**CONTENTS**

ACKNOWLDGEMENT

**CONTENTS**

SYNOPSIS

1. INTRODUCTION

1.1 SYSTEM SPECIFICATION

1.1.1 HARDWARE CONFIGURATION

1.1.2 SOFTWARE SPECIFICATION

2. SYSTEM STUDY

2.1 EXISTING SYSTEM

2.1.1 DRAWBACKS

2.2 PROPOSED SYSTEM

2.2.1 FEATURES

3. SYSTEM DESIGN AND DEVELOPMENT

3.1 FILE DESIGN

3.2 INPUT DESIGN

3.3 OUTPUT DESIGN

3.4 DATABASE DESIGN

3.5 SYSTEM DEVELOPMENT

3.5.1 DESCRIPTION OF MODULES

4. TESTING AND IMPLEMENTATION

5. CONCLUSION

BIBLIOGRAPHY

APPENDICES

A. DATA FLOW DIAGRAM

B. TABLE STRUCTURE

C. SAMPLE CODING

D. SAMPLE INPUT

E. SAMPLE OUTPUT

**SYNOPSIS**

**SYNOPSIS**

This Android **“BANK ACCOUNT TRACKER”** application allows user to keep track of current balance in different bank accounts held by the user and the transactions of those accounts.

**BANK ACCOUNT TRACKER** explains about implementing an app for android mobiles which will help users to know about bank balances in different banks and their transactions information.

In present trend usage of apps had become a new trend because of availability of web services on mobiles.

By considering these improvements in mobile technology knowing information of money transactions through mobile in less time can be useful application for users.

In this application initially users need to install app and update details like listing out different banks and adding new bank accounts.

**1. INTRODUCTION**

**“BANK ACCOUNT TRACKER”** is anAndroid application allows user to keep track of current balance in different bank accounts held by the user and the transactions of those accounts

Bank Account tracker explains about implementing an app for android mobiles which will help users to know about bank balances in different banks and there transactions information.

In present trend usage of apps had become a new trend because of availability of web services on mobiles.

By considering these improvements in mobile technology knowing information of money transactions through mobile in less time can be useful application for users.

In this application initially users need to install app and update details like listing out different banks and adding new bank accounts.

**SYSTEM SPECIFICATION**

**1.1 SYSTEM SPECIFICATION**

**1.1.1 HARDWARE CONFIGURATION**

PROCESSOR : Pentium(R) Dual-Core E5300

MEMORY : 2 GB DDR2 RAM

HARD DISK : 320 GB

KEYBOARD : 105 keys

MOUSE : Optical

**1.1.2 SOFTWARE SPECIFICATION**

* Operating System : Windows 8 professional 64-bit
* Eclipse Oxygen IDE
* Android SDK (Standard Development Kit)
* ADT (Android Development Tool) plug-in
* JRE (Java Runtime Environment) 8
* AVD (Android Virtual Device)

**Various operations performed by the application:**

* List of bank accounts
* Adding new bank accounts
* Updating an existing bank account
* Deleting a bank account
* List of transactions related to bank account
* Search transaction by date and Amount
* Deleting a transaction

**List of Bank accounts:**

Displays all the accounts which are created by the user.

**Adding new bank accounts:**

The application adds new accounts of a new bank. For adding a new bank account the user has to click onto the ad account button in the application.

**Updating an existing bank account:**

Whenever the user wishes to modify the amount and also the contents of a particular account he can do all this by using the update option.

**Deleting a bank account:**

The account in bank is closed or it is not necessary for the user, then he can delete the existing account.

**List of transactions related to bank account**

As many transactions will occur in many accounts, then it becomes very difficult to identify the recent transaction for a single account. So by clicking the list of transactions option the user may feel easy to identify the correct transaction.

**Search transaction by date and Amount**

When many transactions arise then the user may find very difficult to find the correct transaction for the right account. So when the user enters the approximate date or the approximate amount then the transactions or the amount between the entered date and amount will be displayed.

**Deleting a transaction**

When the particular transaction is not needed then the user can easily delete the particular transaction.

**SYSTEM STUDY**

**2. SYSTEM STUDY**

**2.1 EXISTING SYSTEM**

Existing system is a general account transaction plan in which the user uses records and notebooks for recording all the transactions for the various accounts of all banks

In present trend usage of android applications had become a new trend because of availability of web services on mobiles also the availability of lots of applications in Google play store.

By considering these improvements in mobile technology knowing information of money transactions through mobile in less time can be useful application for users.

By using this android mobile application the user can easily find all the details of his accounts in various banks in his pocket. He can easily access the details whenever the need arises.

**2.1.1 DRAWBACKS**

While maintaining records in notebooks then there may be a chance of getting lost of those important documents. Also searching an individual transaction from a big group of data may be tedious task

**2.2 PROPOSED SYSTEM**

In order to overcome existing system problems new system is developed using this system any account can be easily searched with better security features.

Adding new bank account updating an existing bank account deleting a bank account List of transactions related to a bank accounts List of recent transactions from all accounts Search Transactions by date and amount showing all details of a single transaction.

**2.2.1 FEATURES**

List of bank accounts adding new bank account updating an existing bank account deleting a bank account List of transactions related to a bank accounts List of recent 10 transactions from all accounts Search Transactions by date and amount Showing all details of a single transaction Deleting a transaction

**ADD NEW BANKACCOUNT**

This module provides new account related functionalities. User can create a new account.

**TRANSACTION LIST**

List the transaction is a very simple and easy to use app for your list of transactions related to a bank accounts List of recent 10 transactions from all accounts.

**SEARCH TRANSACTION**

This module is about search transaction account of this portal. We have to search New or transaction account, then add the deposit and withdraw amount.

**Search Transactions by date and amount:**

This module searches the transaction made to a particular account when we enter the approximate amount of the transaction and the known date

**UPDATE OR DELETE TRANSACTION ACCOUNT**

This module provides update or deletes transaction account functionalities showing all details of a single transaction updating an existing bank account deleting a bank account and transaction

By using this android mobile application the user can easily find all the details of his accounts in various banks in his pocket. He can easily access the details whenever the need arises.

**SYSTEM DESIGN AND DEVELOPMENT**

**3. SYSTEM DESIGN AND DEVELOPMENT**

**3.1 FILE DESIGN**

Once the input data is captured in the system, these may to be preserved either for a short or long period. These data will generally be stored in files in a logical manner. The designer will have to devise the techniques of storing and retrieving data from these files.

**3.2 INPUT DESIGN**

**List Account:**

Displays all the accounts of all the banks along with the current balances of the accounts.

**Add Account:**

Adds a new account to the application with account number, customer number, amount, cheque or cash transaction etc.

**Search Transaction:**

Searches all the transactions of an account when the approximate date and amount is entered

**Recent transactions:**

Displays all the recent transactions which had made to various bank accounts

**Add transaction:**

Adds necessary transaction to the various created bank accounts. The various available transactions are deposit and withdrawal. Only these two can be possible on the basis of cash and cheque.

**About:**

Contacting details about the author are displayed

**3.3 OUTPUT DESIGN**

One of the most important features of a system for users is the output it produces. Output design should improve the system’s relationship with the user and help in decision making. Considering the future use of output required, and depending on the nature, it is displayed on the monitor for immediate need of obtaining the hard copy. The objective of output design is to define the controls and format of all printed documents and of screens that will be produced by the system. Computer output is the most important and direct source of information to the user and the customer receives a corresponding message according to the actions performed in their accounts.

**3.5 SYSTEM DEVELOPMENT**

**3.5.1 DESCRIPTION OF MODULES**

**ADD NEW BANKACCOUNT**

This module provides new account related functionalities. User can create a new account.

**TRANSACTION LIST**

List the transaction is a very simple and easy to use app for your list of transactions related to a bank accounts List of recent 10 transactions from all accounts.

**SEARCH TRANSACTION**

This module is about search transaction account of this portal. We have to search New or transaction account, then add the deposit and withdraw amount.

**Search Transactions by date and amount:**

This module searches the transaction made to a particular account when we enter the approximate amount of the transaction and the known date

**UPDATE OR DELETE TRANSACTION ACCOUNT**

This module provides update or deletes transaction account functionalities showing all details of a single transaction updating an existing bank account deleting a bank account and transaction

By using this android mobile application the user can easily find all the details of his accounts in various banks in his pocket. He can easily access the details whenever the need arises.

**TESTING AND IMPLEMENTATION**

**4. TESTING AND IMPLEMENTATION**

The proposed system is tested at various levels in software testing. Unit, system and user acceptance testing are often performed. This is a gray areas many different opinions exist as to what the stages of testing are and how much if any iteration occurs.

Testing is the process of execution a program with the intent of finding an error and correcting the same error. A good test case is one that has a probability of finding a yet undiscovered error. A successful test is one that answers a yet undiscovered error. Testing is vital to the success of the system. It makes a logical assumption that if all the parts of the system are correct, the goal will be successfully achieved. Testing demonstrates that the software function according to the specifications and the performance requirements has been met. A series of testing are performed for the proposed system before the system is ready for the user acceptance test in which the user’s demands are satisfied. The steps in the system testing can be categorized as Unit Testing, Acceptance Testing, and also the Test Case are included.

**UNIT TESTING**

This testing has tested as per the Unit wise. The user takes all the data’s which includes data’s from the text fields, date, number and tested all use case testing, looping testing, process testing, dollar testing, integer and string testing. This is also known as ‘Module Testing’. In the **Remote System Manager** tool the modules are tested separately. Here in this system includes the unit testing such as command format testing, sms date testing, admin mobile number testing, etc. The local data structures are examined to ensure that data stored temporarily maintains its integrity during all steps in an algorithm execution. Boundary conditions are tested to ensure that the module operates properly at boundaries established to limit or restrict processing. All independent

Paths through the control structures are exercised to ensure that all statement in a module had been executed at least once. Finally all Error-Handling are tested.

**ACCEPTANCE TESTING**

User acceptance testing of a system is the key factor for the success of any system. The system under consideration tested for user acceptance by constantly keeping in touch for the respective system users, at the time of developing and making changes whenever required.

**VALIDATION TEST**

At the culmination of integration testing, software is completely assembled as a package, interfacing errors have been uncovered and corrected, and validation testing begins.

Software Validation is achieved through a series of black box tests that demonstrates conformity with requirements. Black box testing methods focus on the functional requirements of the software.

Black box testing attempts to find errors in the following categories: (i) incorrect or missing functions, (ii) interface errors, (iii) errors in data structures or external database access, (iv) performance errors, and (v) initialization and termination errors.

**IMPLEMENTATION**

Implementation is the stage of the project when the theoretical design is turned out into a working system. Thus it can be considered to be the most critical stage in achieving a successful new system and in giving the user, confidence that the new system will work and be effective. The implementation stage involves careful planning, investigation of the existing system and it’s constraints on implementation, designing of methods to achieve changeover and evaluation of changeover methods.

**CONCLUSION**

**5. CONCLUSION**

The project has been carried out to develop an android smart phone application related to “Bank Account Tracker”. It has been developed to satisfy the needs of the users. The main purpose of the project is to develop a well GUI (Graphical User Interface) **“BANK ACCOUNT TRACKER”** to get details of all the bank accounts.

The user will be pleased by its efficiency performance and will be satisfied with the requirements. It allows user to satisfy their needs

**BIBILIOGRAPHY**

**Reference Text-Books:**

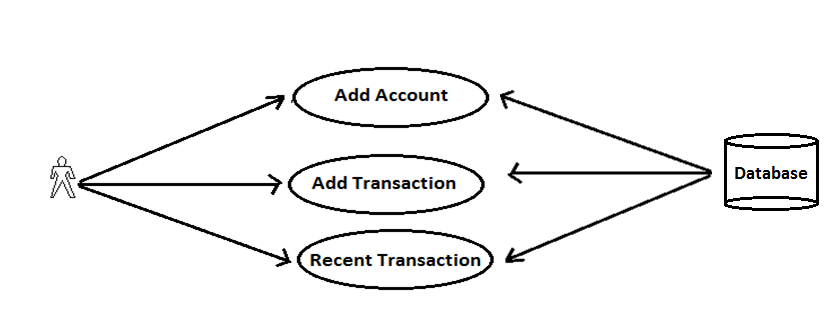
* Android for dummies - Donn Felker
* Android for professionals – Reto Mier
* Eclipse IDE Getting Started

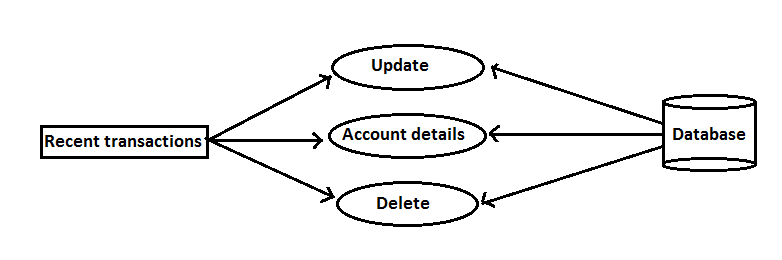
**Websites**

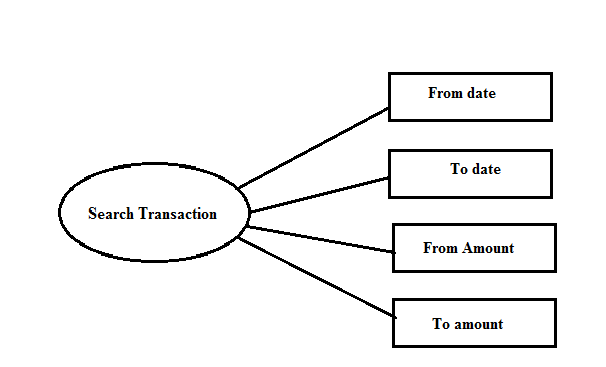
* [www.it-ebooks.com](http://www.it-ebooks.com)
* [www.google.com](http://www.google.com)
* [www.pdfebooks.com](http://www.pdfebooks.com)

**APPENDIX**

**DATA FLOW DIAGRAMS**

****

****

****

**SOURCE CODE**

**JAVA**

**AddAccount.java**

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();

}

}

**AddTransaction.java**

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) {

spinnerAccounts = (Spinner) this.findViewById(R.id.*spinnerAccounts*);

Database.*populateAccounts*(spinnerAccounts);

}

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

**Database.java :**

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

}

**XML**

**Add\_transaction.xml**

<?xml version=*"1.0"* encoding=*"utf-8"*?>

<ScrollView xmlns:android=*"http://schemas.android.com/apk/res/android"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"match\_parent"*

android:background=*"@drawable/windows"* >

<TableLayout

android:layout\_width=*"match\_parent"*

android:layout\_height=*"match\_parent"*>

<TableRow >

<TextView

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:textSize=*"30dp"*

android:textColor=*"#FFFF00"*

android:text=*"Account :"* />

<Spinner

android:id=*"@+id/spinnerAccounts"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:prompt=*"@string/accountsprompt"* >

</Spinner>

</TableRow>

<TableRow >

<TextView

android:layout\_width=*"match\_parent"*

android:layout\_height=*"wrap\_content"*

android:textSize=*"30dp"*

android:textColor=*"#FFFF00"*

android:text=*"Type"* />

<RadioGroup

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:orientation=*"horizontal"* >

<RadioButton

android:id=*"@+id/radioDeposit"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:checked=*"false"*

android:textSize=*"30dp"*

android:textColor=*"#000000"*

android:text=*"Deposit"* />

<RadioButton

android:id=*"@+id/radioWithdraw"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:textSize=*"30dp"*

android:textColor=*"#000000"*

android:text=*"Withdraw"* />

</RadioGroup>

</TableRow>

<TableRow >

<TextView

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:textSize=*"30dp"*

android:textColor=*"#FFFF00"*

android:text=*"Date :"* />

<LinearLayout

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:padding=*"8dp"*

android:layout\_marginBottom=*"8dp"* >

<TextView

android:id=*"@+id/textTransDate"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:textSize=*"30dp"*

android:textColor=*"#FFFF00"*

android:text=*"sysdate"* />

<ImageButton

android:id=*"@+id/buttonDateDialog"*

android:layout\_marginLeft=*"10dp"*

android:layout\_width=*"20dp"*

android:layout\_height=*"20dp"*

android:src=*"@drawable/datepicker"*

android:onClick=*"showDateDialog"* >

</ImageButton>

</LinearLayout>

</TableRow>

<TableRow >

<TextView

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:textSize=*"30dp"*

android:textColor=*"#FFFF00"*

android:text=*"Amount :"* />

<EditText

android:id=*"@+id/editTransAmount"*

android:layout\_width=*"120dp"*

android:layout\_height=*"wrap\_content"*

android:inputType=*"numberDecimal"* >

</EditText>

</TableRow>

<TableRow >

<TextView

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:textSize=*"30dp"*

android:textColor=*"#FFFF00"*

android:text=*"Cheque No."* />

<EditText

android:id=*"@+id/editChequeNo"*

android:layout\_width=*"120dp"*

android:layout\_height=*"wrap\_content"*

android:inputType=*"number"* >

</EditText>

</TableRow>

<TableRow >

<TextView

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:textSize=*"30dp"*

android:textColor=*"#FFFF00"*

android:text=*"Cheque Party :"* />

<EditText

android:id=*"@+id/editChequeParty"*

android:layout\_width=*"120dp"*

android:layout\_height=*"wrap\_content"*

android:inputType=*"text"* >

</EditText>

</TableRow>

<TableRow >

<TextView

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:textSize=*"30dp"*

android:textColor=*"#FFFF00"*

android:text=*"Cheque Details :"* />

<EditText

android:id=*"@+id/editChequeDetails"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"wrap\_content"*

android:inputType=*"text"* >

</EditText>

</TableRow>

<TableRow >

<TextView

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:textSize=*"30dp"*

android:textColor=*"#FFFF00"*

android:text=*"Remarks "* />

<EditText

android:id=*"@+id/editRemarks"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"wrap\_content"*

android:layout\_weight=*"1"*

android:inputType=*"text"* >

</EditText>

</TableRow>

<TableRow >

<Button

android:id=*"@+id/buttonAdd"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"60dp"*

android:onClick=*"addTransaction"*

android:textSize=*"30dp"*

android:textColor=*"#000000"*

android:text=*"Add Transaction"* >

</Button>

</TableRow>

</TableLayout>

</ScrollView>

**Add\_account.xml**

<?xml version=*"1.0"* encoding=*"utf-8"*?>

<ScrollView xmlns:android=*"http://schemas.android.com/apk/res/android"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"match\_parent"*

android:background=*"@drawable/back"*

android:gravity=*"center"*>

<TableLayout

android:layout\_width=*"match\_parent"*

android:layout\_height=*"match\_parent"*>

<TableRow>

<TextView

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:textSize=*"30sp"*

android:textColor=*"#FFFF00"*

android:text=*"Account Number"* />

<EditText

android:id=*"@+id/editAcno"*

android:layout\_width=*"150dp"*

android:layout\_height=*"wrap\_content"*

android:inputType=*"number"*>

<requestFocus />

</EditText>

</TableRow>

<TableRow>

<TextView

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:textSize=*"30sp"*

android:textColor=*"#FFFF00"*

android:text=*"Customer Number"* />

<EditText

android:id=*"@+id/editCno"*

android:layout\_width=*"150dp"*

android:layout\_height=*"wrap\_content"*

android:inputType=*"number"* >

</EditText>

</TableRow>

<TableRow>

<TextView

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:textSize=*"30sp"*

android:textColor=*"#FFFF00"*

android:text=*"Account Holder(s)"* />

<EditText

android:id=*"@+id/editHolders"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"wrap\_content"*

android:inputType=*"textPersonName"*

android:layout\_weight=*"1"* >

</EditText>

</TableRow>

<TableRow>

<TextView

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:textSize=*"30sp"*

android:textColor=*"#FFFF00"*

android:text=*"Bank Name "* />

<EditText

android:id=*"@+id/editBankName"*

android:layout\_width=*"150dp"*

android:layout\_height=*"wrap\_content"*

android:inputType=*"textPersonName"* >

</EditText>

</TableRow>

<TableRow>

<TextView

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:textSize=*"30sp"*

android:textColor=*"#FFFF00"*

android:text=*"Branch Name "* />

<EditText

android:id=*"@+id/editBranchName"*

android:layout\_width=*"150dp"*

android:layout\_height=*"wrap\_content"*

android:inputType=*"textPersonName"* >

</EditText>

</TableRow>

<TableRow>

<TextView

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:textSize=*"30sp"*

android:textColor=*"#FFFF00"*

android:text=*"Branch Address "* />

<EditText

android:id=*"@+id/editAddress"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"wrap\_content"*

android:inputType=*"textPostalAddress"*

android:layout\_weight=*"1"* />

</TableRow>

<TableRow>

<TextView

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:textSize=*"30sp"*

android:textColor=*"#FFFF00"*

android:text=*"IFSC"* />

<EditText

android:id=*"@+id/editIFSC"*

android:layout\_width=*"150dp"*

android:layout\_height=*"wrap\_content"*

android:inputType=*"number"* >

</EditText>

</TableRow>

<TableRow>

<TextView

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:textSize=*"30sp"*

android:textColor=*"#FFFF00"*

android:text=*"MICR"* />

<EditText

android:id=*"@+id/editMICR"*

android:layout\_width=*"150dp"*

android:layout\_height=*"wrap\_content"*

android:inputType=*"number"* >

</EditText>

</TableRow>

<TableRow>

<TextView

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:textSize=*"30sp"*

android:textColor=*"#FFFF00"*

android:text=*"Current Balance"* />

<EditText

android:id=*"@+id/editBalance"*

android:layout\_width=*"150dp"*

android:layout\_height=*"wrap\_content"*

android:inputType=*"numberDecimal"* >

</EditText>

</TableRow>

<TableRow>

<TextView

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:textSize=*"30sp"*

android:textColor=*"#FFFF00"*

android:text=*"Remarks "* />

<EditText

android:id=*"@+id/editRemarks"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"wrap\_content"*

android:layout\_weight=*"1"* >

</EditText>

</TableRow>

<TableRow>

<Button

android:id=*"@+id/buttonAdd"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:onClick=*"addAccount"*

android:textSize=*"30sp"*

android:textColor=*"#000000"*

android:text=*"Add Account"* >

</Button>

</TableRow>

</TableLayout>

</ScrollView>

**Search\_transaction.xml**

<?xml version=*"1.0"* encoding=*"utf-8"*?>

<LinearLayout xmlns:android=*"http://schemas.android.com/apk/res/android"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"match\_parent"*

android:orientation=*"vertical"* >

<TableLayout

android:layout\_width=*"match\_parent"*

android:layout\_height=*"wrap\_content"*>

<TableRow >

<TextView

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:text=*"From Date :"*

android:textColor=*"#008000"*

android:textSize=*"30sp"* />

<EditText

android:id=*"@+id/editFromDate"*

android:layout\_width=*"120dip"*

android:layout\_height=*"wrap\_content"*

android:inputType=*"date"* >

<requestFocus />

</EditText>

<ImageButton

android:id=*"@+id/buttonFromDateDialog"*

android:layout\_marginLeft=*"5dp"*

android:onClick=*"showFromDateDialog"*

android:src=*"@drawable/datepicker"* />

</TableRow>

<TableRow>

<TextView

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:text=*"To Date :"*

android:textColor=*"#008000"*

android:textSize=*"30sp"* />

<EditText

android:id=*"@+id/editToDate"*

android:layout\_width=*"120dip"*

android:layout\_height=*"wrap\_content"*

android:inputType=*"date"* />

<ImageButton

android:id=*"@+id/buttonToDateDialog"*

android:layout\_marginLeft=*"5dp"*

android:onClick=*"showToDateDialog"*

android:src=*"@drawable/datepicker"* />

</TableRow>

<TableRow >

<TextView

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:text=*"From Amount:"*

android:textColor=*"#008000"*

android:textSize=*"30sp"* />

<EditText

android:id=*"@+id/editFromAmount"*

android:layout\_width=*"120dip"*

android:layout\_height=*"wrap\_content"*

android:inputType=*"numberDecimal"* />

</TableRow>

<TableRow>

<TextView

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:text=*"To Amount :"*

android:textColor=*"#008000"*

android:textSize=*"30sp"*/>

<EditText

android:id=*"@+id/editToAmount"*

android:layout\_width=*"120dip"*

android:layout\_height=*"wrap\_content"*

android:inputType=*"numberDecimal"* />

</TableRow>

</TableLayout>

<LinearLayout android:layout\_width=*"match\_parent"*

android:layout\_height=*"wrap\_content"*

android:orientation=*"horizontal"*>

<Button

android:id=*"@+id/btnSearch"*

android:layout\_width=*"80dip"*

android:layout\_height=*"wrap\_content"*

android:onClick=*"searchTransactions"*

android:text=*"Search"*

android:textColor=*"#000000"*

android:textSize=*"30sp"* >

</Button>

<Button

android:id=*"@+id/btnClear"*

android:layout\_width=*"80dip"*

android:layout\_height=*"wrap\_content"*

android:text=*"Clear"*

android:textSize=*"30sp"*

android:textColor=*"#000000"*

android:onClick=*"clearFields"* />

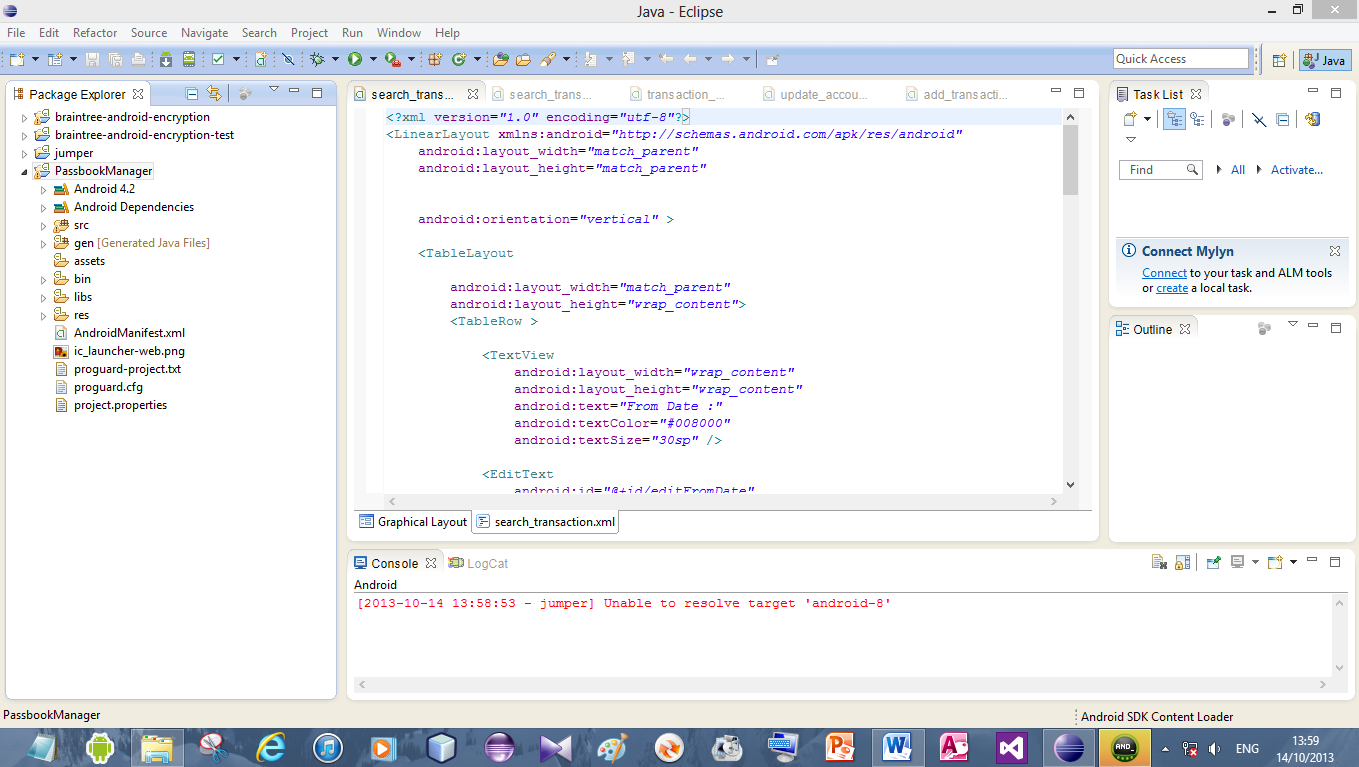
</LinearLayout>

</LinearLayout>

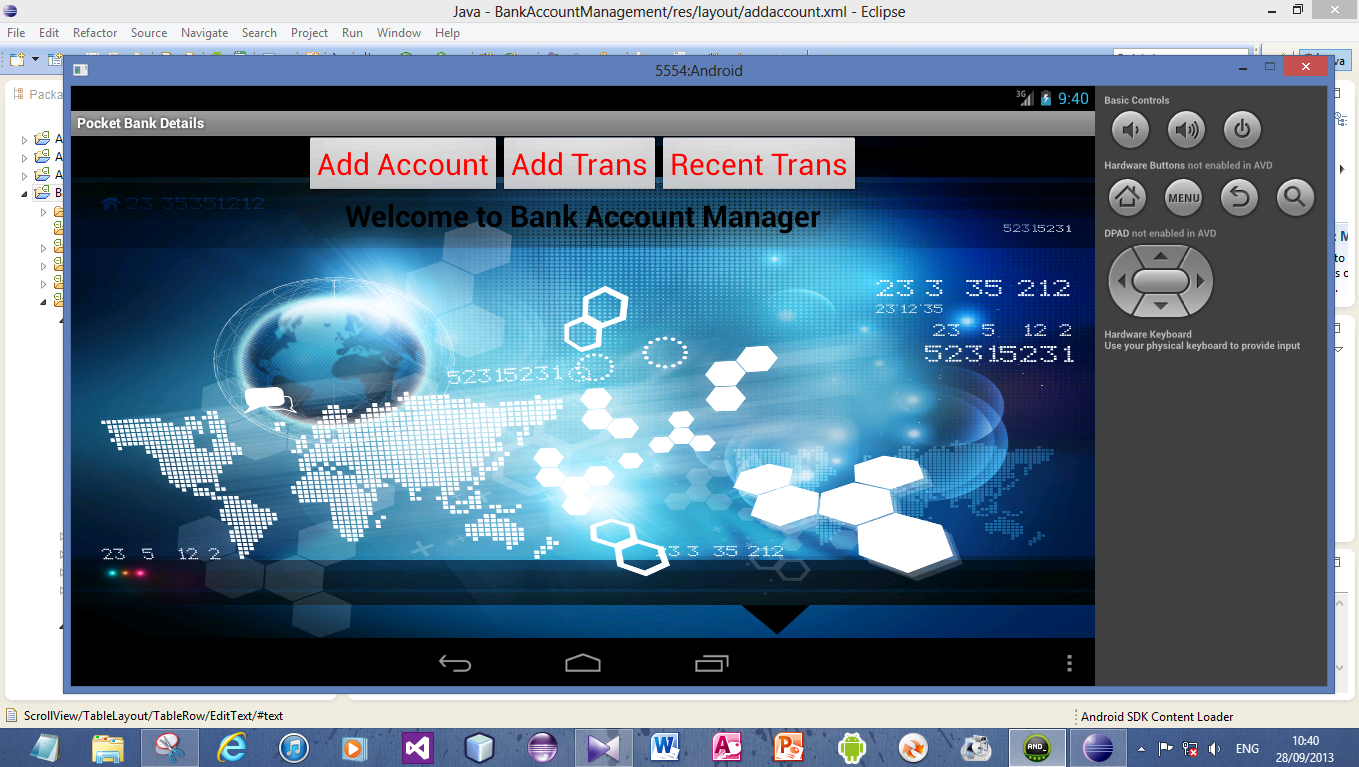
**SCREENSHOTS**

**SCREENSHOTS**

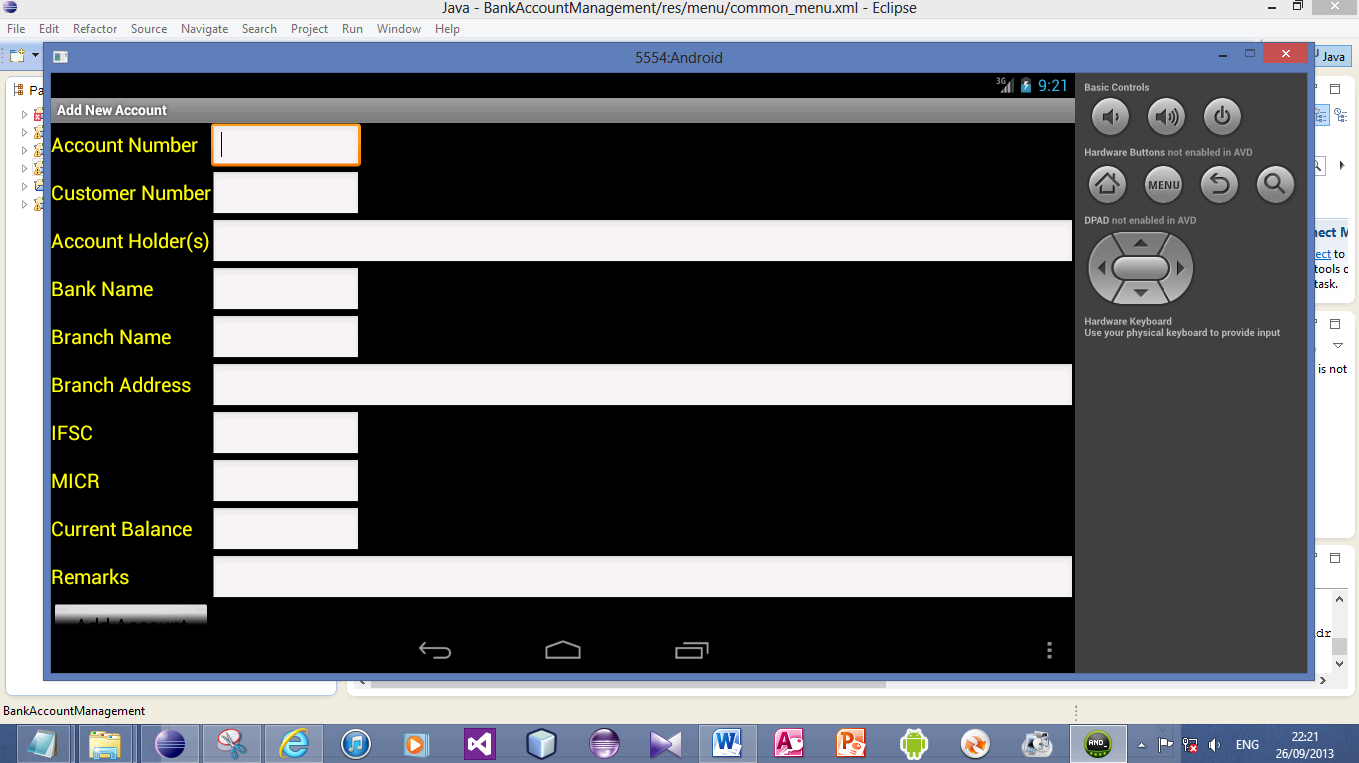
**Eclipse IDE:**

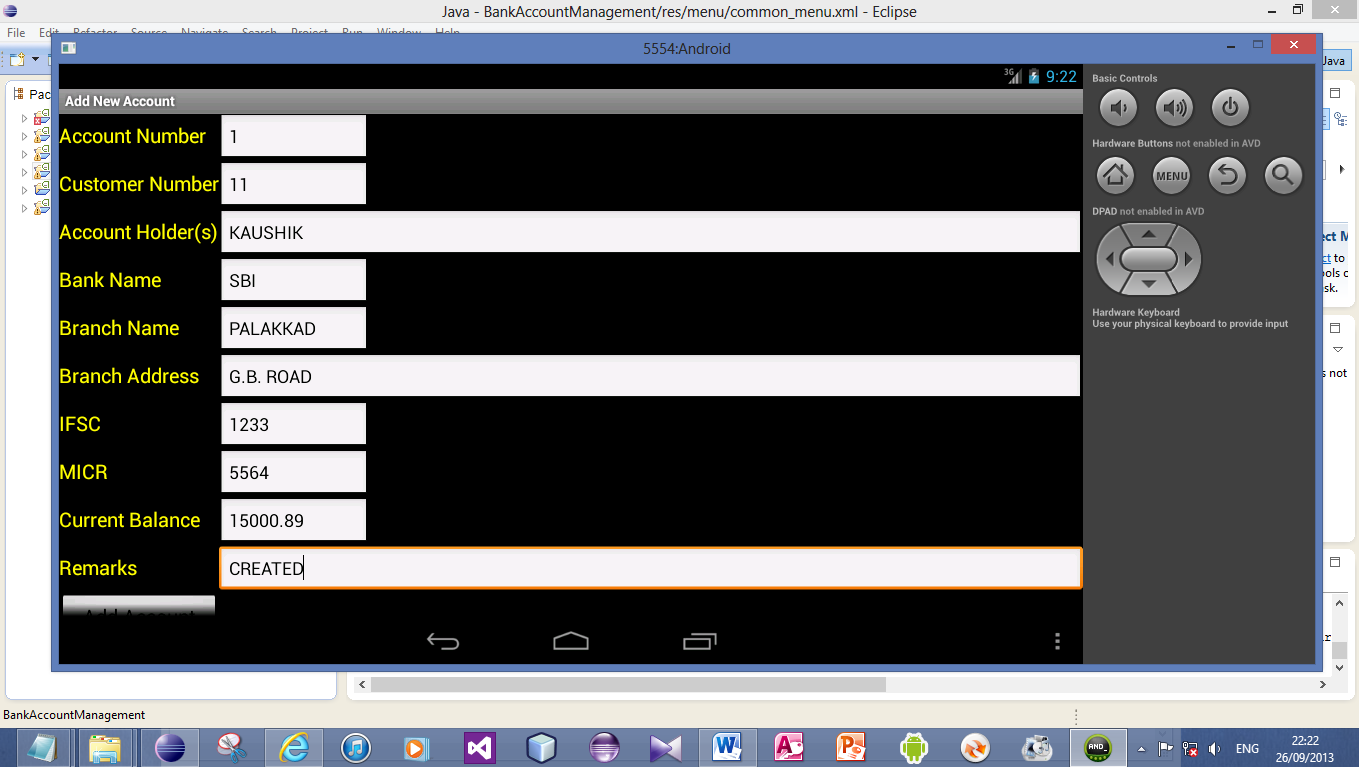
****

**List Accounts:**

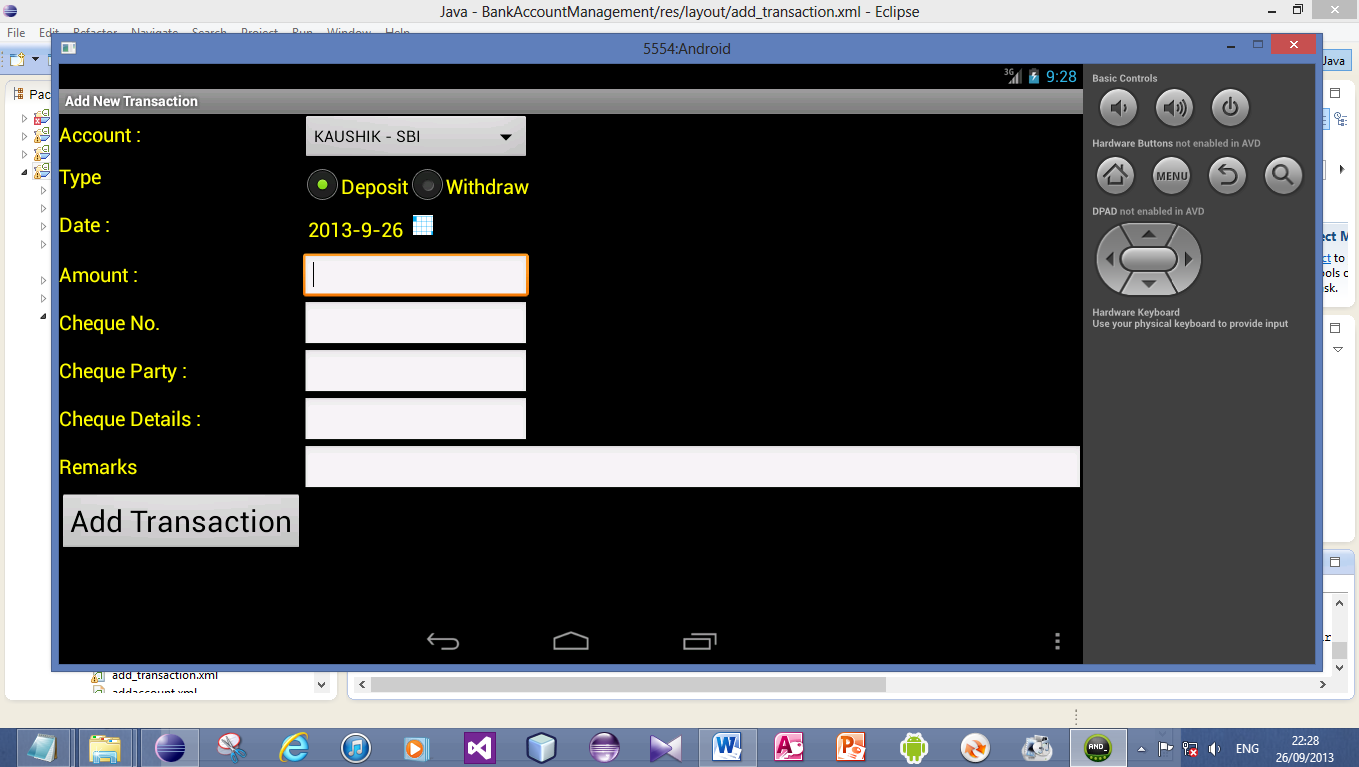
****

**Add Account:**

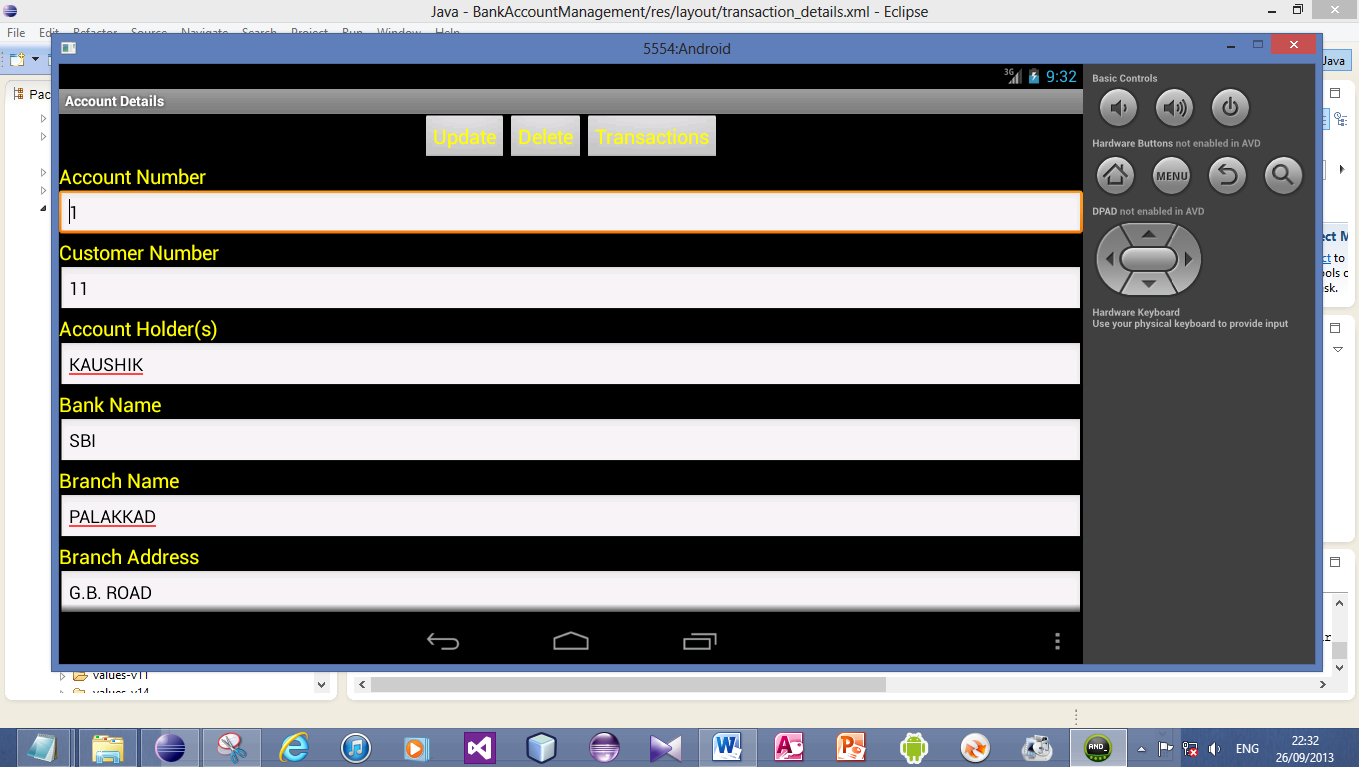
****

****

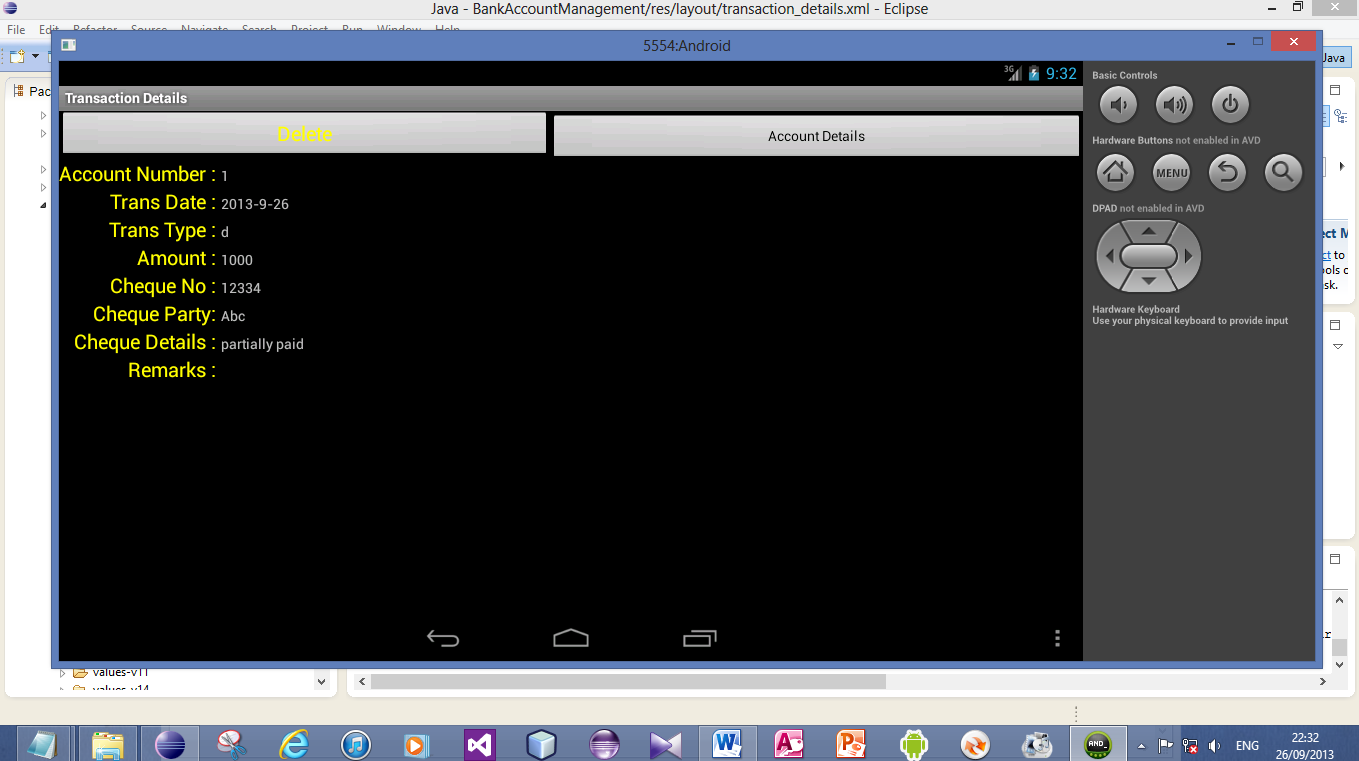
**Add Transactions:**

****

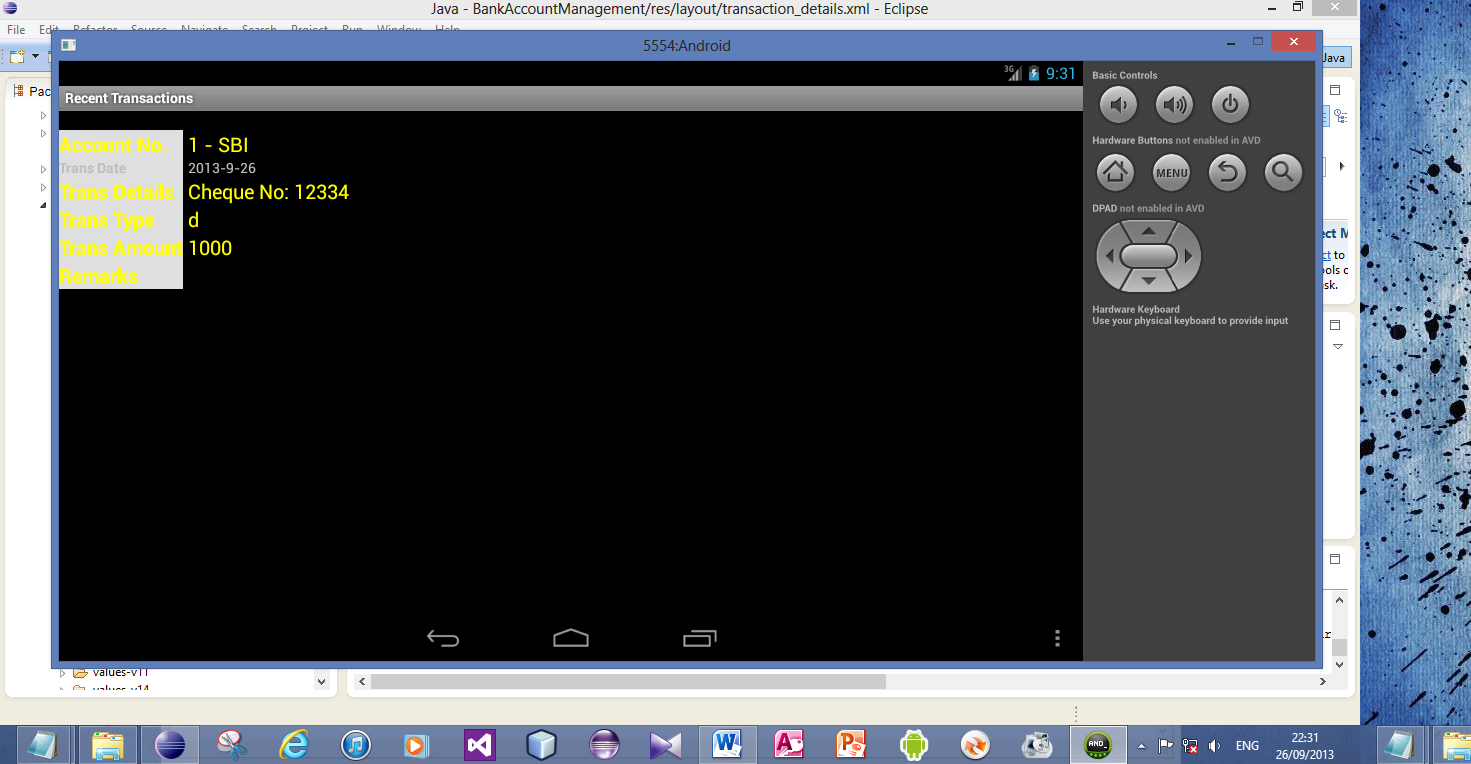
**Account Details:**

****

**Transaction Details:**

****

**Recent Transactions:**

****