ANDROID APPLICATION ON BANK ACCOUNT TRACKING SYSTEM

101231

RAJAT DIKSHIT

Dr NITIN CHANDERWAL

May 2014

Submitted in partial fulfilment of the Degree of

Bachelor of Technology

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

& INFORMATION TECHNOLOGY

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY,

WAKNAGHAT

CERTIFICATE

This is to certify that the work titled **Android Application on Bank Account Tracking** submitted by **Rajat Dikshit** in partial fulfilment for the award of degree of Bachelor of technology ofJaypee University of Information Technology, Waknaghat has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.

Signature of Supervisor:

Name of Supervisor : Dr. Nitin Chanderwal

Designation : Assistant Professor (Senior Grade)

Date :

ACKNOWLEDGEMENT

I would like to express my gratitude to all those who gave us the possibility to complete this project. I want to thank the Department of CSE & IT in JUIT for giving us the permission to commence this project in the first instance, to do the necessary research work.

I am deeply indebted to my project guide Dr. Nitin Chanderwal, whose help, stimulating suggestions and encouragement helped me in all the time of research on this project. I feel motivated and encouraged every time I get his encouragement. For his coherent guidance throughout the tenure of the project, I feel fortunate to be taught by him, who gave me his unwavering support.

Rajat Dikshit

TABLE OF CONTENT

ABSTRACT

This android app allows user to keep track of their respective bank accounts regardless of what kind of bank they are involved with and also the transactions carried out in the banks by the user.

Account trackin app is about implementing an app for android mobile devices so as to help its user to manage their account balances and keep a watch on their finances. In today's scenario using apps for daily needs and other financial and business purposes has become a healthy trend because of the availability of web services on mobile devices. By considering these technological advancements in mobile technology, an awareness of monetary transactions through mobile in less time can be a useful asset to the user. In this application the user first needs to install the app and save all the relevant bank details and then they can start using it for transactional and memo purposes. The key challenges that arise in this context is include dealing with huge amount of data entry and managing processor cycles at the same time for the efficient working of the application. This document makes contributions toward the solution of these problems.The efficacy of the techniques proposed was demonstrated through extensive experiments, both in theory as well as in practice.

CHAPTER 1

INTRODUCTION

1.1 Problem Statement

I have done a thorough study of bank account requirements as well as the technicalities of building and implementing an android app.

A bank account comes with various entities such as account number, cheque deposit system, cash deposit and withdrawal system and also various parts such as branch and ifsc codes of these branches.All these viable entities have been taken into conideration as the data entry process is the first hands-on experience which the user is supposed to do. So it is a challenge to design a good database system so as to encompass the data.

Before I design the functionalities of the application, i need to create a user interface so that the user gets a hassle free medium to communicate with the device and also taking into consideration , the user's awareness for the android operating system the application needs to be very simplistic and minimalstic in approach.

The project proposes a minimalistic model of android app making with the help of java and other android software development tools running alongside. Along with this, background functionality has not been taken into consideration to save cpu cycles.

1.2 INTRODUCTION TO ANDROID

Android was recognized in the year 2007 along with the founding of OHA (Open Handset Alliance) for the upliftment and betterment of smart experiences with mobile devices and since then has been the most installed and accepted operating system in the world.

Android is an open source operating system from Google which is known to be cross device ie. it is available for a variety of devices such as phones, tablets and even televisions. It is a linux based kernel operating system and can be programmed in various languages such as c,c++ etc. but most of the application programming is done in Java (Java accesses the C library through JNI which is Java Native Interface). Android running devices all support Bluetooth, WIFI, 3g and

some 4g data services. Android's basic interface works on the principle of direct manipulation, using touch operations which loosely correspond to real-world actions.

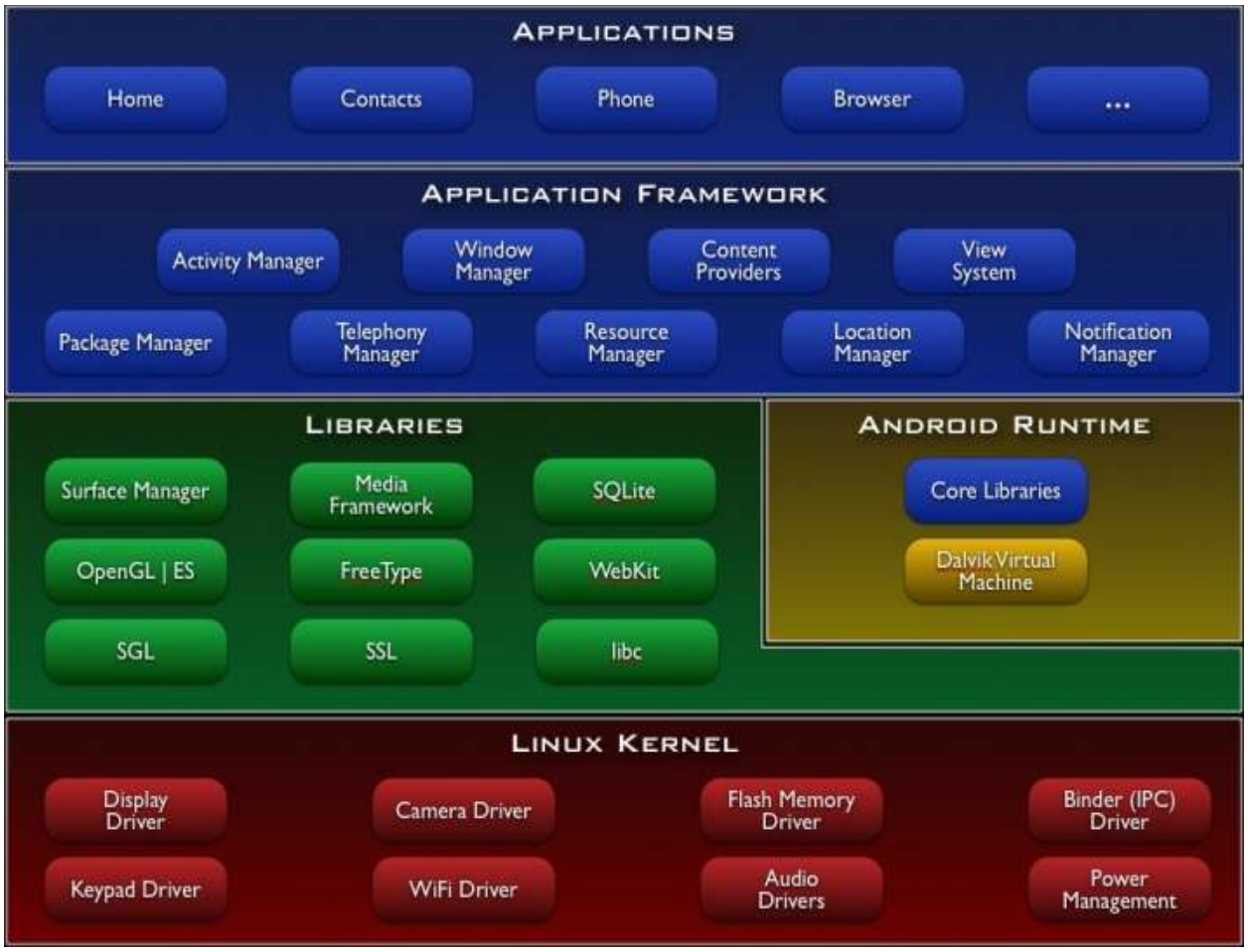
1.3 FEATURES

Android is a very efficient operating system and after the bootup of the device, the interface goes straight to the homescreen. The android homescreen can consist of different app tiles and widgets. Users can enter these apps directly from the homescreen and use widgets for different purposes such as weather, time, news etc. This homescreen can be made up of several pages so that the user is left with enough space to go back and forth.

CHAPTER 2

APPLICATION LAYER AND LIBRARIES

2.1 LAYER DISTRIBUTION



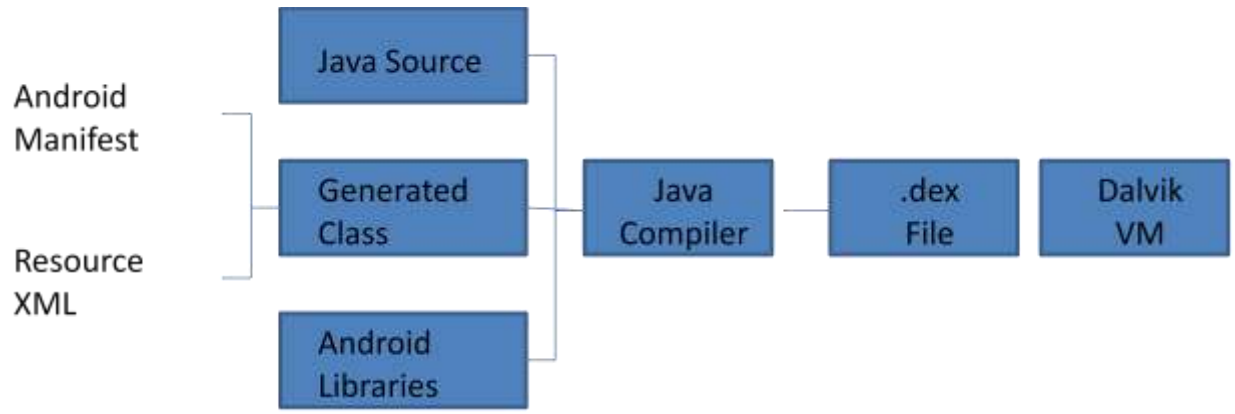
2.2 Application framework

The application framework of this operating system allows the users to reuse and even replace some of the components used in the building or use of the apps. The components involved in the framework are responsible for the session building and working of an application. These components are :-

- Activity manager - manages the lifecycle of applications.
- Contents provider - store and retrieve data and make it accessible to all the application.

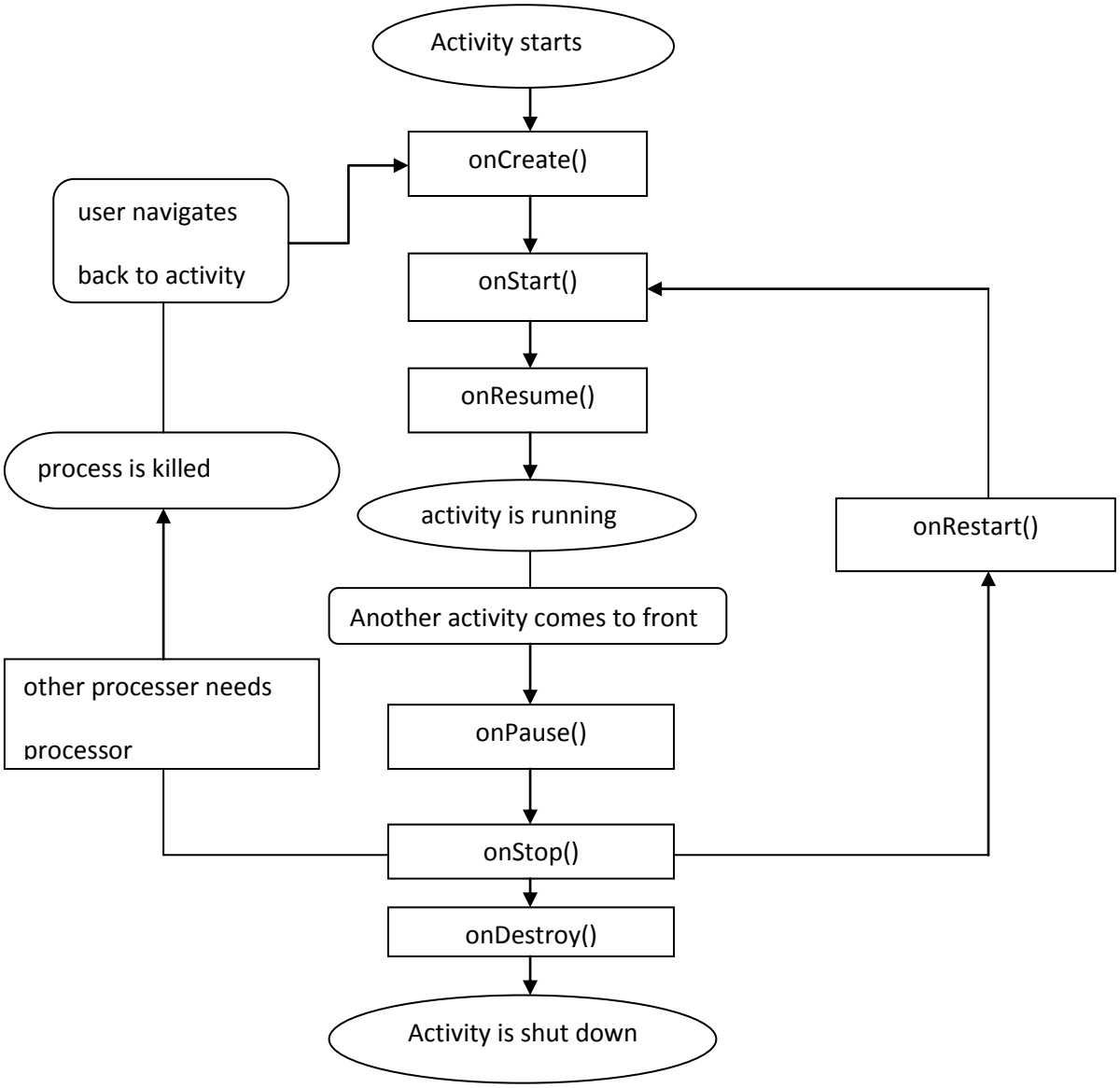
- View system - handles GUI related tasks.
- Package manager - retrieve various kinds of information related to the currently installed applicationon the device.
- Resource manager - provides access to the non-code data such as icons etc.
- Location manager - location-based and related service.
- Notification manager - executes and manages all notification etc.

2.3 Android development phase model



2.4 Application life cycle

An activity is a single screen with a standard UI. We can say that an email application can have one activity that displays a roster of new emails, another activity to make up an email, and another activity for reading purposes.For more than one activity in an application, one of them should be marked as the activity that is existing when the application is to be launched.



The Activity class exhibits the following callbacks known to us as events. wedo not need to apply all the callback methods. However, it is of utmost importance that we understand each one and implement those that guaranteethat the app behaves the way users are expecting it to.

Callback	**Description**
onCreate()	Invoked for first callback and called when the activity is created fopr the first time
onStart()	This is called when the activity becomes noticeable to the user.
onResume()	This is called when the client starts interacting with the application.
onPause()	The paused activity is not ready for input by the user and cannot execute any code and invoked when the current activity is being paused and the preceding activity is being resumed.
onStop()	Called when the activity in consideration is no longer visible.
onDestroy()	The given callback is invokedbefore the activity is killed by the system.
onRestart()	This callback is called whenever the activity has to restart after being stopped for some other application.

CHAPTER 3

APPLICATION LAYOUT AND RESULTS

3.1 Application Layout

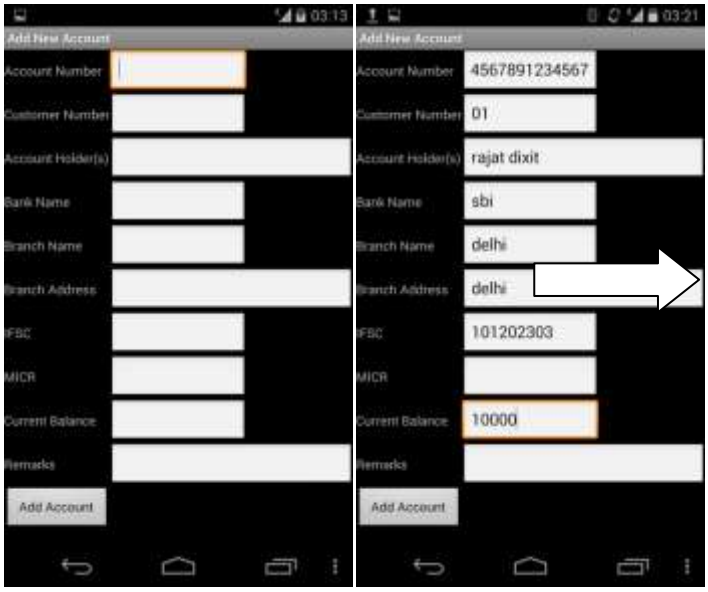
The functionalities of the application consists different attributes related to managing ones bank account. these functionalities are listed below :

- Add Account
- Add Transaction
- Search Transactions
- List Accounts
- Recent Transactions

3.1.1 Homescreen

3.1.2 Add Account

On choosing this option, the user is asked to enter certain details of theirs in the given text boxes corresponding to the required data of the user. The data entered in these boxes are automatically stored in a database.

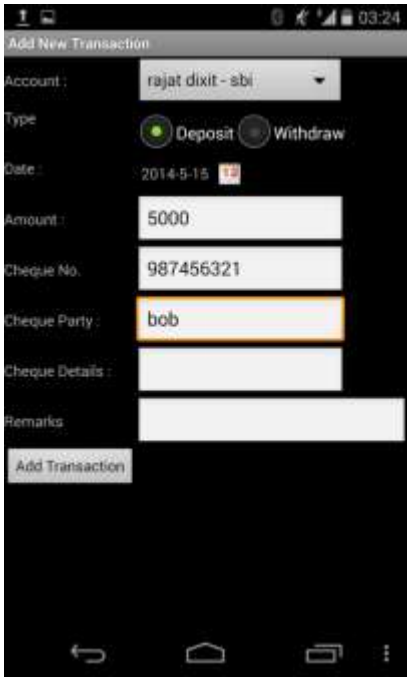


When the Add Account button is pressed, the data entered above is stored under the name Rajat Dixit and is displayed on the homescreen with all the details for further transactions.

3.1.3 Add Transaction

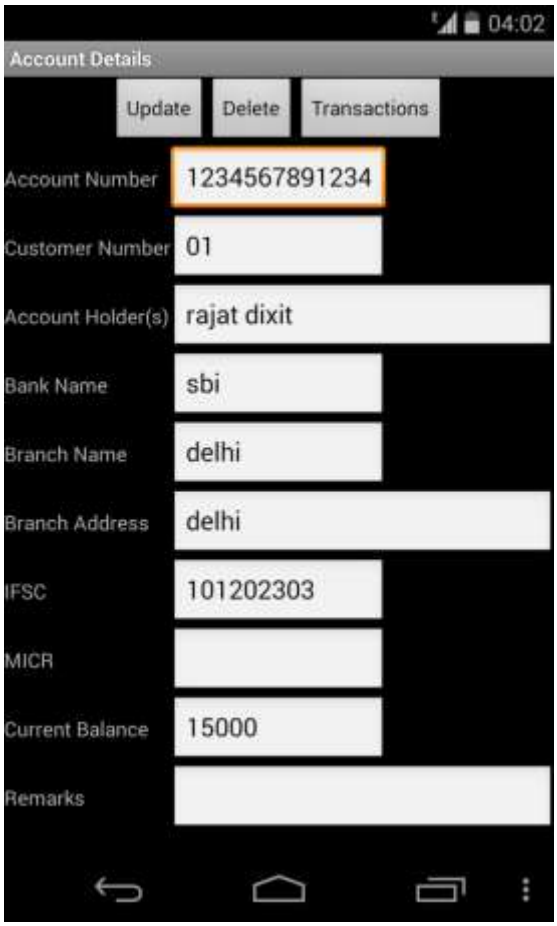
After selecting this option from the homescreen the transaction detais are added to a specified account chosen by the user from the drop down menu and the transaction details are given in the correponding text boxes and then from there stored in the database for the current session.

After the data is entered into the text boxes :



3.1.4 List Accounts

On selecting the list accounts options from the homepage, the control fetches the account details entered above and displays it in a separate page in the application under the list of accounts. The relevant account details are also displayed corresponding to the account holder.



Here the options of Update, delete and transactions are used to modify the account details or delete the account. The transactions button invokes the search transactions function and shows all the transactions done by the user.

3.1.5 Search Transactions

The search transaction functionality searches through the database if any deposit or withdrawal has been done by the registered account between certain dates as per selected by the user.

It shows whether any money was transacted between two amounts of one or two different accounts or not. On invoking the date function the **date and time**api of the android system is run and the date is set by the user.

CHAPTER 4

CODE IMPLEMENTATION

CODE PART

4.1 Account

```
package com.st.accounts;

public class Account {
private String id,acno,bank,branch,holder;

public String getId() {
	return id;
}

public void setId(String id) {
	this.id = id;
}

public String getAcno() {
	return acno;
}

public void setAcno(String acno) {
	this.acno = acno;
```

```
}

public String getBank() {
	return bank;
}

public void setBank(String bank) {
	this.bank = bank;
}

public String getBranch() {
	return branch;
}

public void setBranch(String branch) {
	this.branch = branch;
}

public String getHolder() {
	return holder;
}

public void setHolder(String holder) {
	this.holder = holder;
}

@Override
```

```
public String toString() {
        return  holder + " - " + bank;
}

}
```

4.2 **ADD ACCOUNT**

```
package com.st.accounts;

import android.app.Activity;
import android.content.ContentValues;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;

public class AddAccount extends Activity {

        @Override
        public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.addaccount);
           }
```

```
@Override
        public boolean onCreateOptionsMenu(Menu menu) {
                return Utils.inflateMenu(this,menu);
        }

        @Override
        public boolean onOptionsItemSelected(MenuItem item) {
                return  Utils.handleMenuOption(this,item);
        }

    public void addAccount(View v) {
            // get access to views
            EditText editAcno = (EditText) this.findViewById(R.id.editAcno);
            EditText editCno = (EditText) this.findViewById(R.id.editCno);
            EditText editHolders = (EditText) this.findViewById(R.id.editHolders);
            EditText editBankName = (EditText) this.findViewById(R.id.editBankName);
            EditText editBranchName = (EditText)
this.findViewById(R.id.editBranchName);
            EditText editAddress = (EditText) this.findViewById(R.id.editAddress);
            EditText editIFSC = (EditText) this.findViewById(R.id.editIFSC);
            EditText editMICR = (EditText) this.findViewById(R.id.editMICR);
            EditText editBalance = (EditText) this.findViewById(R.id.editBalance);
            EditText editRemarks = (EditText) this.findViewById(R.id.editRemarks);

                try {
                        DBHelper dbhelper = new DBHelper(this);
```

```
                    SQLiteDatabase db = dbhelper.getWritableDatabase();
Log.d("Account","Got Writable database");
                    // execute insert command


                    ContentValues values = new ContentValues();
                    values.put( Database.ACCOUNTS_ACNO,
editAcno.getText().toString());
                    values.put( Database.ACCOUNTS_CNO,
editCno.getText().toString());
                    values.put( Database.ACCOUNTS_HOLDERS,
editHolders.getText().toString());
                    values.put( Database.ACCOUNTS_BANK,
editBankName.getText().toString());
                    values.put( Database.ACCOUNTS_BRANCH,
editBranchName.getText().toString());
                    values.put( Database.ACCOUNTS_ADDRESS,
editAddress.getText().toString());
                    values.put( Database.ACCOUNTS_IFSC,
editIFSC.getText().toString());
                    values.put( Database.ACCOUNTS_MICR,
editMICR.getText().toString());
                    values.put( Database.ACCOUNTS_BALANCE,
editBalance.getText().toString());
                    values.put( Database.ACCOUNTS_REMARKS,
editRemarks.getText().toString());



                    long rows = db.insert(Database.ACCOUNTS_TABLE_NAME,
null, values);
                    db.close();
                    if ( rows > 0)  {
```

```
                    Toast.makeText(this, "Added Account Successfully!",
        Toast.LENGTH_LONG).show();

                    this.finish();

                    }

                    else

                        Toast.makeText(this, "Sorry! Could not add account!",
        Toast.LENGTH_LONG).show();


                } catch (Exception ex) {

                    Toast.makeText(this, ex.getMessage(),
Toast.LENGTH_LONG).show();

                }



        }

}
```

4.3ADD TRANSACTION

```
package com.st.accounts;


import java.util.Calendar;


import android.app.Activity;

import android.app.DatePickerDialog;

import android.app.Dialog;

import android.os.Bundle;

import android.view.Menu;

import android.view.MenuItem;

import android.view.View;
```

```
import android.widget.DatePicker;
import android.widget.EditText;
import android.widget.RadioButton;
import android.widget.Spinner;
import android.widget.TextView;
import android.widget.Toast;

public class AddTransaction extends Activity {
        private Spinner spinnerAccounts;
        private TextView textTransDate;
        private int day, month, year;
        private final int DATE_DIALOG = 1;

        @Override
        public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.add_transaction);
                spinnerAccounts = (Spinner) this.findViewById(R.id.spinnerAccounts);
                Database.populateAccounts(spinnerAccounts);
                textTransDate = (TextView) this.findViewById(R.id.textTransDate);
                 // get the current date
final Calendar c = Calendar.getInstance();
year = c.get(Calendar.YEAR);
month = c.get(Calendar.MONTH);
day = c.get(Calendar.DAY_OF_MONTH);
updateDateDisplay();
        }
```

```
        private DatePickerDialog.OnDateSetListener dateSetListener =
new DatePickerDialog.OnDateSetListener() {

public void onDateSet(DatePicker view, int pYear,int pMonth, int pDay) {
year = pYear;
month = pMonth;
day = pDay;
updateDateDisplay();
            }
        };


        @Override
        public void onStart() {
                super.onStart();

        }

        public void showDateDialog(View v) {
                showDialog(DATE_DIALOG);
        }

        @Override
        protected Dialog onCreateDialog(int id) {
                super.onCreateDialog(id);
```

```
switch (id) {
case DATE_DIALOG:
return new DatePickerDialog(this,
dateSetListener, year, month, day);
   }
return null;
}


@Override
public boolean onCreateOptionsMenu(Menu menu) {
        return Utils.inflateMenu(this,menu);
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
        return  Utils.handleMenuOption(this,item);
}

private void updateDateDisplay() {
// Month is 0 based so add 1
textTransDate.setText( String.format("%d-%d-%d",year,month + 1,day));
}

public void addTransaction(View v) {
        // get access to views
                        String accountId = Database.getAccountId(spinnerAccounts);
```

```
                    RadioButton  radioDeposit = (RadioButton)
this.findViewById(R.id.radioDeposit);


                    EditText editTransAmount = (EditText)
this.findViewById(R.id.editTransAmount);
                    EditText editChequeNo = (EditText)
this.findViewById(R.id.editChequeNo);
                    EditText editChequeParty = (EditText)
this.findViewById(R.id.editChequeParty);
                    EditText editChequeDetails = (EditText)
this.findViewById(R.id.editChequeDetails);
                    EditText editRemarks = (EditText)
this.findViewById(R.id.editRemarks);


                    boolean done = Database.addTransaction(this,
                                    accountId,
                                    radioDeposit.isChecked() ? "d" : "w",   // trans type
                                    textTransDate.getText().toString(),
                                    editTransAmount.getText().toString(),
                                    editChequeNo.getText().toString(),
                                    editChequeParty.getText().toString(),
                                    editChequeDetails.getText().toString(),
                                    editRemarks.getText().toString());


                    if ( done )
                            Toast.makeText(this,"Added Transaction Successfully!",
Toast.LENGTH_LONG).show();
                    else
                            Toast.makeText(this, "Sorry Could Not Add Transaction!",
Toast.LENGTH_LONG).show();
        } // addDeposit
```

}

4.4DATABASE

```
package com.st.accounts;

import java.util.ArrayList;

import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.util.Log;
import android.widget.ArrayAdapter;
import android.widget.DatePicker;
import android.widget.Spinner;

public class Database {
        public static final String ACCOUNTS_TABLE_NAME = "accounts";
        public static final String ACCOUNTS_ID = "_id";
        public static final String ACCOUNTS_ACNO = "acno";
        public static final String ACCOUNTS_HOLDERS = "holders";
        public static final String ACCOUNTS_CNO = "customerno";
        public static final String ACCOUNTS_BANK = "bank";
        public static final String ACCOUNTS_BRANCH = "branch";
```

```
	public static final String ACCOUNTS_ADDRESS = "address";
	public static final String ACCOUNTS_IFSC = "ifsc";
	public static final String ACCOUNTS_MICR = "micr";
	public static final String ACCOUNTS_BALANCE = "balance";
	public static final String ACCOUNTS_LASTTRANS = "last_tran_date";
	public static final String ACCOUNTS_REMARKS  = "remarks";

	public static final String TRANSACTIONS_TABLE_NAME = "transactions";
	public static final String TRANSACTIONS_ID = "_id";
	public static final String TRANSACTIONS_ACCOUNT_ID = "account_id";
	public static final String TRANSACTIONS_TRANSDATE = "transdate";
	public static final String TRANSACTIONS_TRANSTYPE = "transtype";
	public static final String TRANSACTIONS_TRANSAMOUNT = "transamount";
	public static final String TRANSACTIONS_CHEQUE_NO = "cheque_no";
	public static final String TRANSACTIONS_CHEQUE_PARTY = "cheque_party";
	public static final String TRANSACTIONS_CHEQUE_DETAILS = "cheque_details";
	public static final String TRANSACTIONS_REMARKS  = "remarks";

	public static Account cursorToAccount(Cursor accounts) {
		Account account = new Account();
		account.setId(
accounts.getString(accounts.getColumnIndex(Database.ACCOUNTS_ID)));

	account.setHolder(accounts.getString(accounts.getColumnIndex(Database.ACCOUNTS_
HOLDERS)));
		account.setBank(
accounts.getString(accounts.getColumnIndex(Database.ACCOUNTS_BANK)));
return account;
	}
```

```
public  static void populateAccounts(Spinner spinnerAccounts) {
        Context context = spinnerAccounts.getContext();
        DBHelper dbhelper = new DBHelper(context);
        SQLiteDatabase db = dbhelper.getReadableDatabase();
        Cursor accounts = db.query(Database.ACCOUNTS_TABLE_NAME, null,
null,null, null, null, null);
        ArrayList<Account> list = new ArrayList<Account>();

      //
adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
        while (accounts.moveToNext()) {
            Account account =  Database.cursorToAccount(accounts);
            list.add(account);
        }
        accounts.close();
        db.close();
        dbhelper.close();

        ArrayAdapter<Account> adapter = new ArrayAdapter<Account>(context,
android.R.layout.simple_spinner_item,list);
        spinnerAccounts.setAdapter(adapter);
    }

    public  static boolean updateAccountBalance(SQLiteDatabase db, String accountId,
String transType, double amount, String transDate) {
    try {
        if ( transType.equals("d"))
```

```
                db.execSQL( " update " + Database.ACCOUNTS_TABLE_NAME + " set
balance = balance + " + amount + " where " + Database.ACCOUNTS_ID + " = " + accountId);
else
                db.execSQL( " update " + Database.ACCOUNTS_TABLE_NAME + " set
balance = balance - " + amount + " where " + Database.ACCOUNTS_ID + " = " + accountId);
return true;
            }
        catch(Exception ex) {
                Log.d("Accounts", "Error in UpdateBalance : " + ex.getMessage());
                return false;
            }
        }

        public static String getAccountId(Spinner spinnerAccounts) {
       Account account = (Account) spinnerAccounts.getSelectedItem();
return account.getId();
        }

        public static String getDateFromDatePicker(DatePicker dp) {
                return dp.getYear() + "-" + dp.getMonth() + 1 +  "-" + dp.getDayOfMonth();
        }

        public static boolean addTransaction(Context context, String accountId, String transType,
String transDate, String transAmount, String chequeNo, String chequeParty,
                        String chequeDetails, String remarks) {

                DBHelper dbhelper = null;
                SQLiteDatabase db = null;
```

```
            try {
                    dbhelper = new DBHelper(context);
                    db = dbhelper.getWritableDatabase();
                    db.beginTransaction();

                    // execute insert command
                    ContentValues values = new ContentValues();
                    values.put(Database.TRANSACTIONS_ACCOUNT_ID, accountId);
                    values.put(Database.TRANSACTIONS_TRANSDATE, transDate);
                    values.put(Database.TRANSACTIONS_TRANSAMOUNT,
transAmount);
                    values.put(Database.TRANSACTIONS_CHEQUE_NO, chequeNo);
                    values.put(Database.TRANSACTIONS_CHEQUE_PARTY,
chequeParty);

        values.put(Database.TRANSACTIONS_CHEQUE_DETAILS,chequeDetails);
                    values.put(Database.TRANSACTIONS_REMARKS, remarks);
                    values.put(Database.TRANSACTIONS_TRANSTYPE, transType);

                    long rowid = db.insert(Database.TRANSACTIONS_TABLE_NAME,
null, values);
                    Log.d("Accounts","Inserted into TRANSACTIONS " + rowid);
                    if ( rowid != -1) {
                            // update Accounts Table
                           boolean done =
Database.updateAccountBalance(db,accountId,transType,
Double.parseDouble(transAmount),transDate);
                    Log.d("Accounts","Updated Account Balance");
                            if ( done ) {
                                    db.setTransactionSuccessful();
```

```
                    db.endTransaction();
                    return true;
                }
                else {
                    db.endTransaction();
                    return false;
                }
            }
            else
                return false;
        }
        catch(Exception ex) {
            Log.d("Account", "Error in addTransaction -->" + ex.getMessage());
            return false;
        }
        finally {
            if ( db != null && db.isOpen()) {
                db.close();
            }
        }
    } // addTransaction

}
```

4.5 **LIST ACCOUNT TRANSACTIONS**

```
package com.st.accounts;

import java.util.ArrayList;
import java.util.LinkedHashMap;
import java.util.Map;

import android.app.Activity;
import android.content.Intent;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.opengl.Visibility;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ListView;
import android.widget.SimpleAdapter;
import android.widget.TextView;
import android.widget.Toast;

public class ListAccountTransactions extends Activity {
        ListView listTransactions;
        String accountId;
```

```
@Override
public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.list_account_transactions);
        accountId = this.getIntent().getStringExtra("accountid");
        listTransactions = (ListView) this.findViewById(R.id.listTransactions);

        listTransactions.setOnItemClickListener(new OnItemClickListener() {
                @Override
                public void onItemClick(AdapterView<?> parent, View selectedView,
                                int arg2, long arg3) {
                        TextView textTransId = (TextView) selectedView
                                        .findViewById(R.id.textTransId);
                        Intent intent = new Intent(ListAccountTransactions.this,
                                        TransactionDetails.class);
                        intent.putExtra("transid", textTransId.getText().toString());
                        startActivity(intent);
                }
        });
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
        return Utils.inflateMenu(this, menu);
}

@Override
```

```
public boolean onOptionsItemSelected(MenuItem item) {
        return Utils.handleMenuOption(this, item);
}

@Override
public void onStart() {
        super.onStart();
        try {
                DBHelper dbhelper = new DBHelper(this);
                SQLiteDatabase db = dbhelper.getReadableDatabase();
                Cursor trans = db.query(Database.TRANSACTIONS_TABLE_NAME,
null,
                                Database.TRANSACTIONS_ACCOUNT_ID + " = ?",
                                new String[] { accountId }, null, null,
                                Database.TRANSACTIONS_TRANSDATE + " desc");

                if (trans.getCount() == 0 ) // no trans found
                {
                        // turn off tablelayout and turnon textview for no transactions
                        // display
                        this.findViewById(R.id.heading).setVisibility(View.INVISIBLE);
                        this.findViewById(R.id.textError).setVisibility(View.VISIBLE);
                } else {
                        this.findViewById(R.id.heading).setVisibility(View.VISIBLE);

this.findViewById(R.id.textError).setVisibility(View.INVISIBLE);
                }
```

```
                ArrayList<Map<String, String>> listTrans = new ArrayList<Map<String,
String>>();

                while (trans.moveToNext()) {


                        // get trans details for display

                        LinkedHashMap<String, String> tran = new
LinkedHashMap<String, String>();

                        tran.put("transid", trans.getString(trans

        .getColumnIndex(Database.TRANSACTIONS_ID)));

                        tran.put("transdate", trans.getString(trans

        .getColumnIndex(Database.TRANSACTIONS_TRANSDATE)));

                        String transType = trans.getString(trans

        .getColumnIndex(Database.TRANSACTIONS_TRANSTYPE));

                        String transAmount = trans.getString(trans

        .getColumnIndex(Database.TRANSACTIONS_TRANSAMOUNT));

                        String chequeno = trans.getString(trans

        .getColumnIndex(Database.TRANSACTIONS_CHEQUE_NO));

                        String transDetails = "Cash";

                        if (!chequeno.trim().equals(""))

                                transDetails = "Cheque No: " + chequeno;

                        tran.put("transdetails", transDetails);

                        tran.put("transtype", transType);

                        tran.put("transamount", transAmount);

                        listTrans.add(tran);

                }
```

```
                trans.close();
                db.close();
                dbhelper.close();

                SimpleAdapter adapter = new SimpleAdapter(this, listTrans,
                        R.layout.account_transaction, new String[] { "transid",
                                "transdate", "transdetails", "transtype",
                                "transamount" }, new int[] {
R.id.textTransId,
                                R.id.textTransDate, R.id.textTransDetails,
                                R.id.textTransType, R.id.textAmount });

                listTransactions.setAdapter(adapter);
            } catch (Exception ex) {
                Toast.makeText(this, ex.getMessage(), Toast.LENGTH_LONG).show();
            }
        }

}
```

4.6 LIST RECENT TRANSACTIONS

```
package com.st.accounts;

import java.util.ArrayList;
import java.util.LinkedHashMap;
import java.util.Map;

import android.app.Activity;
```

```
import android.content.Intent;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ListView;
import android.widget.SimpleAdapter;
import android.widget.TextView;
import android.widget.Toast;

public class ListRecentTransactions extends Activity {
	ListView listTransactions;
	String  fromDate,toDate,fromAmount,toAmount;
	String  condition = " 1 = 1 ";
	@Override
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.list_transactions);
		listTransactions = (ListView) this.findViewById(R.id.listTransactions);

		listTransactions.setOnItemClickListener( new OnItemClickListener() {
			@Override
			public void onItemClick(AdapterView<?> parent, View selectedView, int
arg2,long arg3) {
```

```
                    TextView  textTransId = (TextView)
selectedView.findViewById(R.id.textTransId);
                    Intent intent = new Intent(ListRecentTransactions.this,
TransactionDetails.class);
                    intent.putExtra("transid", textTransId.getText().toString());
                    startActivity(intent);
                }
            });
        }

        @Override
        public boolean onCreateOptionsMenu(Menu menu) {
            return Utils.inflateMenu(this,menu);
        }

        @Override
        public boolean onOptionsItemSelected(MenuItem item) {
            return  Utils.handleMenuOption(this,item);
        }



        @Override
        public void onStart() {
            super.onStart();
            try {
                DBHelper dbhelper = new DBHelper(this);
                SQLiteDatabase db = dbhelper.getReadableDatabase();
```

```
                Cursor trans = db.rawQuery("select t._id, acno,bank,
transdate,transamount,transtype,cheque_no,cheque_party,cheque_details, t.remarks from
transactions t inner join accounts a  on ( a._id = t.account_id) order by  transdate desc   LIMIT
10",null);

                if ( trans.getCount() == 0 )

                        this.findViewById(R.id.textError).setVisibility(View.VISIBLE);

                else


        this.findViewById(R.id.textError).setVisibility(View.INVISIBLE);


                ArrayList<Map<String,String>> listTrans = new
ArrayList<Map<String,String>>();

while ( trans.moveToNext()) {

        // get trans details for display

        LinkedHashMap<String,String> tran = new LinkedHashMap<String,String>();

        tran.put("transid",
trans.getString(trans.getColumnIndex(Database.TRANSACTIONS_ID)));

        tran.put("acno",  trans.getString(trans.getColumnIndex(Database.ACCOUNTS_ACNO))
+ " - " +   trans.getString(trans.getColumnIndex(Database.ACCOUNTS_BANK)));

        tran.put("transdate",trans.getString(trans.getColumnIndex(Database.TRANSACTIONS_
TRANSDATE)));

        tran.put("transtype",trans.getString(trans.getColumnIndex(Database.TRANSACTIONS_
TRANSTYPE)));

        tran.put("transamount",trans.getString(trans.getColumnIndex(Database.TRANSACTION
S_TRANSAMOUNT)));

        tran.put("transremarks",trans.getString(trans.getColumnIndex(Database.TRANSACTIO
NS_REMARKS)));


        String chequeno =
trans.getString(trans.getColumnIndex(Database.TRANSACTIONS_CHEQUE_NO));

        String transDetails = "Cash";

        if (! chequeno.trim().equals(""))
```

```
                transDetails = "Cheque No: " + chequeno;


        tran.put("transdetails",transDetails);
        listTrans.add(tran);
        }
trans.close();
db.close();
                dbhelper.close();


                SimpleAdapter  adapter = new SimpleAdapter(this,
                                listTrans,
                                R.layout.transaction,
                                new String [] {"transid", "acno", "transdate", "transdetails",
"transtype", "transamount" ,"transremarks"},
                                new  int [] {  R.id.textTransId, R.id.textAcno,  R.id.textTransDate,
R.id.textTransDetails, R.id.textTransType, R.id.textTransAmount, R.id.textTransRemarks});


                listTransactions.setAdapter(adapter);
                } catch (Exception ex) {
                        Toast.makeText(this, ex.getMessage(), Toast.LENGTH_LONG).show();
                }
        }


}
```

4.7LIST TRANSACTIONS

```
package com.st.accounts;


import java.util.ArrayList;
```

```
import java.util.LinkedHashMap;
import java.util.Map;

import android.app.Activity;
import android.content.Intent;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ListView;
import android.widget.SimpleAdapter;
import android.widget.TextView;
import android.widget.Toast;

public class ListTransactions extends Activity {
	ListView listTransactions;
	String  fromDate,toDate,fromAmount,toAmount;
	String  condition = " 1 = 1 ";
	@Override
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.list_transactions);
		fromDate = this.getIntent().getStringExtra("fromdate");
```

```
toDate = this.getIntent().getStringExtra("todate");
fromAmount = this.getIntent().getStringExtra("fromamount");
toAmount = this.getIntent().getStringExtra("toamount");
listTransactions = (ListView) this.findViewById(R.id.listTransactions);

// form condition based on input
if ( fromDate.length() > 0)
    condition += " and  " + Database.TRANSACTIONS_TRANSDATE + "
>= '" + fromDate + "'";

if ( toDate.length() > 0)
    condition += " and  " + Database.TRANSACTIONS_TRANSDATE + "
<= '" + toDate + "'";

if ( fromAmount.length() > 0)
    condition += " and  " + Database.TRANSACTIONS_TRANSAMOUNT +
" >= " + fromAmount;

if ( toAmount.length() > 0)
    condition += " and  " + Database.TRANSACTIONS_TRANSAMOUNT +
" <= " + toAmount;

listTransactions.setOnItemClickListener( new OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> parent, View selectedView, int
arg2,long arg3) {
```

```
                    TextView  textTransId = (TextView)
selectedView.findViewById(R.id.textTransId);
                    Intent intent = new Intent(ListTransactions.this,
TransactionDetails.class);
                    intent.putExtra("transid", textTransId.getText().toString());
                    startActivity(intent);
                }
            });
        }

        @Override
        public boolean onCreateOptionsMenu(Menu menu) {
            return Utils.inflateMenu(this,menu);
        }

        @Override
        public boolean onOptionsItemSelected(MenuItem item) {
            return  Utils.handleMenuOption(this,item);
        }


        @Override
        public void onStart() {
            super.onStart();
            try {
                DBHelper dbhelper = new DBHelper(this);
                SQLiteDatabase db = dbhelper.getReadableDatabase();
```

```
Cursor trans = db.rawQuery("select t._id, acno, bank,
transdate,transamount,transtype,cheque_no,cheque_party,cheque_details, t.remarks from
transactions t inner join accounts a  on ( a._id = t.account_id) where " + condition,null);

if ( trans.getCount() == 0 )

this.findViewById(R.id.textError).setVisibility(View.VISIBLE);

else


this.findViewById(R.id.textError).setVisibility(View.INVISIBLE);



ArrayList<Map<String,String>> listTrans = new
ArrayList<Map<String,String>>();

while ( trans.moveToNext()) {

// get trans details for display

LinkedHashMap<String,String> tran = new LinkedHashMap<String,String>();

tran.put("transid",
trans.getString(trans.getColumnIndex(Database.TRANSACTIONS_ID)));

tran.put("acno",  trans.getString(trans.getColumnIndex(Database.ACCOUNTS_ACNO))
+ " - " +   trans.getString(trans.getColumnIndex(Database.ACCOUNTS_BANK)));

tran.put("transdate",trans.getString(trans.getColumnIndex(Database.TRANSACTIONS_
TRANSDATE)));

tran.put("transtype",trans.getString(trans.getColumnIndex(Database.TRANSACTIONS_
TRANSTYPE)));

tran.put("transamount",trans.getString(trans.getColumnIndex(Database.TRANSACTION
S_TRANSAMOUNT)));

tran.put("transremarks",trans.getString(trans.getColumnIndex(Database.TRANSACTIO
NS_REMARKS)));



String chequeno =
trans.getString(trans.getColumnIndex(Database.TRANSACTIONS_CHEQUE_NO));

String transDetails = "Cash";

if (! chequeno.trim().equals(""))

transDetails = "Cheque No: " + chequeno;
```

```
        tran.put("transdetails",transDetails);
        listTrans.add(tran);
        }
trans.close();
db.close();
                dbhelper.close();

                SimpleAdapter  adapter = new SimpleAdapter(this,
                                listTrans,
                                R.layout.transaction,
                                new String [] {"transid", "acno", "transdate", "transdetails",
"transtype", "transamount" ,"transremarks"},
                                new  int [] {  R.id.textTransId, R.id.textAcno,  R.id.textTransDate,
R.id.textTransDetails, R.id.textTransType, R.id.textTransAmount, R.id.textTransRemarks});

                listTransactions.setAdapter(adapter);
                } catch (Exception ex) {
                        Toast.makeText(this, ex.getMessage(), Toast.LENGTH_LONG).show();
                }
        }

}
```

4.8 SEARCH TRANSACTIONS

```
package com.st.accounts;

import java.util.Calendar;
```

```
import android.app.Activity;
import android.app.DatePickerDialog;
import android.app.Dialog;
import android.content.Intent;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.DatePicker;
import android.widget.EditText;
import android.widget.RadioButton;
import android.widget.Spinner;
import android.widget.TextView;
import android.widget.Toast;

public class SearchTransactions extends Activity {
	private EditText editFromDate,editToDate,editFromAmount,editToAmount;
	private int fromDay, fromMonth, fromYear;
	private int toDay, toMonth, toYear;
	private final int FROM_DATE_DIALOG = 1;
	private final int TO_DATE_DIALOG = 2;

	@Override
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.search_transactions);
		editFromDate = (EditText) this.findViewById(R.id.editFromDate);
```

```
                editToDate = (EditText) this.findViewById(R.id.editToDate);

                editFromAmount = (EditText) this.findViewById(R.id.editFromAmount);
                editToAmount = (EditText) this.findViewById(R.id.editToAmount);

                 // get the current date
final Calendar c = Calendar.getInstance();
fromYear = toYear = c.get(Calendar.YEAR);
fromMonth  = toMonth = c.get(Calendar.MONTH);
toDay = c.get(Calendar.DAY_OF_MONTH);

fromDay  = 1;  // from is set to 1st of the current month

updateToDateDisplay();
updateFromDateDisplay();
        }
        @Override
        public boolean onCreateOptionsMenu(Menu menu) {
                return Utils.inflateMenu(this,menu);
        }

        @Override
        public boolean onOptionsItemSelected(MenuItem item) {
                return  Utils.handleMenuOption(this,item);
        }
```

```
        private DatePickerDialog.OnDateSetListener fromDateSetListener =
new DatePickerDialog.OnDateSetListener() {
public void onDateSet(DatePicker view, int pYear,int pMonth, int pDay) {
fromYear = pYear;
fromMonth = pMonth;
fromDay = pDay;
updateFromDateDisplay();
            }
        };

        private DatePickerDialog.OnDateSetListener toDateSetListener =
new DatePickerDialog.OnDateSetListener() {
public void onDateSet(DatePicker view, int pYear,int pMonth, int pDay) {
toYear = pYear;
toMonth = pMonth;
toDay = pDay;
updateToDateDisplay();
                }
            };

        public void showFromDateDialog(View v) {
                showDialog(FROM_DATE_DIALOG);
        }

        public void showToDateDialog(View v) {
                showDialog(TO_DATE_DIALOG);
        }
```

```
        @Override
        protected Dialog onCreateDialog(int id) {
        switch (id) {
        case FROM_DATE_DIALOG:
        return new DatePickerDialog(this,
        fromDateSetListener, fromYear, fromMonth, fromDay);
        case TO_DATE_DIALOG:
        return new DatePickerDialog(this,
        toDateSetListener, toYear, toMonth, toDay);
            }
        return null;
        }

        private void updateToDateDisplay() {
        // Month is 0 based so add 1
        editToDate.setText( String.format("%d-%d-%d",toYear,toMonth + 1,toDay));
        }
        private void updateFromDateDisplay() {
      // Month is 0 based so add 1
editFromDate.setText( String.format("%d-%d-%d",fromYear,fromMonth + 1,fromDay));
}

        public void searchTransactions(View v) {
                    Intent intent = new Intent(this, ListTransactions.class);
                intent.putExtra("fromdate", editFromDate.getText().toString());
                intent.putExtra("todate", editToDate.getText().toString());
```

```
            intent.putExtra("fromamount", editFromAmount.getText().toString());
            intent.putExtra("toamount", editToAmount.getText().toString());
            startActivity(intent);
        }

        public void clearFields(View v) {

editFromDate.setText("");
editToDate.setText("");
editFromAmount.setText("");
editToAmount.setText("");
        }

}
```

4.9 TRANSACTION DETAILS

```
package com.st.accounts;

import android.app.Activity;
import android.app.AlertDialog;
import android.content.DialogInterface;
import android.content.Intent;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
```

```
import android.view.View;
import android.widget.TextView;
import android.widget.Toast;

public class TransactionDetails extends Activity {
        private String transId;
        private String accountId;
        private TextView textAcno;
        @Override
        public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.transaction_details);

                transId = this.getIntent().getStringExtra("transid");
                Log.d("Account", "Trans id : " + transId);

                textAcno = (TextView) this.findViewById(R.id.textAcno);
                TextView textTransDate = (TextView) this.findViewById(R.id.textTransDate);
                TextView textTransType = (TextView) this.findViewById(R.id.textTransType);
                TextView textTransAmount = (TextView)
this.findViewById(R.id.textTransAmount);
                TextView textChequeNo = (TextView) this.findViewById(R.id.textChequeNo);
                TextView textChequeParty = (TextView)
this.findViewById(R.id.textChequeParty);
                TextView textChequeDetails = (TextView)
this.findViewById(R.id.textChequeDetails);
                TextView textRemarks  = (TextView)
this.findViewById(R.id.textTransRemarks);
```

```
DBHelper dbhelper = new DBHelper(this);

SQLiteDatabase db = dbhelper.getReadableDatabase();

Cursor tran = db.rawQuery("select
acno,account_id,transdate,transamount,transtype,cheque_no,cheque_party,cheque_details,
t.remarks from transactions t inner join accounts a  on ( a._id = t.account_id) where t._id = ?",

                new String[] {transId });



if (tran.moveToFirst()) {

        accountId  =
tran.getString(tran.getColumnIndex(Database.TRANSACTIONS_ACCOUNT_ID));

        textAcno.setText(
tran.getString(tran.getColumnIndex(Database.ACCOUNTS_ACNO)));

        textTransDate.setText(
tran.getString(tran.getColumnIndex(Database.TRANSACTIONS_TRANSDATE)));

        textTransType.setText(
tran.getString(tran.getColumnIndex(Database.TRANSACTIONS_TRANSTYPE)));

        textTransAmount.setText(
tran.getString(tran.getColumnIndex(Database.TRANSACTIONS_TRANSAMOUNT)));

        textChequeNo.setText(
tran.getString(tran.getColumnIndex(Database.TRANSACTIONS_CHEQUE_NO)));

        textChequeParty.setText(
tran.getString(tran.getColumnIndex(Database.TRANSACTIONS_CHEQUE_PARTY)));

        textChequeDetails.setText(
tran.getString(tran.getColumnIndex(Database.TRANSACTIONS_CHEQUE_DETAILS)));


    textRemarks.setText(tran.getString(tran.getColumnIndex(Database.TRANSACTIONS_R
EMARKS)));

        }

        else

                Log.d("Accounts","No transaction found!");
```

```
        db.close();
        dbhelper.close();
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
        return Utils.inflateMenu(this,menu);
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
        return  Utils.handleMenuOption(this,item);
}

public void deleteTransaction(View v) {
        AlertDialog.Builder builder = new AlertDialog.Builder(this);
        builder.setMessage("Are you sure you want to delete this transaction?")
            .setCancelable(false)
            .setPositiveButton("Yes", new DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int id) {
        deleteCurrentTransaction();
              }
            })
            .setNegativeButton("No", new DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int id) {
        dialog.cancel();
```

```
                    }
                });
            AlertDialog alert = builder.create();
            alert.show();
        }



public void deleteCurrentTransaction() {
        try {
                    DBHelper dbhelper = new DBHelper(this);
                    SQLiteDatabase db = dbhelper.getWritableDatabase();
                    int rows = db.delete(Database.TRANSACTIONS_TABLE_NAME,
"_id=?", new String[] { transId});
                    dbhelper.close();
                    if ( rows == 1) {
                        Toast.makeText(this, "Transaction Deleted Successfully!",
Toast.LENGTH_LONG).show();
                        this.finish();
                    }
                    else
                        Toast.makeText(this, "Could not delet transaction!",
Toast.LENGTH_LONG).show();
            }
        catch (Exception ex) {
                    Toast.makeText(this, ex.getMessage(), Toast.LENGTH_LONG).show();
            }
        }
```

```
public void showAccountDetails(View v) {
      Intent intent = new Intent(this,UpdateAccount.class);
         intent.putExtra("accountid", accountId);
         startActivity(intent);
         }
}
```

4.10 UPDATE ACCOUNT

```
package com.st.accounts;


import android.app.Activity;
import android.app.AlertDialog;
import android.content.ContentValues;
import android.content.DialogInterface;
import android.content.Intent;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.util.Log;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;


public class UpdateAccount extends Activity {
         private String accountId;
```

```
private EditText editAcno, editCno, editHolders, editBankName,
                editBranchName, editAddress, editIFSC, editMICR, editBalance,
                editRemarks;

@Override
public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.update_account);
        editAcno = (EditText) this.findViewById(R.id.editAcno);
        editCno = (EditText) this.findViewById(R.id.editCno);
        editHolders = (EditText) this.findViewById(R.id.editHolders);
        editBankName = (EditText) this.findViewById(R.id.editBankName);
        editBranchName = (EditText) this.findViewById(R.id.editBranchName);
        editAddress = (EditText) this.findViewById(R.id.editAddress);
        editIFSC = (EditText) this.findViewById(R.id.editIFSC);
        editMICR = (EditText) this.findViewById(R.id.editMICR);
        editBalance = (EditText) this.findViewById(R.id.editBalance);
        editRemarks = (EditText) this.findViewById(R.id.editRemarks);
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
        return Utils.inflateMenu(this,menu);
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
```

```
        return  Utils.handleMenuOption(this,item);
}



@Override
public void onStart() {
        super.onStart();
        accountId = this.getIntent().getStringExtra("accountid");
        Log.d("Accounts", "Account Id : " + accountId);
        DBHelper dbhelper = new DBHelper(this);
        SQLiteDatabase db = dbhelper.getReadableDatabase();
        Cursor account = db.query(Database.ACCOUNTS_TABLE_NAME, null,
                        " _id = ?", new String[] { accountId }, null, null, null);
        //startManagingCursor(accounts);
        if (account.moveToFirst()) {
                // update view
                editAcno.setText(account.getString(account
                                .getColumnIndex(Database.ACCOUNTS_ACNO)));
                editCno.setText(account.getString(account
                                .getColumnIndex(Database.ACCOUNTS_CNO)));
                editHolders.setText(account.getString(account
                                .getColumnIndex(Database.ACCOUNTS_HOLDERS)));
                editBankName.setText(account.getString(account
                                .getColumnIndex(Database.ACCOUNTS_BANK)));
                editBranchName.setText(account.getString(account
                                .getColumnIndex(Database.ACCOUNTS_BRANCH)));
```

```
                editAddress.setText(account.getString(account
                        .getColumnIndex(Database.ACCOUNTS_ADDRESS)));
                editIFSC.setText(account.getString(account
                        .getColumnIndex(Database.ACCOUNTS_IFSC)));
                editMICR.setText(account.getString(account
                        .getColumnIndex(Database.ACCOUNTS_MICR)));
                editBalance.setText(account.getString(account
                        .getColumnIndex(Database.ACCOUNTS_BALANCE)));
                editRemarks.setText(account.getString(account
                        .getColumnIndex(Database.ACCOUNTS_REMARKS)));
        }
        account.close();
        db.close();
        dbhelper.close();

}

public void updateAccount(View v) {
        try {
                DBHelper dbhelper = new DBHelper(this);
                SQLiteDatabase db = dbhelper.getWritableDatabase();
                // execute insert command
                ContentValues values = new ContentValues();
                values.put(Database.ACCOUNTS_ACNO, editAcno.getText().toString());
                values.put(Database.ACCOUNTS_CNO, editCno.getText().toString());
                values.put(Database.ACCOUNTS_HOLDERS, editHolders.getText()
                        .toString());
```

```
                values.put(Database.ACCOUNTS_BANK, editBankName.getText()
                        .toString());
                values.put(Database.ACCOUNTS_BRANCH, editBranchName.getText()
                        .toString());
                values.put(Database.ACCOUNTS_ADDRESS, editAddress.getText()
                        .toString());
                values.put(Database.ACCOUNTS_IFSC, editIFSC.getText().toString());
                values.put(Database.ACCOUNTS_MICR,
editMICR.getText().toString());
                values.put(Database.ACCOUNTS_BALANCE, editBalance.getText()
                        .toString());
                values.put(Database.ACCOUNTS_REMARKS, editRemarks.getText()
                        .toString());

                long rows = db.update(Database.ACCOUNTS_TABLE_NAME, values,
                        "_id = ?", new String[] { accountId });

                db.close();
                if (rows > 0)
                    Toast.makeText(this, "Updated Account Successfully!",
                            Toast.LENGTH_LONG).show();
                else
                    Toast.makeText(this, "Sorry! Could not update account!",
                            Toast.LENGTH_LONG).show();
            } catch (Exception ex) {
                Toast.makeText(this, ex.getMessage(), Toast.LENGTH_LONG).show();
            }
        }
```

```
public void deleteAccount(View v) {
                AlertDialog.Builder builder = new AlertDialog.Builder(this);
                builder.setMessage("Are you sure you want to delete this account?")
                    .setCancelable(false)
                    .setPositiveButton("Yes", new DialogInterface.OnClickListener() {
                public void onClick(DialogInterface dialog, int id) {
                deleteCurrentAccount();
                    }
                    })
                    .setNegativeButton("No", new DialogInterface.OnClickListener() {
                public void onClick(DialogInterface dialog, int id) {
                dialog.cancel();
                    }
                    });
                AlertDialog alert = builder.create();
                alert.show();
        }



public void deleteCurrentAccount() {
        try {
                DBHelper dbhelper = new DBHelper(this);
                SQLiteDatabase db = dbhelper.getWritableDatabase();
                int rows = db.delete(Database.ACCOUNTS_TABLE_NAME, "_id=?",
new String[] { accountId});
                dbhelper.close();
```

```
                    if ( rows == 1) {
                        Toast.makeText(this, "Account Deleted Successfully!",
Toast.LENGTH_LONG).show();
                        this.finish();
                    }
                    else
                        Toast.makeText(this, "Could not delet account!",
Toast.LENGTH_LONG).show();

                } catch (Exception ex) {
                    Toast.makeText(this, ex.getMessage(), Toast.LENGTH_LONG).show();
                }

        }

public void listAccountTransactions(View v) {
        Intent intent = new Intent(this,ListAccountTransactions.class);
        intent.putExtra("accountid", accountId);
        startActivity(intent);
        }
}
```

4.11 UTILS

```
package com.st.accounts;

import android.app.Activity;
import android.content.Intent;
import android.view.Menu;
import android.view.MenuInflater;
```

```
import android.view.MenuItem;

public class Utils {

public static boolean  inflateMenu(Activity activity, Menu menu) {
        MenuInflater inflater = activity.getMenuInflater();
                inflater.inflate( R.menu.common_menu, menu);
                return true;
   }

public static boolean handleMenuOption(Activity activity, MenuItem item) {
        Intent intent;
                switch(item.getItemId()) {
                case R.id.optAddAccount :
                        intent = new Intent(activity,AddAccount.class);
                        activity.startActivity(intent);
                        break;
                case R.id.optAddTransaction :
                        intent = new Intent(activity,AddTransaction.class);
                        activity.startActivity(intent);
                        break;

                case R.id.optSearchTransactions :
                        intent = new Intent(activity,SearchTransactions.class);
                        activity.startActivity(intent);
                        break;
                case R.id.optListAccounts :
```

```
            intent = new Intent(activity,ListAccounts.class);
            activity.startActivity(intent);
            break;

    case R.id.optRecentTransactions :
            intent = new Intent(activity,ListRecentTransactions.class);
            activity.startActivity(intent);
            break;
    }
    return true;    }}
```

CONCLUSION

The java code blocks and the android software development packages were thoroughly studied and implemented after a theoritical and several practical test runs. All the functionalities mentioned above run real time with real time data entry. The overall success rate of all the functionalities were tested with different data and an approximate success of 98% was exhibited by the application.

It is also seen that the battery usage of the application was with level of a device on standard display and idle battery time.

REFERENCES

- http://developer.android.com
- Head first android, editor : Brian Sayer
- http://www.stackoverflow.com
- android application development, authors : Lauren Darcy, Shane Conder