بسم الله الرحمن الرحيم

ISLAMIC UNIVERSITY OF TECHNOLOGY (IUT)

THE ORGANIZATION OF ISLAMIC COOPERATION (OIC)

Personal Expense Assistant Management: An Android Based Application

By:

Moussa Sali (134308)

Abdel Salam Abbo (134310)

Supervised By:

Mr. Ashraful Alam Khan

Lecturer

Department of Computer Science and Engineering (CSE), IUT

Gazipur-1704, Dhaka, Bangladesh

November 2016

DECLARATION OF AUTHORSHIP

We, Moussa Sali and Abdel Salam Abbo, declare that this project entitled as "Personal Expense Assistant Management: An Android Based Application" and all its related documentations presented here are our own work. We hereby confirm that:

- This work is a subject of our Higher Diploma degree Completion and has not been submitted elsewhere for the award of another degree.

- All external resources mentioned here were carefully verified and authorized by their owners.

Submitted by:

_____

Moussa Sali (134308)

_____

Abdel Salam Abbo (134310)

Personal Expense Assistant Management (PAM): An android Based Application

Approved by:

Prof. Dr. M. A. Mottalib

Head of the Department,

Computer Science and Engineering,

Islamic University of Technology

Mr. Ashraful Alam Khan

Project Supervisor

Lecturer,

Department of Computer Science and Engineering,

Islamic University of Technology

ABSTRACT

Personal Expense Assistant Management is an application aiming to manage our daily expenses in a more efficient and manageable way. The application attempts to free the user with as much as possible the burden of manual calculation and to keep the track of his expenditure. Instead of keeping a dairy or a log of the expenses on the smartphones or laptops, it enables the user to not just keep the tab on the expenses but also to plan ahead keeping the past budget in mind. With the help of this application, a user may be able to add, delete or change the current entered bill entry efficiently. The graphical representation of the budget is the lucrative part of the system as it appeals the user more and is easy to understand and incorporate for future planning. The user interface of the system ticks the boxes of consistency, easy readable dialogue boxes, easy exit and easy to get used to requirements for any ideal user interface.

ACKNOWLEDGMENTS

The successful completion of this thesis results from the kind supports of many individuals. We would like to extend our sincere thanks to all of them.

We are grateful to **Prof Dr. M. A. Mottalib,** Head of the Department of Computer Science and Engineering for his incredible support towards the completion of our simulation.

We are indebted to **Mr. Ashraful Alam Khan**, Lecturer, CSE Department, IUT whose expertise, understanding, generous guidance and support enabled us to successfully work on a topic that was of great interest to us. His trust and support inspired us in the most important moments of making right decisions and for that we are happy to work with him.

Lastly but not the least, we would like to thank our parents, who taught us the value of hard work by their own example.

TABLE OF CONTENTS

The Page is left blank

CHAPTER 1: INTRODUCTION

At the instant, there is no as such complete solution present easily or we should say free of cost which enables a person to keep a track of his/her daily expenditure easily. To do so a person has to keep a log in a diary or in a computer, also all the calculations need to be done by the user which may sometimes result in errors leading to losses. Due to the lack of a complete tracking system, there is a constant overload to rely on the daily entry of the expenditure and total estimation till the end of the month.

In an effort to fix the above addressed problems, we tried to design a system that would make the task of keeping the expenses in check, efficient and a delight one. This system will include an android application that will allow users to maintain a digital automated diary. The user can add his/her transaction details. The user can also add the information about the loans he/she make. The option to attach the information about his/her loans helps him/her to remember when, where and with whom the payment operation was made. As soon as the entry is made about the transaction, the database is updated and according to the nature of the bill deduction or addition to the total balance in the user's pocket is made. In order to make the user aware about the average rate of the expenses, an alluring graphical statistics are also provided. In addition to the daily tracking of the expenditure, it is also beneficial to have a quick access to the past records of the expense habit; keeping in mind there is also an option of monthly budget to provide a quick glance of the past spending.

CHAPTER 2: MOTIVATION

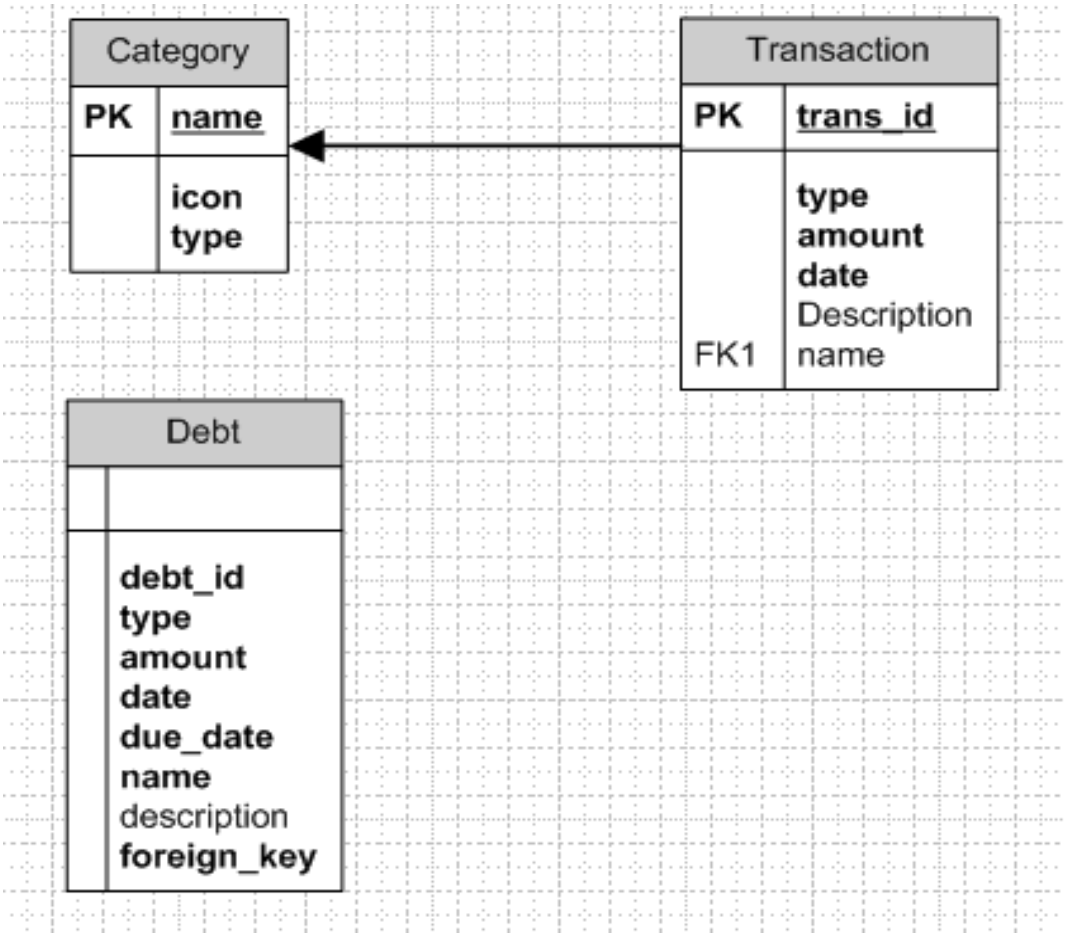
The main motivation of this application is to learn mobile application development.

Design a system for keeping good track of our daily expenses.

Allow users to manage their debts.

CHAPTER 3: REQUIREMENTS ANALYSIS

ER Diagram

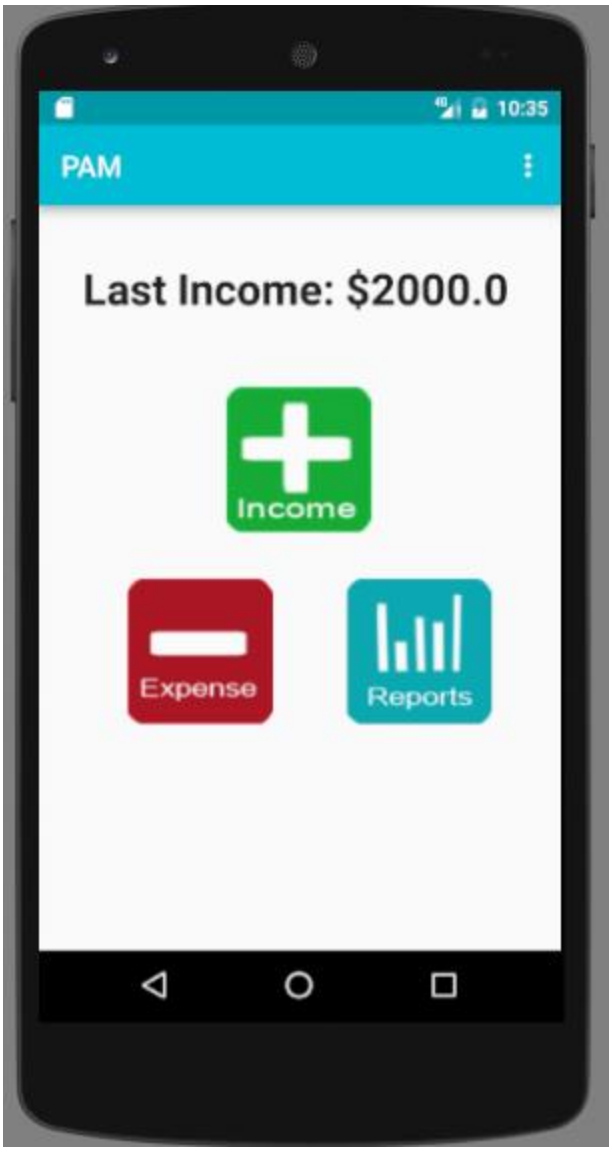


CHAPTER 4: SCREEN SHOTS AND MODULE DESCRIPTIONS

4.1 Main menu

```
package com.example.musa.pam;

import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.TextView;
import android.widget.Toast;

import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Locale;

public class MainActivity extends AppCompatActivity {
    private TextView notification;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        notification = (TextView) findViewById(R.id.message);
        notification.setSelected(true);
        populateMessage();
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.home_menu, menu);
        return super.onCreateOptionsMenu(menu);
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        Intent intent;
        switch (item.getItemId()) {
            case R.id.menu_category:
                intent = new Intent(this, CategoryDisplay.class);
                startActivity(intent);
                break;
            case R.id.menu_transaction:
                intent = new Intent(this, HistoryDisplay.class);
                startActivity(intent);
                break;
            case R.id.menu_debt:
                intent = new Intent(this, DebtDisplay.class);
                startActivity(intent);
                break;
        }
        return super.onOptionsItemSelected(item);
    }
```

```
    @Override
    protected void onResume() {
        super.onResume();
        populateMessage();
    }

    public void onIncome(View view) {
        Intent incomeIntent = new Intent(this, TransactionActivity.class);
        incomeIntent.putExtra(TableData.category.TYPE, 1);
        incomeIntent.putExtra("parent", "Main");
        startActivity(incomeIntent);
    }

    public void onExpense(View view) {
        Intent incomeIntent = new Intent(this, TransactionActivity.class);
        incomeIntent.putExtra(TableData.category.TYPE, 0);
        incomeIntent.putExtra("parent", "main");
        startActivity(incomeIntent);
    }

    public void onReports(View view) {
        Intent chartIntent = new Intent(this, chartActivity.class);
        startActivity(chartIntent);
    }

    protected void populateMessage() {
        DatabaseOperations databaseOperations = new DatabaseOperations(this);
        Double lastIncome = databaseOperations.getLastIncome();
        Double lastExpense = databaseOperations.getLastExpense();
        Double balance = databaseOperations.getBalance();
        notification.setText("Last Income: $"+lastIncome + "        Last Expense: $"+
lastExpense +
                "         Total Balance: $"+balance);
        databaseOperations.close();
    }
}
```


10:35
PAM
Last Income: $2000.0
Income
Expense
Reports

The user is notified his last transaction details along with his current balance.

He can also insert new transaction or view the summary/reports of his expenses.

A manage menu allows him to use more features.

4.2 Income

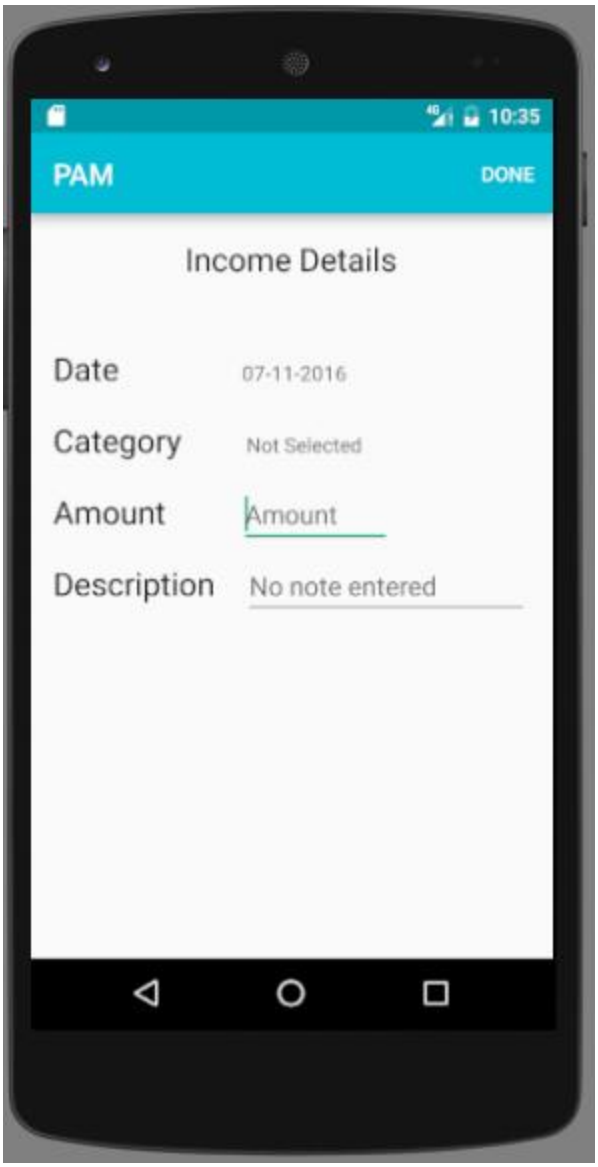
```
package com.example.musa.pam;

import android.app.Activity;
import android.app.DatePickerDialog;
import android.content.Context;
import android.content.Intent;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.text.InputType;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.DatePicker;
import android.widget.EditText;
import android.widget.RadioButton;
import android.widget.RadioGroup;
import android.widget.TextView;
import android.widget.Toast;

import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Locale;

public class TransactionActivity extends AppCompatActivity implements
View.OnClickListener {

    public static final int CATEGORY_REQUEST_CODE = 1001;
    //UI References
    private TextView fromDateEtxt;
    private TextView categoryTview;
    private TextView transactionDetails;
    private EditText amountEText;
    private EditText descriptionEText;
    private DatePickerDialog fromDatePickerDialog;
    private SimpleDateFormat dateFormatter;
    private int type;
    private int tran_id = 0;
    private String date = null;
    private String catname = null;
    private double amount = 0;
    private String description = null;
    private String method = null;
    private String parent = null;
    private Button deleteBtn;
    private RadioGroup radioGroup;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_transaction);

        type = getIntent().getIntExtra(TableData.category.TYPE, 0);
        dateFormatter = new SimpleDateFormat("dd-MM-yyyy", Locale.US);

        findViewsById();
        initViews();
    }
```

```
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.done_menu, menu);
        return super.onCreateOptionsMenu(menu);
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        if (item.getItemId() == R.id.menu_transaction_done) {
            if (isCategoryValid() && isAmountValid()) {
                if ((("Insert".equalsIgnoreCase(method)) ||
("Main".equalsIgnoreCase(parent)))) {
                    if (("Insert".equalsIgnoreCase(method))) {
                        setType();
                    }
                    Insert();
                } else {
                    Update();
                }
            } else {
                if (!isCategoryValid() && !isAmountValid()) {
                    Toast.makeText(TransactionActivity.this, "Category and Amount must
not be empty!", Toast.LENGTH_LONG).show();
                } else if (!isCategoryValid()) {
                    Toast.makeText(TransactionActivity.this, "Category must be
selected", Toast.LENGTH_LONG).show();
                } else {
                    Toast.makeText(TransactionActivity.this, "Amount must be entered",
Toast.LENGTH_LONG).show();
                }
            }
        }
        return super.onOptionsItemSelected(item);
    }

    @Override
    public void onClick(View view) {
        if (view == fromDateEtxt) {
            fromDatePickerDialog.show();
        }
    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        if (requestCode == CATEGORY_REQUEST_CODE) {
            if (resultCode == RESULT_OK) {
                categoryTview.setText(data.getStringExtra("categoryName"));
            }
        }
    }

    public void initViews() {
        method = getIntent().getStringExtra("method");
        parent = getIntent().getStringExtra("parent");
        type = getIntent().getIntExtra(TableData.transaction.TYPE, 0);
        if(type == 0){
            transactionDetails.setText("Expense Details");
        } else{
            transactionDetails.setText("Income Details");
        }
```

```
        if ((("Update".equalsIgnoreCase(method)) ||
("Main".equalsIgnoreCase(parent)))) {
            radioGroup.setVisibility(View.INVISIBLE);
        }

        if ((("Insert".equalsIgnoreCase(method)) ||
("Main".equalsIgnoreCase(parent)))) {
            deleteBtn.setVisibility(View.INVISIBLE);
            setDefaultDate();
            setDateTimeField();
        } else {
            tran_id = getIntent().getIntExtra(TableData.transaction.TRAN_ID, 0);
            date = getIntent().getStringExtra(TableData.transaction.DATE);
            catname = getIntent().getStringExtra(TableData.transaction.CAT_NAME);
            amount = getIntent().getDoubleExtra(TableData.transaction.AMOUNT, 0);
            description =
getIntent().getStringExtra(TableData.transaction.DESCRIPTION);

            fromDateEtxt.setText(date);
            categoryTview.setText(catname);
            amountEText.setText("$ " + Double.toString(amount));
            descriptionEText.setText(description);
        }
    }

    private void findViewsById() {
        fromDateEtxt = (TextView) findViewById(R.id.date);
        fromDateEtxt.setInputType(InputType.TYPE_NULL);
        fromDateEtxt.requestFocus();

        transactionDetails = (TextView) findViewById(R.id.transactionDetails);
        radioGroup = (RadioGroup) findViewById(R.id.typeRbtn);
        categoryTview = (TextView) findViewById(R.id.category);
        amountEText = (EditText) findViewById(R.id.amount);
        descriptionEText = (EditText) findViewById(R.id.description);
        deleteBtn = (Button) findViewById(R.id.btndelete);
    }

    private void setDefaultDate() {
        Calendar newDate = Calendar.getInstance();
        Calendar newCalendar = Calendar.getInstance();
        newDate.set(newCalendar.get(Calendar.YEAR), newCalendar.get(Calendar.MONTH),
newCalendar.get(Calendar.DAY_OF_MONTH));
        fromDateEtxt.setText(dateFormatter.format(newDate.getTime()));
    }

    public void setType() {
        RadioButton expRb = (RadioButton) findViewById(R.id.expenseRBtn);
        // Is the button now checked?
        boolean checked = ((RadioButton) expRb).isChecked();

        // Check which radio button was clicked
        if (checked)
            type = 0;
        else type = 1;
    }

    public void onRadioButtonClicked(View view) {
        // Is the button now checked?
        boolean checked = ((RadioButton) view).isChecked();
```

```
        // Check which radio button was clicked
        switch (view.getId()) {
            case R.id.expenseRBtn:
                if (checked)
                    type = 0;
                break;
            case R.id.incomeRBtn:
                if (checked)
                    type = 1;
                break;
        }
    }

    private void setDateTimeField() {
        fromDateEtxt.setOnClickListener(this);

        Calendar newCalendar = Calendar.getInstance();
        fromDatePickerDialog = new DatePickerDialog(this, new
DatePickerDialog.OnDateSetListener() {
            Calendar newDate = Calendar.getInstance();

            public void onDateSet(DatePicker view, int year, int monthOfYear, int
dayOfMonth) {
                newDate.set(year, monthOfYear, dayOfMonth);
                fromDateEtxt.setText(dateFormatter.format(newDate.getTime()));
            }

        }, newCalendar.get(Calendar.YEAR), newCalendar.get(Calendar.MONTH),
newCalendar.get(Calendar.DAY_OF_MONTH));
    }

    public void categoryHandler(View view) {
        Intent incomeIntent = new Intent(this, CategoryActivity.class);
        incomeIntent.putExtra(TableData.category.TYPE, type);
        startActivityForResult(incomeIntent, CATEGORY_REQUEST_CODE);
    }

    public boolean isCategoryValid() {
        if (categoryTview.getText().toString().trim().isEmpty()) {
            return false;
        } else {
            return true;
        }
    }

    public boolean isAmountValid() {
        if (amountEText.getText().toString().trim().isEmpty()) {
            return false;
        } else {
            return true;
        }
    }
```

```
    public void Insert() {
        DatabaseOperations databaseOperations = new DatabaseOperations(this);
        Transaction transaction = new Transaction();
        transaction.setType(type);
        transaction.setAmount(Integer.parseInt(amountEText.getText().toString()));
        transaction.setDate(fromDateEtxt.getText().toString());
        transaction.setCat_name(categoryTview.getText().toString());
        transaction.setDescription(descriptionEText.getText().toString());
        boolean success = databaseOperations.insertTransaction(transaction);

        if (success) {
            if(type == 0){
                Toast.makeText(TransactionActivity.this, "New Expense added!",
Toast.LENGTH_LONG).show();
            }else{
                Toast.makeText(TransactionActivity.this, "New Income added!",
Toast.LENGTH_LONG).show();
            }
        } else {
            Toast.makeText(TransactionActivity.this, "Error!",
Toast.LENGTH_LONG).show();
        }
        finish();
    }

    private void Update() {
        DatabaseOperations databaseOperations = new DatabaseOperations(this);
        databaseOperations.updateTransaction(tran_id,
fromDateEtxt.getText().toString(), categoryTview.getText().toString(),
Double.parseDouble(amountEText.getText().toString().replace("$ ", "")),
descriptionEText.getText().toString());
        Toast.makeText(TransactionActivity.this, "One transaction edited!",
Toast.LENGTH_SHORT).show();
        databaseOperations.close();
        finish();
    }

    public void onDelete(View view) {
        DatabaseOperations databaseOperations = new DatabaseOperations(this);
        databaseOperations.DeleteTranscation(tran_id);
        Toast.makeText(TransactionActivity.this, "One transaction deleted!",
Toast.LENGTH_LONG).show();
        databaseOperations.close();
        finish();
    }
}
```



The user provides the details of his income.

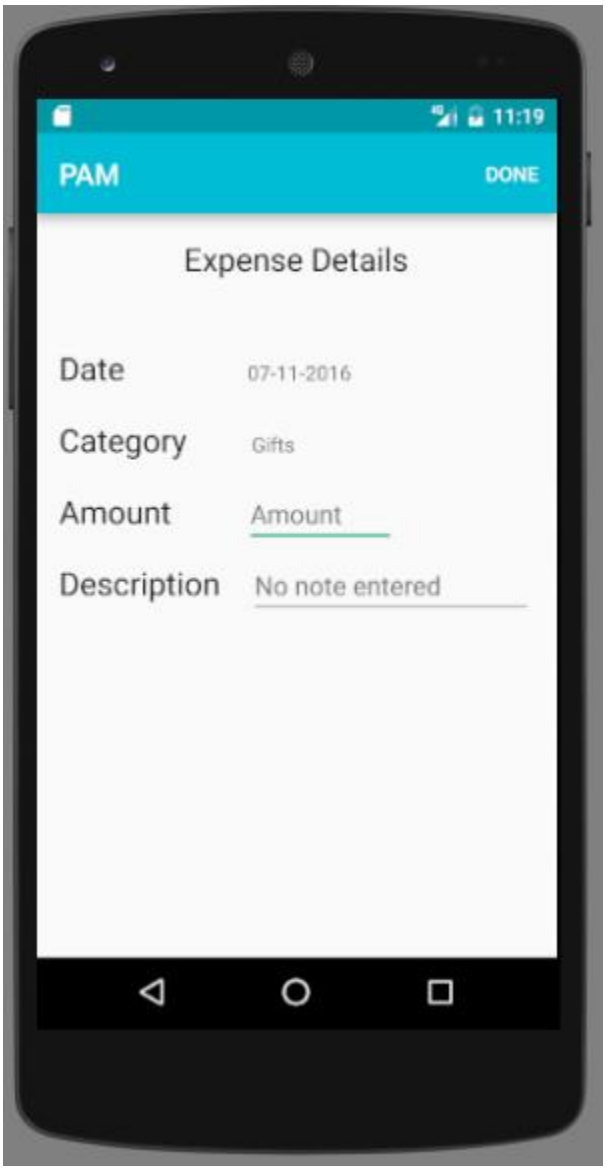
He needs to enter:

the date

the category

the amount the description

4.3 Expense



The user provides the details of his expense.

He needs to enter:

The date

The category

The amount and the description

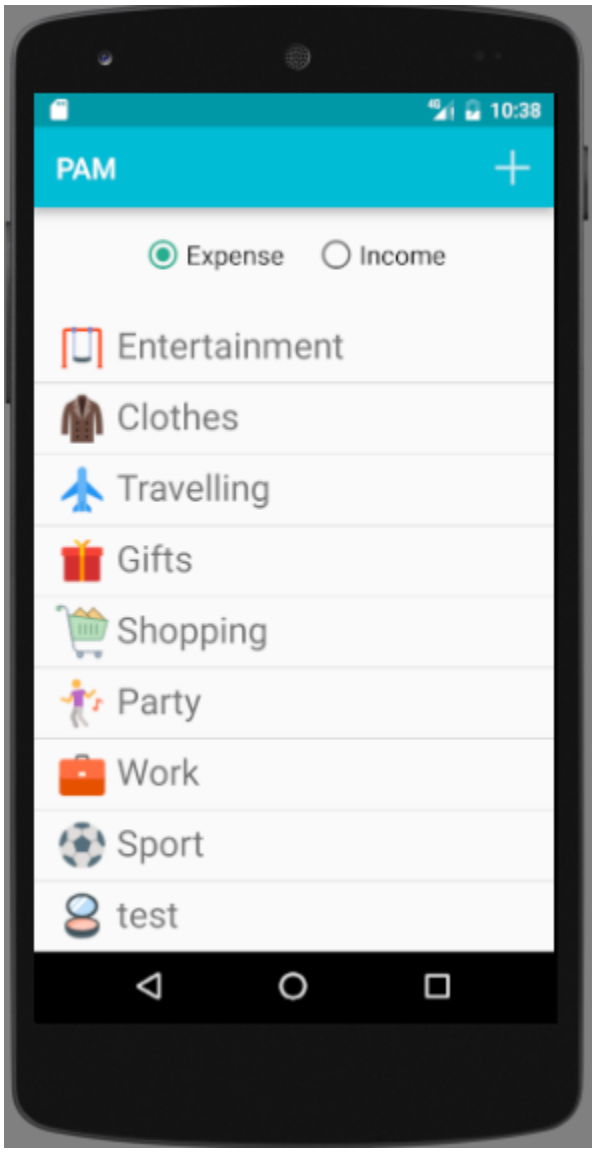
4.4 Reports

```
package com.example.musa.pam;

import android.graphics.Color;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.MenuItem;
import android.view.View;
import android.widget.LinearLayout;
import android.widget.TextView;
import android.widget.Toast;

import com.github.mikephil.charting.charts.BarChart;
import com.github.mikephil.charting.data.BarData;
import com.github.mikephil.charting.data.BarDataSet;
import com.github.mikephil.charting.data.BarEntry;
import com.github.mikephil.charting.utils.ColorTemplate;

import com.github.mikephil.charting.components.AxisBase;
import com.github.mikephil.charting.data.Entry;
import com.github.mikephil.charting.utils.ViewPortHandler;

import java.text.DecimalFormat;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;

public class chartActivity extends AppCompatActivity {
    LinearLayout la;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_chart);

        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        la = (LinearLayout) findViewById(R.id.chart);

        drawChart();

    }

    private void drawChart() {

        DatabaseOperations databaseOperations = new DatabaseOperations(this);
        ArrayList<Transaction> transactionList =
databaseOperations.getExpenseTransactionList();
        int transaction_size = transactionList != null ? transactionList.size() : 0;
        if(transaction_size == 0){
            TextView tv = new TextView(this);
            tv.setText("Empty...");
            tv.setTextSize(30);
            la.addView(tv);
            return;
        }
        Collections.sort(transactionList, new Comparator<Transaction>() {
            @Override
            public int compare(Transaction t1, Transaction t2) {
                return Double.compare(t2.getAmount(), t1.getAmount());
            }
        });
```

```
        Double max = transactionList.get(0).getAmount();

        ArrayList<BarEntry> entries = new ArrayList<>();
        ArrayList<String> labels = new ArrayList<String>();
        for (int i = 0; i < transaction_size; i++) {
            entries.add(new BarEntry((int) (transactionList.get(i).getAmount()), i));
            labels.add(transactionList.get(i).getCat_name());
        }

        BarDataSet data_set = new BarDataSet(entries, "# of Categories");
        data_set.setColors(ColorTemplate.COLORFUL_COLORS);
        data_set.setValueFormatter(null);

        BarChart chart = new BarChart(this);
        setContentView(chart);

        BarData data = new BarData(labels, data_set);
        chart.setData(data);
        databaseOperations.close();
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        int id = item.getItemId();
        if (id == android.R.id.home) {
            finish();
        }
        return super.onOptionsItemSelected(item);
    }
}
```

Shows the chart diagram of the user's expenses based on category names.

4.5 Category Menu

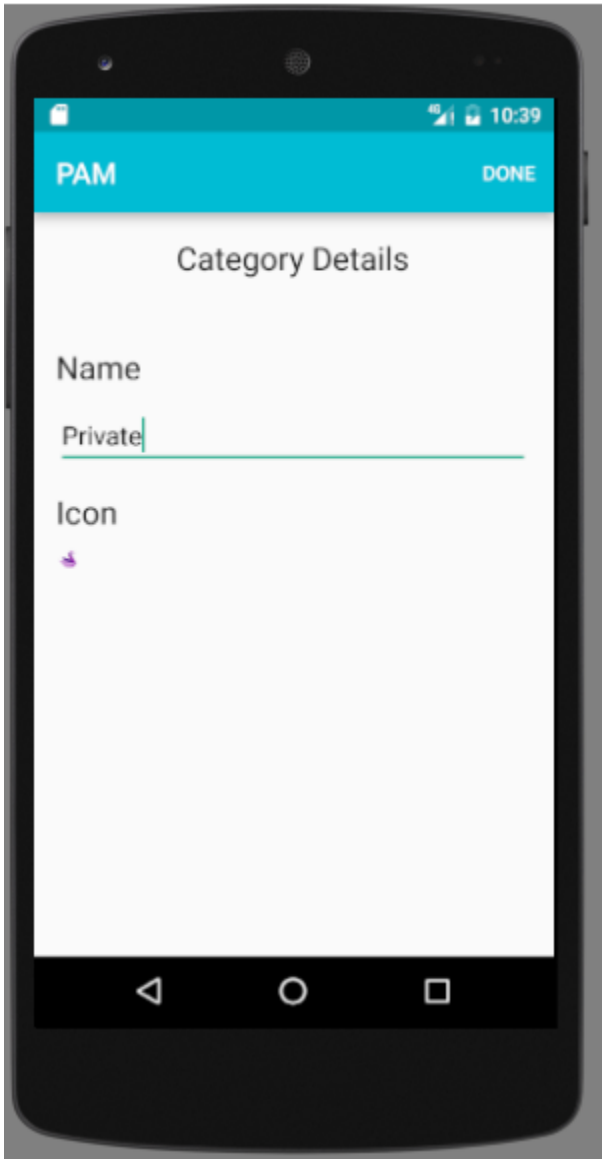


User can see the list of the categories available.

Allows the user to select the list based on category type (income or expense).

User can also edit or delete a category by clicking on it.

4.6 Category details



This activity allows the user to create a new category for transactions.

He needs to provide:

Category Name

Icon

4.7 Transaction List

```
package com.example.musa.pam;

import android.content.DialogInterface;
import android.content.Intent;
import android.content.res.Configuration;
import android.support.v7.app.AlertDialog;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.ListAdapter;
import android.widget.ListView;
import android.widget.RelativeLayout;
import android.widget.TextView;
import android.widget.Toast;

import com.github.mikephil.charting.charts.BarChart;
import com.github.mikephil.charting.data.BarData;
import com.github.mikephil.charting.data.BarDataSet;
import com.github.mikephil.charting.data.BarEntry;
import com.github.mikephil.charting.utils.ColorTemplate;

import org.w3c.dom.Text;

import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Collections;
import java.util.Comparator;
import java.util.Iterator;
import java.util.Locale;
import java.util.StringTokenizer;

public class HistoryDisplay extends AppCompatActivity {

    public static ArrayList<Transaction> transactionList;
    String[] description;
    int[] icons;
    private ListView TransactionListView;
    private TextView incomeTextView;
    private TextView expenseTextView;
    private TextView periodTv;
    private String month = null;
    private Boolean isWeekly = false;
    private Boolean isMonthly = true;
    private Boolean isYearly = false;
    private Boolean landscape = false;
    private int periodFinder = 0;
    int width = 0;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_history_display);

        findViewById();
```

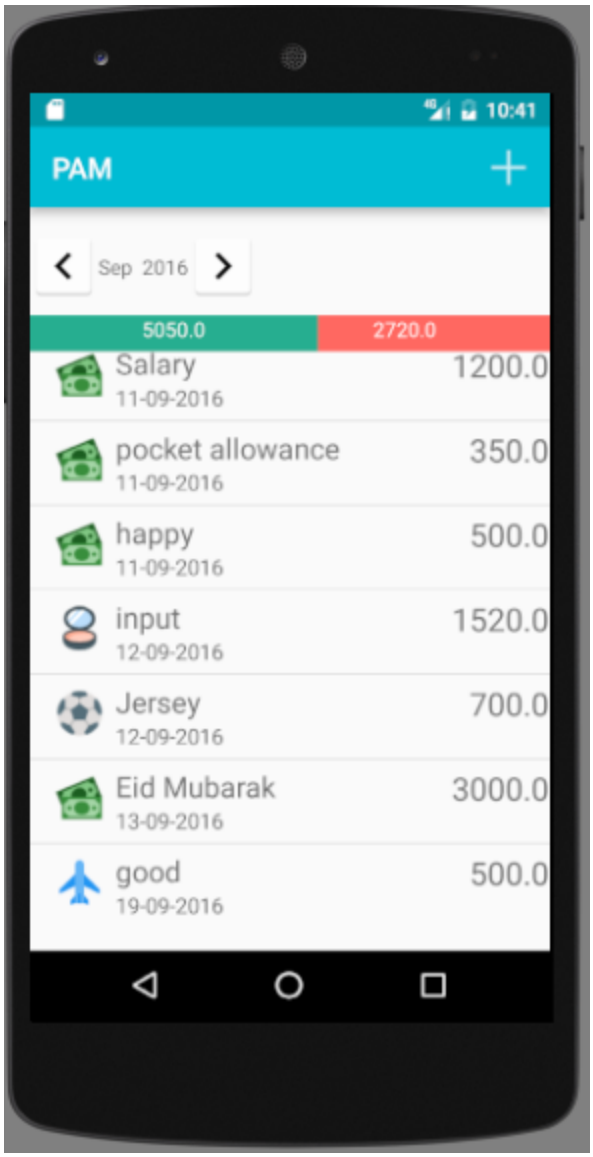
```
        setOnItemClickListener();
        SetTextView();

        if (savedInstanceState != null) {
            isWeekly = savedInstanceState.getBoolean("isWeek");
            isMonthly = savedInstanceState.getBoolean("isMonth");
            isYearly = savedInstanceState.getBoolean("isYear");
            periodFinder = savedInstanceState.getInt("periodeFinder");
            landscape = savedInstanceState.getBoolean("l");
        }
        populateListView();
        if (savedInstanceState != null) {
            if (landscape) {
                drawChart();
                //landscape = true;
            } else {
               // landscape = false;
            }
        }
    }

    @Override
    protected void onResume() {
        populateListView();
        SetTextView();
        super.onResume();
    }

    @Override
    public void onSaveInstanceState(Bundle savedInstanceState) {
        landscape = (landscape) ? false : true;
        savedInstanceState.putBoolean("isWeek", isWeekly);
        savedInstanceState.putBoolean("isMonth", isMonthly);
        savedInstanceState.putBoolean("isYear", isYearly);
        savedInstanceState.putBoolean("l", landscape);
        savedInstanceState.putInt("periodeFinder", periodFinder);

        super.onSaveInstanceState(savedInstanceState);
    }

    /*@Override
    protected void onRestoreInstanceState(Bundle savedInstanceState) {
        super.onRestoreInstanceState(savedInstanceState);
        isWeekly = savedInstanceState.getBoolean("isWeek");
        isMonthly = savedInstanceState.getBoolean("isMonth");
        isYearly = savedInstanceState.getBoolean("isYear");
        periodFinder = savedInstanceState.getInt("periodeFinder");
    }*/

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.main_menu, menu);
        return super.onCreateOptionsMenu(menu);
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case R.id.menu_new_category:
                Intent detailIntent = new Intent(HistoryDisplay.this,
TransactionActivity.class);
                detailIntent.putExtra("method", "Insert");
                startActivity(detailIntent);
```

```
                break;
        }
        return super.onOptionsItemSelected(item);
    }

    /*@Override
    public void onConfigurationChanged(Configuration newConfig) {
        super.onConfigurationChanged(newConfig);

        // Checks the orientation of the screen
        if (newConfig.orientation == Configuration.ORIENTATION_LANDSCAPE) {
            landscape = false;
        } else if (newConfig.orientation == Configuration.ORIENTATION_PORTRAIT) {
            landscape = true;
        }
    }*/

    private void findViewById() {
        TransactionListView = (ListView) findViewById(R.id.transListView);
        incomeTextView = (TextView) findViewById(R.id.incomeTv);
        expenseTextView = (TextView) findViewById(R.id.expenseTv);
        periodTv = (TextView) findViewById(R.id.tvPeriod);
    }

    public void setOnItemClickListener() {
        TransactionListView.setOnItemClickListener(
                new AdapterView.OnItemClickListener() {
                    @Override
                    public void onItemClick(AdapterView<?> parent, View view, int
position, long id) {
                        int tran_id = transactionList.get(position).getTran_id();
                        String date = transactionList.get(position).getDate();
                        int type = transactionList.get(position).getType();
                        String catname = transactionList.get(position).getCat_name();
                        Double amount = transactionList.get(position).getAmount();
                        String description =
transactionList.get(position).getDescription();
                        Intent detailIntent = new Intent(HistoryDisplay.this,
TransactionActivity.class);
                        detailIntent.putExtra(TableData.transaction.TRAN_ID, tran_id);
                        detailIntent.putExtra(TableData.transaction.DATE, date);
                        detailIntent.putExtra(TableData.transaction.TYPE, type);
                        detailIntent.putExtra(TableData.transaction.CAT_NAME,
catname);
                        detailIntent.putExtra(TableData.transaction.AMOUNT, amount);
                        detailIntent.putExtra(TableData.transaction.DESCRIPTION,
description);
                        detailIntent.putExtra("method", "Update");
                        startActivity(detailIntent);
                    }
                }
        );
    }

    protected void populateListView() {
        Calendar newDate = Calendar.getInstance();
        SimpleDateFormat dateFormatter = null;
        Calendar newCalendar = Calendar.getInstance();
        String first = null;
        String last = null;

        if (isWeekly) {
            dateFormatter = new SimpleDateFormat("dd MMM", Locale.US);
```

```
            newDate.set(Calendar.DAY_OF_WEEK, newDate.getFirstDayOfWeek());
            newDate.add(Calendar.DATE, periodFinder);
            first = dateFormatter.format(newDate.getTime());
            newDate.add(Calendar.DATE, 6);
            last = dateFormatter.format(newDate.getTime());
            dateFormatter = new SimpleDateFormat("yyyy", Locale.US);

            periodTv.setText(first + " - " + last + ", " +
dateFormatter.format(newDate.getTime()));
            dateFormatter = new SimpleDateFormat("dd-MM-yyyy", Locale.US);
            last = dateFormatter.format(newDate.getTime());
            newDate.add(Calendar.DATE, -6);
            first = dateFormatter.format(newDate.getTime());
        } else if (isMonthly) {
            dateFormatter = new SimpleDateFormat("MMM  yyyy", Locale.US);
            newDate.set(newCalendar.get(Calendar.YEAR),
newCalendar.get(Calendar.MONTH) + periodFinder,
newCalendar.get(Calendar.DAY_OF_MONTH));
            periodTv.setText(dateFormatter.format(newDate.getTime()));
            dateFormatter = new SimpleDateFormat("MM-yyyy", Locale.US);
        } else if (isYearly) {
            dateFormatter = new SimpleDateFormat("yyyy", Locale.US);
            newDate.set(newCalendar.get(Calendar.YEAR) + periodFinder,
newCalendar.get(Calendar.MONTH), newCalendar.get(Calendar.DAY_OF_MONTH));
            periodTv.setText(dateFormatter.format(newDate.getTime()));
        }
        DatabaseOperations databaseOperations = new DatabaseOperations(this);
        if (isWeekly) {
            transactionList = databaseOperations.getTransactionListByWeek(first,
last);
        } else {
            month = dateFormatter.format(newDate.getTime());
            transactionList = databaseOperations.getTransactionListByMonth(month);
        }
        if (transactionList != null) {
            description = new String[transactionList.size()];
            icons = new int[transactionList.size()];

            for (int i = 0; i < transactionList.size(); i++) {
                description[i] = transactionList.get(i).getDescription();
                icons[i] =
databaseOperations.getCategoryIcon(transactionList.get(i).getCat_name());
            }
            ListAdapter transactionAdapter = new HistorySelection(this, description,
icons);
            TransactionListView.setAdapter(transactionAdapter);
        }
        databaseOperations.close();
    }

    private void drawChart() {

        int transaction_size = transactionList != null ? transactionList.size() : 0;
        if (transaction_size == 0) {
            TextView tv = new TextView(this);
            tv.setText("Empty...");
            tv.setTextSize(30);
            return;
        }
        ArrayList<Transaction> l = new ArrayList<>();
        for (Transaction t : transactionList) {
            if (l.isEmpty()) {
                if (t.getType() == 0)
```

```
                    l.add(t);
            } else {
                int i = 0;
                for (i = 0; i < l.size(); i++) {
                    if (l.get(i).getCat_name().equals(t.getCat_name())) {
                        l.get(i).setAmount(l.get(i).getAmount() + t.getAmount());
                        break;
                    }
                }
                if (i == l.size()) {
                    if (t.getType() == 0)
                        l.add(t);
                }
            }
        }
        Collections.sort(l, new Comparator<Transaction>() {
            @Override
            public int compare(Transaction t1, Transaction t2) {
                return Double.compare(t2.getAmount(), t1.getAmount());
            }
        });
        //Double max = transactionList.get(0).getAmount();

        ArrayList<BarEntry> entries = new ArrayList<>();
        ArrayList<String> labels = new ArrayList<String>();
        for (int i = 0; i < l.size(); i++) {
            entries.add(new BarEntry((int) (l.get(i).getAmount()), i));
            labels.add(l.get(i).getCat_name());
        }

        BarDataSet data_set = new BarDataSet(entries, "# of Categories");
        data_set.setColors(ColorTemplate.COLORFUL_COLORS);
        data_set.setValueFormatter(null);

        BarChart chart = new BarChart(this);
        setContentView(chart);

        BarData data = new BarData(labels, data_set);
        chart.setData(data);
    }

    protected void SetTextView() {
        RelativeLayout rl = (RelativeLayout) findViewById(R.id.historyRL);
        /*rl.measure(0, 0);
        width = rl.getMeasuredWidth();*/
        double totalIncome = 0;
        double totalExpense = 0;
        int t_size = 0;
        t_size = (transactionList != null) ? transactionList.size() : 0;
        for (int i = 0; i < t_size; i++) {
            if (transactionList.get(i).getType() == 1)
                totalIncome += transactionList.get(i).getAmount();
            else
                totalExpense += transactionList.get(i).getAmount();
        }
        incomeTextView.setText(Double.toString(totalIncome));
        incomeTextView.setWidth(width / 2);
        expenseTextView.setText(Double.toString(totalExpense));
        expenseTextView.setWidth(width / 2);
        //Toast.makeText(this, "Width = "+ width, Toast.LENGTH_LONG).show();
    }

    public void onClickPeriod(View view) {
```

```
        String list[] = {"Weekly", "Monthly", "Yearly"};
        AlertDialog.Builder builder = new AlertDialog.Builder(HistoryDisplay.this);
        builder.setTitle("Show Spending")
                .setItems(list, new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int which) {
                        if (which == 0) {
                            isWeekly = true;
                            isMonthly = false;
                            isYearly = false;
                        } else if (which == 1) {
                            isWeekly = false;
                            isMonthly = true;
                            isYearly = false;
                        } else {
                            isWeekly = false;
                            isMonthly = false;
                            isYearly = true;
                        }
                        periodFinder = 0;
                        populateListView();
                        SetTextView();
                    }
                });
        AlertDialog dialog = builder.create();
        dialog.show();
    }

    public void onPreviousClick(View view) {
        if (isWeekly) {
            periodFinder -= 7;
        } else {
            periodFinder--;
        }
        populateListView();
        SetTextView();
    }

    public void onNextClick(View view) {
        if (isWeekly) {
            periodFinder += 7;
        } else {
            periodFinder++;
        }
        populateListView();
        SetTextView();
    }
}
```



Lists all of the transactions

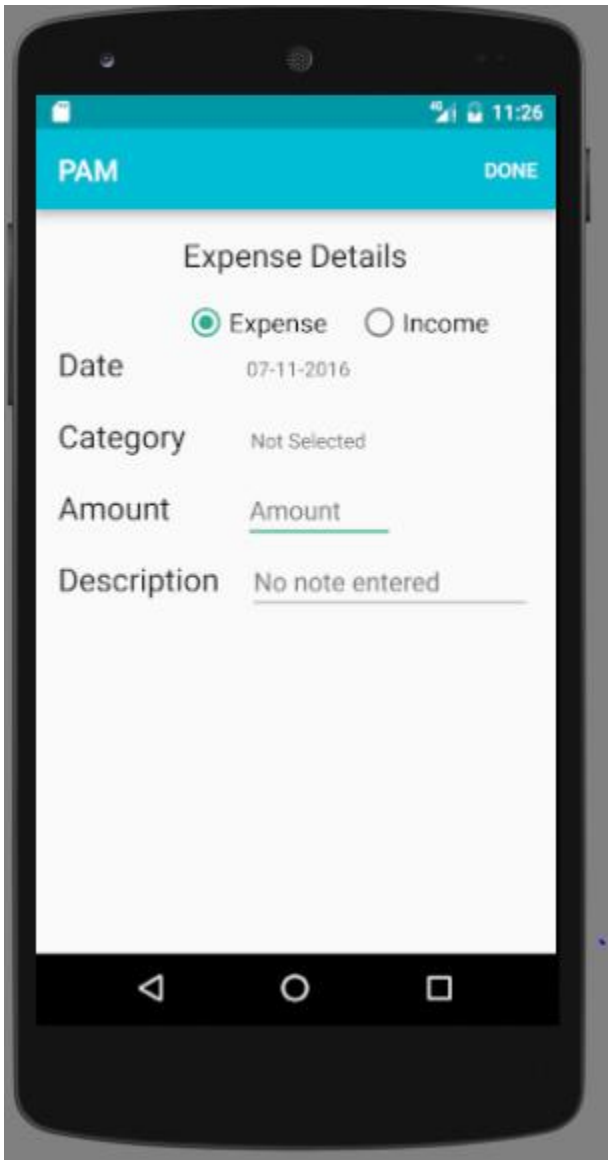
By default it shows the current month's transaction details.

Allows the user to filter the list weekly, monthly and yearly.

User can also edit or delete a transaction.

An option to insert a new transaction is provided by "+".

4.8 Transaction



The user chooses the type of transaction.

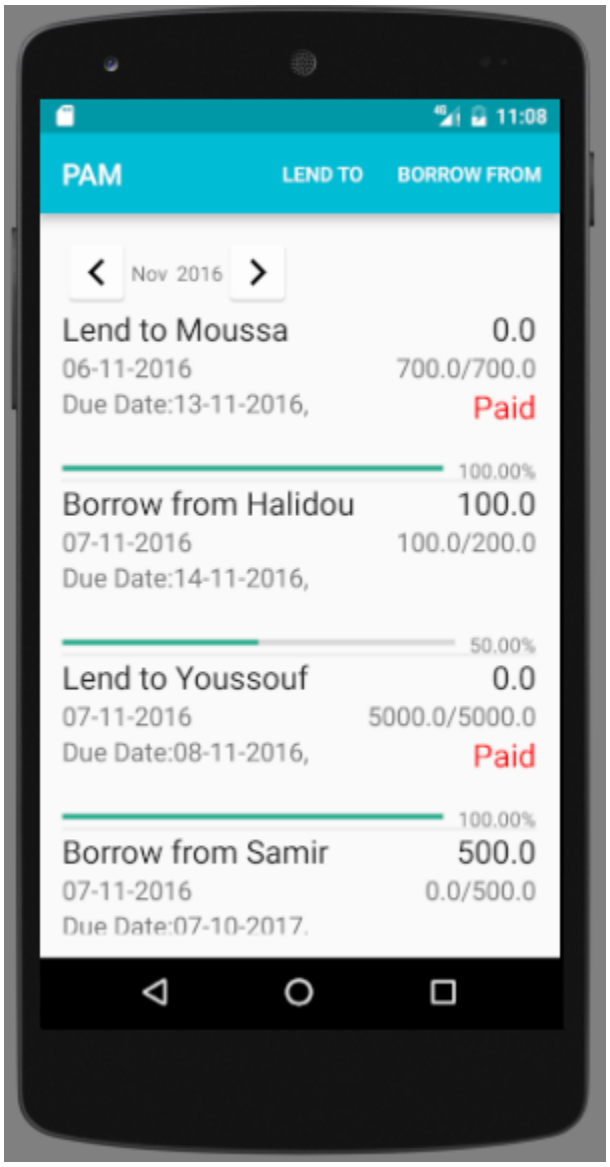
He fills up the fields:

Date

Category name

Amount and the Description

4.9 Debt List


11:08
PAM
LEND TO
BORROW FROM
Nov 2016
Lend to Moussa
0.0
06-11-2016
700.0/700.0
Due Date:13-11-2016,
Paid
100.00%
Borrow from Halidou
100.0
07-11-2016
100.0/200.0
Due Date:14-11-2016,
50.00%
Lend to Youssouf
0.0
07-11-2016
5000.0/5000.0
Due Date:08-11-2016,
Paid
100.00%
Borrow from Samir
500.0
07-11-2016
0.0/500.0
Due Date:07-10-2017.

List of debts.

Borrow From:

View Transaction

Edit or Delete

Pay to

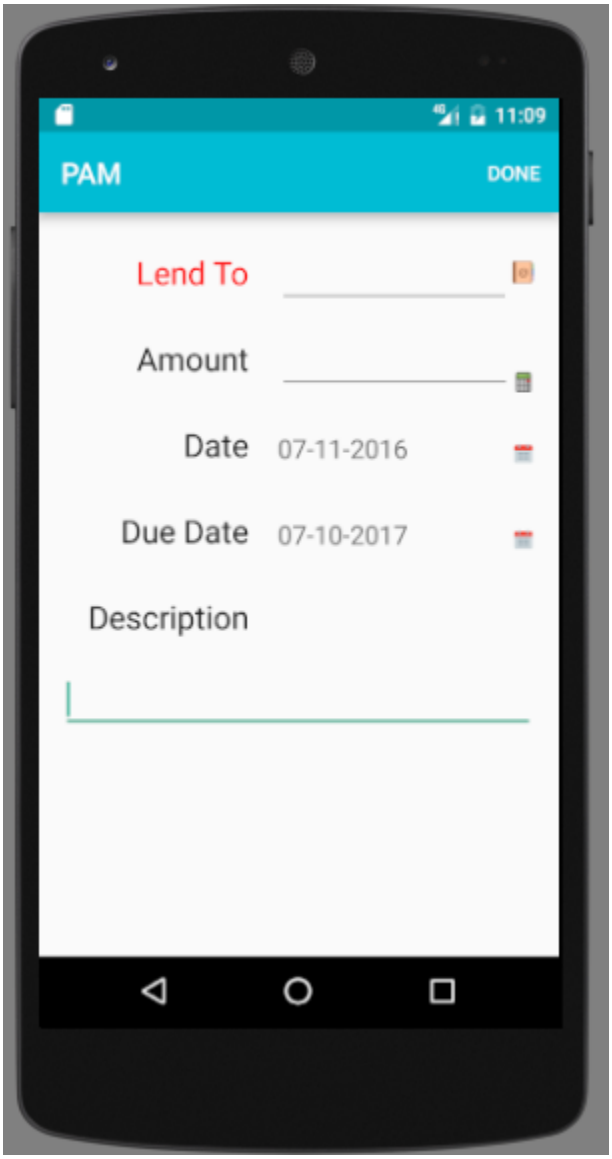
Lend To:

View Transaction

Edit or Delete

Receive from

4.10 Lend to


11:09
PAM
DONE
Lend To
Amount
Date
07-11-2016
Due Date
07-10-2017
Description

Provides:

The receiver’s name.

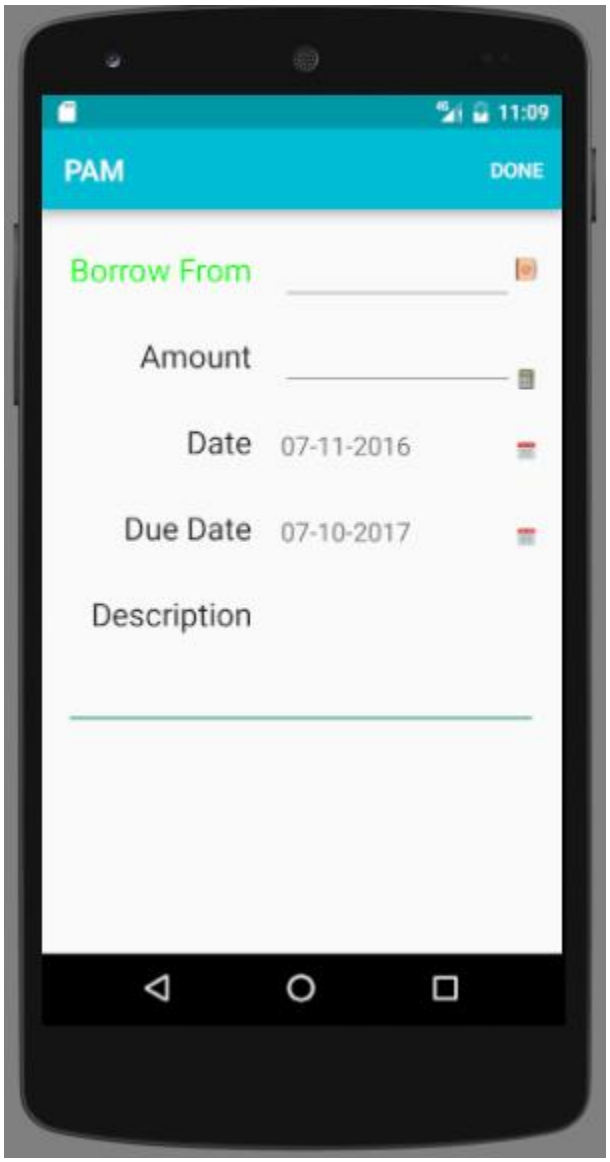
The amount

The date of the operation

The due date

The description about the operation

4.11 Borrow from



Inserts:

The borrower's name.

The amount

The date of the operation

The due date and the description about the operation.

CHAPTER 5: CONCLUSION

After making this application we assure that this application will help its users to manage the cost of their daily expenditure. It will prove to be helpful for the people who are frustrated with their daily budget management, irritated because of high amount of expenses and wishes to manage money and to preserve the record of their daily costs which may be useful to change their way of spending money.

In short this application will help its users to overcome the wastage of money.

REFERENCES

[1] Android Coding

http://stackoverflow.com/

[2] Android Developer Guide

http://developer.android.com/

[3] Expense Manager – Android Apps on Google Play

https://play.google.com/store/apps/details?id=com.expensemanager&hl=en

[4] Spending Tracker – Android Apps on Google Play

https://play.google.com/store/apps/details?id=com.mhriley.spendingtracker&hl=en

[5] Daily Expense Manager – Android Apps on Google Play

https://play.google.com/store/apps/details?id=com.techahead.ExpenseManagers&hl=en