Android Application Framework

Submitted by

Kumaravel S

Roll No.: 1201FOSS0023

Reg. No.: 7581210027

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

FEBRUARY 2014

CENTRE FOR DISTANCE EDUCATION

ANNA UNIVERSITY CHENNAI 600 025

BONA FIDE CERTIFICATE

Certified that this Project report titled "**Android Application Framework**" is the bona fide work of **Mr. Kumaravel S** 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.

S.Kumaravel — Dr. Sanjeev Kumar Saini

Student — Guide

Project in-charge

CERTIFICATE OF VIVA-VOCE-EXAMINATION

This is to certify that Mr/~~Thiru/Ms./Tmt~~. S.Kumaravel. (Roll No.1201FOSS0023; Register No. 75812100027) has been subjected to Viva-voce-Examination on ……………………(Date) at …………..(Time) at the Study Centre **The AU-KBC Research Centre, Madras Institute of Technology, Anna University, Chromepet, Chennai 600044.**

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 :

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.xml. Android Application Framework project will read the dynamic_fields.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, ADT plugin for Eclipse.

திட்ட பணிச்சுருக்கம்

திட்டத்த்தின் நோக்கம், வாடிக்கையாளர் தேர்வு செய்யும் தரவுகளை மையமாகக் கொண்டு தகவல்களை ஏற்ற, ஓர் தகவல் சேகரிப்பு செய்யும் ஆண்ட்ராய்ட் சாதனத்தை தாமாக உருவாக்குவதாகும்.

தற்பொழுது கீழ்காணும் வகையான தகவல்களை சேகரிப்பு செய்ய முடியும்:

1. சாதாரண தரவு வகை
2. எண்கள் மட்டும்
3. இரண்டில் ஒன்றை தேர்வு செய்யும் வகை
4. தேதி மட்டும்
5. ஏதேனும் ஒன்றை தேர்வு செய்யும் வகை
6. பல தரவு தேர்வு செய்யும் வகை
7. தொலைபேசி எண்கள் மட்டும்
8. பல வாக்கியங்கள்
9. இரகசிய சொல்
10. மின் அஞ்சல் மட்டும்
11. இணையத் தளம் மட்டும்

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.

Kumaravel S

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

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

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 needs. 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.

1.1 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.

1.2 Sub Tasks

This project contain the following three sub tasks

1.2.1 PHP Web Page

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

- Application Name (mandatory)

- Screen Name
- Field Name (mandatory)
- Field Type (mandatory)
- 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.xml file used for generating the UI dynamically.

1.2.2 Android Application Framework to generate dynamic UI

Android Application framework will parse the dynamic_fields.xml file and generate the UI screen based upon the field name, field type and field values. The following are the sub tasks that needs 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.

1.2.3 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.

1.3 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

1.4 Future Visions

Currently around 11 types of controls have been covered. In future, all the possible controls will be included in the android application framework.

List of records are shown in the page when the user clicks the “View” button. The “Edit” button will enable the user to edit a particular record and hence the Edit option is very important. It will be handled in future versions.

Export the data into csv, pdf will be provided in future versions to enable the user to generate reports.

CHAPTER 2
REQUIREMENT ANALYSIS & DESIGN

2.1 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 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.xml

Generating Dynamic Screen for data entry

Android Application Framework will read the dynamic.xml file and generate the dynamic screen which will is used to enter the values in each field.

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.

2.2 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.

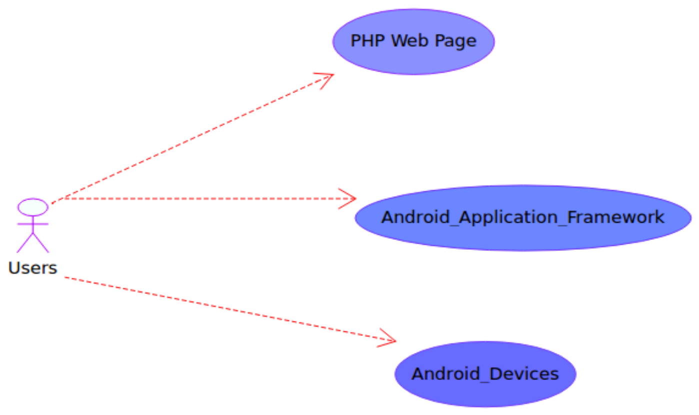


Figure 2.1. Use case diagram

2.3 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.

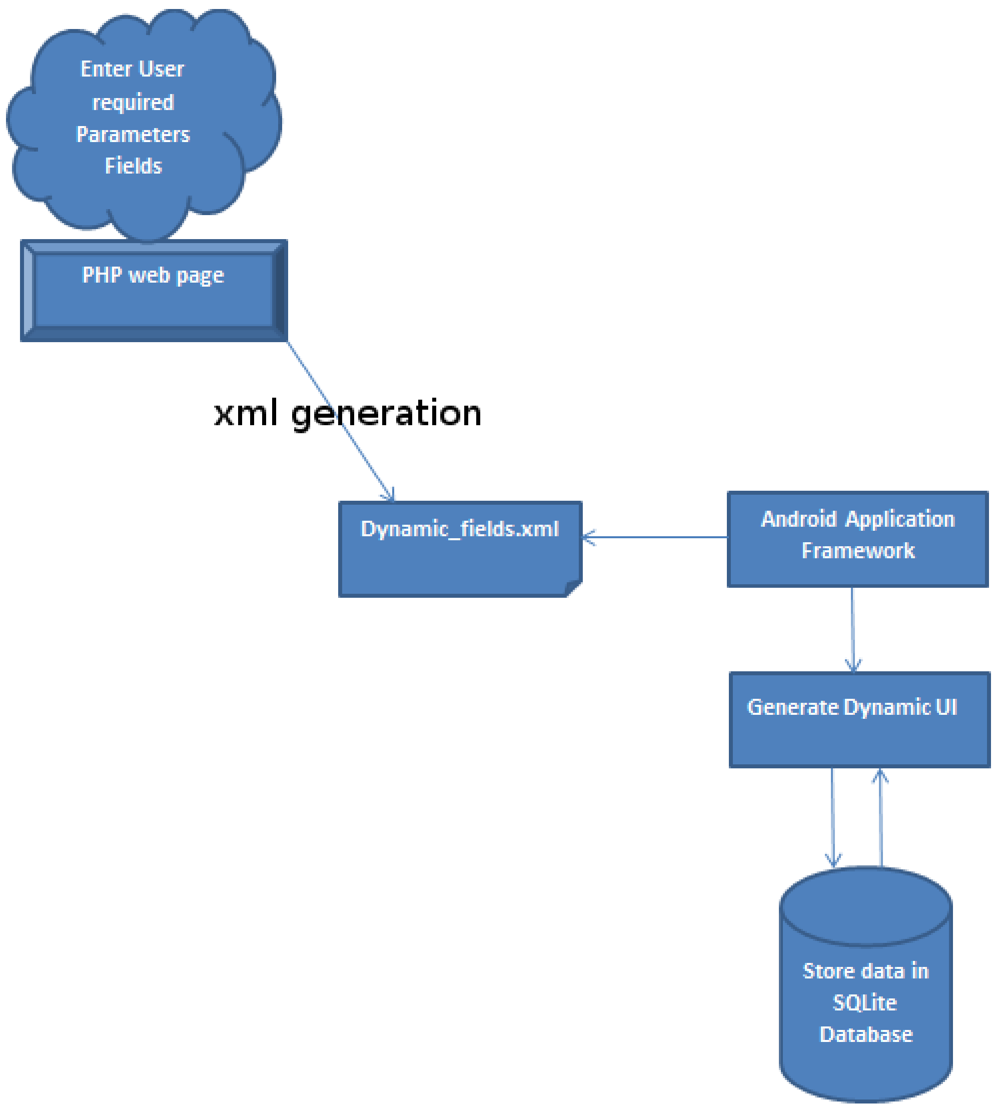

Enter User required Parameters Fields
PHP web page
xml generation
Dynamic_fields.xml
Android Application Framework
Generate Dynamic UI
Store data in SQLite Database

Figure 2.2 Android Application Framework – Architecture Diagram

2.4 Class Diagram

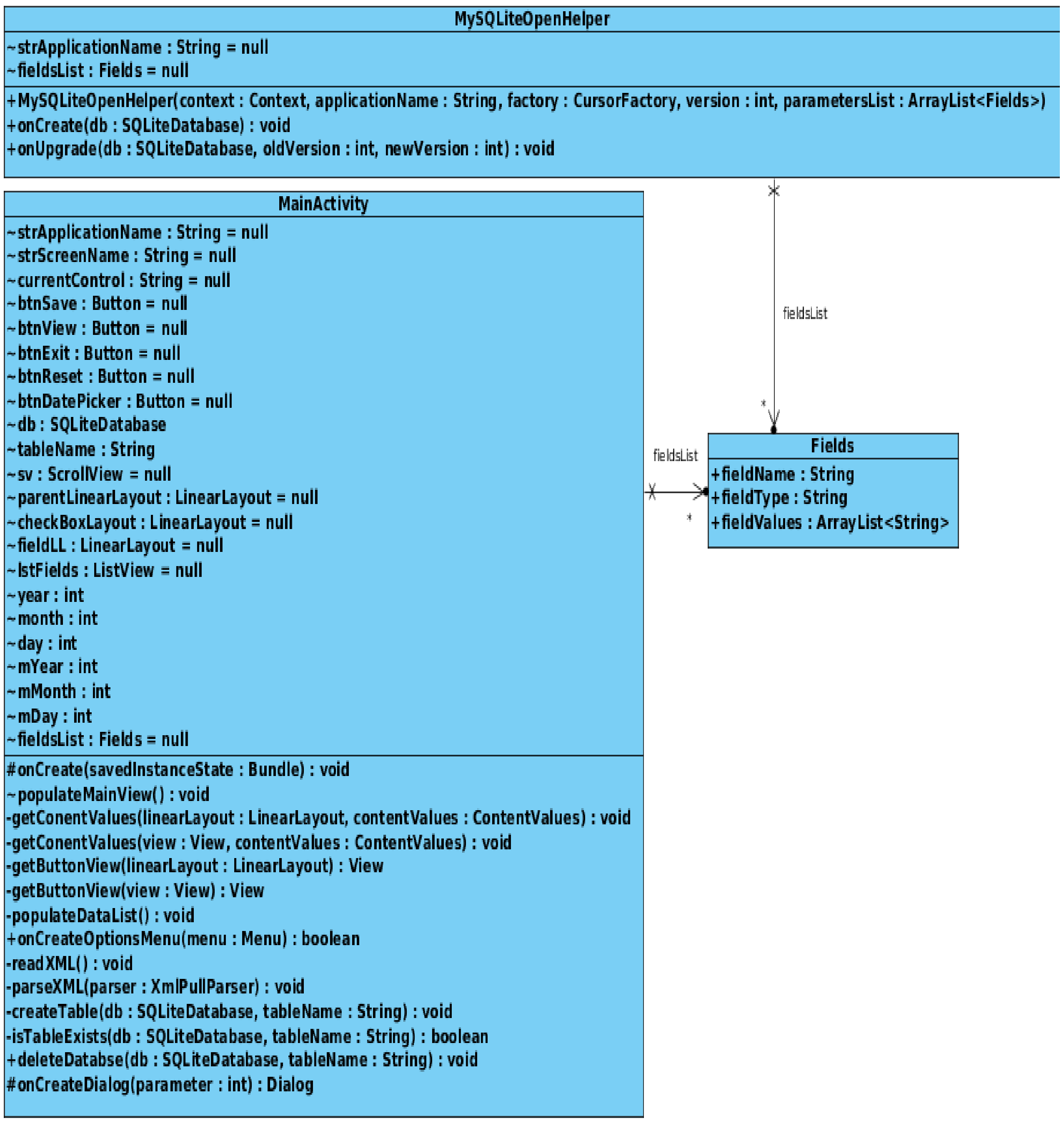

MySQLiteOpenHelper
~strApplicationName : String = null
~fieldsList : Fields = null
+MySQLiteOpenHelper(context : Context, applicationName : String, factory : CursorFactory, version : int, parametersList : ArrayList<Fields>)
+onCreate(db : SQLiteDatabase) : void
+onUpgrade(db : SQLiteDatabase, oldVersion : int, newVersion : int) : void
MainActivity
~strApplicationName : String = null
~strScreenName : String = null
~currentControl : String = null
~btnSave : Button = null
~btnView : Button = null
~btnExit : Button = null
~btnReset : Button = null
~btnDatePicker : Button = null
~db : SQLiteDatabase
~tableName : String
~sv : ScrollView = null
~parentLinearLayout : LinearLayout = null
~checkBoxLayout : LinearLayout = null
~fieldLL : LinearLayout = null
~lstFields : ListView = null
~year : int
~month : int
~day : int
~mYear : int
~mMonth : int
~mDay : int
~fieldsList : Fields = null
#onCreate(savedInstanceState : Bundle) : void
~populateMainView() : void
-getConentValues(linearLayout : LinearLayout, contentValues : ContentValues) : void
-getConentValues(view : View, contentValues : ContentValues) : void
-getButtonView(linearLayout : LinearLayout) : View
-getButtonView(view : View) : View
-populateDataList() : void
+onCreateOptionsMenu(menu : Menu) : boolean
-readXML() : void
-parseXML(parser : XmlPullParser) : void
-createTable(db : SQLiteDatabase, tableName : String) : void
-isTableExists(db : SQLiteDatabase, tableName : String) : boolean
+deleteDatabse(db : SQLiteDatabase, tableName : String) : void
#onCreateDialog(parameter : int) : Dialog
fieldsList
*
fieldsList
*
Fields
+fieldName : String
+fieldType : String
+fieldValues : ArrayList<String>

Figure 2.3. Android Application Framework – Class Diagram

2.4.1 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.

Creates database dynamic application based upon the application name if it was not created yet. It internally calls the isTableExist() method to verify whether the table exist or not.

Generate 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.

2.4.2 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.

2.4.3 MySQLiteOpenHelper Class

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

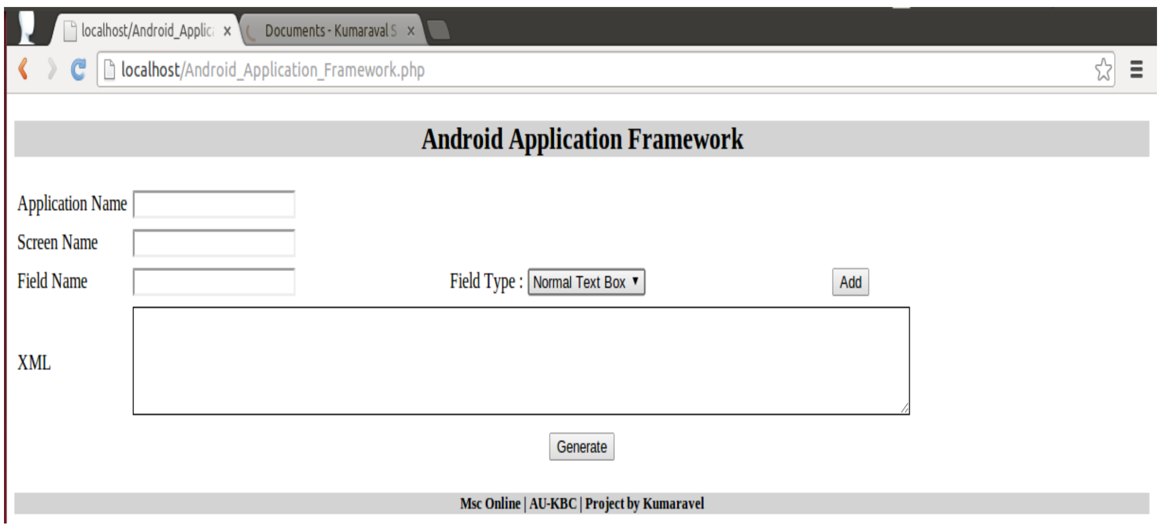
CHAPTER 3
PHP

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 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.

3.1 PHP Web Page



Android Application Framework

Application Name

Screen Name

Field Name

Field Type : Normal Text Box

Add

XML

Generate

Msc Online | AU-KBC | Project by Kumaravel

Figure 3.1. PHP Web Page to capture the field parameters

Application Name

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

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

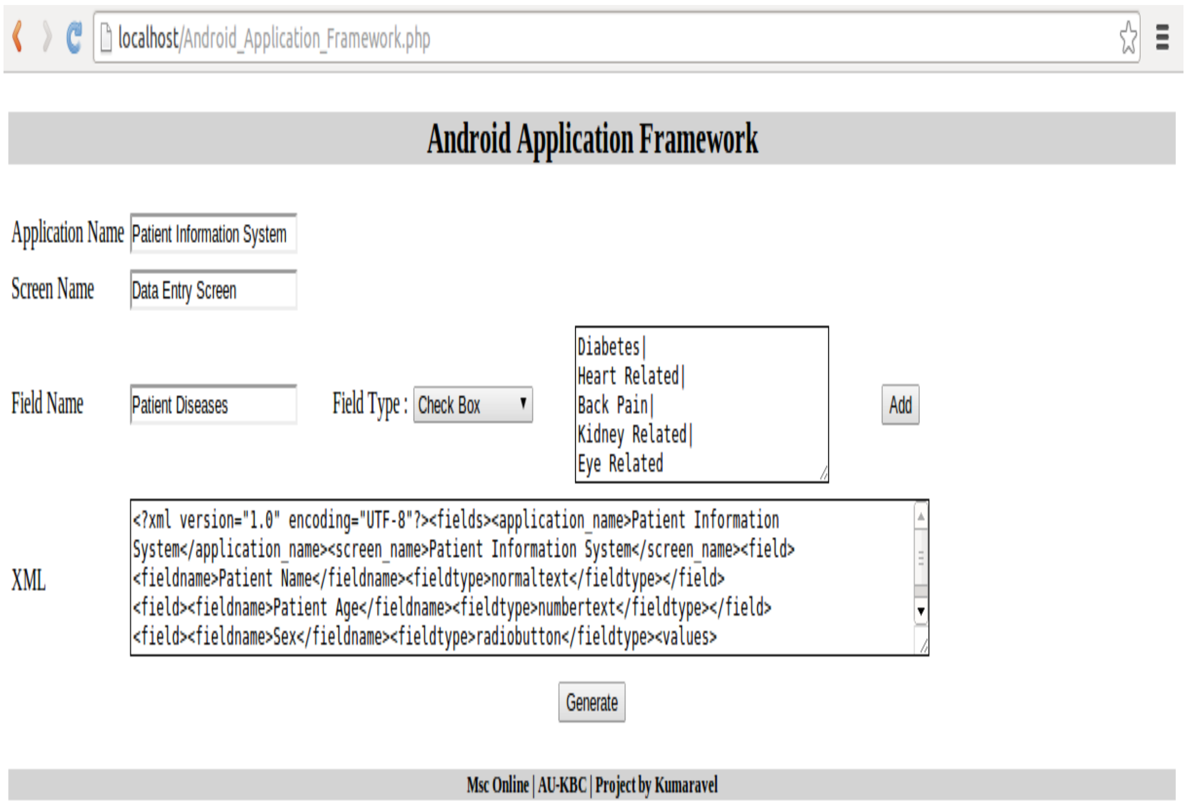
Field type represents the control type like normal text box, number text box, radio button, date picker, drop down, check box, phone text box, multiline text box, password text box,

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

Field Value

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


localhost/Android_Application_Framework.php

Android Application Framework

Application Name: Patient Information System

Screen Name: Data Entry Screen

Field Name: Patient Diseases

Field Type : Check Box

```
Diabetes|
Heart Related|
Back Pain|
Kidney Related|
Eye Related
```

XML

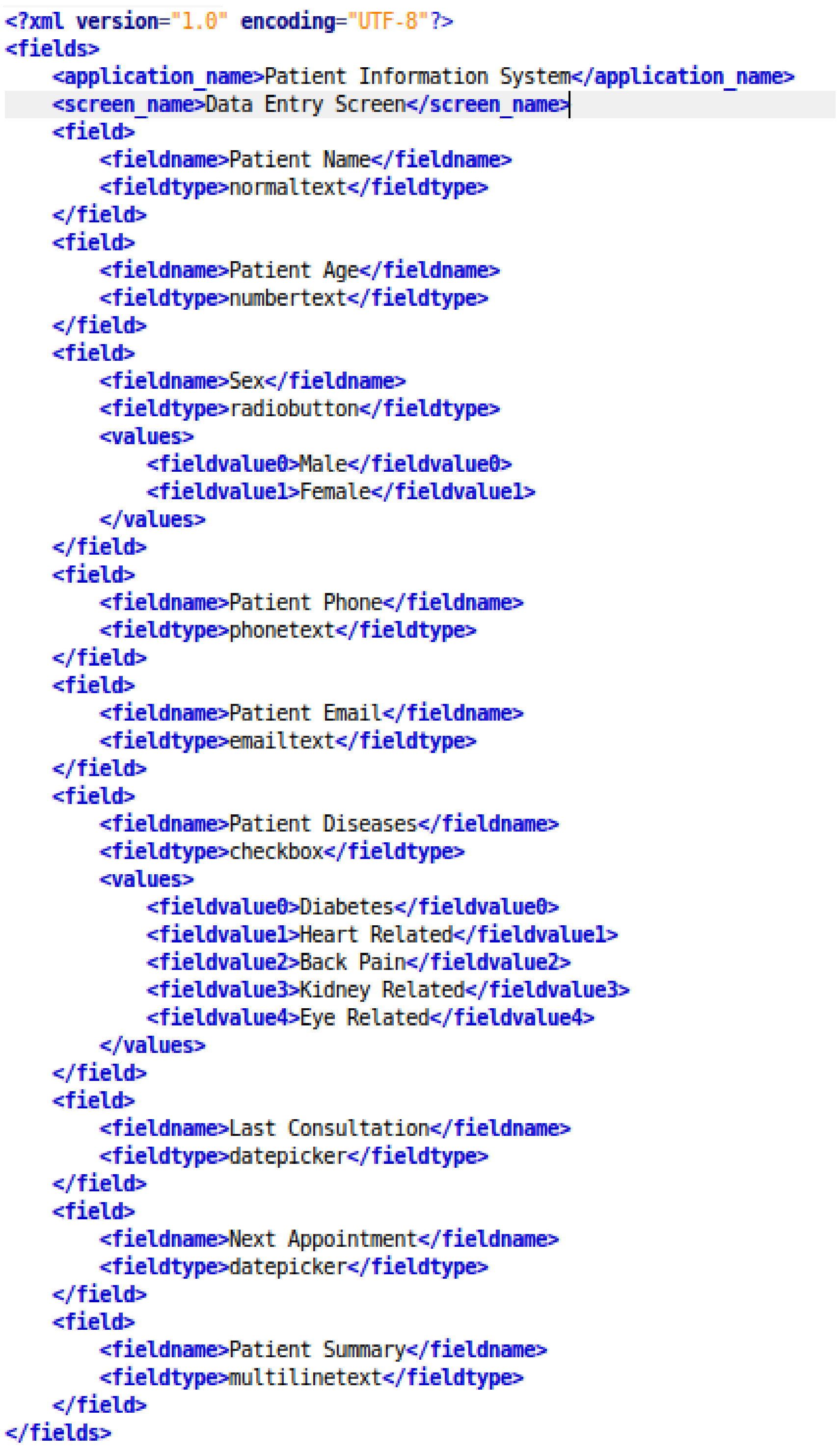
```
<?xml version="1.0" encoding="UTF-8"?><fields><application_name>Patient Information
System</application_name><screen_name>Patient Information System</screen_name><field>
<fieldname>Patient Name</fieldname><fieldtype>normaltext</fieldtype></field>
<field><fieldname>Patient Age</fieldname><fieldtype>numbertext</fieldtype></field>
<field><fieldname>Sex</fieldname><fieldtype>radiobutton</fieldtype><values>
```

Figure 3.2. Check Box - Field Type

3.2 Generate XML

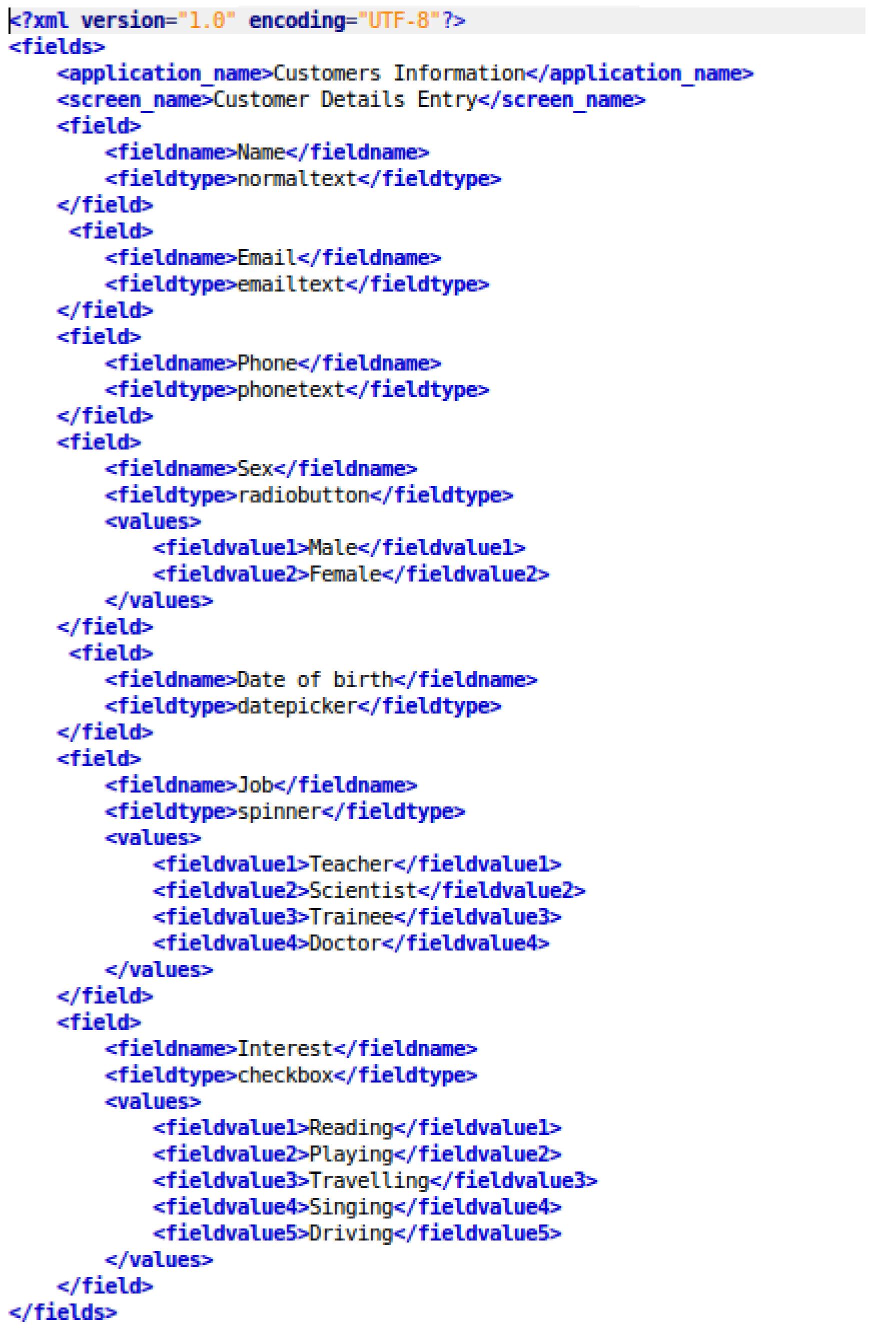
Whenever the user clicks on the "Add" button for the each field details, XML string will be populated in the XML textbox and is shown in Figure 5.

When the user clicks the "Generate" button, it will create the dynamic_fields.xml file as shown in Figure 3.3 and Figure 3.4



```xml
<?xml version="1.0" encoding="UTF-8"?>
<fields>
    <application_name>Patient Information System</application_name>
    <screen_name>Data Entry Screen</screen_name>
    <field>
        <fieldname>Patient Name</fieldname>
        <fieldtype>normaltext</fieldtype>
    </field>
    <field>
        <fieldname>Patient Age</fieldname>
        <fieldtype>numbertext</fieldtype>
    </field>
    <field>
        <fieldname>Sex</fieldname>
        <fieldtype>radiobutton</fieldtype>
        <values>
            <fieldvalue0>Male</fieldvalue0>
            <fieldvalue1>Female</fieldvalue1>
        </values>
    </field>
    <field>
        <fieldname>Patient Phone</fieldname>
        <fieldtype>phonetext</fieldtype>
    </field>
    <field>
        <fieldname>Patient Email</fieldname>
        <fieldtype>emailtext</fieldtype>
    </field>
    <field>
        <fieldname>Patient Diseases</fieldname>
        <fieldtype>checkbox</fieldtype>
        <values>
            <fieldvalue0>Diabetes</fieldvalue0>
            <fieldvalue1>Heart Related</fieldvalue1>
            <fieldvalue2>Back Pain</fieldvalue2>
            <fieldvalue3>Kidney Related</fieldvalue3>
            <fieldvalue4>Eye Related</fieldvalue4>
        </values>
    </field>
    <field>
        <fieldname>Last Consultation</fieldname>
        <fieldtype>datepicker</fieldtype>
    </field>
    <field>
        <fieldname>Next Appointment</fieldname>
        <fieldtype>datepicker</fieldtype>
    </field>
    <field>
        <fieldname>Patient Summary</fieldname>
        <fieldtype>multilinetext</fieldtype>
    </field>
</fields>
```

Figure 3.3. Dynamic_Fields.xml for patient information system



```xml
<?xml version="1.0" encoding="UTF-8"?>
<fields>
    <application_name>Customers Information</application_name>
    <screen_name>Customer Details Entry</screen_name>
    <field>
        <fieldname>Name</fieldname>
        <fieldtype>normaltext</fieldtype>
    </field>
     <field>
        <fieldname>Email</fieldname>
        <fieldtype>emailtext</fieldtype>
    </field>
    <field>
        <fieldname>Phone</fieldname>
        <fieldtype>phonetext</fieldtype>
    </field>
    <field>
        <fieldname>Sex</fieldname>
        <fieldtype>radiobutton</fieldtype>
        <values>
            <fieldvalue1>Male</fieldvalue1>
            <fieldvalue2>Female</fieldvalue2>
        </values>
    </field>
     <field>
        <fieldname>Date of birth</fieldname>
        <fieldtype>datepicker</fieldtype>
    </field>
    <field>
        <fieldname>Job</fieldname>
        <fieldtype>spinner</fieldtype>
        <values>
            <fieldvalue1>Teacher</fieldvalue1>
            <fieldvalue2>Scientist</fieldvalue2>
            <fieldvalue3>Trainee</fieldvalue3>
            <fieldvalue4>Doctor</fieldvalue4>
        </values>
    </field>
    <field>
        <fieldname>Interest</fieldname>
        <fieldtype>checkbox</fieldtype>
        <values>
            <fieldvalue1>Reading</fieldvalue1>
            <fieldvalue2>Playing</fieldvalue2>
            <fieldvalue3>Travelling</fieldvalue3>
            <fieldvalue4>Singing</fieldvalue4>
            <fieldvalue5>Driving</fieldvalue5>
        </values>
    </field>
</fields>
```

Figure 3.4. Dynamic_Fields.xml for Customer information

3.3 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

4.1 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.

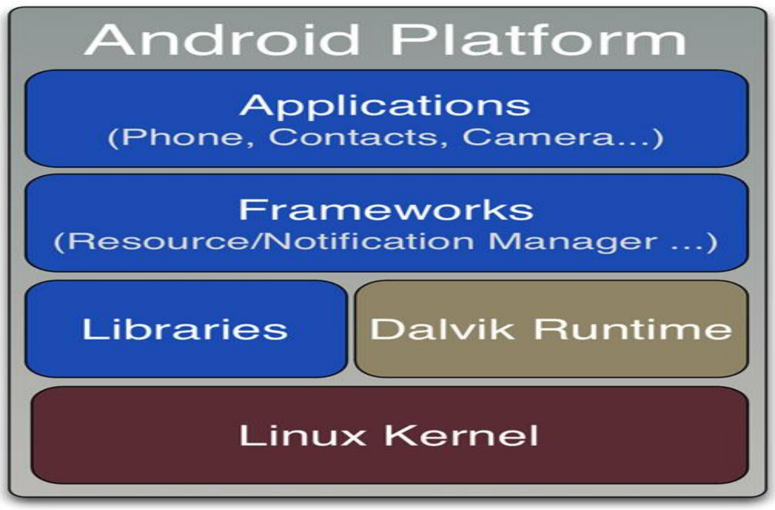


Figure 4.1. Android Architecture

4.2 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.

4.2.1 Read & Parse XML

After generation of the dynamic_fields.xml from php web page, it will be placed under /AndroidApplicationFramework/assets folder.

Android application framework readXML() method read the dynamic_fields.xml (/AndroidApplicationFramework/assets).

The parseXML() method will parse the dynamic_fields.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.

4.2.2 Generate Dynamic Data Entry Screen

- 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
        - 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.

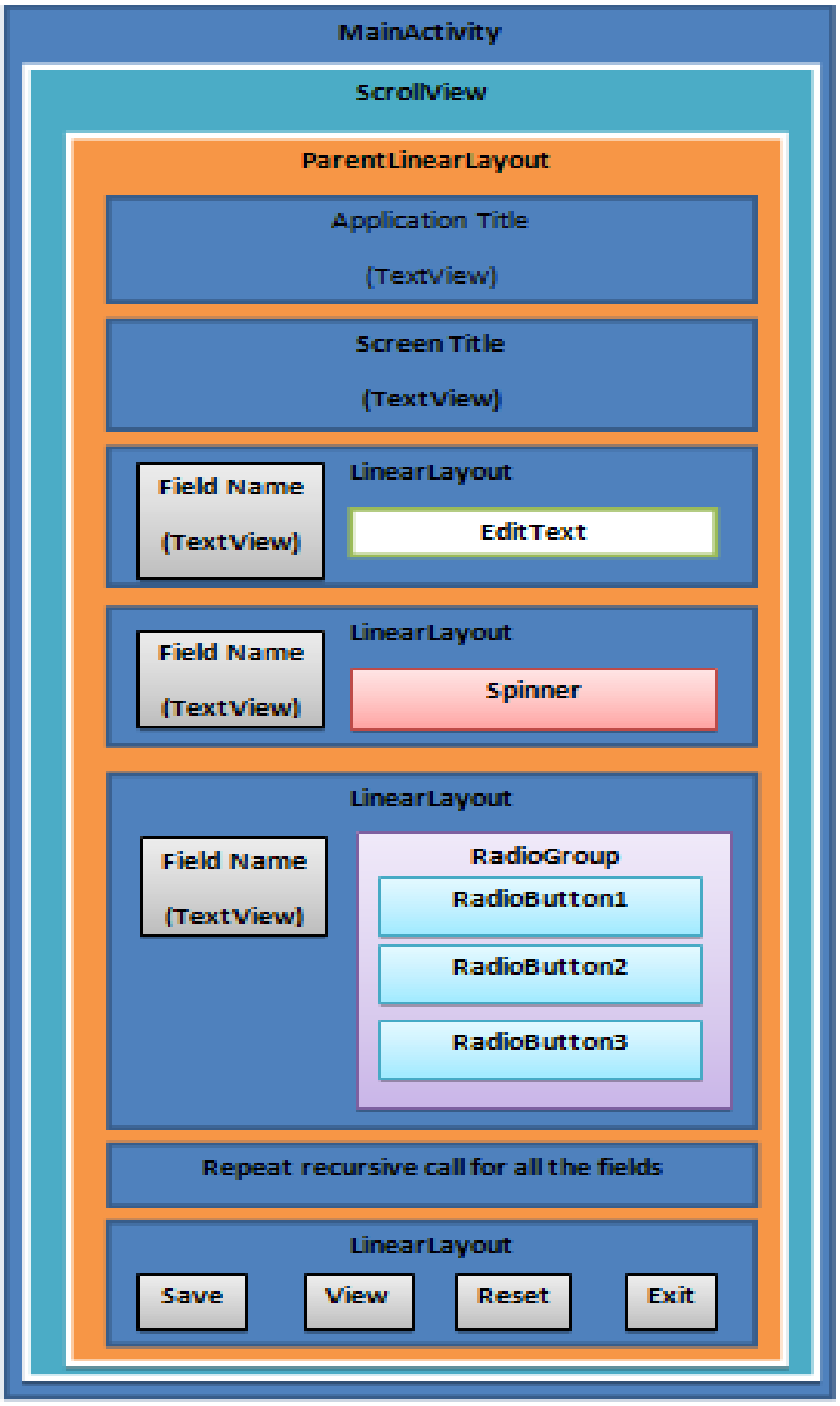
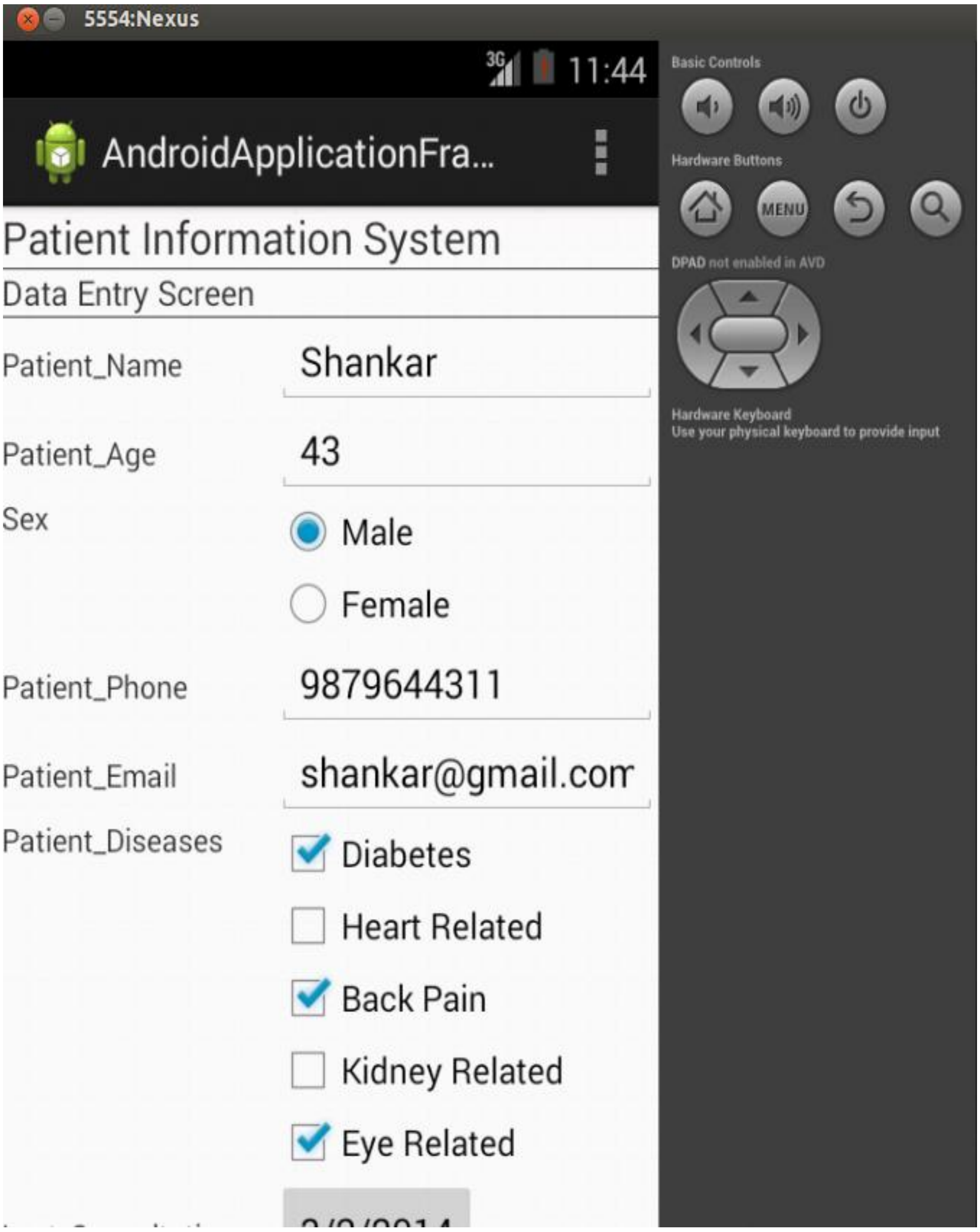


Figure 4.2. Dynamic controls creation - overview

The dynamic screen will be generated after parsing the xml.


5554:Nexus

11:44

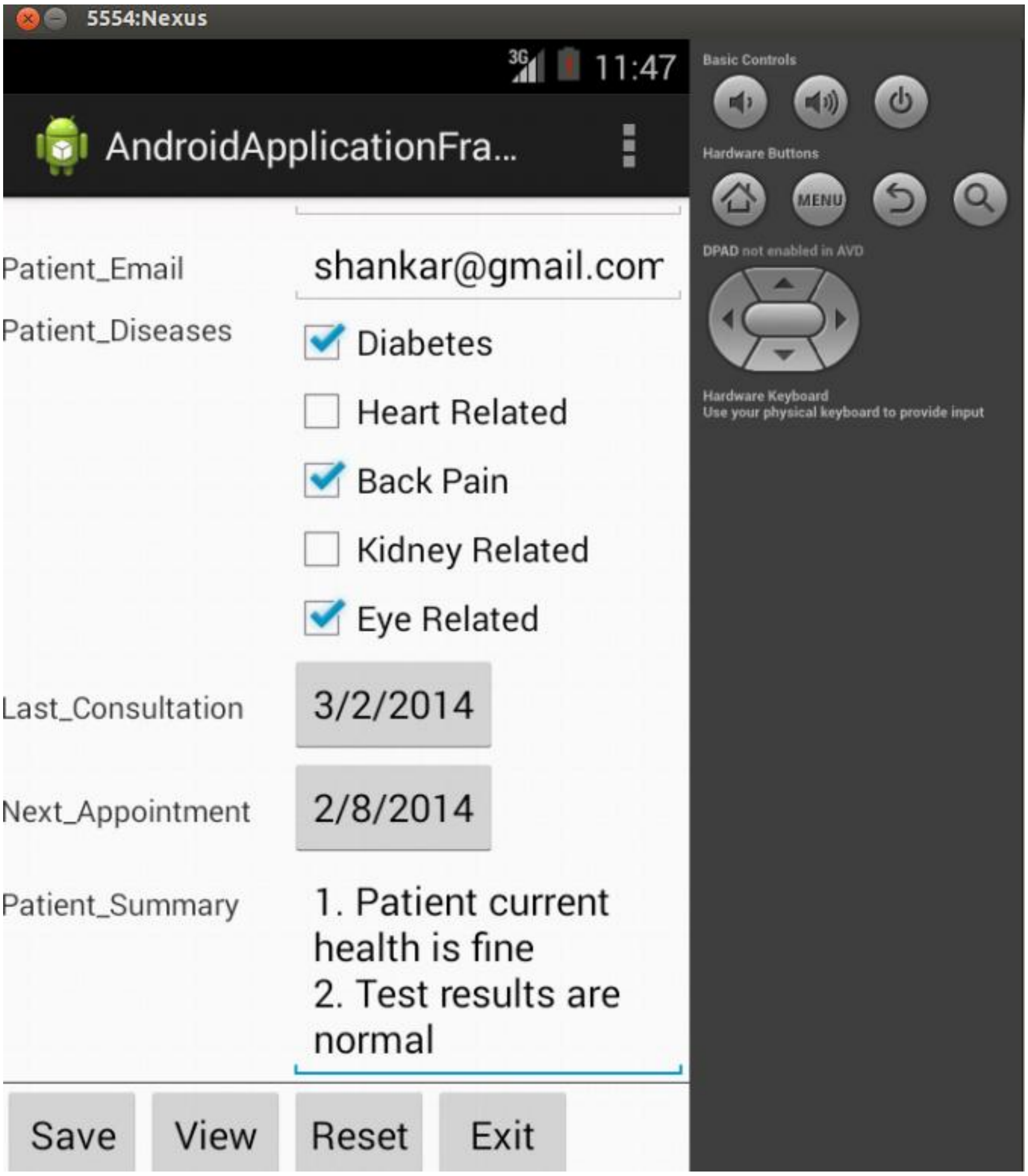
AndroidApplicationFra...

Patient Information System

Data Entry Screen

Patient_Name	Shankar
Patient_Age	43
Sex	[✓ Male] Female
Patient_Phone	9879644311
Patient_Email	shankar@gmail.com
Patient_Diseases	[✓ Diabetes] Heart Related [✓ Back Pain] Kidney Related [✓ Eye Related]

Figure 4.3. Dynamic data entry screen - part1


5554:Nexus

11:47

AndroidApplicationFra...

Patient_Email shankar@gmail.com

Patient_Diseases
- [x] Diabetes
- [ ] Heart Related
- [x] Back Pain
- [ ] Kidney Related
- [x] Eye Related

Last_Consultation 3/2/2014

Next_Appointment 2/8/2014

Patient_Summary 1. Patient current health is fine
2. Test results are normal

Figure 4.4. Dynamic data entry screen - part2

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.

For example, Patient_Diseases is a Checx box control type. It has the following field values:

Diabetes

Heart Related

Back Pain

Kidney Related

Eye Related

At the time of save, if Diabetes, Back Pain and Eye Related check boxes have checked, then Patient_Diseses field will store Diabetes, Back Pain, Eye Related. Refer Figure 4.4 & 4.5.

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.

4.2.3 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 Patient Information System, user specified application name as "Patient Information System" then the database will be created as below:

Database	**:**	dynamic_db_Patient_Information_System
Table	**:**	Patient_Information_System
Columns	**:**	COL_ID (Primary Key, Auto generated unique number)
	:	COL_Patient_Name (Text)
	:	COL_Patient_Age (Text)
	:	COL_Sex (Text)
	:	COL_Patient_Phone (Text)
	:	COL_Patient_Email (Text)
	:	COL_Patient_Diseases (Text)

: COL_Last_Consultation (Text)

: COL_Next_Consultation (Text)

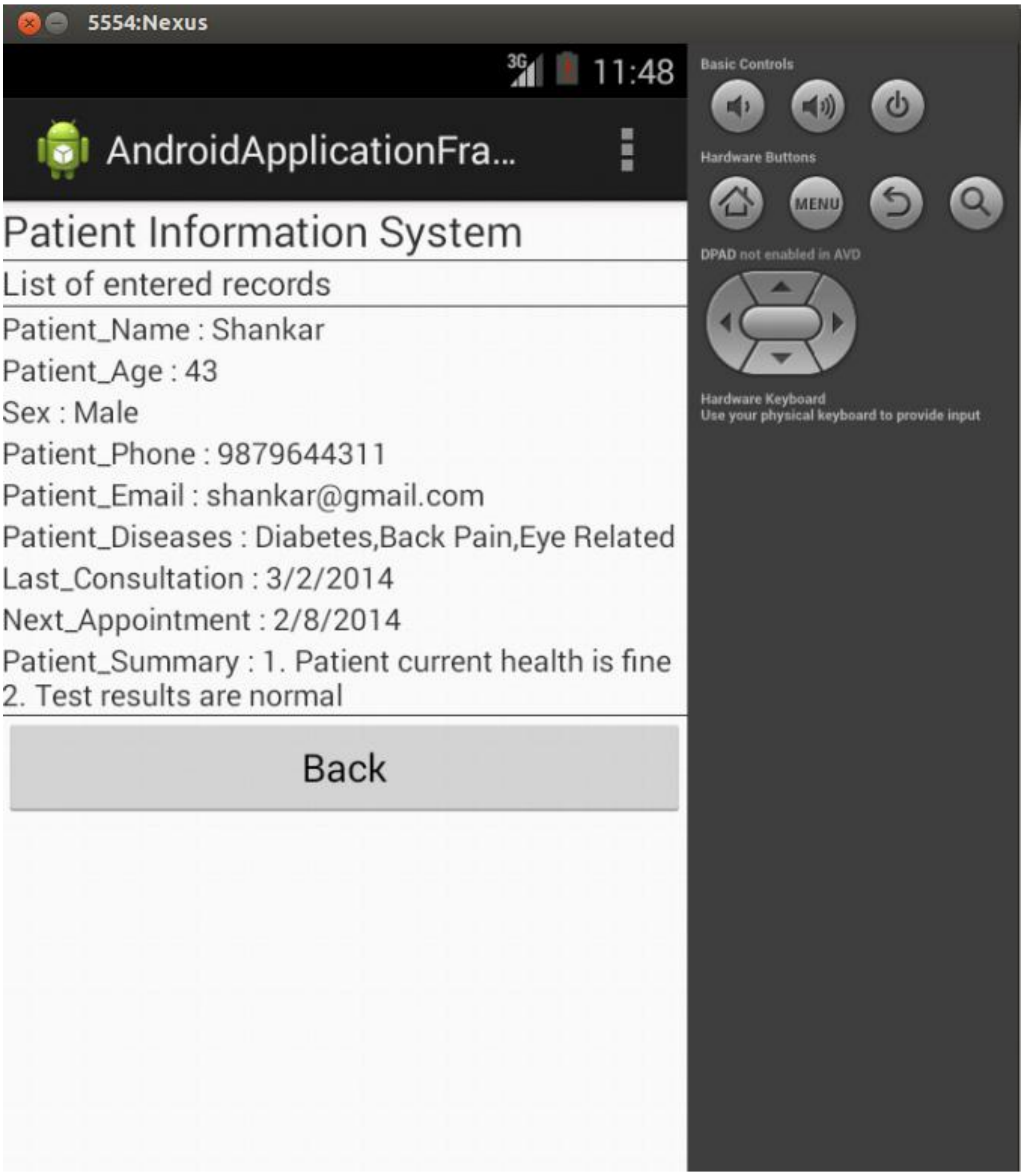
: COL_Patient_Summary (Text)

4.2.4 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



Patient Information System

List of entered records

Patient_Name : Shankar
Patient_Age : 43
Sex : Male
Patient_Phone : 9879644311
Patient_Email : shankar@gmail.com
Patient_Diseases : Diabetes,Back Pain,Eye Related
Last_Consultation : 3/2/2014
Next_Appointment : 2/8/2014
Patient_Summary : 1. Patient current health is fine
2. Test results are normal

Figure 4.5. Dynamic List Screen

When press on Back Button, then home page will be shown for entering further data.

4.2.5 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.

CHAPTER 5
GENERATE AND DEPLOYING APK FILE

5.1 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:

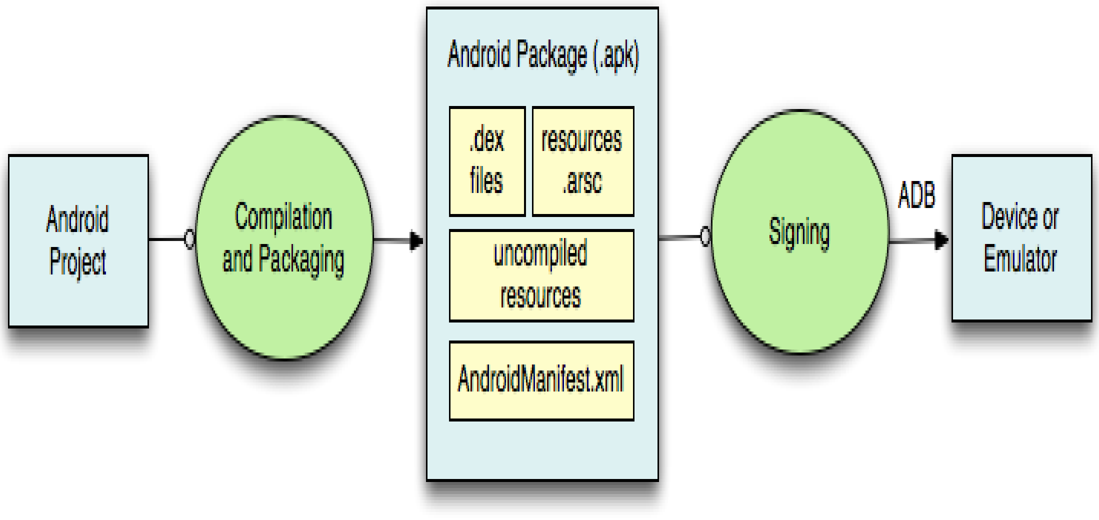


Figure 5.1. Apk file generation

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.

5.2 Deploy apk file in Android device

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.

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

- Customer Information

Name	: Normal Text Edit Control type
Email	: Email Text Edit Control type
Phone	: Phone Text Edit Control type
Sex	: Radio button control type
Date of birth	: Date picker control type

o Job	: Spinner control type
o Interest	: Check box control type

6.1 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 is an issue when two date picker fields are used. If we select date for date picker1 field, then it updates the date with other date picker.

Identified the issue through debugging and noticed dynamic event mapping done on date picker control. Now dynamically event mapping also wired with each date picker control. So issue got resolved.

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 needs 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.

- List of records are showing in the page when the “View” button is clicked. “Edit” button will be required to help to edit the particular record. So, Edit option is very important and needs to be handled in future versions.

- Export the data into csv, pdf formats will be provided in future option to enable the user to generate report.

APPENDIX I

SOURCE CODE

Android_Application_Framework.php

```
<!DOCTYPE html>
<head>
        <script type="text/javascript">

                /*
                * Function used to create XML for all the controls and populate in
txtXML.
                */
                function addText()
                {
                        //Variables decalaration
                        var applicationTitle, screenTitle,
selectedControls,controlName,controlType,controlValue, xmlText;

                        //Assign values
                        applicationTitle =
document.getElementById("txtApplicationName").value;
                        screenTitle =
document.getElementById("txtScreenName").value;
                        selectedControls =
document.getElementById("txtXml").value;
                        controlName =
document.getElementById("controlName").value;
                        controlType =
document.getElementById("controlType").value;
                        controlValue =
document.getElementById("controlValue").value;

                        //Framing xml
                        xmlText =  '<?xml version="1.0" encoding="UTF-
8"?><fields>';

                        //Application Name
                        xmlText = xmlText +
"<application_name>"+applicationTitle+"</application_name>";

                        //Screen Name
                        xmlText = xmlText +
"<screen_name>"+applicationTitle+"</screen_name>";

                        if ( selectedControls.length > 0 )
                        {
                                xmlText = selectedControls.replace("</fields>","") +
"\n" + getXml(controlName,controlType,controlValue);
                        }
                        else
                        {
                                xmlText = xmlText +
getXml(controlName,controlType,controlValue);
                        }
                        xmlText = xmlText + "</fields>";
                        document.getElementById("txtXml").innerHTML = xmlText;

                        //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 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;
		}

	</script>
</head>
<body>
```

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

                        </table>
                        <div align="center"><input type="submit" name="submit"
value="Generate" align="center"></div>
                        <p style="background-color:lightgrey" align="center">
                        <?php
                                if(isset($_POST['submit']))
                                {
                                        $txtXml  = $_POST['txtXml'];
                                        file_put_contents("dynamic_fields.xml",
$txtXml);
                                        echo "dynamic_fields.xml has been generated
successfully";
                                }
                        ?>
                        </p>
                </form>
                <div  style="background-color:lightgrey"><h5 align="center">Msc
```

```
Online | AU-KBC | Project by Kumaravel</h2></div>
        </body>
</html>
```

dynamic_fields.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<fields>
        <application_name>Patient Information System</application_name>
        <screen_name>Data Entry Screen</screen_name>
        <field>
                <fieldname>Patient Name</fieldname>
                <fieldtype>normaltext</fieldtype>
        </field>
        <field>
                <fieldname>Patient Age</fieldname>

                <fieldtype>numbertext</fieldtype>
        </field>
        <field>
                <fieldname>Sex</fieldname>
                <fieldtype>radiobutton</fieldtype>
                <values>
                        <fieldvalue0>Male</fieldvalue0>
                        <fieldvalue1>Female</fieldvalue1>
                </values>
        </field>
        <field>
                <fieldname>Patient Phone</fieldname>
                <fieldtype>phonetext</fieldtype>
        </field>
        <field>
                <fieldname>Patient Email</fieldname>
                <fieldtype>emailtext</fieldtype>
        </field>
        <field>
                <fieldname>Patient Diseases</fieldname>
                <fieldtype>checkbox</fieldtype>
                <values>
                        <fieldvalue0>Diabetes</fieldvalue0>
                        <fieldvalue1>Heart Related</fieldvalue1>
                        <fieldvalue2>Back Pain</fieldvalue2>
                        <fieldvalue3>Kidney Related</fieldvalue3>
                        <fieldvalue4>Eye Related</fieldvalue4>
                </values>
        </field>
        <field>
                <fieldname>Last Consultation</fieldname>
                <fieldtype>datepicker</fieldtype>
        </field>
        <field>
                <fieldname>Next Appointment</fieldname>
                <fieldtype>datepicker</fieldtype>
        </field>
        <field>
                <fieldname>Patient Summary</fieldname>
                <fieldtype>multilinetext</fieldtype>
        </field>
</fields>
```

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_GRAV
ITY);
        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_A
DDRESS);
               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_GRAV
ITY);
        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();
       }
    }
```

REFERENCES

1. http://developer.android.com/index.html

2. Ed Burnette, “Hello Android: Introducing Google's Mobile Development Platform”, The Pragmatic Programmers, 3rd edition, 2010