G.S.Mandal's

MAHARASHTRA INSTITUTE OF TECHNOLOGY,

CHHATRAPATI SAMBHAJI NAGAR

(An Autonomous Institute)

PROJECT REPORT

ON

"Books App "

SUBMITTED

BY

AKASH VILAS RANPISE

Roll No: 87

UNDER THE GUIDANCE

OF

Ms. Seema Kute

DEPARTMENT OF COMPUTER APPLICATIONS

2024-25

G.S.Mandal's

MAHARASHTRA INSTITUTE OF TECHNOLOGY,

CHHATRAPATI SAMBHAJI NAGAR

(An Autonomous Institute)

Department of Computer Applications

<u>Certificate</u>

This is to certify that, Mr. **AKASH VILAS RANPISE,** Student of First Year of MCA (part-II) has satisfactorily completed the project on **"Books App "** as partial fulfillment of Project work for the subject Mobile Application Development Lab in the course Master of Computer Applications during the academic year **2024-25**

Ms. Seema Kute **Dr. Prashant Chintal Internal Guide** **Head of Department**

External Examiner

<u>DECLARATION</u>

To,

The Head,

Computer Applications Department,

Maharashtra Institute of Technology, Chh.Sambhajinagar.

Respected Sir/Ma'am,

I **Mr.Akash Vilas Ranpise,** hereby declare that, the project titled **"Books App"** developed and submitted as the partial fulfillment of submission for MCA Final year in the subject of Mobile Application Development Lab under the guidance of **Ms. Seema Kute** (Assistant Professor), and is my original work and has not duplicated from any other sources.

Date:

Place:

You're sincerely

Mr.AKASH VILAS RANPISE

INDEX

1. Project Description

The Financial Manager Android app is a comprehensive tool designed to help users manage and track their personal finances efficiently.

The app offers a range of features to monitor income, expenses, budgets, and financial goals, empowering users to make informed decisions and achieve financial stability.

Students, working professionals, freelancers, and anyone interested in tracking their income and expenses.

The primary objectives of the BMI Calculator application are:

- The primary goal of this app is to simplify personal finance management for everyday users by providing a userfriendly, secure, and feature-rich platform.

- The app seeks to help users achieve financial awareness and control, making it easier to save money, pay off debts, and track financial goals.

- Ensure a simple, clean, and easy-to-navigate user interface that appeals to users of all financial literacy levels.

-Allow users to set financial goals and track their progress toward those goals.

- Help users set budgets for different categories and track their progress against these budgets.

-Allow users to record and categorize their income, expenses, and savings, giving them a clear picture of their financial status.

1

2. Detailed Description of Technology Used

a. **Android Studio:-**

   i. Android Studio is the primary Integrated Development Environment (IDE) used for developing this application. It provides a comprehensive platform for Android development, with a powerful code editor, debugging tools, and an emulator for testing. Android Studio supports Java programming and offers resources to optimize app performance and ensure compatibility with various Android devices.

b. **Java:**

   i. Java is the core programming language used to build the application's functionality. Known for its stability, scalability, and widespread use in Android development, Java enables developers to implement complex logic, handle user interactions, and manage background processes seamlessly. It is particularly suited to applications requiring high levels of reliability and maintainability, like this ride-sharing app.

c. **XML (Extensible Markup Language):**

XML is used to design the app's user interface within Android Studio. It enables structured, readable layouts and allows developers to define UI elements, such as buttons, lists, and images, in a clear, hierarchical format. XML helps maintain consistency across different screen sizes and resolutions, ensuring the app's interface is responsive and accessible.

These technologies collectively contribute to creating a robust, user-friendly, and efficient ridesharing application that delivers smooth performance and a reliable user experience.

3. System Requirement

1. Mobile Device Requirements

- Operating System: Android 7.0 or higher
- Processor: Quad-core 1.5 GHz or faster
- RAM: Minimum 2 GB (4 GB or more recommended for optimal performance)
- Storage: At least 100 MB of free storage for installation and data caching
- Display: 720p or higher screen resolution for clear UI visibility
- Internet Connectivity: Active internet connection (4G, 5G, or Wi-Fi) required for accessing realtime vehicle availability and online payments

2. OS Requirements

- Operating System: Linux (Ubuntu 20.04+ or CentOS 7+ recommended) or Windows Server 2016+
- Processor: Dual-core 2.5 GHz or higher (Quad-core recommended for handling multiple requests)
- RAM: Minimum 4 GB (8 GB or more recommended for better performance)
- Storage: 10 GB minimum, with SSD storage recommended for faster read/write operations
- Database: SQLite for local storage; optional MySQL or PostgreSQL for remote storage
- Network: High-speed internet connection to handle API requests from mobile clients

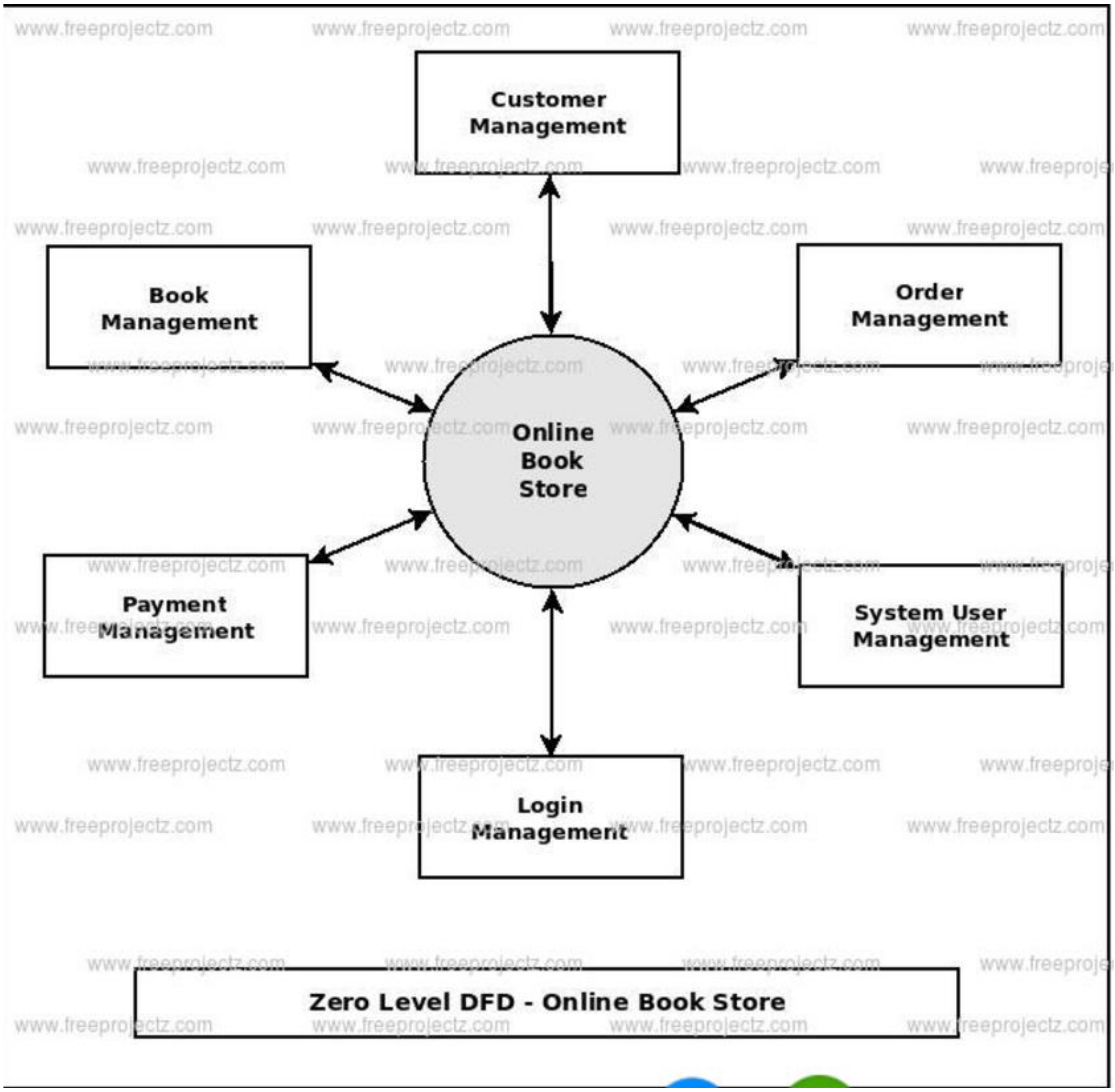
3. Development Environment Requirements (for app development)

- IDE: Android Studio 4.0 or higher
- Java Development Kit (JDK): JDK 8 or higher
- Operating System: Windows 10, macOS 10.14 (Mojave) or higher, or a Linux distribution
- RAM: Minimum 8 GB (16 GB recommended for smoother development)
- Storage: At least 2 GB for Android Studio installation and SDK tools
- Emulator: Android Emulator (comes with Android Studio) for testing across different device configurations.

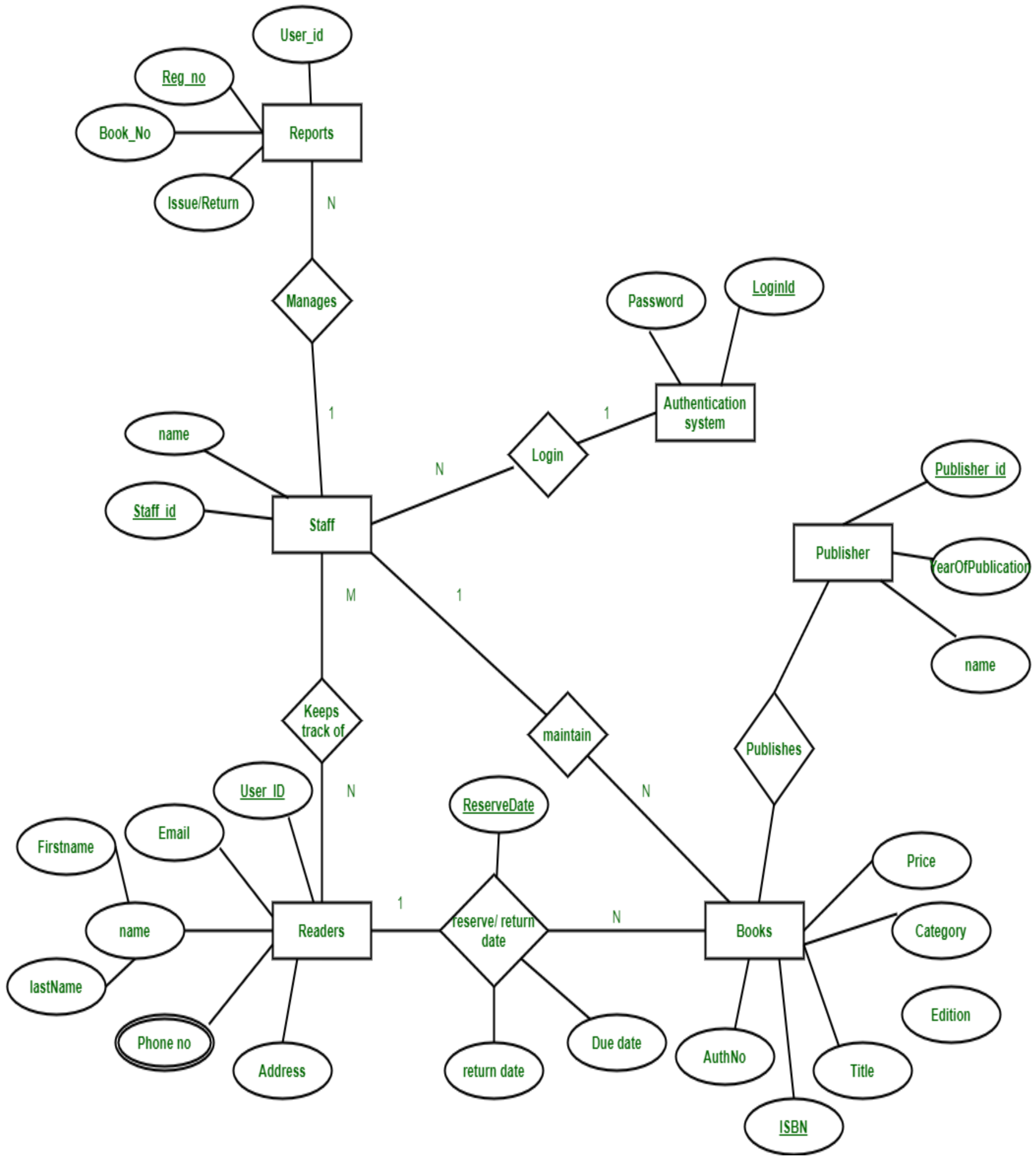
These requirements ensure the ride-sharing app operates smoothly, providing users with a reliable, fast, and user-friendly experience.

3

o 4. Data Flow Diagram (DFD)



5. Entity Relationship Diagram (ER-Diagram)



6. Table Design, Data Dictionary

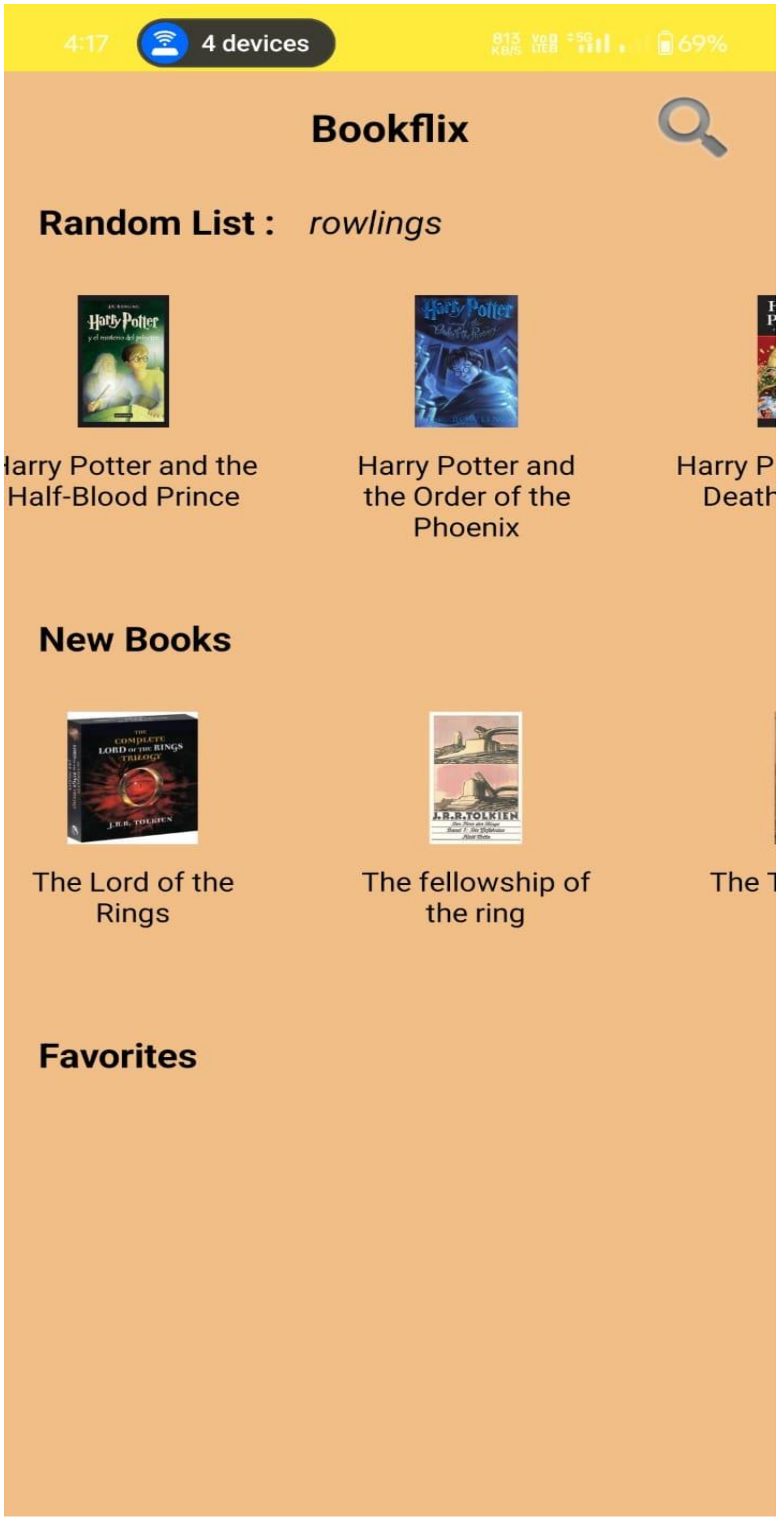
Books App is a mechanism in Android that allows you to store small amounts of data in key-value pairs. It is often used to save simple data such as settings, preferences, flags, or simple app-related data. SharedPreferences stores this data in XML files and makes it easy to read and write data persistently without the need for a full database.
Key Characteristics of SharedPreferences:

1. Simple Data Storage: SharedPreferences is best used for storing primitive data types such as strings, integers, booleans, and floating-point values.
2. File-based Storage: Data is stored in an XML file, which is located in the app's private storage area. This makes it secure because the data is only accessible by the app.
3. Key-Value Pair: Data is stored in key-value pairs, where each key is a unique identifier for the data, and the value is the actual data (e.g., a string, int, boolean, etc.).
4. Persistent Data: Data stored in SharedPreferences remains available even after the app is closed or the device is restarted.
5. No Complex Data: It's not suitable for large amounts of data or complex data structures (like lists or custom objects). For complex data storage, you would typically use a database (like SQLite).

Common Use Cases for SharedPreferences:

- Saving User Preferences: For example, whether a user has enabled dark mode or light mode.
- Storing App Settings: Such as language settings, app configuration, or whether a user has completed an onboarding tutorial.
- Saving Session Information: For example, storing whether the user is logged in or not, and the session token.
- Storing Small Amounts of Data: Such as scores in a game, currency conversion history, or flags indicating app features that the user has interacted with.

7. Input Forms With Data


4:17 4 devices 69%

Bookflix

Random List : *rowlings*

Harry Potter and the Half-Blood Prince

Harry Potter and the Order of the Phoenix

Harry P Death

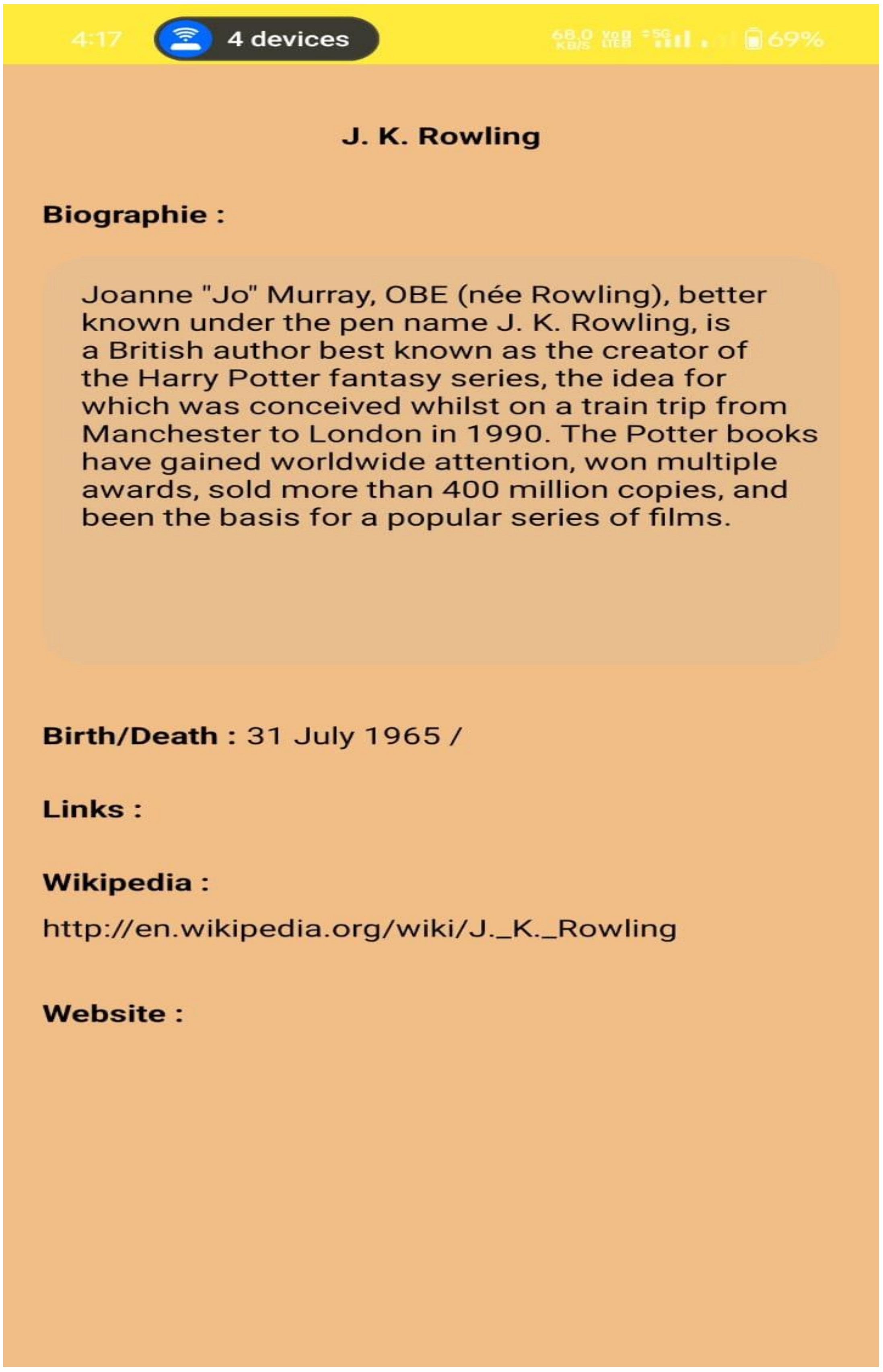
New Books

The Lord of the Rings

The fellowship of the ring

The T

Favorites



J. K. Rowling

Biographie :

Joanne "Jo" Murray, OBE (née Rowling), better known under the pen name J. K. Rowling, is a British author best known as the creator of the Harry Potter fantasy series, the idea for which was conceived whilst on a train trip from Manchester to London in 1990. The Potter books have gained worldwide attention, won multiple awards, sold more than 400 million copies, and been the basis for a popular series of films.

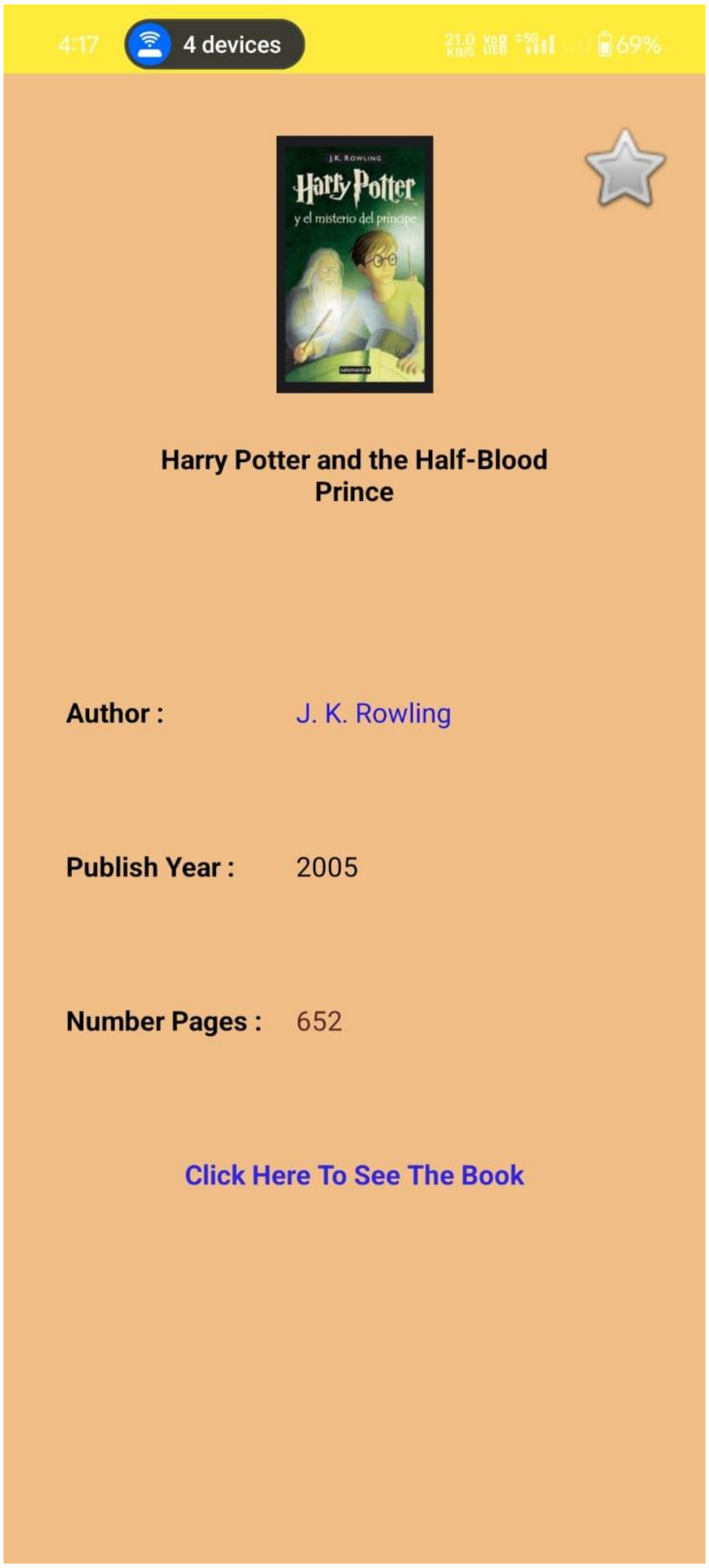
Birth/Death : 31 July 1965 /

Links :

Wikipedia :

http://en.wikipedia.org/wiki/J._K._Rowling

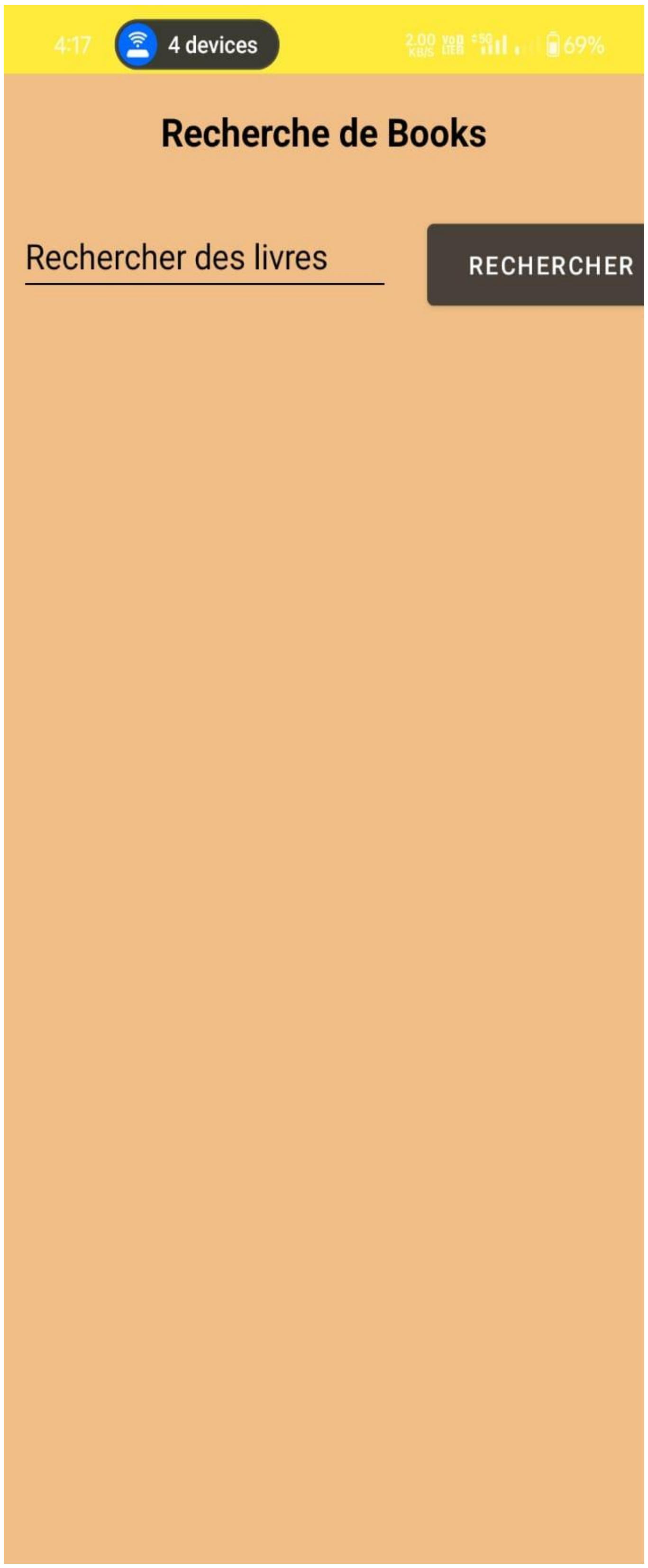
Website :



Harry Potter and the Half-Blood Prince

Author :	J. K. Rowling
Publish Year :	2005
Number Pages :	652

Click Here To See The Book


4:17
4 devices
69%
Recherche de Books
Rechercher des livres
RECHERCHER

8. Coding with heading for each program

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="com.hongjolim.mfmanager">

  <uses-permission android:name="android.permission.INTERNET"/>
  <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>

  <application
    android:allowBackup="true"
android:icon="@mipmap/ic_launcher"
android:label="@string/app_name"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/AppTheme">
    <activity
      android:name="com.hongjolim.mfmanager.MainActivity"
      android:label="@string/app_name"
      android:theme="@style/AppTheme.NoActionBar" />

    <provider
      android:name="com.hongjolim.mfmanager.database.DataProvider"
android:authorities="com.hongjolim.mfmanager.dataprovider"         android:exported="false"
/>

    <activity
      android:name="com.hongjolim.mfmanager.SplashScreen"
      android:theme="@style/AppTheme.NoActionBar">
      <intent-filter>
        <action android:name="android.intent.action.MAIN" />

        <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
    </activity>
    <activity
      android:name="com.hongjolim.mfmanager.SettingAccountsActivity"
android:label="Add Accounts"
      android:parentActivityName="com.hongjolim.mfmanager.ShowingAccountsActivity" />
    <activity
      android:name="com.hongjolim.mfmanager.ShowingAccountsActivity"
android:label="My Accounts"
      android:parentActivityName="com.hongjolim.mfmanager.MainActivity" />
    <activity
```

```
            android:name="com.hongjolim.mfmanager.BudgetActivity"
    android:label="Budget"
    android:parentActivityName="com.hongjolim.mfmanager.MainActivity" />
        <activity
            android:name="com.hongjolim.mfmanager.ShowingExpensesActivity"
    android:label="Expense"
            android:parentActivityName="com.hongjolim.mfmanager.MainActivity" />
        <activity
            android:name="com.hongjolim.mfmanager.AddingExpenseActivity"
    android:label="Add Expense"
            android:parentActivityName="com.hongjolim.mfmanager.MainActivity" />
        <activity
            android:name="com.hongjolim.mfmanager.AddingIncomeActivity"
    android:label="Add Income"
            android:parentActivityName="com.hongjolim.mfmanager.ShowingIncomeActivity" />
        <activity
            android:name="com.hongjolim.mfmanager.ShowingIncomeActivity"
    android:label="Income"
            android:parentActivityName="com.hongjolim.mfmanager.MainActivity" />
        <activity
            android:name="com.hongjolim.mfmanager.ShowingExpensesDetailActivity"
    android:label="Expense Detail"
            android:parentActivityName="com.hongjolim.mfmanager.ShowingExpensesActivity" />
        <activity
            android:name="com.hongjolim.mfmanager.ShowingIncomeDetailActivity"
    android:label="Income Detail"
            android:parentActivityName="com.hongjolim.mfmanager.ShowingIncomeActivity" />
        <activity
            android:name="com.hongjolim.mfmanager.TransferActivity"
    android:label="@string/transfer"
            android:parentActivityName="com.hongjolim.mfmanager.MainActivity" />
        <activity
            android:name="com.hongjolim.mfmanager.IncomeCategoryActivity"
    android:label="Income Category"
            android:parentActivityName="com.hongjolim.mfmanager.MainActivity" />
        <activity
            android:name="com.hongjolim.mfmanager.ShowingCreditActivity"
    android:label="Credit Cards"
            android:parentActivityName="com.hongjolim.mfmanager.MainActivity" />
        <activity
            android:name="com.hongjolim.mfmanager.AddingCreditActivity"
    android:label="Adding Cards"
            android:parentActivityName="com.hongjolim.mfmanager.ShowingCreditActivity" />
        <activity
            android:name="com.hongjolim.mfmanager.ShowingCreditDetailActivity"
    android:label="Credit Detail"
            android:parentActivityName="com.hongjolim.mfmanager.ShowingCreditActivity" />
        <activity
           android:name="com.hongjolim.mfmanager.LoginActivity"
android:theme="@style/AppTheme.NoActionBar" />
        <activity
           android:name="com.hongjolim.mfmanager.PreferencesActivity"            android:label="Settings"
           android:parentActivityName="com.hongjolim.mfmanager.MainActivity"    />
<activity                         android:name="com.hongjolim.mfmanager.RegisterActivity"
android:theme="@style/AppTheme.NoActionBar"/>
    </application>

</manifest>
```

MainActivity.java

```
package com.hongjolim.mfmanager;

import android.animation.ObjectAnimator; import
android.annotation.SuppressLint; import
android.app.AlertDialog; import
android.app.LoaderManager; import
android.content.ContentValues; import
android.content.CursorLoader; import
android.content.DialogInterface; import
android.content.Intent; import android.content.Loader; import
android.content.SharedPreferences; import
android.database.Cursor; import android.graphics.Color;
import android.os.Bundle; import
android.preference.PreferenceManager; import
android.support.annotation.NonNull; import
android.support.design.widget.FloatingActionButton; import
android.support.design.widget.NavigationView; import
android.support.v4.view.GravityCompat; import
android.support.v4.widget.DrawerLayout; import
android.support.v7.app.ActionBarDrawerToggle; import
android.support.v7.app.AppCompatActivity; import
android.support.v7.widget.CardView; import
android.support.v7.widget.Toolbar; import
android.view.MenuItem; import android.view.View;
import android.view.animation.DecelerateInterpolator;
import android.widget.EditText; import
android.widget.ProgressBar;
import android.widget.TextView;

import com.github.mikephil.charting.charts.PieChart; import
com.github.mikephil.charting.data.PieData; import
com.github.mikephil.charting.data.PieDataSet; import
com.github.mikephil.charting.data.PieEntry; import
com.github.mikephil.charting.formatter.PercentFormatter; import
com.google.android.gms.ads.AdRequest; import
com.google.android.gms.ads.AdView; import
com.google.android.gms.ads.MobileAds;
import com.hongjolim.mfmanager.database.AccountsTable;
import com.hongjolim.mfmanager.database.DataProvider;
import com.hongjolim.mfmanager.database.DataSource; import
com.hongjolim.mfmanager.database.ExCategoryTable; import
com.hongjolim.mfmanager.database.InCategoryTable; import
com.hongjolim.mfmanager.database.TransactionTable; import
com.hongjolim.mfmanager.model.Account; import
com.hongjolim.mfmanager.model.ExCategory; import
com.hongjolim.mfmanager.model.InCategory; import
com.hongjolim.mfmanager.tools.BigDecimalCalculator; import
com.hongjolim.mfmanager.tools.CurrencyFormatter;

import java.math.BigDecimal; import
java.math.RoundingMode;
import java.util.ArrayList;

import static android.view.View.GONE; import
static android.view.View.INVISIBLE; import
static android.view.View.VISIBLE;

public class MainActivity extends AppCompatActivity
        implements                                        NavigationView.OnNavigationItemSelectedListener,
LoaderManager.LoaderCallbacks<Cursor>{

    private DataSource mDataSource;

    //static final fields for sharedPreferences
    private static final String CHECK_IF = "IS_FROM_MAIN_FAB";

    //the name of the shared preferences set
    private static final String FIRST_START = "FIRST_START";

    //the key to get value for default currency setting from sharedpreference
public static final String CURRENCY_KEY = "CURRENCY";

    private static final int REQUEST_CODE = 1;     private
static final int IS_FROM_MAIN_FAB = 1001;     private
Cursor cursor;

    private BigDecimal totalMonthlyIn, totalMonthlyEx;

    private SharedPreferences prefs;
    String currencyCode;
    AdView mAdview;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);         setContentView(R.layout.activity_main);
MobileAds.initialize(this, "ca-app-pub-1465820537677658/8541972644");

        mAdview = (AdView) findViewById(R.id.adView);
AdRequest adRequest = new AdRequest.Builder().build();
mAdview.loadAd(adRequest);


        mDataSource = new DataSource(this);
mDataSource.open();

        //get Shared preferences to check whether this app is launching for the first time or not
prefs = PreferenceManager.getDefaultSharedPreferences(this);         boolean firstStart =
prefs.getBoolean(FIRST_START, true);

        currencyCode = prefs.getString(CURRENCY_KEY, "Canada");

        if(firstStart){
            setDefaultCurrencySetting();
setDefaultExCategory();
            setDefaultInCategory();
        }

        Toolbar toolbar = findViewById(R.id.toolbar);
setSupportActionBar(toolbar);

        setContentForMain();

        FloatingActionButton fab = findViewById(R.id.fab);
fab.setOnClickListener(listener);

        DrawerLayout drawer = findViewById(R.id.drawer_layout);
        ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(
                this, drawer, toolbar, R.string.navigation_drawer_open, R.string.navigation_drawer_close);
drawer.addDrawerListener(toggle);
        toggle.syncState();

        NavigationView navigationView = findViewById(R.id.nav_view);
navigationView.setNavigationItemSelectedListener(this);

        getLoaderManager().initLoader(0, null, this);
    }
```

```
   private void setDefaultCurrencySetting() {
       AlertDialog.Builder builder = new AlertDialog.Builder(this);          final
CharSequence [] items = getResources().getStringArray(R.array.currency);
builder.setTitle("Choose Currency").
             setItems(items, new DialogInterface.OnClickListener(){
@Override
                public void onClick(DialogInterface dialogInterface, int position) {
SharedPreferences.Editor editor = prefs.edit();
                   editor.putString("CURRENCY", items[position].toString()).apply();

                   //calls the default cash balance setting method
showDefaultAccountSettingDialog();
                }
             }).setPositiveButton("OK", null).setCancelable(false).create().show();

   }

   //let the user to set the default cash balance, if nothing entered, the balance is set to 0
private void showDefaultAccountSettingDialog(){

       final View view = getLayoutInflater().inflate(R.layout.set_default_account, null);

       final EditText defaultAmount = view.findViewById(R.id.default_balance);

       AlertDialog.Builder builder = new AlertDialog.Builder(this);
builder.setTitle("Default Account Setting");
       builder.setMessage("Please set your Cash balance. If not provided, it will be set to 0");
builder.setView(view);
       builder.setPositiveButton("OK", new DialogInterface.OnClickListener(){
          @Override
          public void onClick(DialogInterface dialogInterface, int i) {

             defaultAmount.setText(String.valueOf(BigDecimalCalculator.roundValue(0, currencyCode)));

             double amount_double = 0.0;

             try {
                amount_double = Double.parseDouble(defaultAmount.getText().toString());
             }catch(Exception e){
                amount_double = BigDecimalCalculator.roundValue(0.0, currencyCode);
             }

             String amount = String.valueOf(BigDecimalCalculator.roundValue(amount_double,
currencyCode));
```

```
                setDefaultAccount("Cash", "default", amount);
dialogInterface.dismiss();

                SharedPreferences.Editor editor = prefs.edit();
                //if the cash amount is set, set the shared preferences value to false
editor.putBoolean(FIRST_START, false);                editor.apply();

            }
        }).setCancelable(false).create().show();
    }

    private void setBudgetSummaryPieChart(BigDecimal totalMonthlyEx) {

        /*
        execute raw query to get data for expenses for category:
        query = SELECT category_id, SUM(CAST(amount AS NUMBER(10))) FROM Transaction
WHERE category_id IS NOT NULL GROUP BY category_id ORDER BY category_id
        */

        String[] columns = {TransactionTable.COL2, TransactionTable.COL5,
"SUM(CAST("+TransactionTable.COL3+" AS REAL))"};
        String[] selectionArgs = {String.valueOf(TransactionTable.TRANS_TYPE1)};
        cursor = mDataSource.query(TransactionTable.TABLE_NAME, columns,
TransactionTable.COL8+"=? AND "+
                TransactionTable.COL2+"<=date('now', 'start of month', '+1 month', '-1 day') AND "+
                    TransactionTable.COL2+" >= date('now', 'start of month')",
selectionArgs, TransactionTable.COL5, null, TransactionTable.COL5);

        ArrayList<ExCategory> exCategories = mDataSource.getAllExCategories();

try{
            while(cursor.moveToNext()){
                for(int i = 0; i<exCategories.size(); i++){

if(cursor.getInt(cursor.getColumnIndex(TransactionTable.COL5))==exCategories.get(i).get_id()){
//3rd column is the value of sum of amount spent monthly

exCategories.get(i).setTotalMonthlySpent(cursor.getFloat(cursor.getColumnIndex(columns[2])));
                    }
                }
            }
        }finally{
            if(cursor!=null&&!cursor.isClosed())
                cursor.close();
        }

        //to get the cursor to retrieve all the names of Ex categories
        cursor      =         getContentResolver().query(DataProvider.EX_CATEGORY_URI,
ExCategoryTable.ALL_COLS,
        null, null, ExCategoryTable.COL1);

        //initialize variables for pie chart
        PieChart budgetSummary = findViewById(R.id.budget_piechart);
budgetSummary.setUsePercentValues(false);
budgetSummary.getDescription().setEnabled(false);        budgetSummary.setExtraOffsets(5,
0, 5, 5);

        budgetSummary.setDragDecelerationFrictionCoef(0.95f);

        //set whether or not there will be a hole in the center of the pie chart
budgetSummary.setDrawHoleEnabled(false);;

        ArrayList<PieEntry> yValues = new ArrayList<>();

        //the size of both array lists(the numbers of categoryNames, categoryIds are NOT identical)
for(int i = 0; i<exCategories.size(); i++) {            ExCategory exCategory = exCategories.get(i);

            /*finds an exCategory that has been spent at all
             * if it has not been spent, it will not be in the chart
             */

            if(exCategory.getTotalMonthlySpent()!=0.0f) {
                yValues.add(new PieEntry(exCategory.getTotalMonthlySpent(), exCategory.getName()));
            }
        }

        TextView noExpense = findViewById(R.id.show_no_expense);
if(yValues.size()>=1){
            //if there is more than 1 expense corresponding any expense category, erase the text saying that
there is no expense
            noExpense.setVisibility(GONE);
}else{
            //if there is no expense spent this month so far, erase the pie chart and show the text saying that
there is no expense
            budgetSummary.setVisibility(GONE);
        }

        PieDataSet dataSet = new PieDataSet(yValues, null);
        dataSet.setSliceSpace(1.0f);
        dataSet.setSelectionShift(10f);

        //set colors for each part of the pie chart
        dataSet.setColors(Color.rgb(244,67,54), Color.rgb(255,193,7), Color.rgb(3,169,244),
Color.rgb(76,175,80), Color.rgb(121,85,72));
```

```
        PieData data = new PieData((dataSet));
data.setValueTextSize(10f);
        data.setValueTextColor(getResources().getColor(R.color.lightPrimary));

        budgetSummary.setData(data);

        TextView budgetTotal = findViewById(R.id.budget_total_amount);
        BigDecimal                    exBudgetTotal                    =                    new
BigDecimal(String.valueOf(BigDecimalCalculator.roundValue(0, currencyCode)));
for(int i = 0; i<exCategories.size(); i++){
            exBudgetTotal          =        BigDecimalCalculator.add(exBudgetTotal.toString(),
exCategories.get(i).getAmount());
            budgetTotal.setText(CurrencyFormatter.format(this, exBudgetTotal.toString()));
        }
        TextView budgetSpent = findViewById(R.id.budget_total_spent);

        if(totalMonthlyEx.doubleValue()>0) {
            budgetSpent.setText(String.format("-%s",     CurrencyFormatter.format(this,
totalMonthlyEx.toString())));
        }else {
            budgetSpent.setText(CurrencyFormatter.format(this, totalMonthlyEx.toString()));
        }
    }

    @Override     public void onActivityResult(int requestCode, int resultCode,
Intent data){
if(requestCode==REQUEST_CODE&&resultCode==RESULT_OK){
reLoad();
        }
    }

    private void reLoad(){
        getLoaderManager().restartLoader(0, null, this);
setContentForMain();
    }

    @SuppressLint("SetTextI18n")
    private void setContentForMain() {

        CardView netEarningMonthlyCardView = findViewById(R.id.netEarnings_monthly_cardView);

        TextView total7daysExpense = findViewById(R.id.expense_total_7days);
        TextView total7daysIncome = findViewById(R.id.income_total_7days);
        TextView totalMonthlyExpense = findViewById(R.id.expense_total_monthly);
        TextView totalMonthlyIncome = findViewById(R.id.income_total_monthly);
        TextView netEarning7days = findViewById(R.id.netEarnings_total_7days);
        TextView netEarningMonthly = findViewById(R.id.netEarnings_total_monthly);

        //build the selection statement to find expenses and income for the 'last 7 days' in the transaction
table
        String selection1 = TransactionTable.COL8 + "=" +
String.valueOf(TransactionTable.TRANS_TYPE1) + " AND " +
            TransactionTable.COL2+"<="+"date('now') AND "+
            TransactionTable.COL2 + ">=" + "date('now', '-7 days')";
        String selection2 = TransactionTable.COL8 + "=" +
String.valueOf(TransactionTable.TRANS_TYPE2) + " AND " +
            TransactionTable.COL2+"<="+"date('now') AND "+
            TransactionTable.COL2 + ">=" + "date('now', '-7 days')";

        //get the cursor to track expenses for the last 7 days
        cursor = getContentResolver().query(DataProvider.TRANSACTION_URI,
TransactionTable.ALL_COLS,
            selection1, null, null);

        /* since the amount column does not use numeric types, it has to be added using customized class
BigDecimalCalculator to get exact currency value */
        BigDecimal total7daysEx = new
BigDecimal(String.valueOf(BigDecimalCalculator.roundValue(0.0, currencyCode)));
try {
          while (cursor.moveToNext()) {           total7daysEx =
BigDecimalCalculator.add(total7daysEx.toString(),
cursor.getString(cursor.getColumnIndex(TransactionTable.COL3)));
          }
        } finally {
          if (cursor != null && !cursor.isClosed()) {
cursor.close();
          }
        }

        //get the cursor to track income for the last 7 days
        cursor = getContentResolver().query(DataProvider.TRANSACTION_URI,
TransactionTable.ALL_COLS,
            selection2, null, null);

        /* since the amount column does not use numeric types, it has to be added using customized class
BigDecimalCalculator to get exact currency value */
        BigDecimal total7daysIn = new
BigDecimal(String.valueOf(BigDecimalCalculator.roundValue(0, currencyCode)));
try {
          while (cursor.moveToNext()) {
            total7daysIn = BigDecimalCalculator.add(total7daysIn.toString(),
cursor.getString(cursor.getColumnIndex(TransactionTable.COL3)));
          }
        } finally {
            if (cursor != null && !cursor.isClosed()) {
cursor.close();
            }
        }

        total7daysExpense.setText(CurrencyFormatter.format(this, total7daysEx.toString()));
total7daysIncome.setText(CurrencyFormatter.format(this, total7daysIn.toString()));

        BigDecimal bigNetEarning7days = BigDecimalCalculator.subtract(total7daysIn.toString(),
total7daysEx.toString());

        //if net Earning is greater than 0, set color 'green', equals 0, 'black(default)', less than, set 'red'
if(bigNetEarning7days.doubleValue()>0) {
            netEarning7days.setTextColor(getResources().getColor(android.R.color.holo_green_dark));
        }else if(bigNetEarning7days.doubleValue()<0) {
            netEarning7days.setTextColor(getResources().getColor(android.R.color.holo_red_dark));
        }

        netEarning7days.setText(CurrencyFormatter.format(this, bigNetEarning7days.toString()));
BigDecimal bigNetEarningMonthly = getMonthlyNetEarnings(0);

        totalMonthlyExpense.setText(CurrencyFormatter.format(this, totalMonthlyEx.toString()));
totalMonthlyIncome.setText(CurrencyFormatter.format(this, totalMonthlyIn.toString()));


        if(bigNetEarningMonthly.doubleValue()>0) {
            netEarningMonthly.setTextColor(getResources().getColor(android.R.color.holo_green_dark));
        }
        else if(bigNetEarningMonthly.doubleValue()<0){
            netEarningMonthly.setTextColor(getResources().getColor(android.R.color.holo_red_dark));
        }

        netEarningMonthly.setText(CurrencyFormatter.format(this, bigNetEarningMonthly.toString()));

        setAssetsPieChart();

        //set earning percentage for this month
setEarningProgress(totalMonthlyIn);

        //set budget summary pie chart in content main layout
setBudgetSummaryPieChart(totalMonthlyEx);

        netEarningMonthlyCardView.setOnClickListener(listener);

    }
    private void setAssetsPieChart(){

        PieChart assetsPieChart = findViewById(R.id.assets_piechart);

        //sets the center void
        assetsPieChart.setDrawHoleEnabled(true);

        ArrayList<PieEntry> yValues = new ArrayList<>();

        //get all the accounts from the DataSource class
        ArrayList<Account> accounts = mDataSource.getAllAccounts();

        //this variable is going to be used for the total amount of assets
        BigDecimal assetTotal = new BigDecimal(String.valueOf(BigDecimalCalculator.roundValue(0,
currencyCode)));

        //loop to get total assets as well as to get each entry for each account
for(int i = 0; i<accounts.size(); i++){

            //get the total amount of current balances of all the accounts
            //if an account's current balance is less than 0, than do not add it to the total assets
if(Float.parseFloat(accounts.get(i).getCurrent_balance())>0.0f) {
                //add the current balance of an account to the total asset
                assetTotal    =         BigDecimalCalculator.add(assetTotal.toString(),
accounts.get(i).getCurrent_balance());
            }

            //if the current amount of an account is no more than 0, do not add it to the pie entry
if(Float.parseFloat(accounts.get(i).getCurrent_balance())>0.0f){
                //the PieEntry Class takes float, String as parameters for its constructor
yValues.add(new PieEntry(Float.parseFloat(accounts.get(i).getCurrent_balance()),
accounts.get(i).getName()));
            }
        }

        //put the total balance at the center of the chart
assetsPieChart.setCenterText(String.format("Total\n%s", assetTotal.toString()));        //set
the size of the center text which in this chart represents the total amount
        assetsPieChart.setCenterTextColor(Color.rgb(63,81,181));
assetsPieChart.setCenterTextSize(12);
        //set the margin of the chart
        assetsPieChart.setExtraOffsets(0, 5, 0, 0);
        //going to use percent values instead of original values
assetsPieChart.setUsePercentValues(true);
        //erase the description of the chart that would appear right side of the card view
assetsPieChart.getDescription().setEnabled(false);
```

```
        //set the size of entry label text
assetsPieChart.setEntryLabelTextSize(10);
        assetsPieChart.setEntryLabelColor(Color.rgb(62,39,35));

        PieDataSet dataSet = new PieDataSet(yValues, "");
        dataSet.setSliceSpace(1.0f);
        dataSet.setSelectionShift(10f);

        //set colors for each part of the pie chart
        dataSet.setColors(Color.rgb(205,220,57), Color.rgb(121,85,72), Color.rgb(63,81,181),
Color.rgb(3,169,244),
                Color.rgb(255,193,7), Color.rgb(244,67,54));

        PieData data = new PieData((dataSet));
data.setValueTextSize(10f);
        data.setValueTextColor(Color.rgb(62,39,35));

        //set the value formatter that sets the value string with '%' at the end
data.setValueFormatter(new PercentFormatter());

        assetsPieChart.setData(data);
assetsPieChart.setHoleRadius(35);
        assetsPieChart.setTransparentCircleRadius(0);

    }

    //this method is to get net Earnings for 1 specified month
private BigDecimal getMonthlyNetEarnings(int offSet){

        String selection1="";
        String selection2="";

        /**
* build the selection statement to find expenses and income for specified period(1 month)
* for example, if offset is 0, that means this method is going to return net earnings for this month
         */

        if(offSet==0) {
            selection1          =          TransactionTable.COL8          +          "="          +
String.valueOf(TransactionTable.TRANS_TYPE1) + " AND " +
                    TransactionTable.COL2 + "<=" + "date('now', 'start of month', '+1 month', '-1 day') AND " +
                    TransactionTable.COL2 + ">=" + "date('now', 'start of month')";
            selection2          =          TransactionTable.COL8          +          "="          +
String.valueOf(TransactionTable.TRANS_TYPE2) + " AND " +
                    TransactionTable.COL2 + "<=" + "date('now', 'start of month', '+1 month', '-1 day') AND " +
                    TransactionTable.COL2 + ">=" + "date('now', 'start of month')";
        }else{
            selection1 = TransactionTable.COL8 + "=" +
String.valueOf(TransactionTable.TRANS_TYPE1) + " AND " +
              TransactionTable.COL2 + "<=" + "date('now', 'start of month', '-" + (offSet-1) + " month', '-1
day') AND " +
              TransactionTable.COL2 + ">=" + "date('now', 'start of month', '-" + offSet + " month')";
            selection2 = TransactionTable.COL8 + "=" +
String.valueOf(TransactionTable.TRANS_TYPE2) + " AND " +
                TransactionTable.COL2 + "<=" + "date('now', 'start of month', '-" + (offSet-1) + " month', '-1
day') AND " +
                TransactionTable.COL2 + ">=" + "date('now', 'start of month', '-" + offSet + " month')";
        }

        //get the cursor to track expenses for a specified month
        cursor = getContentResolver().query(DataProvider.TRANSACTION_URI,
TransactionTable.ALL_COLS,
              selection1, null, null);

        /* since the amount column does not use numeric types, it has to be added using customized class
        BigDecimalCalculator to get exact currency value */
        totalMonthlyEx = new BigDecimal(String.valueOf(BigDecimalCalculator.roundValue(0,
currencyCode)));
        try {
          while (cursor.moveToNext()) {                    totalMonthlyEx =
BigDecimalCalculator.add(totalMonthlyEx.toString(),
cursor.getString(cursor.getColumnIndex(TransactionTable.COL3)));
          }
        } finally {
          if (cursor != null && !cursor.isClosed()) {
cursor.close();
          }
        }

        //get the cursor to track income for the last 7 days
        cursor = getContentResolver().query(DataProvider.TRANSACTION_URI,
TransactionTable.ALL_COLS,
              selection2, null, null);

        /* since the amount column does not use numeric types, it has to be added using customized class
        BigDecimalCalculator to get exact currency value */
        totalMonthlyIn = new BigDecimal(String.valueOf(BigDecimalCalculator.roundValue(0,
currencyCode)));
        try {
          while (cursor.moveToNext()) {          totalMonthlyIn =
BigDecimalCalculator.add(totalMonthlyIn.toString(),
cursor.getString(cursor.getColumnIndex(TransactionTable.COL3)));
          }
        } finally {
           if (cursor != null && !cursor.isClosed()) {
cursor.close();
           }
       }

       return BigDecimalCalculator.subtract(totalMonthlyIn.toString(),
totalMonthlyEx.toString());
   }

   @SuppressLint("SetTextI18n")
   private void setEarningProgress(BigDecimal totalMonthlyIn) {

       CardView earningProgressView = findViewById(R.id.earning_progress_view);
earningProgressView.setOnClickListener(listener);

       TextView expectedEarning = findViewById(R.id.earning_expectation);
TextView earningStatus = findViewById(R.id.earning_status);

       ArrayList<InCategory> inCategories = mDataSource.getAllInCategories();

        BigDecimal                    totalExpectedEarning                    =                    new
BigDecimal(String.valueOf(BigDecimalCalculator.roundValue(0,    currencyCode)));
for(int  i  =  0;  i<inCategories.size();  i++){                   InCategory  inCategory  =
inCategories.get(i);
           /**
*  add all the amount of each inCategory (the amount is in a string value)
*  so the customized class BigDecimalCalculator is needed here to get the exact sum of the amount
            */
           totalExpectedEarning  =         BigDecimalCalculator.add(totalExpectedEarning.toString(),
inCategory.getAmount());
       }

       expectedEarning.setText(CurrencyFormatter.format(this, totalExpectedEarning.toString()));

       BigDecimal earningPercent = new BigDecimal("0.00");
       //calculate the (exact earning) / (expected earning) using BigDecimal Class rounding half up

       TextView show_no_earnings = findViewById(R.id.show_no_earnings);
       ProgressBar earningProgressBar = findViewById(R.id.earning_progress_bar);

       try {
           earningProgressBar.setVisibility(VISIBLE);           earningStatus.setVisibility(VISIBLE);
show_no_earnings.setVisibility(GONE);           earningPercent =
totalMonthlyIn.divide(totalExpectedEarning, 3, RoundingMode.HALF_UP).
                 multiply(new BigDecimal("100"));
       }catch(ArithmeticException e){
earningProgressBar.setVisibility(INVISIBLE);
```

```
earningStatus.setVisibility(INVISIBLE);
show_no_earnings.setVisibility(VISIBLE);
        }

        //set the big decimal number to have 2 digits after decimal point
        earningStatus.setText(String.format("%s%%", earningPercent.setScale(2)));

        ObjectAnimator anim = ObjectAnimator.ofInt(earningProgressBar, "progress", 0,
earningPercent.intValue());

        anim.setInterpolator(new DecelerateInterpolator());
anim.setDuration(2000);
        anim.start();
    }

    //when the app is launching for the first time, this method sets the info of default account
private void setDefaultAccount(String name, String type, String balance) {

            ContentValues values = new ContentValues();
values.put(AccountsTable.COL2, name);             values.put(AccountsTable.COL3,
type);             values.put(AccountsTable.COL4, balance);
values.put(AccountsTable.COL5, balance);
            getContentResolver().insert(DataProvider.ACCOUNTS_URI, values);
    }

    @Override     public
void onPause() {
super.onPause();
        mDataSource.close();
    }

    @Override     public void
onResume() {
super.onResume();
mDataSource.open();
        reLoad();
    }

    private void setDefaultExCategory() {

        ContentValues values = new ContentValues();
values.put(ExCategoryTable.COL2, "Others");
        values.put(ExCategoryTable.COL3, String.valueOf(BigDecimalCalculator.roundValue(0,
currencyCode)));
        getContentResolver().insert(DataProvider.EX_CATEGORY_URI, values);

    }
    private void setDefaultInCategory() {

        ContentValues values = new ContentValues();
values.put(InCategoryTable.COL2, "Others");
        values.put(InCategoryTable.COL3, String.valueOf(BigDecimalCalculator.roundValue(0,
currencyCode)));
        getContentResolver().insert(DataProvider.IN_CATEGORY_URI, values);

    }

    @Override
    public void onBackPressed() {
        DrawerLayout drawer = findViewById(R.id.drawer_layout);
if (drawer.isDrawerOpen(GravityCompat.START)) {
drawer.closeDrawer(GravityCompat.START);
        } else {
            super.onBackPressed();
        }
    }

    @SuppressWarnings("StatementWithEmptyBody")
    @Override     public boolean onNavigationItemSelected(@NonNull
MenuItem item) {         // Handle navigation view item clicks here.
        switch (item.getItemId()) {

            case R.id.nav_main:
                Intent mainIntent = new Intent(MainActivity.this, MainActivity.class);
startActivity(mainIntent);
                finish();
break;             case
R.id.nav_expense:
                Intent expenseIntent = new Intent(MainActivity.this, ShowingExpensesActivity.class);
startActivity(expenseIntent);                break;             case R.id.nav_income:
                Intent incomeIntent = new Intent(MainActivity.this, ShowingIncomeActivity.class);
startActivity(incomeIntent);                break;             case R.id.nav_accounts:
                Intent accountIntent = new Intent(MainActivity.this, ShowingAccountsActivity.class);
startActivity(accountIntent);                break;             case R.id.nav_credit:
                Intent creditIntent = new Intent(MainActivity.this, ShowingCreditActivity.class);
startActivity(creditIntent);
                break;             case
R.id.nav_budget:
                Intent budgetIntent = new Intent(MainActivity.this, BudgetActivity.class);
startActivity(budgetIntent);                break;             case R.id.nav_income_category:
                Intent             incomeCategoryIntent             =             new             Intent(MainActivity.this,
IncomeCategoryActivity.class);
```

```
                startActivity(incomeCategoryIntent);
break;              case R.id.nav_transfer:
                Intent transferActivity = new Intent(MainActivity.this, TransferActivity.class);
startActivity(transferActivity);                  break;              case R.id.nav_settings:
                Intent prefsActivity = new Intent(MainActivity.this, PreferencesActivity.class);
startActivity(prefsActivity);                  break;              case R.id.nav_exit:
                AlertDialog.Builder builder = new AlertDialog.Builder(MainActivity.this);
builder.setMessage("Do you want to exit?");                  builder.setPositiveButton("Yes",
new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialogInterface, int i) {
dialogInterface.dismiss();
                        finish();
                    }
                });
                builder.setNegativeButton("No", new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialogInterface, int i) {
dialogInterface.dismiss();
                    }
                });
                AlertDialog dialog = builder.create();
dialog.show();
                break;
        }

        DrawerLayout drawer = findViewById(R.id.drawer_layout);
drawer.closeDrawer(GravityCompat.START);          return true;
    }

    @Override      public Loader<Cursor> onCreateLoader(int i,
Bundle bundle) {
        return new CursorLoader(this, DataProvider.TRANSACTION_URI,
null, null, null, null);
    }

    @Override
    public void onLoadFinished(Loader<Cursor> loader, Cursor cursor) {}

    @Override
    public void onLoaderReset(Loader<Cursor> loader) {}

    View.OnClickListener listener = new View.OnClickListener(){
        @Override
        public void onClick(View view){
            switch(view.getId()){
case R.id.fab:
```

```
                Intent addIntent = new Intent(MainActivity.this, AddingExpenseActivity.class);
                addIntent.putExtra(CHECK_IF, IS_FROM_MAIN_FAB);
startActivityForResult(addIntent, REQUEST_CODE);
break;                case R.id.netEarnings_monthly_cardView:
                MonthlyNetEarningsFragment fragment = new MonthlyNetEarningsFragment();
                Bundle b = new Bundle();
                double[] chart_yValues = new double[6];

                for(int i = 0; i<6; i++) {                    chart_yValues[i] =
getMonthlyNetEarnings(i).doubleValue();
                }

                b.putDoubleArray("Y_VALUES", chart_yValues);
                fragment.setArguments(b);
fragment.setCancelable(true);
                fragment.show(getFragmentManager(), "CHART");
break;                case R.id.earning_progress_view:
                Intent incomeIntent = new Intent(MainActivity.this, ShowingIncomeActivity.class);
startActivity(incomeIntent);                    break;
            }
        }
    };
```

```
}                                                                    implements
```

LoginActivity.java

```
package com.hongjolim.mfmanager;

import android.app.LoaderManager;
import android.content.CursorLoader;
import android.content.Loader; import
android.content.Intent; import
android.database.Cursor; import
android.os.Bundle;
import android.support.v7.app.AlertDialog; import
android.support.v7.app.AppCompatActivity; import
android.view.View; import android.widget.Button;
import android.widget.EditText; import
android.widget.TextView;
import android.widget.Toast;

import com.hongjolim.mfmanager.database.DataProvider;
import com.hongjolim.mfmanager.database.DataSource; import
com.hongjolim.mfmanager.model.User;

//this app only accepts 1 user account because it is run in local Database
public      class      LoginActivity      extends      AppCompatActivity
LoaderManager.LoaderCallbacks<Cursor>,
ShowingSecurityQuestionFragment.FromCallBack{

    //key for shared preference
    public static final String ENABLE_LOGIN = "ENABLE_LOGIN";

    private DataSource mDataSource;

    private User user;

    private EditText emailEdt, passwordEdt;

    private String email, password;

    private TextView forgotPassword;

    /**
* This variable is used as the key, when passing a security question string
* as an argument to the ShowingSecurityQuestionFragment.java class
     */
    public static final String SECURITY_QUESTION = "security_question";
```

```
        @Override
        protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.activity_login);

        mDataSource = new DataSource(this);

        /**
    * Find user data from the database,
    * if the system cannot find the user data,
    * the exception would be handled in DataSource class.
    * But just to be save, handle the exception here as well
    * */
        try {
            user = mDataSource.getUser();
        }catch(Exception e){
            Toast.makeText(this, R.string.login_no_user, Toast.LENGTH_SHORT).show();
        }

        emailEdt = findViewById(R.id.email);
        passwordEdt = findViewById(R.id.password);

        forgotPassword = findViewById(R.id.forgot_password);

        Button signIn = findViewById(R.id.sign_in_button);
        Button register = findViewById(R.id.register);

        signIn.setOnClickListener(listener);
    register.setOnClickListener(listener);
        forgotPassword.setOnClickListener(listener);

        //init the LoaderManager
        getLoaderManager().initLoader(0, null, this);

      }

      View.OnClickListener listener = new View.OnClickListener(){

        @Override
        public void onClick(View view){

            switch(view.getId()){
    case R.id.sign_in_button:
    checkValidity();                    break;
    case R.id.register:
                  //if user data already exists, show this alert dialog
    if(user!=null){
                        AlertDialog.Builder builder = new AlertDialog.Builder(LoginActivity.this);
    builder.setTitle(R.string.register_error);
```

```
                    builder.setMessage(R.string.sign_up_error_user_already_in_database);
builder.setPositiveButton("OK", null).create().show();                    return;
```

```
                }                                                                    new      account",
                Intent intent = new Intent(LoginActivity.this, RegisterActivity.class);
startActivity(intent);
                finish();
break;          case
R.id.forgot_password:
                //if the user forgets the password, send it to the user's email
//TO DO: let the user answer his/her security question
if(user!=null) {
                  promptSecurityQuestion();
                }else{
                   Toast.makeText(LoginActivity.this,      "Please     create     a
Toast.LENGTH_SHORT).show();
                }
break;
            }
        }
    };

    //if the user wants to sign in, this method is called
private void checkValidity(){
        email = emailEdt.getText().toString();
        password = passwordEdt.getText().toString();

        //get the user object and store it into the instance variable
user = mDataSource.getUser();

        AlertDialog.Builder builder = new AlertDialog.Builder(this);
builder.setTitle(R.string.sign_in_error);

        if(email.isEmpty()||password.isEmpty()){
builder.setMessage(R.string.sign_in_error_prompt_email_password);
builder.setCancelable(false).setPositiveButton("OK", null).create().show();
return;
        }

        //if there is no user in the database
if(user==null){
            builder.setMessage(R.string.sign_in_error_no_user_in_database);
builder.setCancelable(false).setPositiveButton("OK", null).create().show();
return;
        }

        if(user.getEmail().equals(email)){
            //there is the matching email and the password data
            if(user.getPassword().equals(password)){
                Intent mainIntent = new Intent(LoginActivity.this, MainActivity.class);
        startActivity(mainIntent);                finish();
            }else{
                //there is the matching email in the database but the password is different
builder.setMessage(R.string.sign_in_error_no_password);
                builder.setCancelable(false).setPositiveButton("OK", null).create().show();
            }
        }else{
            //if there is no user who has the same email in the database
builder.setMessage(R.string.sign_in_error_no_email_in_database);
builder.setCancelable(false).setPositiveButton("OK", null).create().show();
        }
    }

    /**
     * This method is to show the security question and get the answer to it.
* It is called when the user forgets the password or id
     */

    private void promptSecurityQuestion(){

        ShowingSecurityQuestionFragment fragment = new ShowingSecurityQuestionFragment();

        int index = user.getSecurityQNum();
        String securityQuestion = getResources().getStringArray(R.array.security_question)[index];

        Bundle bundle = new Bundle();
        bundle.putString(SECURITY_QUESTION, securityQuestion);
fragment.setArguments(bundle);
        fragment.show(getSupportFragmentManager(), "PROMPT_SECURITY_QUESTION");

    }

    private void reLoad(){
        getLoaderManager().restartLoader(0, null, this);
user = mDataSource.getUser();
    }

    @Override
    public void onResume(){
super.onResume();         mDataSource.open();
        reLoad();
    }

    @Override     public
void onPause(){
super.onPause();
        mDataSource.close();
    }
```

@Override

```
    public Loader<Cursor> onCreateLoader(int i, Bundle bundle) {
        return new CursorLoader(this, DataProvider.USER_URI,
null, null, null, null);
    }

    @Override
    public void onLoadFinished(android.content.Loader<Cursor> loader, Cursor cursor) {

    }

    @Override
    public void onLoaderReset(android.content.Loader<Cursor> loader) {

    }

    @Override
    public void securityQuestionAnswered(String securityAnswer) {

        if(securityAnswer.toLowerCase().equals(user.getSecuirtyAnswer().toLowerCase())){
            Toast.makeText(this,     "Your  password     is:     "+user.getPassword()
Toast.LENGTH_LONG).show();
        }else{
            Toast.makeText(this, "Wrong security answer!", Toast.LENGTH_SHORT).show();
        }
    }
}
```

<u>Activitymain.xml</u>

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v4.widget.DrawerLayout
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto"
xmlns:tools="http://schemas.android.com/tools"
android:id="@+id/drawer_layout"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:fitsSystemWindows="true"    tools:openDrawer="start">

    <include
        layout="@layout/app_bar_main"
android:layout_width="match_parent"
android:layout_height="match_parent" />
```

```
   <android.support.design.widget.NavigationView
android:id="@+id/nav_view"      android:layout_width="wrap_content"
android:layout_height="match_parent"
```

```
        android:layout_gravity="start"
android:fitsSystemWindows="true"
app:headerLayout="@layout/nav_header_main"
        app:menu="@menu/activity_main_drawer" />

</android.support.v4.widget.DrawerLayout>
```

Login.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"     android:gravity="center_horizontal"
android:theme="@style/AppTheme.NoActionBar"
    android:orientation="vertical"
    tools:context=".LoginActivity">

    <LinearLayout
android:padding="24dp"
android:gravity="center"
        android:background="@color/colorPrimaryDark"
android:layout_width="match_parent"         android:layout_height="wrap_content"
        android:orientation="vertical">

        <TextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_marginBottom="48dp"
android:layout_marginTop="48dp"
android:text="@string/app_name"
android:textAlignment="center"
android:textColor="@android:color/white"
            android:textSize="48sp"
            android:textStyle="italic|bold"/>

    </LinearLayout>

    <ScrollView
android:padding="16dp"
android:id="@+id/login_form"
android:layout_width="match_parent"
        android:layout_height="match_parent">

        <LinearLayout
android:layout_marginTop="24dp"
```

```
android:layout_width="match_parent"
android:layout_height="wrap_content"
```

```
            android:gravity="center"
            android:orientation="vertical">

            <android.support.design.widget.TextInputLayout
    android:layout_width="match_parent"
                android:layout_height="wrap_content">

                <AutoCompleteTextView
    android:id="@+id/email"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="@string/prompt_email"
    android:inputType="textEmailAddress"
    android:maxLines="1"
                    android:singleLine="true" />

            </android.support.design.widget.TextInputLayout>

            <android.support.design.widget.TextInputLayout
    android:layout_width="match_parent"
                android:layout_height="wrap_content">

                <EditText
    android:id="@+id/password"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="@string/prompt_password"
                    android:imeActionId="6"
    android:imeActionLabel="@string/action_sign_in"
    android:imeOptions="actionUnspecified"
    android:inputType="textPassword"                    android:maxLines="1"
                    android:singleLine="true" />

            </android.support.design.widget.TextInputLayout>

            <LinearLayout
    android:layout_marginTop="32dp"
    android:layout_width="match_parent"
    android:orientation="horizontal"
    android:gravity="center"
                android:layout_height="wrap_content">

                <Button
                    android:id="@+id/sign_in_button"
    android:layout_width="120dp"                    android:layout_marginEnd="16dp"
    android:layout_height="wrap_content"
    android:textAlignment="center"
```

```
android:backgroundTint="@android:color/white"
android:text="@string/sign_in"
          android:textSize="@dimen/text_size"/>

        <Button
android:id="@+id/register"
android:layout_width="120dp"
android:layout_height="wrap_content"
android:textAlignment="center"
          android:backgroundTint="@android:color/white"
          android:text="@string/register"
          android:textSize="@dimen/text_size"/>


      </LinearLayout>

      <TextView
android:id="@+id/forgot_password"
android:layout_marginTop="32dp"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="forgot password"
        android:layout_gravity="center"/>

      <TextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_gravity="center"
android:layout_marginTop="16dp"
        android:text="Disable log in function in Settings menu" />


    </LinearLayout>
  </ScrollView>
</LinearLayout>
```

9. Advantages & Limitations of System

Advantages:

User-friendly Financial Management:

The app will simplify personal finance management for users by centralizing income, expenses, budgets, and financial goals into a single platform.

Users can easily track and categorize their financial activities, which helps in improving financial awareness and control.

Budget Tracking and Planning:

Helps users set budgets for different categories (e.g., food, entertainment, utilities) and monitor their progress throughout the month or year.

The ability to track budgets in real-time can help users avoid overspending and stay financially disciplined.

Goal Setting and Progress Monitoring:

The app allows users to set financial goals (e.g., saving for a vacation or paying off debt) and track their progress toward those goals.

This can be motivating and lead to better savings habits or financial planning.

Reports and Insights:

Automatic generation of detailed reports (e.g., monthly spending, income vs. expenses, progress toward goals).

Visual reports (charts, graphs) can help users quickly understand their financial behavior, identify trends, and make informed decisions.

Expense Categorization:

Categorizing expenses allows users to see where their money is going (e.g., food, transportation, entertainment).

This can help identify areas where users can cut back or adjust spending to save more.

Reminders and Alerts:

The app can send reminders for upcoming bills, payments, or financial goals.

Alerts for overspending or when nearing budget limits can help users avoid financial misstep

Limitations:

Data Security and Privacy Concerns:

Storing sensitive financial data can be a security risk. If not properly encrypted, it could be vulnerable to hacking, data breaches, or misuse.

Users may be hesitant to input detailed financial information due to concerns about security and privacy, which may require extensive measures to secure.

Complexity for Beginners:

While the app can be user-friendly, users with little financial literacy might find some of the features (e.g., goal setting, budgeting, reports) overwhelming or hard to understand.

You might need to provide educational materials or tutorials to help beginners get started, which could add to the development time.

Dependence on User Input:

The app relies heavily on users to input accurate and timely data (transactions, budgets, goals). If users neglect to enter data regularly, the app's utility and accuracy will be diminished.

The app will be less effective if the user does not consistently update their financial information.

Limited Integration with Financial Institutions:

If your app doesn't integrate with banks or external financial APIs (e.g., for automatic transaction imports or syncing with bank accounts), users will need to manually enter transactions, which can be cumbersome.

A lack of integration with financial institutions or real-time data feeds may make the app less convenient for users who want an automated experience.

Device Dependency:

The app being Android-based means it will be limited to users with Android devices. If users switch to iOS or another platform, they will lose access to their data unless there's a crossplatform solution or cloud synchronization.

10.Reports Conclusion and References, Bibliography

Conclusion:

The Android project was developed to [state the primary purpose, e.g., enhance operational efficiency, improve customer engagement, or streamline internal processes]. Its core functionality includes [list key features or applications, e.g., user management, real-time updates, or secure payment integration], which will help achieve [specific goals, e.g., faster service delivery, cost reduction, or revenue growth].

This project positions us to leverage mobile technology for better scalability and improved outcomes..

References:

The some of the websites we used as reference:

www.google.com
chatgpt.com
youtube.com

□

□

□

41