamic\_fields\_admin.xml file and generate the dynamic screen which will be used to enter the values in each field. If successful validation of user name and password for admin then it displays all the controls with edit and delete access to the user. If normal user it displays only the viewable screen for the entered data for the controls. Also it reads the styles.xml to bind the style properties to the controls while displaying in the screen.

## Data Storage

Sqlite is used to create the database for storing the data.

## Listing the entered data

All the entered data in each field will be displayed as a list view.

## Reset Data

Reset option will delete the previous data.

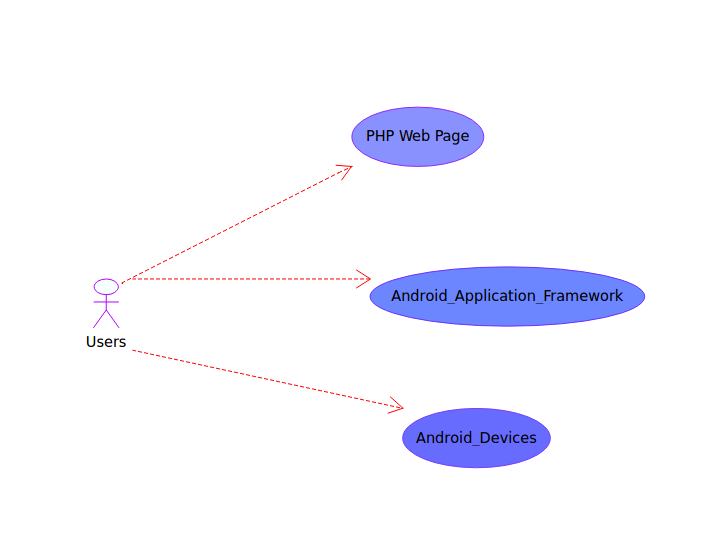
## Deploy the APK file in Android Devices

Finally, Android application framework will generate an APK file through eclipse ADT. It is directly deployed into the Android devices.

## Use Case Diagram

The following use cases are captured in the use case diagram shown in Figure 2.1.

* Users access the PHP page and create the XML.
* Users generate dynamic android application by using android application framework project.
* Users deploy the APK file in android device.



**Figure 2.1. Use case diagram**

## Architecture

Overall architecture of Android Application Framework is as follows:

* + PHP Web Page generates xml file
  + Android Application Framework reads the xml file and generate dynamic screen
  + Sqlite database is used to store the data.

PHP Web Page

Users Enter the

Required fields

Data with admin

Credentials

Creation of

dynamic\_fields\_admin.xml

And styles.xml files

Android Application

Framework project

If not usccessfully validated,

The user is given access

Only to view the data for

The required controls.

If Successfully validated, the

User is given access to

Edit, view , delete the data

For the required controls.

Validation of

Admin

Credentials

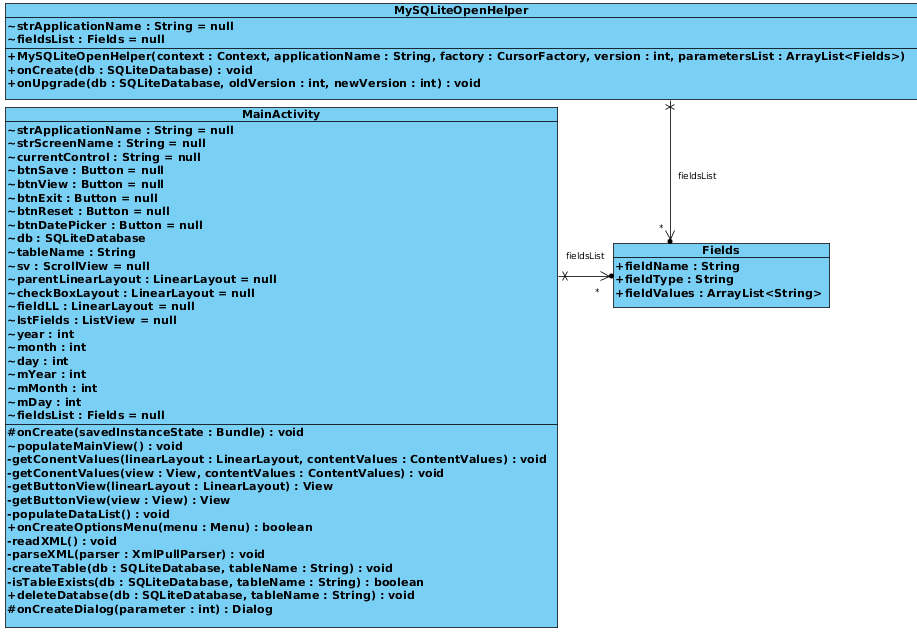
Store data in SQLite database

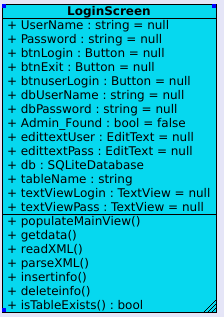
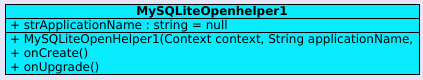
Generation of Apk

file

**Figure 2.2 Android Application Framework – Architecture Diagram**

## Class Diagram



****

**Figure 2.3. Android Application Framework – Class Diagram**

### Main Activity Class

**PopulateMainView()**

PopulateMainView() is the first and main method which calls the other sub methods. It initiates all the global variables.

**ReadXML ()**

This method is used to read the dynamic\_fields.xml from the assets folder.

**ParseXML()**

In this method, dynamic\_fields xml file is parsed and generate a list which hold the dynamic field details.

**GetContentValues()**

Two overloaded methods are used to construct dynamic controls structure list.

**PopulateDataList()**

Method used to create controls dynamically and added to parent linear layout control. It creates database dynamic application based upon the application name if it was not created yet. It also generates dynamic list screen after storing data in sqlite database.

**GetButtonView()**

Two overloaded methods are used to generate Button for date picker control type.

**DeleteDatabase()**

Method used to delete the current database of the application.

### Fields Class

The Field class is used to hold the dynamic field parameters like field name, field type and list of field values. In the MainActivity and MySQLiteOpenHelper classes, the fields class is used as a list.

### MySQLiteOpenHelper Class

MySQLiteOpenHelper class is extended class of SQLiteOpenHelper. It is used to create database and table.

### LoginScreen Class

**PopulateMainView()**

PopulateMainView() is the first and main method which calls the other sub methods. It initiates all the global variables.

**ReadXML()**

This method is used to read the dynamic\_fields.xml from the assets folder.

**ParseXML()**

In this method, dynamic\_fields xml file is parsed and generate a list which hold the dynamic field details.

**GetData()**

Fetches the data from SQLite database and loads it into the String varialbes for validation.

**IsTableExists()**

The function checks if the table exists or not.

**Delete Info()**

The function is called each time when the application framework is created to clear all the junk data from SQLite database.

**CHAPTER 3**

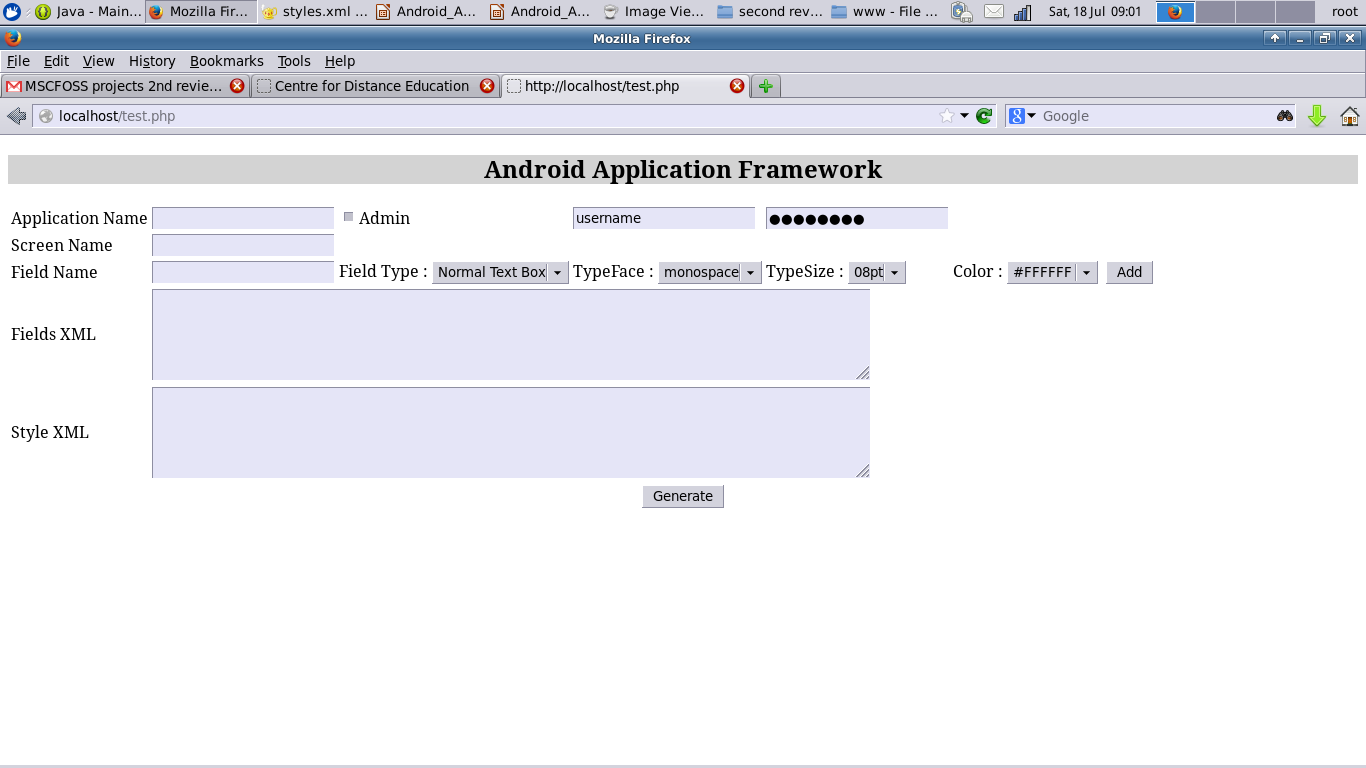
**PHP**

PHP is a server-side scripting language designed for web development but also used as a general-purpose programming language. PHP originally stood for Personal Home Page, it now stands for PHP: Hypertext Preprocessor.

PHP code is interpreted by a web server with a PHP processor module, which generates the resulting web page: PHP commands can be embedded directly into an HTML source document rather than calling an external file to process data.

It has also evolved to include a command-line interface capability and can be used in [standalone](http://en.wikipedia.org/wiki/Computer_software) graphical applications. PHP is free software released under the PHP License. PHP can be deployed on most web servers and also as a standalone shell on almost every operating system and platform, free of charge.

## PHP Web Page



**Figure 3.1. PHP Web Page to capture the field parameters**

**Application Name**

Application name will represent the title of the screen in Android.

**Admin**

If this check box is enables two new text boxes appear to enter the admin user name and password.

**Screen Name**

Screen Name parameter represents functionality of the screen in Android.

**Field Name**

Field Name used to declare the unique field name.

**Field Type**

Field type represents the control type like normal text box, number text box, radio button, date picker, drop down, check box, phone text box, multiline text box, password text box, Email text and web URL text box. If select on the following control type, then Field Value control will be visible.

* Radio Button
* Drop Down
* Checkbox

**TypeFace**

This is a drop down field for choosing the type face for the user defined control.

**TypeSize**

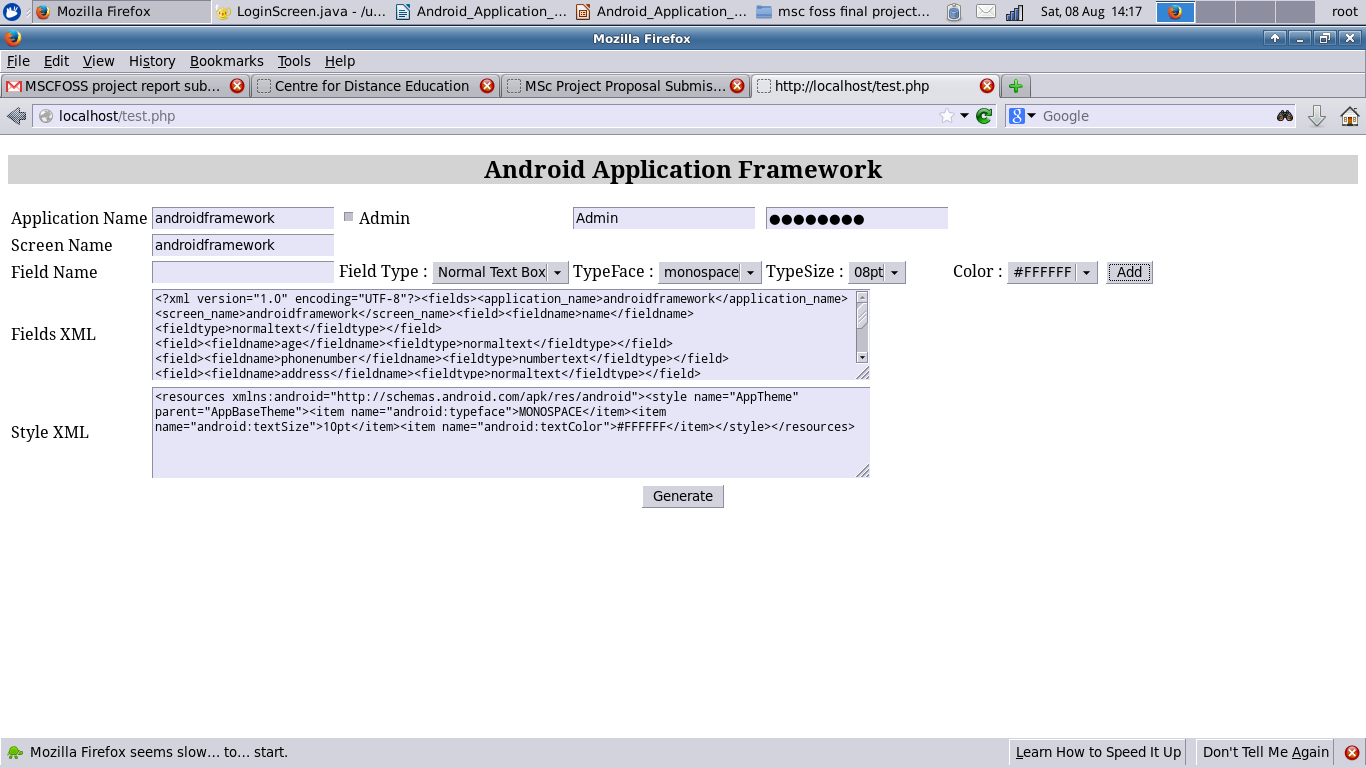
This is a drop down field for choosing the user defined control's size.

**Type Color**

This is a drop down field for choosing the user defined control's color.

**Field Value**

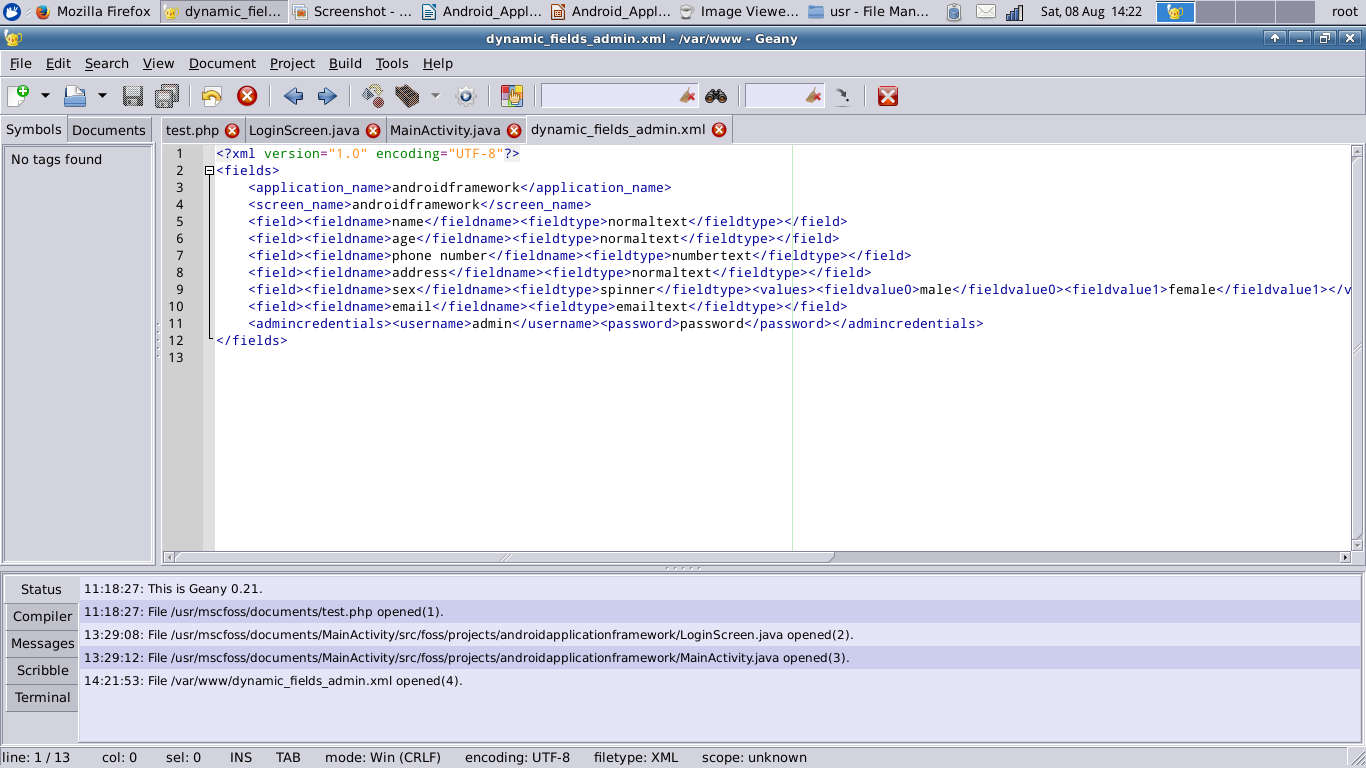
Field Value will be used to capture the values of radio button captions, drop down items or Check box captions.

****

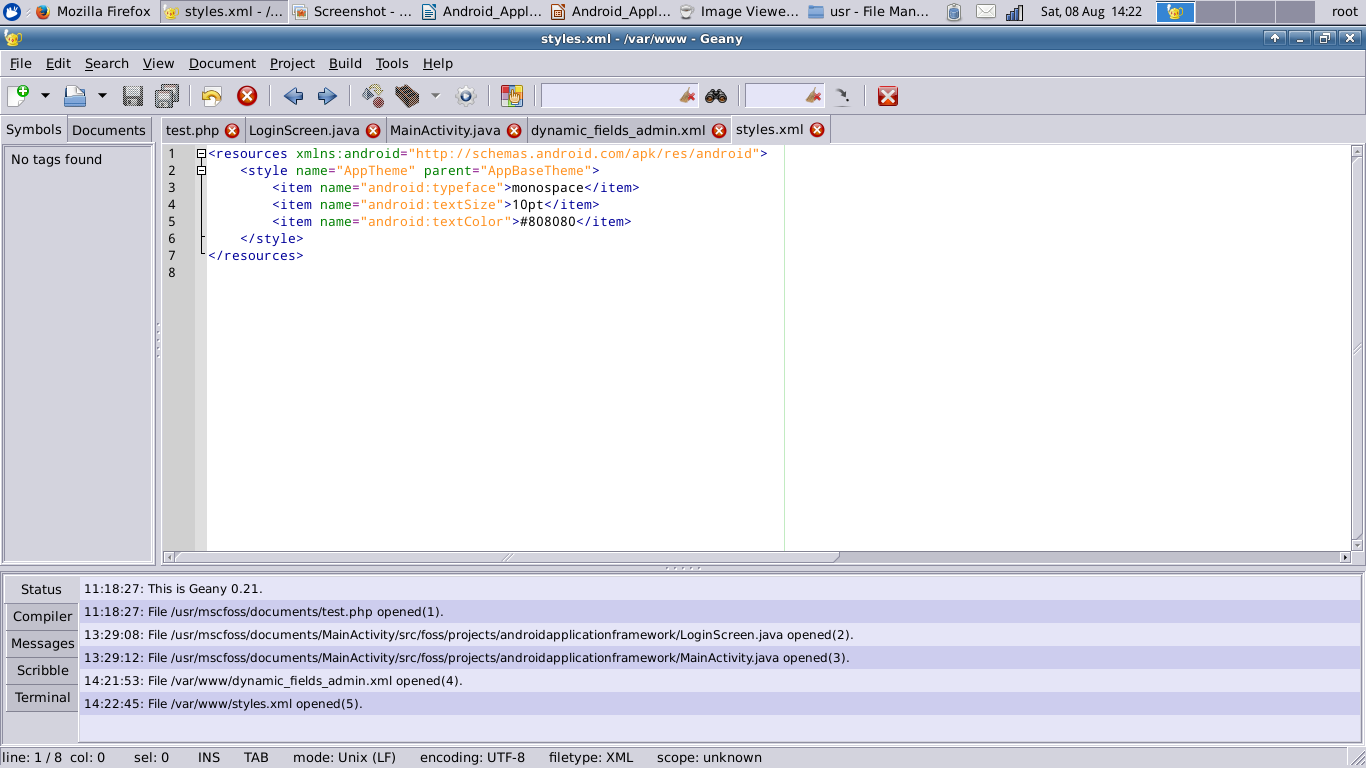
**Figure 3.2. Check Box - Field Type**

## Generate XML

Whenever the user clicks on the “Add” button for the each field details, XML string will be populated in the XML textbox and is shown in Figure 5. When the user clicks the “Generate” button, it will create the dynamic\_fields\_admin.xml file as well as styles.xml shown in Figure 3.3 and Figure 3.4



**Figure 3.3. Dynamic\_Fields\_admin.xml for Android Framework**

****

**Figure 3.4. Styles.xml for Android Framework**

## Apache Server

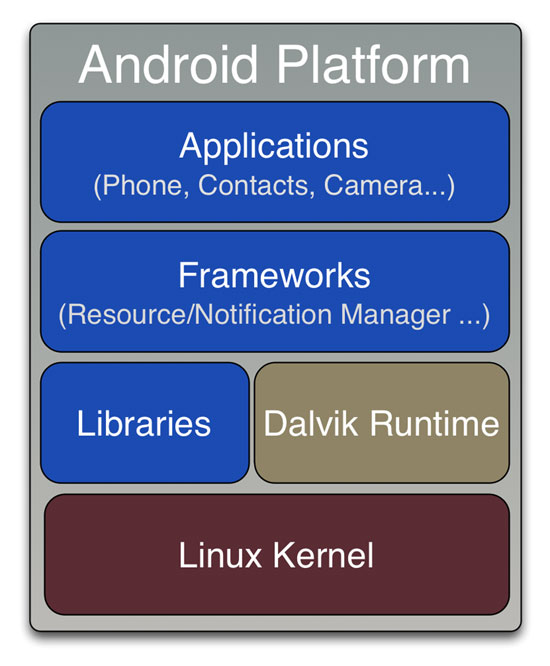
Apache web server is widely used and is the most popular open source web server. PHP Web page - AndroidApplictionFramework.php is hosted in the Apache Web server.

**CHAPTER 4**

**ANDROID APPLICATION FRAMEWORK**

## Android

Android is an operating system based on the Linux kernel, and designed primarily for touchscreen mobile devices such as smartphones and tablet computers. The user interface of Android is based on direct manipulation, using touch inputs that loosely correspond to real-world actions, like swiping, tapping, pinching and reverse pinching to manipulate on-screen objects. Internal hardware such as accelerometers, gyroscopes and proximity sensors are used by some applications to respond to additional user actions, for example adjusting the screen from portrait to landscape depending on how the device is oriented. Android allows users to customize their home screens with shortcuts to applications and widgets, which allow users to display live content, such as emails and weather information, directly on the home screen. Applications can further send notifications to the user to inform them of relevant information, such as new emails and text messages.



**Figure 4.1. Android Architecture**

## Android Application Framework

The Android Application Framework is created based upon the default android project. After creating a new project in eclipse, it will generate one common template project with main activity.java.

Dynamic UI generation logic is based upon the xml, so all the logic has been moved into main activity.java. There are five main components i.e., Parse XML, Generate UI Screen, Sqlite Database creation, Rest Database and Listing the records.

### Read & Parse XML

After generation of the dynamic\_fields\_admin.xml and styles.xml from php web page, it will be placed under /AndroidApplicationFramework/assets and /AndroidapplicationFramework/res/values/ folders respectively.

Android application framework readXML() method read the dynamic\_fields\_admin.xml (/AndroidApplicationFramework/assets) and styles.xml (/AndroidapplicationFramework/res/values/).

The parseXML() method will parse the dynamic\_fields\_admin.xml for the following elements:

* application\_name
* screen\_name
* field
  + fieldname
  + fieldtype
  + values
    - fieldvalue

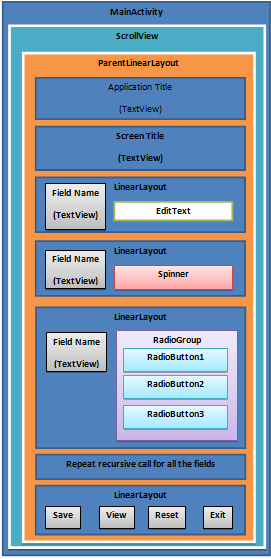
At the time of parsing xml, one list will be created that holds the field details through fields class.

The Login Screen class also parses the dynamic\_fields\_admin.xml file to read the admin credentials with field name admincredentials and store it in the database. Once the data is validated successfully, through intent the MainActivity class is called.

### Generate Dynamic Data Entry Screen

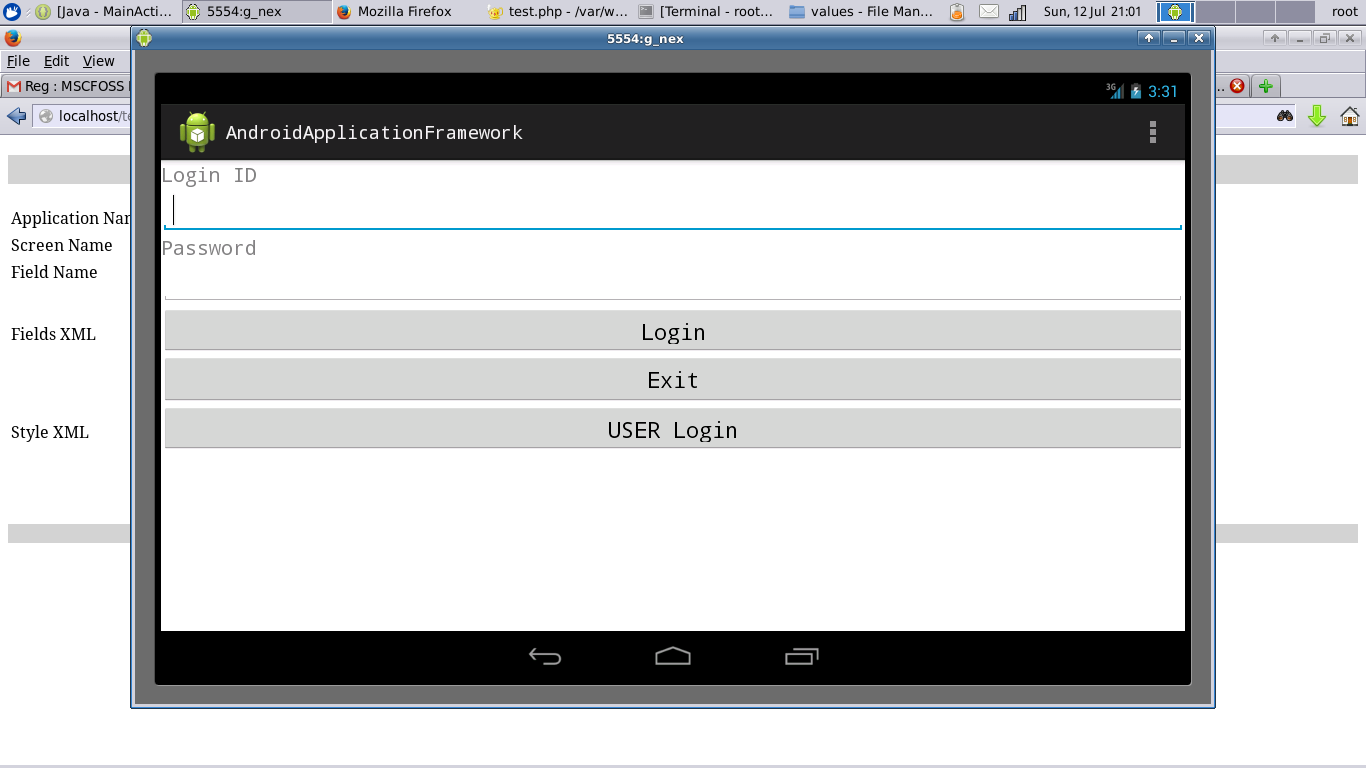
* Login Screen is the base view.
  + Once the admin credentials are successfully validated through intent the MainActivity is called.
  + MainActivity
  + Scroll View has been created and added with MainActivity.
  + Parent Layout has been created and add the following controls:
    - 1. Text view with Application Name
      2. Text view with Screen Name
      3. Read the fields list recursively and create the linear control for each control
         * Create Text view for field name
         * Create dynamic control based upon the control type

For example, if control type is Normal Text box, then edit view is created. If control type is radio button, then radio buttons will be created and added with one radio group.



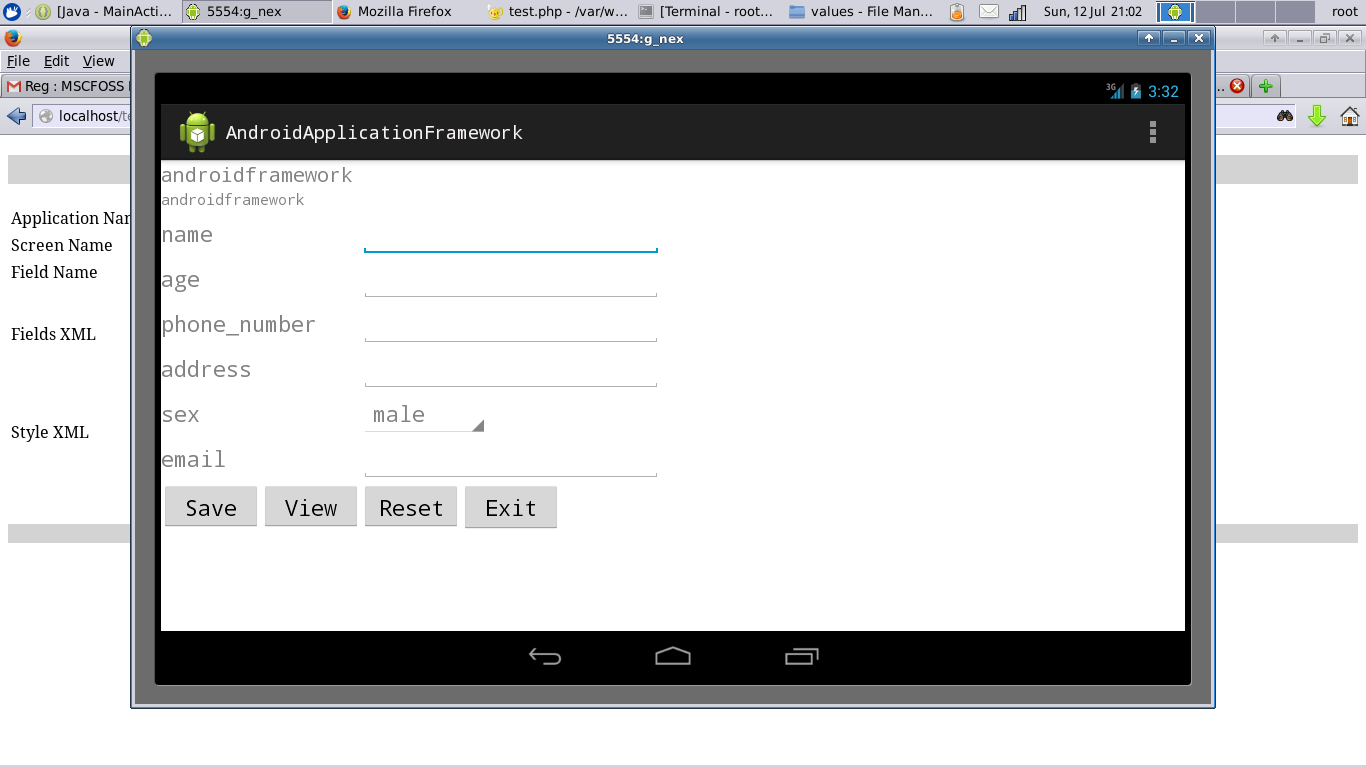
**Figure 4.2. Dynamic controls creation - overview**

The Login Screen will be displayed as given in the Figure 4.3.



**Figure 4.3. Login Screen**

The dynamic screen will be generated after parsing the xml for admin user.

****

**Figure 4.4. Dynamic data entry screen**

**Edit Text Control**

Edit Text Control is common control and is used for various types. In Android there is option for Input Type, by using the input type various control type can be created based upon the user requirements.

**Normal Edit Text**

Control used to enter common text without restricting.

**Number Text**

Control used to enter only numbers.

InputType.TYPE\_CLASS\_NUMBER.

**Email Text**

Control used to enter email type fields.

InputType.TYPE\_TEXT\_VARIATION\_EMAIL\_ADDRESS

**Phone Text**

Control used to enter phone numbers

InputType.TYPE\_CLASS\_PHONE

**URI Text**

Control used to capture web site url details.

InputType.TYPE\_TEXT\_VARIATION\_URI

**Multiline Text**

Control used to enter multiline text contents. The following Editor Info details have to provide at the time of setting the input type:

EditorInfo.TYPE\_CLASS\_TEXT | EditorInfo.TYPE\_TEXT\_FLAG\_MULTI\_LINE | EditorInfo.TYPE\_TEXT\_FLAG\_IME\_MULTI\_LINE

**Password Text**

Control used to enter mask characters.

InputType.TYPE\_MASK\_VARIATION

**Check Box Control**

Check box control can be used for multiple option selection. Field Value list is holding the each check boxes caption. At the time of save, selected check box caption will be stored against the check box field name with comma separated values.

**Radio Button Control**

Radio Button control can be used to select at least one option among the various radio buttons. So while adding radio button controls, first of all, one radio group control created and then all the radio buttons have added into radio group control. So that user can able to select at least one radio button. Field Value list is holding the each radio button caption. At the time of save, selected radio button (0n) caption will be stored against the field name of radio button.

**Drop Down Control (Spinner)**

Drop down (Spinner) normally used to select one value from the list of values. Field Value list is holding all the list of values. At the time of save, the selected drop down value will be stored against the drop down field name.

**Date Picker Control**

Date Picker control is used to select the date from calendar. Button dynamically created with current date and added with parent linear layout. At the time of data entry, we can click on the button, so it will populate the calendar to select the required date. After date selection in the calendar, selected date value is assigned as button’s text.

At the time of save, button’s text (selected date value) will be stored against the date picker control field name.

### Sqlite Database Creation

Android contains a set of C and C++ libraries that are responsible for performance optimization and efficiency. This includes SQLite that is a lightweight relational database engine.

At the time of controls creation, database has been created with the following name format:

**dynamic\_db\_<application name>**

Table has been created with <application name>. In this table, columns have been created with COL\_<user specified field name>. If user specifies space between words, then space will be replaced by underscore (“\_”) to avoid issues.

For example of Android Framework, user specified application name as “androidframework” then the database will be created as below:

**Database :** dynamic\_db\_**androidframework**

**Table :** **androidframework**

**Columns :** COL\_ID (Primary Key, Auto generated unique number)

: COL\_Name (Text)

: COL\_age (Text)

: COL\_phone number (Text)

: COL\_address (Text)

: COL\_sex (Text)

: COL\_email (Text)

**Database :** dynamic\_db\_**androidframework**

**Table :** **Android\_Admin**

**Columns :** COL\_ID (Primary Key, Auto generated unique number)

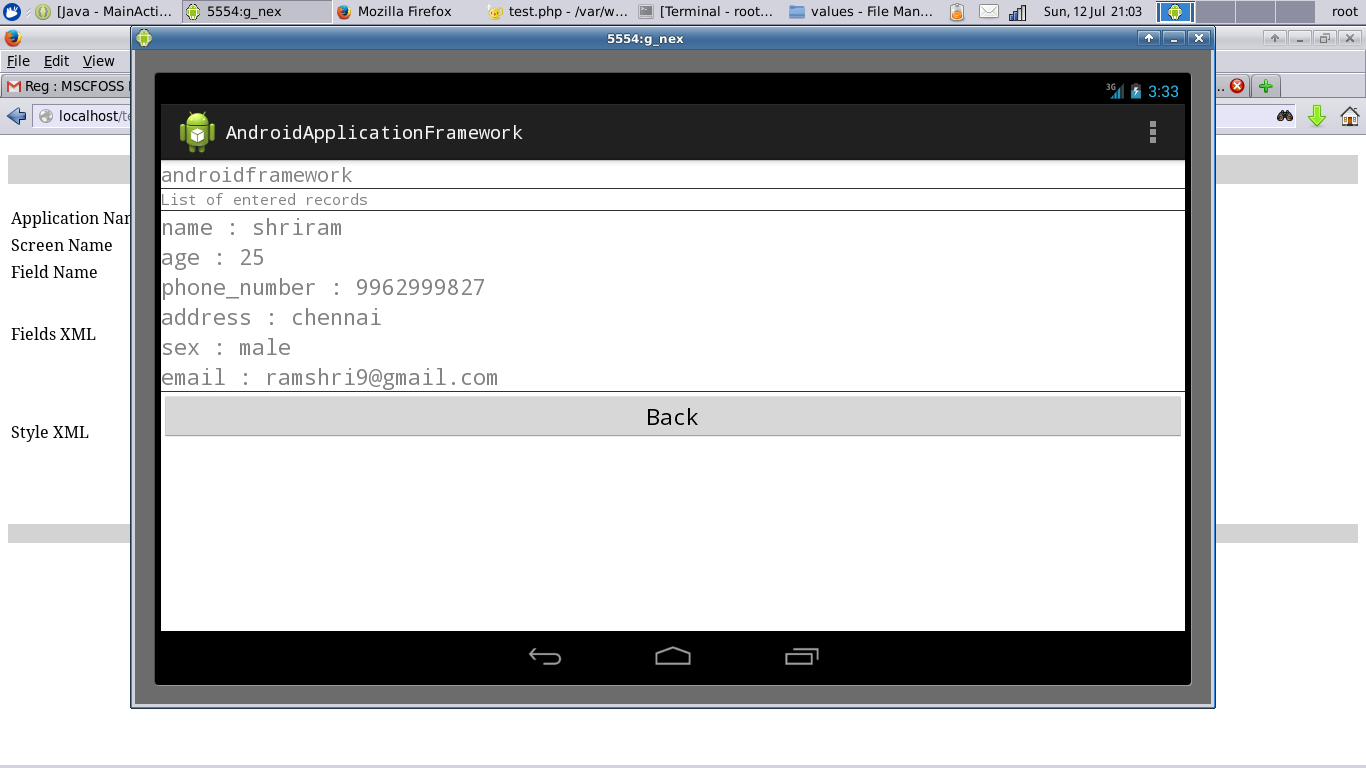
: COL\_Username (Text)

: COL\_Password (Text)

### Listing the stored data

List the stored the data by creating the TextView controls for each field name and its value.

* MainActivity is the base view.
* Scroll View has been created and added with MainActivity.
* Parent Layout has been created and add the following controls:
* Text view with Application Name
* Text view with Screen Name
* Read the fields list recursively and create the linear control for each control
* Create Text view for field name : field value
* Back button

****

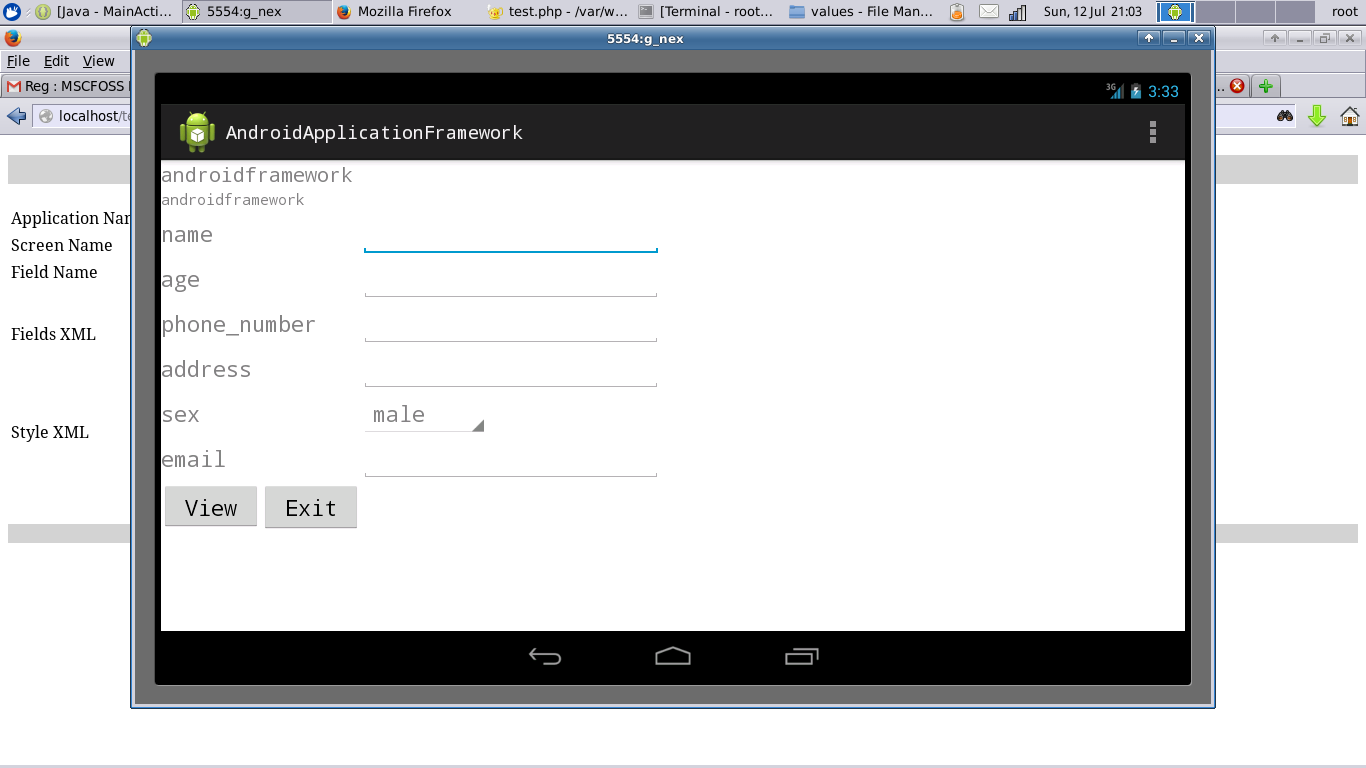
**Figure 4.5. Dynamic List Screen**

### Reset the stored data

When press on the Reset Button, then Delete the dynamic database, tables for the application. Exit button used to quit from the application.

### Dynamic Fields for Normal User

When the login credentials are not correct, the user can login as a normal user. The data for the created controls will be fetched from SQLite database and displayed to the user. Only the View option will be enabled for the normal users whereas for the Admin user Save and Reset Options will be enabled. After loggin as a normal user, the screen will be displayed as given in the figure 4.6 below.



**Figure 4.6. Normal User**

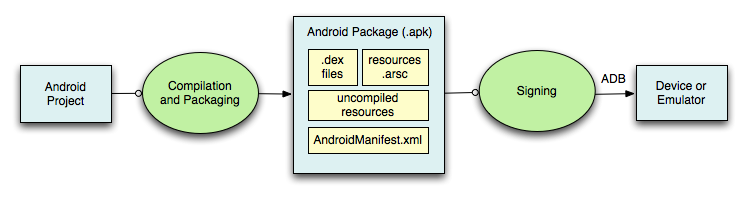
**Figure 4.6. Normal User**

**CHAPTER 5**

**GENERATE AND DEPLOYING APK FILE**

## Generate APK file

Android applications are written in the Java language, compiled into byte codes which will be converted to a .dex file (Dalvik executable file) using the dx converter. This will further be compiled in to android application package file (APK file), that can be installed on the android devices. The following diagram depicts the components involved in building and running an application:



**Figure 5.1. APK file generation**

In Eclipse, the ADT plugin incrementally builds your project as you make changes to the source code. Eclipse outputs an .APK file automatically to the bin folder of the project, so you do not have to do anything extra to generate the .APK.

## Deploy APK file in Android devices

Android Apps Installer is used to install the APK in devices. The following steps have to follow to deploy the APK into android device:

* Copy the AndroidApplicationFramework.apk file into android’s device memory. Download and install the Apps Installer application from the Android Market. Once installed, the Apps Installer will display the APK files on the memory card.
* Double Click on the AndroidApplicationFramework.apk file, it will be installed on the Android device.

**CHAPTER 6**

**TESTING**

Unit testing is very important at the time of development. All the unit test cases have been identified at the time of requirement analysis. So, it was easy to test the unit test cases and fix the issues quickly. In Android application framework project, all the possible control type has been used and tested. There are unique scenarios for Radio button, Date Picker, Check box and Drop down control type. But all the controls have been used in two kind of sample application. Also the application was tested with different attributes for color, typeface and type size. Each scenario was tested with correct username password for admin credentials and wrong credentials. To test Android application framework, there are two sample application has been created and tested.

* Patient Information System
  + Patient Name : Normal Text Edit control type
  + Patient Age : Number Text Edit control type
  + Sex : Radio button control type
  + Patient Phone : Phone Text Edit control type
  + Patient Email : Email Text Edit control type
  + Patient Diseases : Check box control type
  + Last Consultation Date : Date picker control type
  + Next Appointment Date : Date picker control type
* AndroidApplicationFramework
  + Name : Normal Text Edit control type
  + Age : Number Text Edit control type
  + Phone : Phone Text Edit control type
  + Address : Normal Text Edit Control type
  + Sex : Radio button control type
  + Email : Email Text Edit control type
* Next Appointment Date : Date picker control type

## Issues & Status

Initially there was issue noticed at the time of saving the data. It was not stored in SQLite database. When debugging, it was identified that creation of database has issues and it was resolved. There were few issues while binding the controls with the layout in the android application. Debugged the same and corrected to correctly bind the controls. There were some issues when calling the main activity through intent. Later it was found out to be a syntactical error and identified and fixed the same.

**CHAPTER 7**

**CONCLUSIONS AND FUTURE WORKS**

The thesis project has covered Android application framework in the Android Platform. It explains briefly the need for the framework and its intended user. The requirement analysis and design and the tools and technologies used are explained. The Basic components of Android applications framework is described briefly. The generation and deployment of the APK file is explained followed by the testing. The goal of the project i.e. creation of an Android Application Framework is achieved. Even though the goal of the project is achieved there are a few more features and challenges that need to be done in the future and these have been identified as below:

* Currently around 11 types of controls have been covered. In future, all the possible controls will be included in the android application framework.
* To enhance the UI the Android Application Framework can be included with voice recognition feature. This way it could be still more appealing to the users for creating easy mobile applications.

APPENDIX I

SOURCE CODE

**Android\_Application\_Framework.php**

<html>

<head>

<script type="text/javascript">

/\*

\* Function used to create XML for all the controls and populate in txtXML.

\*/

var main\_xml;

function addText()

{

//Variables decalaration

var applicationTitle, screenTitle, selectedControls,controlName,controlType,controlValue, xmlText,typeface,typesize,styleText,adminuser,adminspass;

//Assign values

applicationTitle =document.getElementById("txtApplicationName").value;

screenTitle = document.getElementById("txtScreenName").value;

selectedControls = document.getElementById("txtXml").value;

selectedStyle = document.getElementById("stylexml").value; controlName = document.getElementById("controlName").value;

controlType = document.getElementById("controlType").value;

controlValue = document.getElementById("controlValue").value;

typeface = document.getElementById("FontFace").value;

typesize = document.getElementById("TypeSize").value;

color = document.getElementById("Color").value;

//Framing xml

xmlText = "\<\?xml version=\"1.0\" encoding=\"UTF-8\"\?\>\<fields\>";

styleText = "\<resources xmlns\:android=\"http://schemas.android.com/apk/res/android\">";

//Application Name

xmlText = xmlText + "<application\_name>"+applicationTitle+"</application\_name>";

//Screen Name

xmlText = xmlText + "<screen\_name>"+applicationTitle+"</screen\_name>";

styleText = styleText + "\<style name=\"AppTheme\" parent=\"AppBaseTheme\">";

if ( selectedControls.length > 0 )

{

xmlText = selectedControls.replace("</fields>","") + "\n" + getXml(controlName,controlType,controlValue);

}

else

{

xmlText = xmlText + getXml(controlName,controlType,controlValue);

}

xmlText = xmlText + "</fields>";

//if(document.getElementById("radadmin").checked)

//{

//adminuser = document.getElementById("adminuser").value;

//adminpass = document.getElementById("adminpass").value;

//xmlText = xmlText + "<admincredentials>";

//xmlText = xmlText + "<username>"+adminuser+"</username>";

//xmlText = xmlText + "<password>"+adminpass+"</password>";

//xmlText = xmlText + "</admincredentials>";

//}

//else

//{

//}

document.getElementById("txtXml").innerHTML = xmlText;

main\_xml = xmlText;

styleText = styleText + "\<item name=\"android:typeface\">" + typeface + "</item>";

styleText = styleText + "\<item name=\"android:textSize\">" + typesize + "</item>";

styleText = styleText + "\<item name=\"android:textColor\">" + color + "</item>";

styleText = styleText + "</style>";

styleText = styleText + "</resources>";

document.getElementById("stylexml").innerHTML = styleText;

//Clear the content of the control

document.getElementById("controlName").value = "";

document.getElementById("controlType").selectedIndex = 0;

document.getElementById("controlValue").value = "";

}

/\*

\* Function used to show or hide the controlValue based upon the control type

\* Usually control values parameter is required only for Drop down, Radio Button and Checkbox\*/

function visibleControl(sel)

{

var controlType = sel.options[sel.selectedIndex].value;

if ( controlType == "spinner" || controlType == "radiobutton" || controlType == "checkbox" )

{

document.getElementById("controlValue").style.display = "block"; }

else

{

document.getElementById("controlValue").style.display = "none";

}

}

function displaytext()

{

//var AdminValue = document.getElementById("radadmin").value;

if(document.getElementById("radadmin").checked)

{

document.getElementById("adminuser").disabled= false;

document.getElementById("adminpass").disabled= false;

document.getElementById("adminuser").style.visibility = "visible";

document.getElementById("adminpass").style.visibility = "visible";

}

else

{

//document.getElementById("adminuser").value= '';

//document.getElementById("adminpass").value= '';

document.getElementById("adminuser").disabled= true;

document.getElementById("adminpass").disabled= true;

document.getElementById("adminuser").style.visibility = "hidden";

document.getElementById("adminpass").style.visibility = "hidden";

}

}

/\*

\* Function used to ammend the | symbol when press enter key

\*/

function fKeyDown(e)

{

var kc = window.event ? window.event.keyCode : e.which;

if (kc == 13)

{

document.getElementById('controlValue').value = document.getElementById('controlValue').value + "|";

}

}

/\*

\* Function used to get the XML based upon each control add

\*/

function getXml(controlName, controlType, controlValue)

{

var xmlText = "<field><fieldname>" + controlName + "</fieldname>"; xmlText = xmlText + "<fieldtype>" + controlType +"</fieldtype>";

if (controlValue.length > 0 )

{

var controlValues = controlValue.split("|");

if (controlValues.length > 0 )

{

xmlText = xmlText + "<values>"; for (var index=0,len=controlValues.length; index<len; index++)

{

xmlText = xmlText + "<fieldvalue"+index+">"+ controlValues[index] +"</fieldvalue"+index+">";

}

xmlText = xmlText + "</values>";

}

}

xmlText=xmlText + "</field>";

return xmlText;

}

function loadcontrols()

{

document.getElementById("adminuser").style.visibility = "hidden";

document.getElementById("adminpass").style.visibility = "hidden";

}

function checkadmin()

{

var adminuser,adminpass;

if(document.getElementById("radadmin").checked)

{

adminuser = document.getElementById("adminuser").value;

adminpass = document.getElementById("adminpass").value;

main\_xml = main\_xml + "<admincredentials>";

main\_xml = main\_xml + "<username>"+adminuser+"</username>";

main\_xml = main\_xml + "<password>"+adminpass+"</password>";

main\_xml = main\_xml + "</admincredentials>";

document.getElementById("txtXml").innerHTML = main\_xml;

}

}

</script>

</head>

<body onload="loadcontrols();">

<div style="background-color:lightgrey"><h2 align="center">Android Application Framework</h2></div>

<form method="post" action="<?php echo $\_SERVER['PHP\_SELF']; ?>">

<table>

<tr>

<td>Application Name</td>

<td><input type="text" name="txtApplicationName" id="txtApplicationName"/></td>

<td><input type="checkbox" name="adminval" id="radadmin" onchange="displaytext();"/>Admin </td>

<td><input type="text" name="username" id="adminuser" value="username"/> </td>

<td><input type="password" name="adminpassword" id="adminpass" value="password"/> </td>

</tr>

<tr>

<td>Screen Name</td><td><input type="text" name="txtScreenName" id="txtScreenName"/></td>

</tr>

<tr border><td>Field Name</td><td><input type="text" name="controlName" id="controlName"></td>

<td>

Field Type :

<select name="controlType" id="controlType" onchange="visibleControl(this);">

<option value="normaltext">Normal Text Box</option>

<option value="numbertext">Number</option>

<option value="radiobutton">Radio Button</option>

<option value="datepicker">Date Picker</option>

<option value="spinner">Drop Down</option>

<option value="checkbox">Check Box</option>

<option value="phonetext">Phone</option>

<option value="multilinetext">Multiline Text</option>

<option value="passwordtext">Password Text</option>

<option value="emailtext">Email</option>

<option value="uritext">Web URL</option>

</select>

</td>

<td>

TypeFace :

<select name="FontFace" id="FontFace" onchange="visibleControl(this);">

<option value="MONOSPACE">monospace</option>

<option value="SERIF">serif</option>

</select>

</td>

<td>

TypeSize :

<select name="TypeSize" id="TypeSize" onchange="visibleControl(this);">

<option value="10pt">08pt</option>

<option value="10pt">09pt</option>

<option value="10pt">10pt</option>

<option value="10pt">11pt</option>

<option value="12pt">12pt</option>

<option value="15pt">13pt</option>

<option value="15pt">14pt</option>

<option value="15pt">15pt</option>

</select>

</td>

<td>

Color :

<select name="Color" id="Color" onchange="visibleControl(this);">

<option value="#FFFFFF">#FFFFFF</option >

<option value="#FFFF00">#FFFF00</option >

<option value="#FF00FF">#FF00FF</option >

<option value="#FF0000">#FF0000</option >

<option value="#C0C0C0">#C0C0C0</option >

<option value="#808080">#808080</option >

<option value="#808000">#808000</option >

<option value="#800080">#800080</option >

<option value="#800000">#800000</option >

<option value="#00FFFF">#00FFFF</option >

<option value="#00FF00">#00FF00</option >

<option value="#008080">#008080</option >

<option value="#008000">#008000</option >

<option value="#0000FF">#0000FF</option >

<option value="#000080">#000080</option >

<option value="#000000">#000000</option >

</select>

</td>

<td>

<textarea name="controlValue" id="controlValue" rows="5" cols="30" style="display:none" onKeyDown=javascript:fKeyDown(event);></textarea>

</td>

<td><input type="button" onclick="addText();" name="btnAdd" value="Add" width="100"/></td>

</tr>

<tr>

<td>Fields XML </td><td colspan="6" align="left"><textarea name="txtXml" id="txtXml" rows="5" cols="100"></textarea>

</td>

</tr>

<tr>

<td>Style XML </td><td colspan="6" align="left"><textarea name="stylexml" id="stylexml" rows="5" cols="100"></textarea>

</td>

</tr>

<tr></tr>

</table>

<div align="center"><input type="submit" name="submit" value="Generate" align="center" onclick="checkadmin();"></div>

<p style="background-color:lightgrey" align="center">

<?php

if(isset($\_POST['submit']))

{

$txtXml = $\_POST['txtXml'];

$var = file\_put\_contents("/var/www/dynamic\_fields\_admin.xml",$txtXml);

$stylexml = $\_POST['stylexml'];

$var1 = file\_put\_contents("/var/www/styles.xml",$stylexml);

echo $var;

echo "dynamic\_fields.xml has been generated successfully"; }

?>

</p>

</form>

</body>

</html>

**dynamic\_fields\_admin.xml**

<?xml version="1.0" encoding="UTF-8"?>

<fields>

<application\_name>androidframework</application\_name>

<screen\_name>androidframework</screen\_name>

<field><fieldname>name</fieldname><fieldtype>normaltext</fieldtype></field>

<field><fieldname>age</fieldname><fieldtype>normaltext</fieldtype></field>

<field><fieldname>phone number</fieldname><fieldtype>numbertext</fieldtype></field>

<field><fieldname>address</fieldname><fieldtype>normaltext</fieldtype></field>

<field><fieldname>sex</fieldname><fieldtype>spinner</fieldtype><values><fieldvalue0>male</fieldvalue0><fieldvalue1>female</fieldvalue1></values></field>

<field><fieldname>email</fieldname><fieldtype>emailtext</fieldtype></field>

<admincredentials><username>admin</username><password>password</password></admincredentials>

</fields>

**styles.xml**

<resources xmlns:android="http://schemas.android.com/apk/res/android">

<style name="AppTheme" parent="AppBaseTheme">

<item name="android:typeface">monospace</item>

<item name="android:textSize">10pt</item>

<item name="android:textColor">#808080</item>

</style>

</resources>

**MainActivity.java**

private void **readXML()**

{

XmlPullParserFactory pullParserFactory;

try

{

pullParserFactory = XmlPullParserFactory.newInstance();

XmlPullParser parser = pullParserFactory.newPullParser();

AssetManager assetManager = getAssets();

InputStream inputStream = assetManager.open("dynamic\_fields.xml");

//InputStream in\_s = getApplicationContext().getAssets().open("dynamic\_fields.xml");

parser.setFeature(XmlPullParser.FEATURE\_PROCESS\_NAMESPACES, false);

parser.setInput(inputStream, null);

parseXML(parser);

}

catch (XmlPullParserException e)

{

e.printStackTrace();

}

catch (IOException e)

{

e.printStackTrace();

}

}

private void **parseXML**(XmlPullParser parser) throws XmlPullParserException,IOException

{

int eventType = parser.getEventType();

Fields currentField = null;

while (eventType != XmlPullParser.END\_DOCUMENT){

String name = null;

switch (eventType){

case XmlPullParser.START\_DOCUMENT:

fieldsList = new ArrayList();

break;

case XmlPullParser.START\_TAG:

name = parser.getName();

if (name.equalsIgnoreCase("application\_name"))

{

this.strApplicationName = parser.nextText();

}

else if (name.equalsIgnoreCase("screen\_name"))

{

this.strScreenName = parser.nextText();

}

else if (name.equalsIgnoreCase("field"))

{

currentField = new Fields();

}

else if (currentField != null)

{

if (name.equalsIgnoreCase("fieldname"))

{

currentField.fieldName = parser.nextText();

currentField.fieldName = currentField.fieldName.replace(' ', '\_');

}

else if (name.equalsIgnoreCase("fieldtype"))

{

currentField.fieldType = parser.nextText();

}

else if (name.equalsIgnoreCase("values"))

{

currentField.fieldValues = new ArrayList<String>();

}

else if (name.startsWith("fieldvalue"))

{

currentField.fieldValues.add(parser.nextText());

}

}

break;

case XmlPullParser.END\_TAG:

name = parser.getName();

if (name.equalsIgnoreCase("field") && currentField != null)

{

fieldsList.add(currentField);

}

}

eventType = parser.next();

}

}

void **populateMainView()**

{

sv = new ScrollView(this);

sv.setLeft(10);

parentLinearLayout = new LinearLayout(this);

parentLinearLayout.setGravity(parentLinearLayout.TEXT\_ALIGNMENT\_GRAVITY);

parentLinearLayout.setOrientation(LinearLayout.VERTICAL);

sv.addView(parentLinearLayout);

parentLinearLayout.setDividerPadding(5);

TextView textViewControl = null;

EditText editTextControl = null;

RadioGroup radioGroup = null;

RadioButton radioButtonControl = null;

Spinner spinnerControl = null;

CheckBox checkBox = null;

//Read the Fields xml file

readXML();

LinearLayout.LayoutParams params = new LinearLayout.LayoutParams(

ViewGroup.LayoutParams.FILL\_PARENT, ViewGroup.LayoutParams.WRAP\_CONTENT);

params.gravity = Gravity.CENTER;

//Set Application Title

textViewControl = new TextView(this);

textViewControl.setText(strApplicationName);

textViewControl.setLeft(10);

textViewControl.setGravity(View.TEXT\_ALIGNMENT\_CENTER);

textViewControl.setTextSize(20);

textViewControl.setLayoutParams(params);

parentLinearLayout.addView(textViewControl);

View line = new View(this);

line.setLayoutParams(new ViewGroup.LayoutParams(ViewGroup.LayoutParams.FILL\_PARENT, 1));

line.setBackgroundColor(Color.rgb(51, 51, 51));

parentLinearLayout.addView(line);

//Set Table Name

tableName = strApplicationName.replace(' ','\_');

//Set Screen heading

textViewControl = new TextView(this);

textViewControl.setText(strScreenName);

textViewControl.setLeft(10);

textViewControl.setTextSize(15);

textViewControl.setLayoutParams(params);

parentLinearLayout.addView(textViewControl);

line = new View(this);

line.setLayoutParams(new ViewGroup.LayoutParams(ViewGroup.LayoutParams.FILL\_PARENT, 1));

line.setBackgroundColor(Color.rgb(51, 51, 51));

parentLinearLayout.addView(line);

//Create all the controls for fields dynamically

if ( fieldsList != null && fieldsList.size() > 0 )

{

for(Fields currentField : fieldsList)

{

fieldLL = new LinearLayout(this);

fieldLL.setOrientation(LinearLayout.HORIZONTAL);

fieldLL.setLeft(10);

if (currentField.fieldType.equalsIgnoreCase("normaltext"))

{

textViewControl = new TextView(this);

textViewControl.setText(currentField.fieldName);

textViewControl.setWidth(200);

fieldLL.addView(textViewControl);

editTextControl = new EditText(this);

editTextControl.setWidth(300);

editTextControl.setTag(currentField.fieldName);

fieldLL.addView(editTextControl);

}

else if (currentField.fieldType.equalsIgnoreCase("emailtext"))

{

textViewControl = new TextView(this);

textViewControl.setText(currentField.fieldName);

textViewControl.setWidth(200);

fieldLL.addView(textViewControl);

editTextControl = new EditText(this);

editTextControl.setWidth(300);

editTextControl.setTag(currentField.fieldName.replace(' ', '\_'));

editTextControl.setInputType(InputType.TYPE\_TEXT\_VARIATION\_EMAIL\_

ADDRESS);

fieldLL.addView(editTextControl);

}

else if (currentField.fieldType.equalsIgnoreCase("phonetext"))

{

textViewControl = new TextView(this);

textViewControl.setText(currentField.fieldName);

textViewControl.setWidth(200);

fieldLL.addView(textViewControl);

editTextControl = new EditText(this);

editTextControl.setWidth(300);

editTextControl.setTag(currentField.fieldName);

editTextControl.setInputType(InputType.TYPE\_CLASS\_PHONE);

fieldLL.addView(editTextControl);

}

else if (currentField.fieldType.equalsIgnoreCase("numbertext"))

{

textViewControl = new TextView(this);

textViewControl.setText(currentField.fieldName);

textViewControl.setWidth(200);

fieldLL.addView(textViewControl);

editTextControl = new EditText(this);

editTextControl.setWidth(300);

editTextControl.setTag(currentField.fieldName);

editTextControl.setInputType(InputType.TYPE\_CLASS\_NUMBER);

fieldLL.addView(editTextControl);

}

else if (currentField.fieldType.equalsIgnoreCase("uritext"))

{

textViewControl = new TextView(this);

textViewControl.setText(currentField.fieldName);

textViewControl.setWidth(200);

fieldLL.addView(textViewControl);

editTextControl = new EditText(this);

editTextControl.setWidth(300);

editTextControl.setTag(currentField.fieldName);

editTextControl.setInputType(InputType.TYPE\_TEXT\_VARIATION\_URI);

fieldLL.addView(editTextControl);

}

else if (currentField.fieldType.equalsIgnoreCase("multilinetext"))

{

textViewControl = new TextView(this);

textViewControl.setText(currentField.fieldName);

textViewControl.setWidth(200);

fieldLL.addView(textViewControl);

editTextControl = new EditText(this);

editTextControl.setWidth(300);

editTextControl.setTag(currentField.fieldName);

editTextControl.setSingleLine(false);

editTextControl.setImeOptions(EditorInfo.IME\_FLAG\_NO\_EXTRACT\_UI);

editTextControl.setFocusableInTouchMode(true);

editTextControl.setInputType(EditorInfo.TYPE\_CLASS\_TEXT | EditorInfo.TYPE\_TEXT\_FLAG\_MULTI\_LINE | EditorInfo.TYPE\_TEXT\_FLAG\_IME\_MULTI\_LINE);

editTextControl.setMaxLines(Integer.MAX\_VALUE);

editTextControl.setHorizontallyScrolling(false);

editTextControl.setTransformationMethod(null);

fieldLL.addView(editTextControl);

}

else if (currentField.fieldType.equalsIgnoreCase("passwordtext"))

{

textViewControl = new TextView(this);

textViewControl.setText(currentField.fieldName);

textViewControl.setWidth(200);

fieldLL.addView(textViewControl);

editTextControl = new EditText(this);

editTextControl.setWidth(300);

editTextControl.setTag(currentField.fieldName);

editTextControl.setInputType(InputType.TYPE\_MASK\_VARIATION);

fieldLL.addView(editTextControl);

}

else if (currentField.fieldType.equalsIgnoreCase("datepicker"))

{

textViewControl = new TextView(this);

textViewControl.setText(currentField.fieldName);

textViewControl.setWidth(200);

fieldLL.addView(textViewControl);

btnDatePicker = new Button(this);

btnDatePicker.setTag(currentField.fieldName.replace(' ', '\_'));

fieldLL.addView(btnDatePicker);

final Calendar c = Calendar.getInstance();

mYear = c.get(Calendar.YEAR);

mMonth = c.get(Calendar.MONTH) + 1;

mDay = c.get(Calendar.DAY\_OF\_MONTH);

btnDatePicker.setText(mDay+"/"+mMonth+"/"+mYear);

// Set ClickListener on btnDatePicker

btnDatePicker.setOnClickListener(new View.OnClickListener() {

public void onClick(View v)

{

currentControl = v.getTag().toString();

// Show the DatePickerDialog

showDialog(0);

}

});

}

else if (currentField.fieldType.equalsIgnoreCase("radiobutton"))

{

textViewControl = new TextView(this);

textViewControl.setText(currentField.fieldName);

textViewControl.setLeft(10);

textViewControl.setWidth(200);

fieldLL.addView(textViewControl);

radioGroup = new RadioGroup(this);

for(String fieldValue : currentField.fieldValues)

{

radioButtonControl = new RadioButton(this);

radioButtonControl.setText(fieldValue);

radioButtonControl.setTag(fieldValue.replace(' ','\_'));

radioGroup.addView(radioButtonControl);

}

fieldLL.addView(radioGroup);

}

else if (currentField.fieldType.equalsIgnoreCase("spinner"))

{

textViewControl = new TextView(this);

textViewControl.setText(currentField.fieldName);

textViewControl.setLeft(10);

textViewControl.setWidth(200);

fieldLL.addView(textViewControl);

spinnerControl = new Spinner(this);

spinnerControl.setTag(currentField.fieldName);

ArrayAdapter<String> aa = new ArrayAdapter<String>( this, android.R.layout.simple\_spinner\_item,

currentField.fieldValues);

spinnerControl.setAdapter(aa);

fieldLL.addView(spinnerControl);

}

else if (currentField.fieldType.equalsIgnoreCase("checkbox"))

{

textViewControl = new TextView(this);

textViewControl.setText(currentField.fieldName);

textViewControl.setLeft(10);

textViewControl.setWidth(200);

fieldLL.addView(textViewControl);

checkBoxLayout = new LinearLayout(this);

checkBoxLayout.setOrientation(LinearLayout.VERTICAL);

checkBoxLayout.setTag("checkbox");

for(String fieldValue : currentField.fieldValues)

{

checkBox = new CheckBox(this);

checkBox.setTag(fieldValue);

checkBox.setText(fieldValue);

checkBoxLayout.addView(checkBox);

}

fieldLL.addView(checkBoxLayout);

}

parentLinearLayout.addView(fieldLL);

}

}

btnSave = new Button(this);

btnSave.setText("Save");

btnSave.setWidth(100);

btnSave.setHeight(30);

btnSave.setPadding(10, 10, 10, 10);

btnView = new Button(this);

btnView.setText("View");

btnView.setWidth(100);

btnView.setHeight(30);

btnView.setPadding(10, 10, 10, 10);

btnReset = new Button(this);

btnReset.setText("Reset");

btnReset.setWidth(100);

btnReset.setHeight(30);

btnReset.setPadding(10, 10, 10, 10);

btnExit = new Button(this);

btnExit.setText("Exit");

btnExit.setWidth(100);

btnExit.setHeight(50);

btnExit.setPadding(10, 10, 10, 10);

fieldLL = new LinearLayout(this);

fieldLL.setOrientation(LinearLayout.HORIZONTAL);

line = new View(this);

line.setLayoutParams(new ViewGroup.LayoutParams(ViewGroup.LayoutParams.FILL\_PARENT, 1));

line.setBackgroundColor(Color.rgb(51, 51, 51));

parentLinearLayout.addView(line);

fieldLL.addView(btnSave);

fieldLL.addView(btnView);

fieldLL.addView(btnReset);

fieldLL.addView(btnExit);

fieldLL.setGravity(View.TEXT\_ALIGNMENT\_CENTER);

parentLinearLayout.addView(fieldLL);

line = new View(this);

line.setLayoutParams(new ViewGroup.LayoutParams(ViewGroup.LayoutParams.FILL\_PARENT, 1));

line.setBackgroundColor(Color.rgb(51, 51, 51));

parentLinearLayout.addView(line);

this.setContentView(sv);

//Getting database instance

db = new MySQLiteOpenHelper(this,

"dynamic\_db\_"+tableName,null,1,fieldsList).getWritableDatabase();

btnSave.setOnClickListener(new View.OnClickListener()

{

@Override

public void onClick(View v)

{

//Creating a row to be inserted in database with the help of ContentValues.

ContentValues cv = new ContentValues();

//Fetching all data for user has entered before clicking save button

for (int i = 0; i < parentLinearLayout.getChildCount(); i++)

{

View view = parentLinearLayout.getChildAt(i);

if (view.getClass().equals(LinearLayout.class))

{

LinearLayout childLL = (LinearLayout)view;

getConentValues(childLL,cv);

}

else

{

getConentValues(view,cv);

}

}

try

{

//Check whether table exist or not

if (!isTableExists(db,tableName))

{

createTable(db,tableName);

}

//Inserting newly created row in table in database

//db.insert(strApplicationName.replace(' ','\_'),null, cv);

StringBuilder sbInsertSql = new StringBuilder();

sbInsertSql.append("Insert into "+tableName );

StringBuilder sbKeys = new StringBuilder();

sbKeys.append(" (");

for(String keyItem : cv.keySet())

{

sbKeys.append(" "+ keyItem + ",");

}

sbInsertSql.append( sbKeys.substring(0, sbKeys.length() -1) + " ) values ( ");

for(String keyItem : cv.keySet())

{

sbInsertSql.append("'" + cv.get(keyItem) + "',");

}

String finalQuery = sbInsertSql.substring(0,sbInsertSql.length()-1) + " );";

db.execSQL(finalQuery);

}

catch (Exception ex)

{

ex.printStackTrace();

}

populateDataList();

}

});

btnView.setOnClickListener(new View.OnClickListener()

{

@Override

public void onClick(View v)

{

populateDataList();

}

});

btnReset.setOnClickListener(new View.OnClickListener()

{

@Override

public void onClick(View v)

{

deleteDatabse(db,tableName);

}

});

btnExit.setOnClickListener(new View.OnClickListener()

{

@Override

public void onClick(View v)

{

MainActivity.this.finish();

}

});

}

private void getConentValues(LinearLayout linearLayout, ContentValues contentValues)

{

for(int controlIndex=0; controlIndex < linearLayout.getChildCount(); controlIndex++)

{

View view = linearLayout.getChildAt(controlIndex);

getConentValues(view,contentValues);

}

}

private void getConentValues(View view, ContentValues contentValues)

{

Class viewClass = view.getClass();

if (viewClass == EditText.class)

{

for (Fields currentField : fieldsList)

{

if (currentField.fieldType.equals("normaltext") || currentField.fieldType.equals("emailtext")

|| currentField.fieldType.equals("phonetext") || currentField.fieldType.equals("numbertext")

|| currentField.fieldType.equals("uritext") || currentField.fieldType.equals("multilinetext")

|| currentField.fieldType.equals("passwordtext")

)

{

if (view.getTag() != null && view.getTag().equals(currentField.fieldName))

{

contentValues.put("COL\_"+currentField.fieldName, ((EditText)view).getText().toString());

break;

}

}

}

}

else if (viewClass == Button.class)

{

for (Fields currentField : fieldsList)

{

if (currentField.fieldType.equals("datepicker"))

{

if (view.getTag() != null && view.getTag().equals(currentField.fieldName))

{

contentValues.put("COL\_"+currentField.fieldName, ((Button)view).getText().toString());

break;

}

}

}

}

else if (viewClass == RadioGroup.class)

{

int radioGroupChildCount = ((RadioGroup)view).getChildCount();

for( int radioButtonIndex =0; radioButtonIndex < radioGroupChildCount; radioButtonIndex++)

{

RadioButton childRadioButton = (RadioButton) ((RadioGroup)view).getChildAt(radioButtonIndex);

for (Fields currentField : fieldsList)

{

if (currentField.fieldType.equals("radiobutton") && !contentValues.containsKey("COL\_"+currentField.fieldName))

{

for(String radioButtonName : currentField.fieldValues)

{

if ( childRadioButton.getTag() != null )

{

if ( childRadioButton.getTag().equals(radioButtonName) && childRadioButton.isChecked())

{

contentValues.put("COL\_"+currentField.fieldName, radioButtonName);

break;

}

}

}

}

}

}

}

else if (viewClass == Spinner.class)

{

for (Fields currentField : fieldsList)

{

if (currentField.fieldType.equals("spinner"))

{

if ( view.getTag() != null && view.getTag().equals(currentField.fieldName) )

{

int positionIndex = ((Spinner)view).getSelectedItemPosition();

contentValues.put("COL\_"+currentField.fieldName,

currentField.fieldValues.get(positionIndex));

break;

}

}

}

}

else if (viewClass == LinearLayout.class && view.getTag() != null && view.getTag().equals("checkbox"))

{

for (Fields currentField : fieldsList)

{

if ( !contentValues.containsKey("COL\_"+currentField.fieldName))

{

int checkBoxChildCount = checkBoxLayout.getChildCount();

StringBuilder sbCheckBoxValues = new StringBuilder();

for(int checkBoxIndex = 0; checkBoxIndex< checkBoxChildCount; checkBoxIndex++)

{

CheckBox currentCheckBox = (CheckBox)checkBoxLayout.getChildAt(checkBoxIndex);

if ( currentCheckBox.isChecked() )

{

sbCheckBoxValues.append(currentCheckBox.getTag());

sbCheckBoxValues.append(",");

}

}

contentValues.put("COL\_"+currentField.fieldName,sbCheckBoxValues.substring(0,sbCheckBoxValues.length()-1));

break;

}

}

}

}

private void populateDataList()

{

try

{

sv = new ScrollView(this);

parentLinearLayout = new LinearLayout(this);

parentLinearLayout.setOrientation(LinearLayout.VERTICAL);

parentLinearLayout.setGravity(parentLinearLayout.TEXT\_ALIGNMENT\_GRAVITY);

sv.addView(parentLinearLayout);

LinearLayout.LayoutParams params = new LinearLayout.LayoutParams(

ViewGroup.LayoutParams.FILL\_PARENT, ViewGroup.LayoutParams.WRAP\_CONTENT);

params.gravity = Gravity.CENTER;

//Set Application Title

TextView textViewControl = new TextView(this);

textViewControl.setText(strApplicationName);

textViewControl.setLeft(10);

textViewControl.setGravity(View.TEXT\_ALIGNMENT\_CENTER);

textViewControl.setTextSize(20);

textViewControl.setLayoutParams(params);

parentLinearLayout.addView(textViewControl);

View line = new View(this);

line.setLayoutParams(new ViewGroup.LayoutParams(ViewGroup.LayoutParams.FILL\_PARENT, 1));

line.setBackgroundColor(Color.rgb(51, 51, 51));

parentLinearLayout.addView(line);

//Set Screen heading

textViewControl = new TextView(this);

textViewControl.setText("List of entered records");

textViewControl.setLeft(10);

textViewControl.setTextSize(15);

textViewControl.setLayoutParams(params);

parentLinearLayout.addView(textViewControl);

line = new View(this);

line.setLayoutParams(new ViewGroup.LayoutParams(ViewGroup.LayoutParams.FILL\_PARENT, 1));

line.setBackgroundColor(Color.rgb(51, 51, 51));

parentLinearLayout.addView(line);

//Getting an instance of database

db = new MySQLiteOpenHelper(this,

"dynamic\_db\_"+strApplicationName.replace(' ','\_'),null,1,fieldsList).getWritableDatabase();

//Check whether table exist or not

if (isTableExists(db,tableName))

{

StringBuilder sbFieldNames = new StringBuilder();

for(Fields currentField : fieldsList)

{

sbFieldNames.append("COL\_"+currentField.fieldName+",");

}

String fieldNames = sbFieldNames.substring(0,sbFieldNames.length()-1);

//Fetching data by executing query on our table.

Cursor cursor = db.query(tableName, new String[]{fieldNames}, null, null, null, null, null);

//Checking wether cursor pointing to first record or not ?

if(!cursor.isAfterLast())

{

cursor.moveToFirst();

int fieldsCount = fieldsList.size();

//Navigating through all records and storing each row in a new //candidate object and then adding it to ArrayList

do

{

for(int fieldIndex = 0; fieldIndex < fieldsCount; fieldIndex++)

{

textViewControl = new TextView(this);

textViewControl.setText(fieldsList.get(fieldIndex).fieldName + " : "+ cursor.getString(fieldIndex));

parentLinearLayout.addView(textViewControl);

}

line = new View(this);

line.setLayoutParams(new ViewGroup.LayoutParams(ViewGroup.LayoutParams.FILL\_PARENT, 1));

line.setBackgroundColor(Color.rgb(51, 51, 51));

parentLinearLayout.addView(line);

cursor.moveToNext();

}

while(!cursor.isAfterLast());

}

else

{

textViewControl = new TextView(this);

textViewControl.setText("No records exist in database");

parentLinearLayout.addView(textViewControl);

}

cursor.close();

}

else

{

textViewControl = new TextView(this);

textViewControl.setText("Table is not created yet");

parentLinearLayout.addView(textViewControl);

}

Button btnBack = new Button(this);

btnBack.setText("Back");

parentLinearLayout.addView(btnBack);

btnBack.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

populateMainView();

}

});

this.setContentView(sv);

}

catch (Exception ex)

{

ex.printStackTrace();

}

}

**LoginScreen.java**

public void populateMainView()

{

TextView textViewLogin = null;

TextView textViewPass = null;

parentLinearLayout = new LinearLayout(this);

//parentLinearLayout.setGravity(parentLinearLayout.TEXT\_ALIGNMENT\_GRAVITY);

parentLinearLayout.setOrientation(LinearLayout.VERTICAL);

parentLinearLayout.setDividerPadding(5);

//Read the Fields xml file

readXML();

LinearLayout.LayoutParams params = new LinearLayout.LayoutParams(

ViewGroup.LayoutParams.FILL\_PARENT, ViewGroup.LayoutParams.WRAP\_CONTENT);

params.gravity = Gravity.CENTER;

textViewLogin = new TextView(this);

textViewLogin.setText("Login ID");

textViewLogin.setLeft(10);

textViewLogin.setGravity(View.TEXT\_ALIGNMENT\_CENTER);

textViewLogin.setTextSize(20);

textViewLogin.setLayoutParams(params);

parentLinearLayout.addView(textViewLogin);

//Set Application Title

edittextUser = new EditText(this);

//edittextUser.setLeft(10);

edittextUser.setGravity(View.TEXT\_ALIGNMENT\_CENTER);

//edittextUser.setTextSize(20);

edittextUser.setLayoutParams(params);

parentLinearLayout.addView(edittextUser);

textViewPass = new TextView(this);

textViewPass.setText("Password");

textViewPass.setLeft(10);

textViewPass.setGravity(View.TEXT\_ALIGNMENT\_CENTER);

textViewPass.setTextSize(20);

textViewPass.setLayoutParams(params);

parentLinearLayout.addView(textViewPass);

edittextPass = new EditText(this);

//edittextPass.setLeft(10);

edittextPass.setGravity(View.TEXT\_ALIGNMENT\_CENTER);

edittextPass.setTextSize(20);

edittextPass.setLayoutParams(params);

parentLinearLayout.addView(edittextPass);

View line = new View(this);

line.setLayoutParams(new ViewGroup.LayoutParams(ViewGroup.LayoutParams.FILL\_PARENT, 1));

parentLinearLayout.addView(line);

//Set Table Name

tableName = "Android\_Admin";

btnLogin = new Button(this);

btnLogin.setText("Login");

btnLogin.setWidth(100);

btnLogin.setHeight(30);

btnLogin.setPadding(10, 10, 10, 10);

btnuserLogin = new Button(this);

btnuserLogin.setText("USER Login");

btnuserLogin.setWidth(100);

btnuserLogin.setHeight(30);

btnuserLogin.setPadding(10, 10, 10, 10);

btnExit = new Button(this);

btnExit.setText("Exit");

btnExit.setWidth(100);

btnExit.setHeight(50);

btnExit.setPadding(10, 10, 10, 10);

line = new View(this);

line.setLayoutParams(new ViewGroup.LayoutParams(ViewGroup.LayoutParams.FILL\_PARENT, 1));

//line.setBackgroundColor(Color.rgb(51, 51, 51));

parentLinearLayout.addView(line);

parentLinearLayout.addView(btnLogin);

parentLinearLayout.addView(btnExit);

parentLinearLayout.addView(btnuserLogin);

line = new View(this);

line.setLayoutParams(new ViewGroup.LayoutParams(ViewGroup.LayoutParams.FILL\_PARENT, 1));

//line.setBackgroundColor(Color.rgb(51, 51, 51));

parentLinearLayout.addView(line);

//setTheme(R.style.AppTheme);

this.setContentView(parentLinearLayout);

//Getting database instance

db = new MySQLiteOpenHelper1(this,

tableName,null,1).getWritableDatabase();

insertinfo(db,tableName);

btnLogin.setOnClickListener(new View.OnClickListener()

{

@Override

public void onClick(View v)

{

getdata();

//Toast.makeText(getApplicationContext(), "After get data", Toast.LENGTH\_SHORT).show();

//Toast.makeText(getApplicationContext(), edittextUser.getText().toString(), Toast.LENGTH\_SHORT).show();

//Toast.makeText(getApplicationContext(), edittextPass.getText().toString(), Toast.LENGTH\_SHORT).show();

//Toast.makeText(getApplicationContext(), dbUserName, Toast.LENGTH\_SHORT).show();

//Toast.makeText(getApplicationContext(), dbPassword, Toast.LENGTH\_SHORT).show();

if(edittextUser.getText().toString().equals(dbUserName) && edittextPass.getText().toString().equals(dbPassword))

{

Admin\_Found = true;

//Toast.makeText(getApplicationContext(), "Correct!!", Toast.LENGTH\_SHORT).show();

Intent launchActivity2 = new Intent(LoginScreen.this, MainActivity.class);

launchActivity2.putExtra("Admin\_Found", Admin\_Found);

startActivity(launchActivity2);

}

else

{

Toast.makeText(getApplicationContext(), "Incorrect admin password!!", Toast.LENGTH\_SHORT).show();

}

}

});

btnuserLogin.setOnClickListener(new View.OnClickListener()

{

@Override

public void onClick(View v)

{

Intent launchActivity2 = new Intent(LoginScreen.this, MainActivity.class);

startActivity(launchActivity2);

}

});

btnExit.setOnClickListener(new View.OnClickListener()

{

@Override

public void onClick(View v)

{

LoginScreen.this.finish();

}

});

}

@SuppressLint("NewApi")

private void getdata()

{

try

{

//Getting an instance of database

db = new MySQLiteOpenHelper1(this,

tableName.replace(' ','\_'),null,1).getWritableDatabase();

//Check whether table exist or not

if (isTableExists(db,tableName))

{

//Toast.makeText(getApplicationContext(), "Inside table exists", Toast.LENGTH\_SHORT).show();

String SelectQuery = "Select COL\_Username,COL\_Password from "+tableName;

Cursor cur = db.rawQuery(SelectQuery, null);

if (cur.moveToFirst()) {

do {

dbUserName = cur.getString(0);

dbPassword = cur.getString(1);

// Toast.makeText(getApplicationContext(), dbUserName, Toast.LENGTH\_SHORT).show();

//Toast.makeText(getApplicationContext(), dbPassword, Toast.LENGTH\_SHORT).show();

break;

} while (cur.moveToNext());

}

db.close();

return;

}

else

{

;

}

}

catch (Exception ex)

{

ex.printStackTrace();

}

}

@Override

public boolean onCreateOptionsMenu(Menu menu) {

// Inflate the menu; this adds items to the action bar if it is present.

getMenuInflater().inflate(R.menu.main, menu);

return true;

}

private void readXML()

{

XmlPullParserFactory pullParserFactory;

try

{

pullParserFactory = XmlPullParserFactory.newInstance();

XmlPullParser parser = pullParserFactory.newPullParser();

InputStream in\_s = getApplicationContext().getAssets().open("dynamic\_fields\_admin.xml");

parser.setFeature(XmlPullParser.FEATURE\_PROCESS\_NAMESPACES, false);

parser.setInput(in\_s, null);

parseXML(parser);

}

catch (XmlPullParserException e)

{

e.printStackTrace();

}

catch (IOException e)

{

e.printStackTrace();

}

}

private void parseXML(XmlPullParser parser) throws XmlPullParserException,IOException

{

int eventType = parser.getEventType();

Fields2 currentField = null;

while (eventType != XmlPullParser.END\_DOCUMENT){

String name = null;

switch (eventType){

case XmlPullParser.START\_DOCUMENT:

//fieldsList = new ArrayList();

break;

case XmlPullParser.START\_TAG:

name = parser.getName();

// Toast.makeText(getApplicationContext(), name, Toast.LENGTH\_SHORT).show();

//if (name.equalsIgnoreCase("admincredentials"))

//{

//parser.nextText();

if (name.equalsIgnoreCase("username"))

{

this.UserName = parser.nextText();

//Toast.makeText(getApplicationContext(), UserName, Toast.LENGTH\_SHORT).show();

//this.UserName = "shriram";

}

if (name.equalsIgnoreCase("password"))

{

this.Password = parser.nextText();

//Toast.makeText(getApplicationContext(), Password, Toast.LENGTH\_SHORT).show();

//this.Password = "Polaris08";

}

//}

}

eventType = parser.next();

}

}

private boolean isTableExists(SQLiteDatabase db, String tableName)

{

Cursor cursor = db.rawQuery("Select count(\*) from sqlite\_master where type = ? AND name = ?",

new String[] {"table", tableName});

if (!cursor.moveToFirst())

{

return false;

}

int count = cursor.getInt(0);

cursor.close();

return count > 0;

}

public void insertinfo(SQLiteDatabase db, String tableName)

{

deleteinfo(db,tableName);

ContentValues initialValues = new ContentValues();

initialValues.put("COL\_UserName",UserName);

initialValues.put("COL\_Password",Password);

db.insert(tableName, null, initialValues);

//Toast.makeText(getApplicationContext(), "Inserted", Toast.LENGTH\_SHORT).show();

}

public void deleteinfo(SQLiteDatabase db, String tableName)

{

String whereClause = "1=1";

db.delete(tableName, whereClause, null);

}

}

class MySQLiteOpenHelper1 extends SQLiteOpenHelper

{

String strApplicationName = null;

public MySQLiteOpenHelper1(Context context, String applicationName,

CursorFactory factory, int version)

{

super(context, applicationName, null, version);

//fieldsList = parametersList;

strApplicationName = applicationName;

}

@Override

public void onCreate(SQLiteDatabase db)

{

StringBuilder sbSql = new StringBuilder();

sbSql.append("CREATE TABLE ");

sbSql.append(strApplicationName.replace(' ','\_'));

sbSql.append("( COL\_ID INTEGER PRIMARY KEY AUTOINCREMENT,");

sbSql.append("COL\_");

sbSql.append("UserName");

sbSql.append(" TEXT,");

sbSql.append("COL\_");

sbSql.append("Password");

sbSql.append(" TEXT");

db.execSQL(sbSql.substring(0,sbSql.length() - 1) +")");

}

@Override

public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {

// TODO Auto-generated method stub

}

}

**REFERENCES**

1. Ed Burnette, “Hello Android: Introducing Google's Mobile Development Platform”, The Pragmatic Programmers, 3rd edition, 2010
2. Reto Meier, “Professional Android 4 Application Development”
3. [http://developer.android.com](http://developer.android.com/)