MOBILE APPLICATION DEVELOPMENT LAB
MINI PROJECT REPORT

205001069
Nishaanth R

205001085
Sabarivasan Velayutham

205001097
Shajith Hameed

Expense Tracker Application

Problem Statement:

Develop a comprehensive Android application that serves as a user-friendly and efficient expense tracker. The primary goal is to help users manage their expenses, visualise spending patterns, and make informed financial decisions. The app should include features such as currency conversion and detailed expense reports for different time periods.

Functional Requirements:

Functional requirements include the features that the system provides to the user.

- **Today's Expense page**
  Contains reports of the current day and displays a list of expenses with the categories. Also displays the total expense.

- **Expense Reports page:**
  It is a summary page for all the expenses by the user.It can be filtered by week, month etc.

- Expense Categories:
  Shows all expense categories and allows the user to edit and add new categories.
- Settings:
  Choose the currency type.

Non-Functional Requirements:

These are requirements that are not functional in nature. Specifically, these are the constraints that the system must work within.

- **UI/UX:**
  The application must provide a clear user experience to enable seamless use.

- **Performance:**
  The system dynamically reflects any changes - addition, deletion or updation.

Functionalities:

- **Today's Page Module:**
  The application features a daily report showcasing a list of categorized expenses incurred on the current day, along with the total expenditure for quick reference.

- **Expense Report Module:**

  The application includes a consolidated summary page displaying all user expenses, with the flexibility to apply filters based on different time frames such as weeks or months.

- **Categories:**
  Displays a comprehensive list of expense categories and provides users with the ability to both edit existing categories and add new ones.
- **Settings:**
  Select the type of currency.

System Design for Expense Tracker App

Architecture:
The Expense Tracker app follows a client-server architecture.

Client Side (Android App):

Frontend: Developed using Android SDK and Kotlin/Java.
User Interface: Activities and Fragments for different screens (Expense logging, Reports, Settings).

Backend :
SQLite Database on the application with local storage.

Database Schema:

Expense Table:
ExpenseID (Primary Key)
CategoryID (Foreign Key referencing Category table)
Amount
Date

Category Table:
CategoryID (Primary Key)
CategoryName

Data Flow:

Expense Logging:
User inputs expense details in the app.
The app validates and sends the data to the SQLite database for insertion.

Reports:
The app queries the SQLite database to fetch expenses based on selected time periods (weeks, months).

Settings (Currency Selection):
User selects the currency type. The app updates the currency preference in the local SQLite database.

Code :

```
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.github.ematiyuk.expensetracer">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">

        <activity android:name=".activities.MainActivity"
            android:label="@string/app_name"
```

```
            android:launchMode="singleTop"
            android:screenOrientation="portrait">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>

                <category
android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>

        <activity android:name=".activities.SettingsActivity"
            android:screenOrientation="portrait"
            android:parentActivityName=".activities.MainActivity">
            <!-- Parent activity meta-data to support Android 4.0 and
lower -->
            <meta-data
android:name="android.support.PARENT_ACTIVITY"
                android:value=".activities.MainActivity"/>
        </activity>

        <activity android:name=".activities.CategoryEditActivity"
            android:screenOrientation="portrait"
            android:windowSoftInputMode="stateVisible"
            android:parentActivityName=".activities.MainActivity">
            <!-- Parent activity meta-data to support Android 4.0 and
lower -->
            <meta-data
android:name="android.support.PARENT_ACTIVITY"
                android:value=".activities.MainActivity"/>
        </activity>
```

```
        <activity android:name=".activities.ExpenseEditActivity"
            android:screenOrientation="portrait"
            android:windowSoftInputMode="stateVisible"
            android:parentActivityName=".activities.MainActivity">
            <!-- Parent activity meta-data to support Android 4.0 and
lower -->
            <meta-data
android:name="android.support.PARENT_ACTIVITY"
                android:value=".activities.MainActivity"/>
        </activity>

        <provider

android:authorities="com.github.ematiyuk.expensetracer.provider"
            android:name=".providers.ExpensesProvider" />

    </application>

</manifest>



package com.github.ematiyuk.expensetracer.activities;

import android.os.Bundle;
import android.support.annotation.LayoutRes;
import android.support.annotation.Nullable;
import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentManager;
```

```
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;

import com.github.ematiyuk.expensetracer.R;

public abstract class BaseFragmentActivity extends
AppCompatActivity {
    protected Toolbar mToolbar;

    @LayoutRes
    protected int getLayoutResId() {
        return R.layout.activity_base;
    }

    @Override
    protected void onCreate(@Nullable Bundle
savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(getLayoutResId());

        // Set a Toolbar to replace the ActionBar
        mToolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(mToolbar);
    }

    protected void insertFragment(Fragment fragment) {
        // Insert the fragment by replacing any existing fragment
        FragmentManager fragmentManager =
getSupportFragmentManager();
        fragmentManager.beginTransaction()
```

```
                .replace(R.id.content_frame, fragment)
                .commit();
    }
}
```

```
package com.github.ematiyuk.expensetracer.activities;

import android.os.Bundle;
import android.support.annotation.Nullable;
import android.support.v7.app.ActionBar;

import
com.github.ematiyuk.expensetracer.fragments.CategoryEditFrag
ment;

public class CategoryEditActivity extends BaseFragmentActivity {

    /* Important: use onCreate(Bundle savedInstanceState)
     * instead of onCreate(Bundle savedInstanceState,
PersistableBundle persistentState) */
    @Override
    protected void onCreate(@Nullable Bundle
savedInstanceState) {
        super.onCreate(savedInstanceState);

        insertFragment(new CategoryEditFragment());
        setupActionBar();
```

```
    }

    private void setupActionBar() {
        ActionBar actionBar = getSupportActionBar();
        if (actionBar != null) {
            // Show the Up button in the action bar (toolbar).
            actionBar.setDisplayHomeAsUpEnabled(true);
        }
    }
}
```

```
package com.github.ematiyuk.expensetracer.activities;

import android.content.Intent;
import android.content.res.Configuration;
import android.os.Bundle;
import android.support.annotation.IdRes;
import android.support.annotation.LayoutRes;
import android.support.annotation.Nullable;
import android.support.design.widget.NavigationView;
import android.support.v4.app.Fragment;
import android.support.v4.view.GravityCompat;
import android.support.v4.widget.DrawerLayout;
import android.support.v7.app.ActionBarDrawerToggle;
import android.view.MenuItem;
```

```
import
com.github.ematiyuk.expensetracer.fragments.CategoryFragment
;
import com.github.ematiyuk.expensetracer.R;
import
com.github.ematiyuk.expensetracer.fragments.ReportFragment;
import
com.github.ematiyuk.expensetracer.fragments.TodayFragment;

public class MainActivity extends BaseFragmentActivity {
    private DrawerLayout mDrawerLayout;
    private NavigationView mNavDrawer;
    private ActionBarDrawerToggle mDrawerToggle;

    @Override
    @LayoutRes
    protected int getLayoutResId() {
        return R.layout.activity_main;
    }

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

        mDrawerLayout = (DrawerLayout)
findViewById(R.id.drawer_layout);
        mNavDrawer = (NavigationView)
findViewById(R.id.nav_drawer);
        mDrawerToggle = setupDrawerToggle();
```

```
        // Tie DrawerLayout events to the ActionBarToggle
        mDrawerLayout.addDrawerListener(mDrawerToggle);

        // Setup drawer view
        setupDrawerContent(mNavDrawer);

        // Select TodayFragment on app start by default
        loadTodayFragment();
    }

    @Override
    protected void onPause() {
        super.onPause();
        closeNavigationDrawer();
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Pass the event to ActionBarDrawerToggle, if it returns
        // true, then it has handled the app icon touch event
        if (mDrawerToggle.onOptionsItemSelected(item)) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }

    @Override
    protected void onPostCreate(@Nullable Bundle
savedInstanceState) {
```

```
        super.onPostCreate(savedInstanceState);
        // Sync the toggle state after onRestoreInstanceState has
occurred.
        mDrawerToggle.syncState();
    }

    @Override
    public void onConfigurationChanged(Configuration newConfig)
{
        super.onConfigurationChanged(newConfig);
        // Pass any configuration change to the drawer toggle
        mDrawerToggle.onConfigurationChanged(newConfig);
    }

    @Override
    public void onBackPressed() {
        if (!closeNavigationDrawer()) {
            Fragment currentFragment =
getSupportFragmentManager()
                    .findFragmentById(R.id.content_frame);
            if (!(currentFragment instanceof TodayFragment)) {
                loadTodayFragment();
            } else {
                // If current fragment is TodayFragment then exit
                super.onBackPressed();
            }
        }
    }

    private ActionBarDrawerToggle setupDrawerToggle() {
```

```
        return new ActionBarDrawerToggle(this, mDrawerLayout,
mToolbar,
                R.string.drawer_open,  R.string.drawer_close);
    }

    private void setupDrawerContent(NavigationView
navigationView) {
        navigationView.setNavigationItemSelectedListener(
                new
NavigationView.OnNavigationItemSelectedListener() {
                    @Override
                    public boolean onNavigationItemSelected(MenuItem
menuItem) {
                        selectDrawerItem(menuItem);
                        return true;
                    }
                });
    }

    private void selectDrawerItem(MenuItem menuItem) {
        closeNavigationDrawer();
        switch(menuItem.getItemId()) {
            case R.id.nav_today:
                loadFragment(TodayFragment.class,
menuItem.getItemId(), menuItem.getTitle());
                break;
            case R.id.nav_report:
                loadFragment(ReportFragment.class,
menuItem.getItemId(), menuItem.getTitle());
                break;
```

```
            case R.id.nav_categories:
                loadFragment(CategoryFragment.class,
menuItem.getItemId(), menuItem.getTitle());
                break;
            case R.id.nav_settings:
                startActivity(new Intent(MainActivity.this,
SettingsActivity.class));
                break;
            default:
                loadFragment(TodayFragment.class,
menuItem.getItemId(), menuItem.getTitle());
        }
    }

    private boolean closeNavigationDrawer() {
        boolean drawerIsOpen =
mDrawerLayout.isDrawerOpen(GravityCompat.START);
        if (drawerIsOpen) {
            mDrawerLayout.closeDrawer(GravityCompat.START);
        }
        return drawerIsOpen;
    }

    public void hideNavigationBar() {
        closeNavigationDrawer();
    }

    private void loadFragment(Class fragmentClass, @IdRes int
navDrawerCheckedItemId,
                              CharSequence toolbarTitle) {
```

```
        Fragment fragment = null;
        try {
            fragment = (Fragment) fragmentClass.newInstance();
        } catch (Exception e) {
            e.printStackTrace();
        }

        insertFragment(fragment);

        // Highlight the selected item
        mNavDrawer.setCheckedItem(navDrawerCheckedItemId);
        // Set action bar title
        setTitle(toolbarTitle);
    }

    private void loadTodayFragment() {
        loadFragment(TodayFragment.class, R.id.nav_today,
                getResources().getString(R.string.nav_today));
    }
}


package com.github.ematiyuk.expensetracer.adapters;

import android.content.Context;
import android.database.Cursor;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;
```

```
import com.github.ematiyuk.expensetracer.R;
import
com.github.ematiyuk.expensetracer.providers.ExpensesContract;
import com.github.ematiyuk.expensetracer.utils.Utils;
import
com.twotoasters.sectioncursoradapter.SectionCursorAdapter;

public class SectionExpenseAdapter extends
SectionCursorAdapter {
    private String mCurrency;

    public SectionExpenseAdapter(Context context) {
        super(context, null, 0);
    }

    public void setCurrency(String currency) {
        mCurrency = currency;
        notifyDataSetChanged();
    }

    @Override
    protected Object getSectionFromCursor(Cursor cursor) {
        String dateStr =
cursor.getString(cursor.getColumnIndexOrThrow(ExpensesContra
ct.Expenses.DATE));
        return Utils.getSystemFormatDateString(mContext, dateStr);
    }

    @Override
```

```
    protected View newSectionView(Context context, Object item,
ViewGroup parent) {
        return
getLayoutInflater().inflate(R.layout.expense_report_section_head
er, parent, false);
    }

    @Override
    protected void bindSectionView(View convertView, Context
context, int position, Object item) {
        ((TextView) convertView).setText((String) item);
    }

    @Override
    protected View newItemView(Context context, Cursor cursor,
ViewGroup parent) {
        return getLayoutInflater().inflate(R.layout.expense_list_item,
parent, false);
    }

    @Override
    protected void bindItemView(View convertView, Context
context, Cursor cursor) {
        // Find fields to populate in inflated template
        TextView tvExpenseValue = (TextView)
convertView.findViewById(R.id.expense_value_text_view);
        TextView tvExpenseCurrency = (TextView)
convertView.findViewById(R.id.expense_currency_text_view);
```

```
        TextView tvExpenseCatName = (TextView)
convertView.findViewById(R.id.expense_category_name_text_vie
w);

        // Extract values from cursor
        float expValue =
cursor.getFloat(cursor.getColumnIndexOrThrow(ExpensesContra
ct.Expenses.VALUE));
        String categoryName =
cursor.getString(cursor.getColumnIndexOrThrow(ExpensesContra
ct.Categories.NAME));

        // Populate views with extracted values
        tvExpenseValue.setText(Utils.formatToCurrency(expValue));
        tvExpenseCatName.setText(categoryName);
        tvExpenseCurrency.setText(mCurrency);
    }
}



package com.github.ematiyuk.expensetracer.db;

import android.content.ContentValues;
import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

import com.github.ematiyuk.expensetracer.R;
```

```
import
com.github.ematiyuk.expensetracer.providers.ExpensesContract.
Categories;
import
com.github.ematiyuk.expensetracer.providers.ExpensesContract.
Expenses;

public class ExpenseDbHelper extends SQLiteOpenHelper {
    private static final int DATABASE_VERSION = 1;
    private static final String DATABASE_NAME =
"expense_tracer.db";

    public static final String CATEGORIES_TABLE_NAME =
"categories";
    public static final String EXPENSES_TABLE_NAME =
"expenses";

    private Context mContext;

    public ExpenseDbHelper(Context ctx) {
        super(ctx, DATABASE_NAME, null, DATABASE_VERSION);
        mContext = ctx;
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL(CategoriesTable.CREATE_TABLE_QUERY);
        // Fill the table with predefined values
        CategoriesTable.fillTable(db, mContext);
```

```
        db.execSQL(ExpensesTable.CREATE_TABLE_QUERY);
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int
newVersion) {
        /* Temporary (dummy) upgrade policy */
        db.execSQL(ExpensesTable.DELETE_TABLE_QUERY);
        db.execSQL(CategoriesTable.DELETE_TABLE_QUERY);
        onCreate(db);
    }

    private static final class CategoriesTable {
        public static final String CREATE_TABLE_QUERY =
                "CREATE TABLE " + CATEGORIES_TABLE_NAME + "
(" +
                Categories._ID + " INTEGER PRIMARY KEY
AUTOINCREMENT, " +
                Categories.NAME + " TEXT NOT NULL);";

        public static final String DELETE_TABLE_QUERY =
                "DROP TABLE IF EXISTS " +
CATEGORIES_TABLE_NAME + ";";

        public static void fillTable(SQLiteDatabase db, Context ctx) {
            String[] predefinedNames =
ctx.getResources().getStringArray(R.array.predefined_categories)
;
            ContentValues values = new ContentValues();
            for (String name : predefinedNames) {
```

```
            values.put(Categories.NAME, name);
            db.insert(CATEGORIES_TABLE_NAME, null, values);
        }
    }
}

private static final class ExpensesTable {
    public static final String CREATE_TABLE_QUERY =
            "CREATE TABLE " + EXPENSES_TABLE_NAME + " ("
+
            Expenses._ID + " INTEGER PRIMARY KEY
AUTOINCREMENT, " +
            Expenses.VALUE + " FLOAT NOT NULL, " +
            Expenses.DATE + " DATE NOT NULL, " +
            Expenses.CATEGORY_ID + " INTEGER NOT NULL);";

    public static final String DELETE_TABLE_QUERY =
            "DROP TABLE IF EXISTS " +
EXPENSES_TABLE_NAME + ";";
}
}
```

```
package com.github.ematiyuk.expensetracer.fragments;

import android.content.ContentUris;
import android.content.ContentValues;
import android.database.Cursor;
import android.net.Uri;
```

```
import android.os.Bundle;
import android.support.annotation.Nullable;
import android.support.v4.app.Fragment;
import android.support.v4.app.LoaderManager;
import android.support.v4.content.CursorLoader;
import android.support.v4.content.Loader;
import android.view.KeyEvent;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.view.ViewGroup;
import android.widget.EditText;
import android.widget.Toast;

import
com.github.ematiyuk.expensetracer.providers.ExpensesContract.
Categories;
import com.github.ematiyuk.expensetracer.R;

public class CategoryEditFragment extends Fragment implements
LoaderManager.LoaderCallbacks<Cursor> {
    public static final String EXTRA_EDIT_CATEGORY =
"com.github.ematiyuk.expensetracer.edit_category";

    private EditText mCatNameEditText;
    private long mExtraValue;

    @Override
```

```
    public void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        setHasOptionsMenu(true);
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup
container, Bundle savedInstanceState) {
        // Inflate layout for this fragment
        View rootView =
inflater.inflate(R.layout.fragment_category_edit, container, false);

        mCatNameEditText = (EditText)
rootView.findViewById(R.id.category_name_edit_text);

        // Set listener on Done (submit) button on keyboard clicked
        mCatNameEditText.setOnKeyListener(new
View.OnKeyListener() {
            @Override
            public boolean onKey(View view, int keyCode, KeyEvent
event) {
                if ((event.getAction() == KeyEvent.ACTION_DOWN) &&
(keyCode == KeyEvent.KEYCODE_ENTER)) {
                    checkEditTextForEmptyField(mCatNameEditText);
                    return true;
                }
                return false;
            }
        });
```

```
        return rootView;
    }

    @Override
    public void onActivityCreated(@Nullable Bundle
savedInstanceState) {
        super.onActivityCreated(savedInstanceState);

        mExtraValue =
getActivity().getIntent().getLongExtra(EXTRA_EDIT_CATEGORY,
-1);
        // Create a new category
        if (mExtraValue < 1) {
            getActivity().setTitle(R.string.add_category);

        // Edit existing category
        } else {
            getActivity().setTitle(R.string.edit_category);
            setCategoryData();
        }
    }

    @Override
    public void onCreateOptionsMenu(Menu menu, MenuInflater
inflater) {
        super.onCreateOptionsMenu(menu, inflater);
        inflater.inflate(R.menu.fragment_category_edit, menu);
    }
```

```
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case R.id.done_category_edit_menu_item:
                if (checkEditTextForEmptyField(mCatNameEditText)) {
                    // Create a new category
                    if (mExtraValue < 1) {
                        insertNewCategory();

                    // Edit existing category
                    } else {
                        updateCategory(mExtraValue);
                    }
                    getActivity().finish();
                }
                return true;
            default:
                return super.onOptionsItemSelected(item);
        }
    }

    private boolean checkEditTextForEmptyField(EditText editText)
{
        String inputText = editText.getText().toString().trim();
        if (inputText.length() == 0) {

editText.setError(getResources().getString(R.string.error_empty_fi
eld));
            mCatNameEditText.selectAll();
            return false;
```

```
        } else {
            return true;
        }
    }

    private void setCategoryData() {
        getLoaderManager().initLoader(0, null, this);
    }

    @Override
    public CursorLoader onCreateLoader(int id, Bundle args) {
        String[] projectionFields = new String[] {
                Categories._ID,
                Categories.NAME
        };

        Uri singleCategoryUri =
ContentUris.withAppendedId(Categories.CONTENT_URI,
mExtraValue);

        return new CursorLoader(getActivity(),
                singleCategoryUri,
                projectionFields,
                null,
                null,
                null
        );
    }

    @Override
```

```
    public void onLoadFinished(Loader<Cursor> loader, Cursor
data) {
        int categoryNameIndex =
data.getColumnIndex(Categories.NAME);
        data.moveToFirst();
        String categoryName = data.getString(categoryNameIndex);
        mCatNameEditText.setText(categoryName);
    }

    @Override
    public void onLoaderReset(Loader loader) {
        mCatNameEditText.setText("");
    }

    private void insertNewCategory() {
        ContentValues insertValues = new ContentValues();
        insertValues.put(Categories.NAME,
mCatNameEditText.getText().toString());

        getActivity().getContentResolver().insert(
                Categories.CONTENT_URI,
                insertValues
        );

        Toast.makeText(getActivity(),
                getResources().getString(R.string.category_added),
                Toast.LENGTH_SHORT).show();
    }

    private void updateCategory(long id) {
```

```
        ContentValues updateValues = new ContentValues();
        updateValues.put(Categories.NAME,
mCatNameEditText.getText().toString());

        Uri categoryUri =
ContentUris.withAppendedId(Categories.CONTENT_URI, id);

        getActivity().getContentResolver().update(
                categoryUri,
                updateValues,
                null,
                null
        );

        Toast.makeText(getActivity(),
                getResources().getString(R.string.category_updated),
                Toast.LENGTH_SHORT).show();
    }
}


package com.github.ematiyuk.expensetracer.fragments;

import android.app.DatePickerDialog;
import android.app.Dialog;
import android.os.Bundle;
import android.support.v4.app.DialogFragment;

import java.util.Calendar;
```

```
public class DatePickerFragment extends DialogFragment {
    private static DatePickerDialog.OnDateSetListener mListener;

    @Override
    public Dialog onCreateDialog(Bundle savedInstanceState) {
        // Use the current date as the default date in the picker
        final Calendar c = Calendar.getInstance();
        int year = c.get(Calendar.YEAR);
        int month = c.get(Calendar.MONTH);
        int day = c.get(Calendar.DAY_OF_MONTH);

        // Create a new instance of DatePickerDialog and return it
        return new DatePickerDialog(getActivity(), mListener, year,
month, day);
    }

    public static DatePickerFragment
newInstance(DatePickerDialog.OnDateSetListener listener) {
        mListener = listener;
        return new DatePickerFragment();
    }
}

package com.github.ematiyuk.expensetracer.fragments;

import android.content.ContentUris;
import android.content.ContentValues;
import android.database.Cursor;
import android.net.Uri;
import android.os.Bundle;
```

```
import android.support.annotation.Nullable;
import android.support.v4.app.Fragment;
import android.support.v4.app.LoaderManager;
import android.support.v4.content.CursorLoader;
import android.support.v4.content.Loader;
import android.support.v4.widget.SimpleCursorAdapter;
import android.support.v7.widget.AppCompatSpinner;
import android.view.KeyEvent;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.EditText;
import android.widget.Toast;

import
com.github.ematiyuk.expensetracer.providers.ExpensesContract.
Categories;
import
com.github.ematiyuk.expensetracer.providers.ExpensesContract.
Expenses;
import com.github.ematiyuk.expensetracer.R;
import com.github.ematiyuk.expensetracer.utils.Utils;

import java.util.ArrayList;
import java.util.Date;
```

```
public class ExpenseEditFragment extends Fragment implements
LoaderManager.LoaderCallbacks<Cursor> {
    public static final String EXTRA_EDIT_EXPENSE =
"com.github.ematiyuk.expensetracer.edit_expense";

    private static final int EXPENSE_LOADER_ID = 1;
    private static final int CATEGORIES_LOADER_ID = 0;

    private EditText mExpValueEditText;
    private AppCompatSpinner mCategorySpinner;
    private SimpleCursorAdapter mAdapter;
    private View mCatProgressBar;
    private long mExtraValue;
    private long mExpenseCategoryId = -1;

    @Override
    public void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        setHasOptionsMenu(true);
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup
container, Bundle savedInstanceState) {
        // Inflate layout for this fragment
        View rootView =
inflater.inflate(R.layout.fragment_expense_edit, container, false);
```

```
        mExpValueEditText = (EditText)
rootView.findViewById(R.id.expense_value_edit_text);
        mCatProgressBar =
rootView.findViewById(R.id.cat_select_progress_bar);
        mCategorySpinner = (AppCompatSpinner)
rootView.findViewById(R.id.category_choose_spinner);

        setEditTextDefaultValue();

        // Set listener on Done (submit) button on keyboard clicked
        mExpValueEditText.setOnKeyListener(new
View.OnKeyListener() {
            @Override
            public boolean onKey(View view, int keyCode, KeyEvent
event) {
                if ((event.getAction() == KeyEvent.ACTION_DOWN) &&
(keyCode == KeyEvent.KEYCODE_ENTER)) {
                    checkValueFieldForIncorrectInput();
                    return true;
                }
                return false;
            }
        });

        mCategorySpinner.setOnItemSelectedListener(new
AdapterView.OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> parent, View
view, int pos, long id) {
                mExpenseCategoryId = id;
```

```
            }

            @Override
            public void onNothingSelected(AdapterView<?> parent) {
            }
        });

        return rootView;
    }

    @Override
    public void onActivityCreated(@Nullable Bundle
savedInstanceState) {
        super.onActivityCreated(savedInstanceState);

        mAdapter = new SimpleCursorAdapter(getActivity(),
                android.R.layout.simple_spinner_item,
                null,
                new String[] { Categories.NAME },
                new int[] { android.R.id.text1 },
                0);
        // Specify the layout to use when the list of choices appears

mAdapter.setDropDownViewResource(android.R.layout.simple_s
pinner_dropdown_item);
        // Apply the adapter to the spinner
        mCategorySpinner.setAdapter(mAdapter);
```

```
        mExtraValue =
getActivity().getIntent().getLongExtra(EXTRA_EDIT_EXPENSE,
-1);
        // Create a new expense
        if (mExtraValue < 1) {
            getActivity().setTitle(R.string.add_expense);
            loadCategories();

            // Edit existing expense
        } else {
            getActivity().setTitle(R.string.edit_expense);
            loadExpenseData();
        }
    }

    @Override
    public void onCreateOptionsMenu(Menu menu, MenuInflater
inflater) {
        super.onCreateOptionsMenu(menu, inflater);
        inflater.inflate(R.menu.fragment_expense_edit, menu);
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case R.id.done_expense_edit_menu_item:
                if (checkForIncorrectInput()) {
                    // Create a new expense
                    if (mExtraValue < 1) {
                        insertNewExpense();
```

```
                // Edit existing expense
                } else {
                    updateExpense(mExtraValue);
                }
                getActivity().finish();
            }
            return true;
        default:
            return super.onOptionsItemSelected(item);
    }
}

private boolean checkForIncorrectInput() {
    if (!checkValueFieldForIncorrectInput()) {
        mExpValueEditText.selectAll();
        return false;
    }
    // Future check of other fields

    return true;
}

private boolean checkValueFieldForIncorrectInput() {
    String etValue = mExpValueEditText.getText().toString();
    try {
        if (etValue.length() == 0) {

mExpValueEditText.setError(getResources().getString(R.string.err
or_empty_field));
```

```
                return false;
            } else if (Float.parseFloat(etValue) == 0.00f) {

mExpValueEditText.setError(getResources().getString(R.string.err
or_zero_value));
                return false;
            }
        } catch (Exception e) {

mExpValueEditText.setError(getResources().getString(R.string.err
or_incorrect_input));
            return false;
        }
        return true;
    }

    private void loadCategories() {
        // Show the progress bar next to category spinner
        mCatProgressBar.setVisibility(View.VISIBLE);


getLoaderManager().initLoader(CATEGORIES_LOADER_ID, null,
this);
    }

    private void loadExpenseData() {
        getLoaderManager().initLoader(EXPENSE_LOADER_ID,
null, this);
        loadCategories();
    }
```

```
    private void setEditTextDefaultValue() {
        mExpValueEditText.setText(String.valueOf(0));
        mExpValueEditText.selectAll();
    }

    @Override
    public CursorLoader onCreateLoader(int id, Bundle args) {
        String[] projectionFields = null;
        Uri uri = null;
        switch (id) {
            case EXPENSE_LOADER_ID:
                projectionFields = new String[] {
                        Expenses._ID,
                        Expenses.VALUE,
                        Expenses.CATEGORY_ID
                };

                uri =
ContentUris.withAppendedId(Expenses.CONTENT_URI,
mExtraValue);
                break;
            case CATEGORIES_LOADER_ID:
                projectionFields = new String[] {
                        Categories._ID,
                        Categories.NAME
                };

                uri = Categories.CONTENT_URI;
                break;
```

```
        }

        return new CursorLoader(getActivity(),
                uri,
                projectionFields,
                null,
                null,
                null
        );
    }

    @Override
    public void onLoadFinished(Loader<Cursor> loader, Cursor
data) {
        switch (loader.getId()) {
            case EXPENSE_LOADER_ID:
                int expenseValueIndex =
data.getColumnIndex(Expenses.VALUE);
                int expenseCategoryIdIndex =
data.getColumnIndex(Expenses.CATEGORY_ID);

                data.moveToFirst();
                mExpenseCategoryId =
data.getLong(expenseCategoryIdIndex);
                updateSpinnerSelection();


mExpValueEditText.setText(String.valueOf(data.getFloat(expense
ValueIndex)));
                mExpValueEditText.selectAll();
```

```
            break;
        case CATEGORIES_LOADER_ID:
            // Hide the progress bar next to category spinner
            mCatProgressBar.setVisibility(View.GONE);

            if (null == data || data.getCount() < 1) {
                mExpenseCategoryId = -1;
                // Fill the spinner with default values
                ArrayList<String> defaultItems = new ArrayList<>();

defaultItems.add(getResources().getString(R.string.no_categories
_string));

                ArrayAdapter<String> tempAdapter = new
ArrayAdapter<String>(getActivity(),
                        android.R.layout.simple_spinner_item,
                        defaultItems);
                mCategorySpinner.setAdapter(tempAdapter);
                // Disable the spinner
                mCategorySpinner.setEnabled(false);
            } else {
                // Set the original adapter
                mCategorySpinner.setAdapter(mAdapter);
                // Update spinner data
                mAdapter.swapCursor(data);
                // Enable the spinner
                mCategorySpinner.setEnabled(true);
                updateSpinnerSelection();
            }
            break;
```

```
        }
    }

    @Override
    public void onLoaderReset(Loader<Cursor> loader) {
        switch (loader.getId()) {
            case EXPENSE_LOADER_ID:
                mExpenseCategoryId = -1;
                setEditTextDefaultValue();
                break;
            case CATEGORIES_LOADER_ID:
                mAdapter.swapCursor(null);
                break;
        }
    }

    private void updateSpinnerSelection() {
        mCategorySpinner.setSelection(0);
        for (int pos = 0; pos < mAdapter.getCount(); ++pos) {
            if (mAdapter.getItemId(pos) == mExpenseCategoryId) {
                // Set spinner item selected according to the value from
db
                mCategorySpinner.setSelection(pos);
                break;
            }
        }
    }

    private void insertNewExpense() {
        ContentValues insertValues = new ContentValues();
```

```
        insertValues.put(Expenses.VALUE,
Float.parseFloat(mExpValueEditText.getText().toString()));
        insertValues.put(Expenses.DATE, Utils.getDateString(new
Date())); // Put current date (today)
        insertValues.put(Expenses.CATEGORY_ID,
mExpenseCategoryId);

        getActivity().getContentResolver().insert(
                Expenses.CONTENT_URI,
                insertValues
        );

        Toast.makeText(getActivity(),
                getResources().getString(R.string.expense_added),
                Toast.LENGTH_SHORT).show();
    }

    private void updateExpense(long id) {
        ContentValues updateValues = new ContentValues();
        updateValues.put(Expenses.VALUE,
Float.parseFloat(mExpValueEditText.getText().toString()));
        updateValues.put(Expenses.CATEGORY_ID,
mExpenseCategoryId);

        Uri expenseUri =
ContentUris.withAppendedId(Expenses.CONTENT_URI, id);

        getActivity().getContentResolver().update(
                expenseUri,
                updateValues,
```

```
                null,
                null
        );

        Toast.makeText(getActivity(),
                getResources().getString(R.string.expense_updated),
                Toast.LENGTH_SHORT).show();
    }
}
```

```
package com.github.ematiyuk.expensetracer.utils;

import android.content.Context;

import java.text.NumberFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Locale;

public class Utils {
    public static String getSystemFormatDateString(Context
context, Date date) {
        java.text.DateFormat dateFormat =
android.text.format.DateFormat.getDateFormat(context);
        return  dateFormat.format(date);
    }
```

```
    public static String getSystemFormatDateString(Context
context, String dateString) {
        java.text.DateFormat dateFormat =
android.text.format.DateFormat.getDateFormat(context);
        return  dateFormat.format(stringToDate(dateString));
    }

    public static String getDateString(Date date) {
        SimpleDateFormat dateFormat = new
SimpleDateFormat("MM/dd/yy", Locale.US);
        try {
            return dateFormat.format(date);
        } catch (Exception pe) {
            pe.printStackTrace();
            return "no_date";
        }
    }

    private static Date stringToDate(String dateString) {
        SimpleDateFormat dateFormat = new
SimpleDateFormat("MM/dd/yy", Locale.US);
        try {
            return dateFormat.parse(dateString);
        } catch (ParseException pe) {
            pe.printStackTrace();
            return null;
        }
    }

    public static String formatToCurrency(float value) {
```

```
        final NumberFormat numberFormat =
NumberFormat.getNumberInstance();
        numberFormat.setMaximumFractionDigits(2);
        numberFormat.setMinimumFractionDigits(2);
        return numberFormat.format(value);
    }
}
```

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:id="@+id/top_parent"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true">

    <!-- The ActionBar (Toolbar) displayed at the top -->
    <include
        android:id="@+id/toolbar"
        layout="@layout/toolbar" />

    <android.support.v4.widget.DrawerLayout
        android:id="@+id/drawer_layout"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_below="@id/toolbar">

        <!-- The main content view where fragments are loaded -->
```

```
        <FrameLayout
            android:id="@+id/content_frame"
            android:layout_width="match_parent"
            android:layout_height="match_parent" />

        <android.support.design.widget.NavigationView
            android:id="@+id/nav_drawer"
            android:layout_width="wrap_content"
            android:layout_height="match_parent"
            android:layout_gravity="start"
            android:background="@android:color/white"
            app:menu="@menu/drawer_view"/>
    </android.support.v4.widget.DrawerLayout>

</RelativeLayout>



<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" >

    <TextView
        android:id="@+id/category_name_list_item"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
```

```
        android:padding="10dp"
        android:textSize="19sp" />

</RelativeLayout>


<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" >

    <TextView
        android:id="@+id/expense_value_text_view"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:paddingLeft="6dp"
        android:paddingRight="6dp"
        android:paddingTop="6dp"
        android:textSize="22sp" />

    <TextView
        android:id="@+id/expense_currency_text_view"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:paddingLeft="6dp"
        android:paddingRight="6dp"
        android:paddingTop="6dp"
```

```
        android:paddingBottom="0dp"
        android:layout_toRightOf="@id/expense_value_text_view"
        android:layout_toEndOf="@id/expense_value_text_view"
        android:layout_alignParentTop="true"
        android:layout_alignParentEnd="true"
        android:layout_alignParentRight="true"
        android:gravity="end"
        android:textSize="20sp" />

    <TextView
        android:id="@+id/expense_category_name_text_view"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:layout_below="@id/expense_value_text_view"
        android:paddingLeft="6dp"
        android:paddingRight="6dp"
        android:paddingBottom="4dp"
        android:textSize="14sp" />

</RelativeLayout>


<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
```

```
<LinearLayout
    android:id="@+id/categories_progress_bar"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_gravity="center"
    android:gravity="center">

    <ProgressBar
        style="@style/Base.Widget.AppCompat.ProgressBar"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:indeterminate="true"/>

</LinearLayout>

<ListView
    android:id="@+id/categories_list_view"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />

<LinearLayout
    android:id="@+id/categories_empty_list_view"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:gravity="center">

    <Button
        android:id="@+id/add_category_button_if_empty_list"
        android:layout_width="wrap_content"
```

```
            android:layout_height="wrap_content"
            android:text="@string/add_category"/>

    </LinearLayout>

</LinearLayout>


<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:layout_weight="0">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical"
            android:layout_marginTop="10dp"
            android:layout_marginLeft="10dp"
            android:layout_marginRight="10dp"
            android:layout_marginBottom="0dp">

            <TextView
```

```
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="2dp"
        android:textSize="18sp"
        android:text="@string/total_string" />

    <View
        android:layout_width="match_parent"
        android:layout_height="1dip"
        android:background="#000000" />

</LinearLayout>

<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="0dp"
    android:layout_marginBottom="0dp"
    android:layout_marginLeft="10dp"
    android:layout_marginRight="10dp">

    <TextView
        android:id="@+id/expenses_report_total_text_view"
        android:layout_width="wrap_content"
        android:layout_height="55dp"
        android:gravity="start|center_vertical"
        android:paddingRight="4dp"
        android:paddingEnd="4dp"
        android:paddingLeft="2dp"
        android:paddingStart="2dp"
```

```
            android:textSize="30sp"
            android:singleLine="true"/>

        <TextView

android:id="@+id/expenses_report_total_currency_text_view"
            android:layout_width="wrap_content"
            android:layout_height="55dp"

android:layout_toEndOf="@id/expenses_report_total_text_view"

android:layout_toRightOf="@id/expenses_report_total_text_view"
            android:gravity="end|center_vertical"
            android:paddingRight="4dp"
            android:paddingEnd="4dp"
            android:paddingLeft="0dp"
            android:paddingStart="0dp"
            android:textSize="22sp"
            android:singleLine="true"
            android:layout_alignParentEnd="true"
            android:layout_alignParentRight="true"/>

    </RelativeLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:layout_marginTop="0dp"
        android:layout_marginLeft="10dp"
```

```
        android:layout_marginRight="10dp"
        android:layout_marginBottom="2dp">

        <View
            android:layout_width="match_parent"
            android:layout_height="1dip"
            android:background="#000000" />

    </LinearLayout>

</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:orientation="vertical"
    android:layout_marginLeft="10dp"
    android:layout_marginRight="10dp"
    android:layout_weight="1">

    <LinearLayout
        android:id="@+id/expenses_report_progress_bar"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_gravity="center"
        android:gravity="center">

        <ProgressBar
            style="@style/Base.Widget.AppCompat.ProgressBar"
            android:layout_width="wrap_content"
```

```
            android:layout_height="wrap_content"
            android:indeterminate="true"/>

        </LinearLayout>

        <ListView
            android:id="@+id/expenses_report_list_view"
            android:layout_width="match_parent"
            android:layout_height="match_parent"/>

        <LinearLayout
            android:id="@+id/expenses_report_empty_list_view"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:orientation="vertical"
            android:gravity="center">

            <TextView

android:id="@+id/expenses_report_empty_list_text_view"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:textSize="20sp"
                android:text="@string/no_expenses"/>

        </LinearLayout>

    </LinearLayout>

</LinearLayout>
```

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="predefined_categories">
        <item>@string/cat_general</item>
        <item>@string/cat_clothes</item>
        <item>@string/cat_entertainment</item>
        <item>@string/cat_food</item>
        <item>@string/cat_health</item>
        <item>@string/cat_household</item>
        <item>@string/cat_hygiene</item>
        <item>@string/cat_pets</item>
        <item>@string/cat_presents</item>
        <item>@string/cat_sports</item>
        <item>@string/cat_transportation</item>
    </string-array>
</resources>
```

```
<resources>
    <string name="app_name">ExpenseTracer</string>

    <string name="nav_today">Today</string>
    <string name="nav_report">Expense reports</string>
    <string name="nav_categories">Expense categories</string>
    <string name="nav_settings">Settings</string>

    <string name="drawer_open">Open navigation drawer</string>
    <string name="drawer_close">Close navigation
drawer</string>
```

```
    <string name="add_category">Add a category</string>
    <string name="edit_category">Edit category</string>
    <string name="delete_category">Delete category</string>
    <string name="delete_cat_dialog_msg">All associated
expenses will be deleted as well. Are you sure you want to delete
the category?</string>
    <string name="category_deleted">Category deleted</string>
    <string name="category_added">Category added</string>
    <string name="category_updated">Category updated</string>
    <string name="category_name_string">Category
name</string>

    <string name="today_total">Today\'s total</string>
    <string name="today_expenses">Today\'s expenses</string>

    <string name="add_expense">New expense</string>
    <string name="edit_expense">Edit expense</string>
    <string name="delete_expense">Delete expense</string>
    <string name="delete_exp_dialog_msg">Are you sure you
want to delete the expense?</string>
    <string name="expense_deleted">Expense deleted</string>
    <string name="expense_added">Expense added</string>
    <string name="expense_updated">Expense updated</string>
    <string
name="expense_category_name_string">Category</string>
    <plurals name="expenses_deleted_plurals_msg">
        <item quantity="one">%d expense deleted</item>
        <item quantity="other">%d expenses deleted</item>
    </plurals>
```

```
    <string name="delete_string">Delete</string>
    <string name="done_string">Done</string>
    <string name="error_empty_field">Empty field</string>
    <string name="error_zero_value">Zero value is invalid</string>
    <string name="error_incorrect_input">Incorrect input</string>

    <string name="filter_expenses">Filter</string>
    <string name="filter_option_today">By Today</string>
    <string name="filter_option_week">By Week</string>
    <string name="filter_option_month">By Month</string>
    <string name="filter_option_date">By Date</string>
    <string name="filter_option_range">By Date range</string>
    <string name="filter_todays_expenses">Today\'s
expenses</string>
    <string name="filter_weeks_expenses">Week\'s
expenses</string>
    <string name="filter_months_expenses">Month\'s
expenses</string>
    <string name="filter_date_expenses">For %s</string>
    <string name="filter_date_range_expenses">%1$s -
%2$s</string>

    <string name="default_string">Default</string>
    <string name="no_categories_string">&amp;#60;No
categories&amp;#62;</string>
    <string name="no_expenses">There are no expenses.</string>
    <string name="total_string">Total</string>

    <string name="pref_currency_title">Currency</string>
```

```
<string name="pref_currency_default">USD</string>
<string-array name="pref_currency_list_titles">
    <item>Indian Rupee</item>
    <item>US Dollar</item>
    <item>Euro</item>
    <item>British Pound</item>
    <item>Canadian Dollar</item>
    <item>Australian Dollar</item>
    <item>Ukrainian Hryvnia</item>
    <item>Russian Rouble</item>
</string-array>
<string-array name="pref_currency_list_values">
    <item>INR</item>
    <item>USD</item>
    <item>EUR</item>
    <item>GBP</item>
    <item>CAD</item>
    <item>AUD</item>
    <item>UAH</item>
    <item>RUB</item>
</string-array>

<!-- Predefined expense categories -->
<string name="cat_general">General</string>
<string name="cat_clothes">Clothes</string>
<string name="cat_entertainment">Entertainment</string>
<string name="cat_food">Food</string>
<string name="cat_health">Health</string>
<string name="cat_household">Household</string>
<string name="cat_hygiene">Hygiene</string>
```

```
    <string name="cat_pets">Pets</string>
    <string name="cat_presents">Presents</string>
    <string name="cat_sports">Sports</string>
    <string name="cat_transportation">Transportation</string>

</resources>
```

Output Screenshots:

Today's total

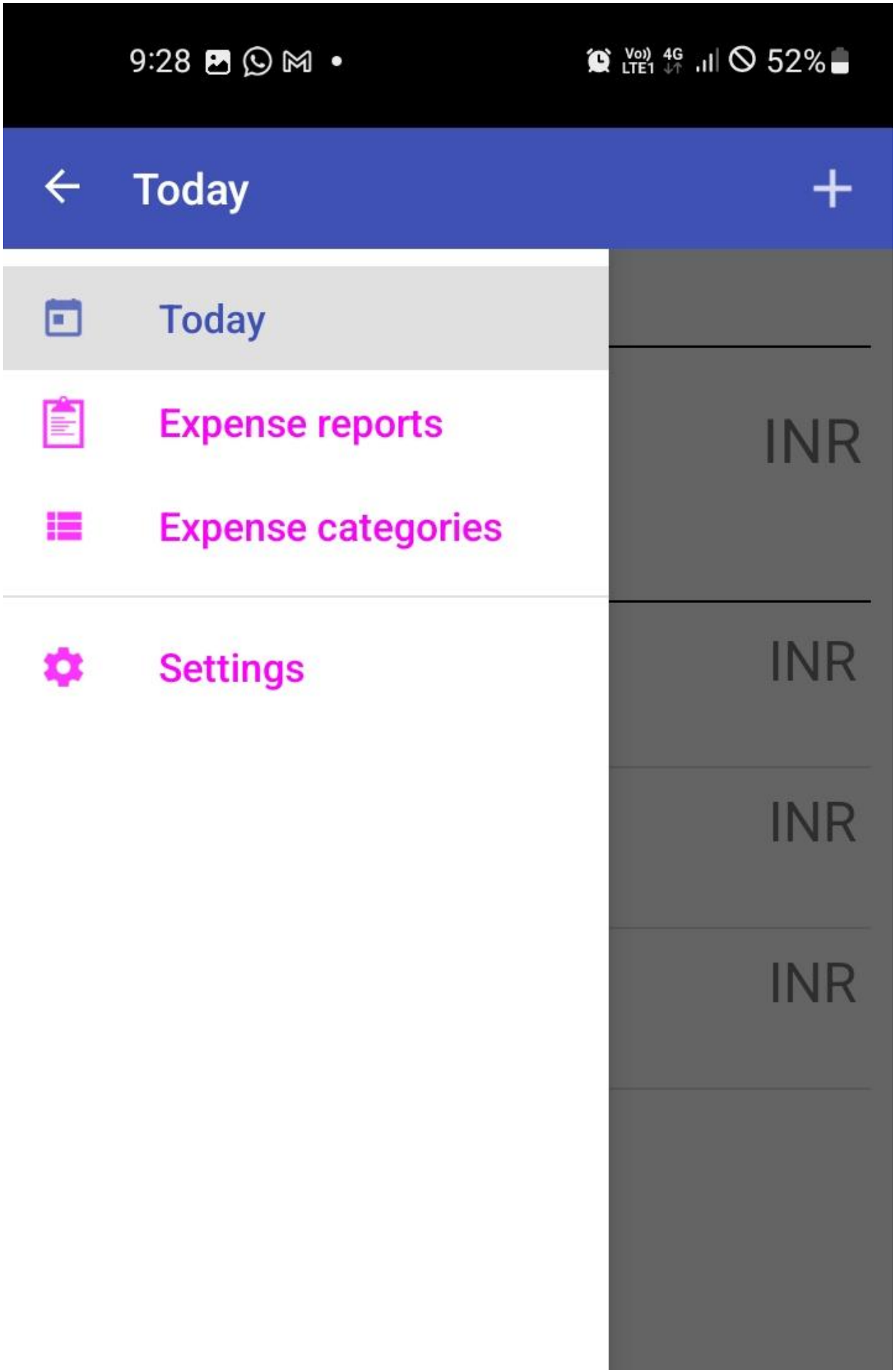
5,032.00 INR

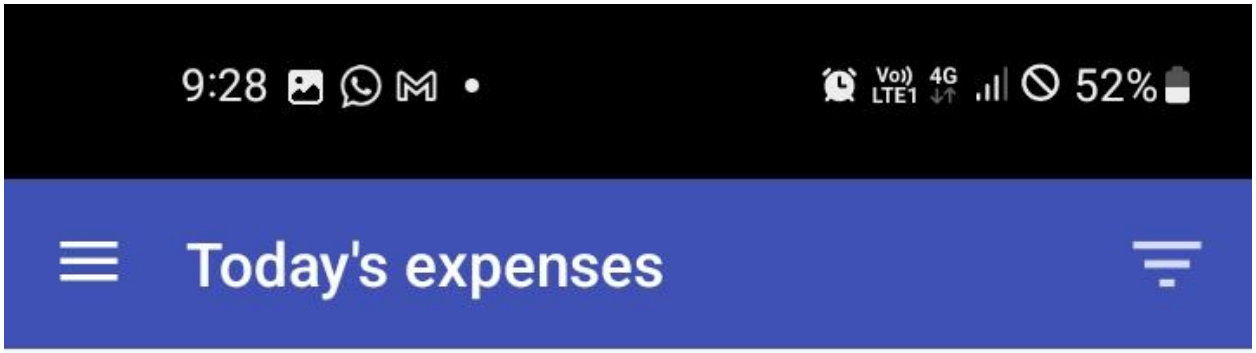
Today's expenses

123.00 INR
General

4,789.00 INR
Hygiene

120.00 INR
Hygiene


9:28
52%
Today
Today
Expense reports
Expense categories
Settings
INR
INR
INR
INR



Total

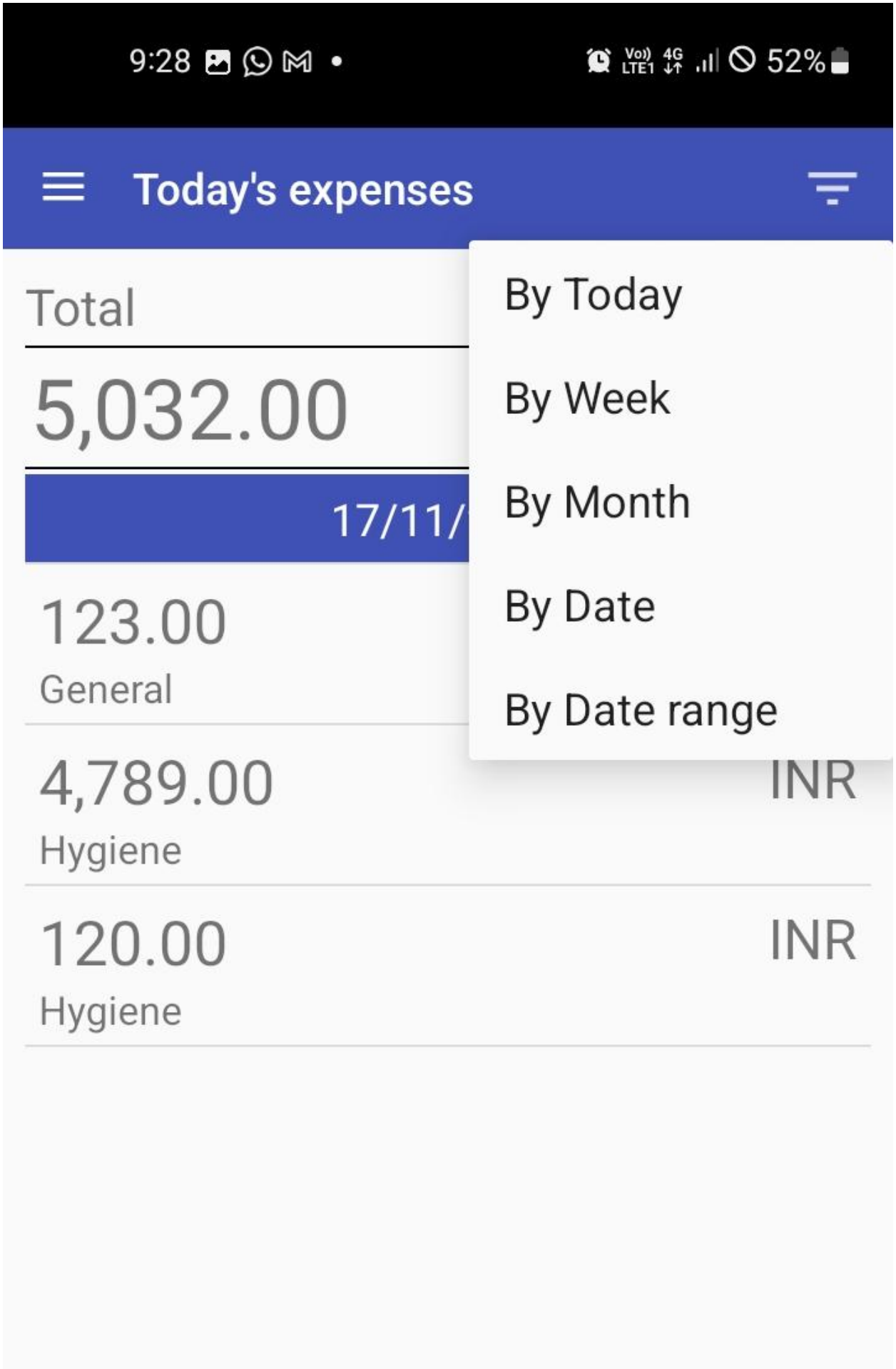
5,032.00 INR

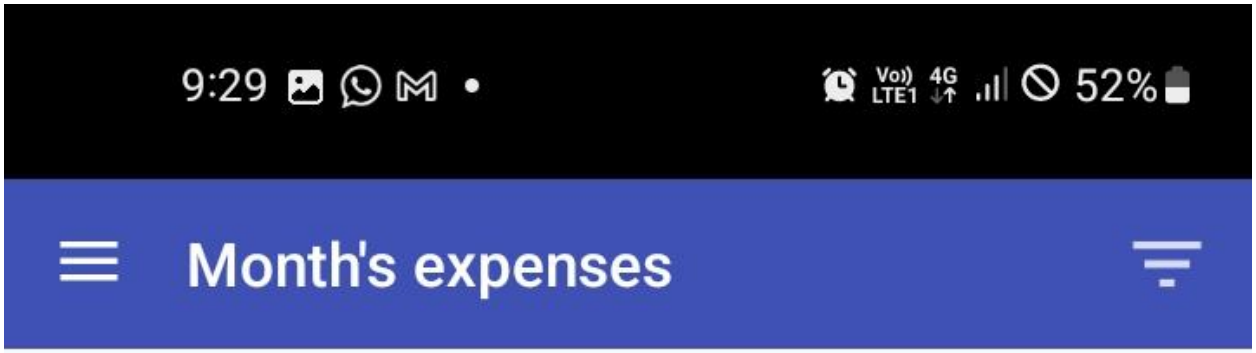
17/11/2023

123.00 INR
General

4,789.00 INR
Hygiene

120.00 INR
Hygiene


9:28
52%
Today's expenses
Total
5,032.00
17/11/
By Today
By Week
By Month
By Date
By Date range
123.00
General
4,789.00
INR
Hygiene
120.00
INR
Hygiene



Month's expenses

Total

7,262.00 INR

16/11/2023

230.00 INR
General

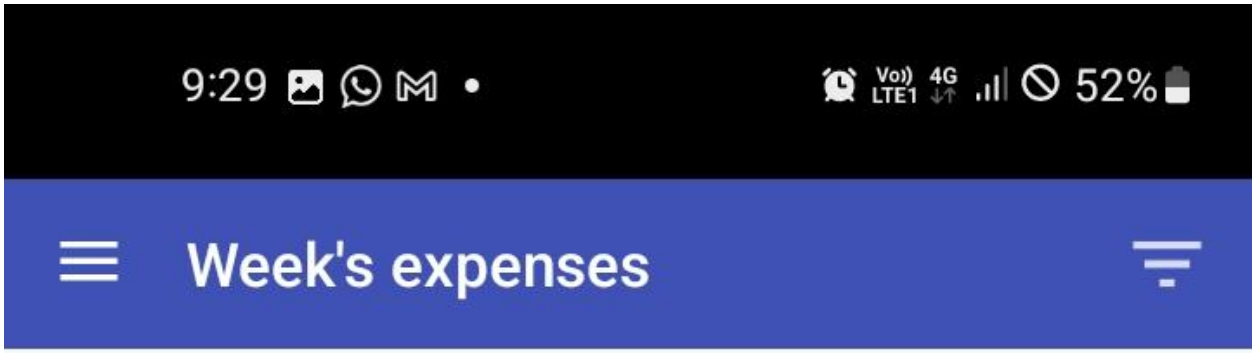
1,000.00 INR
Pets

1,000.00 INR
Sabari

17/11/2023

123.00 INR
General

4,789.00 INR



Week's expenses

Total

7,262.00 INR

16/11/2023

230.00 INR
General

1,000.00 INR
Pets

1,000.00 INR
Sabari

17/11/2023

123.00 INR
General

4,789.00 INR

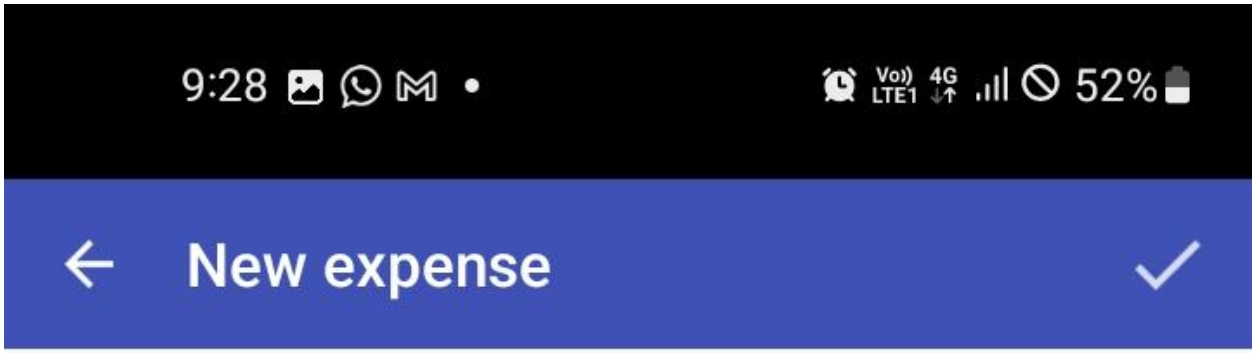
Today's total

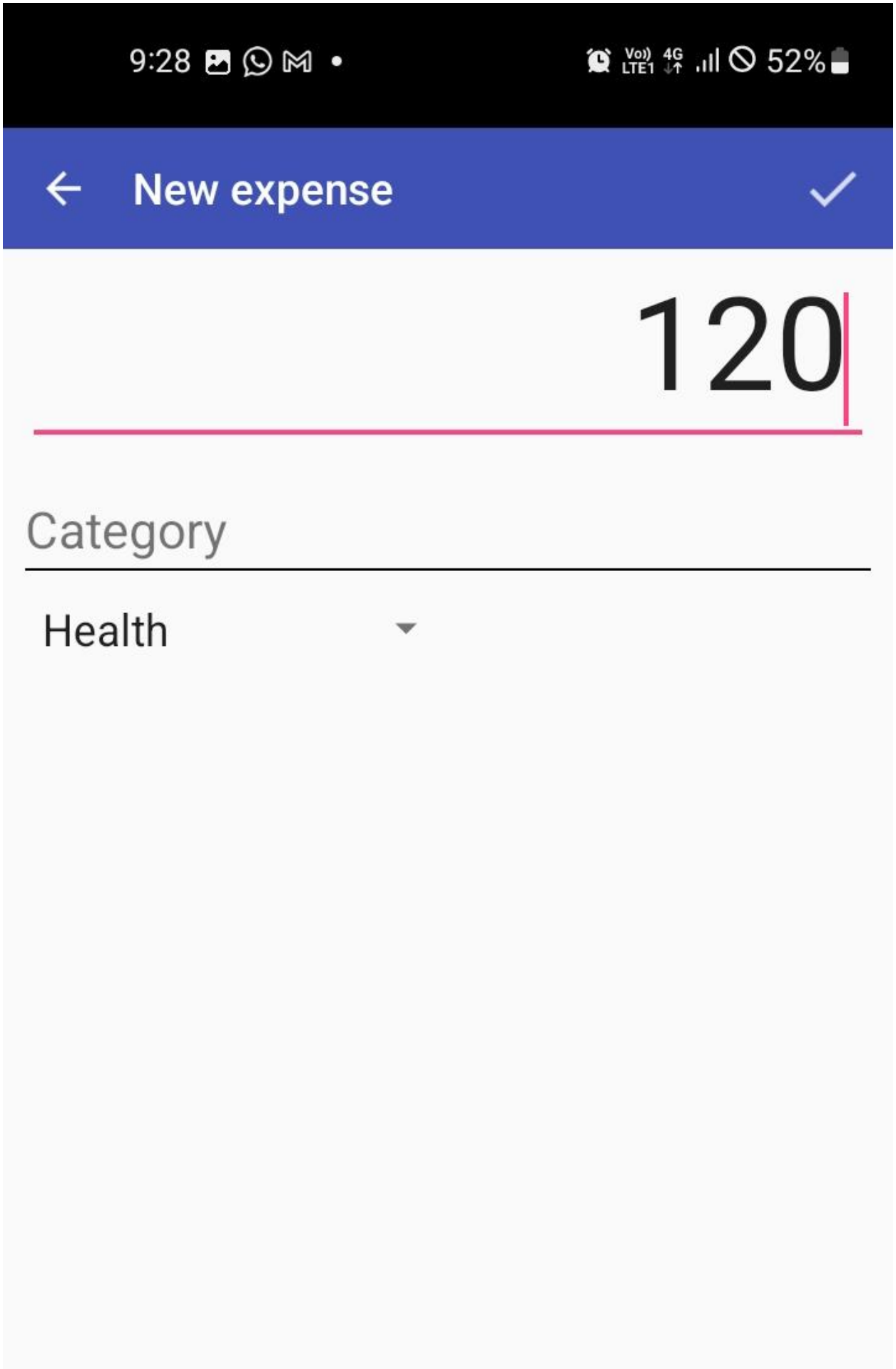
4,912.00 INR

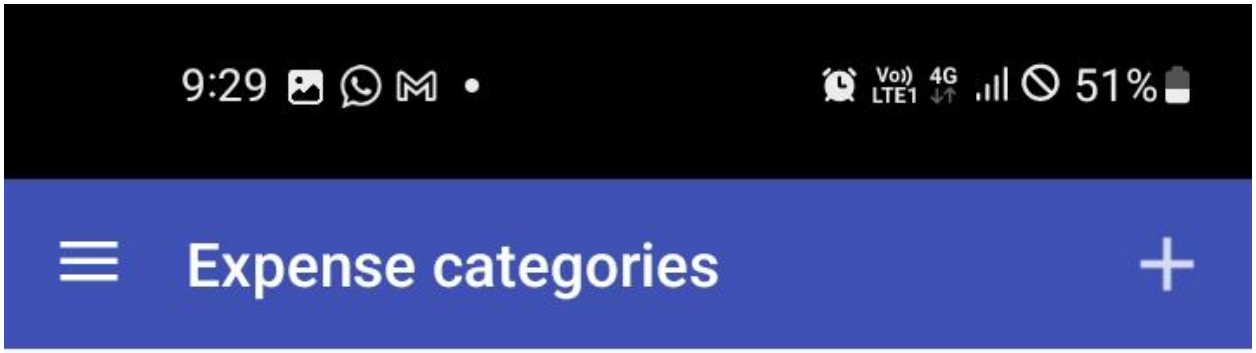
Today's expenses

123.00 INR
General

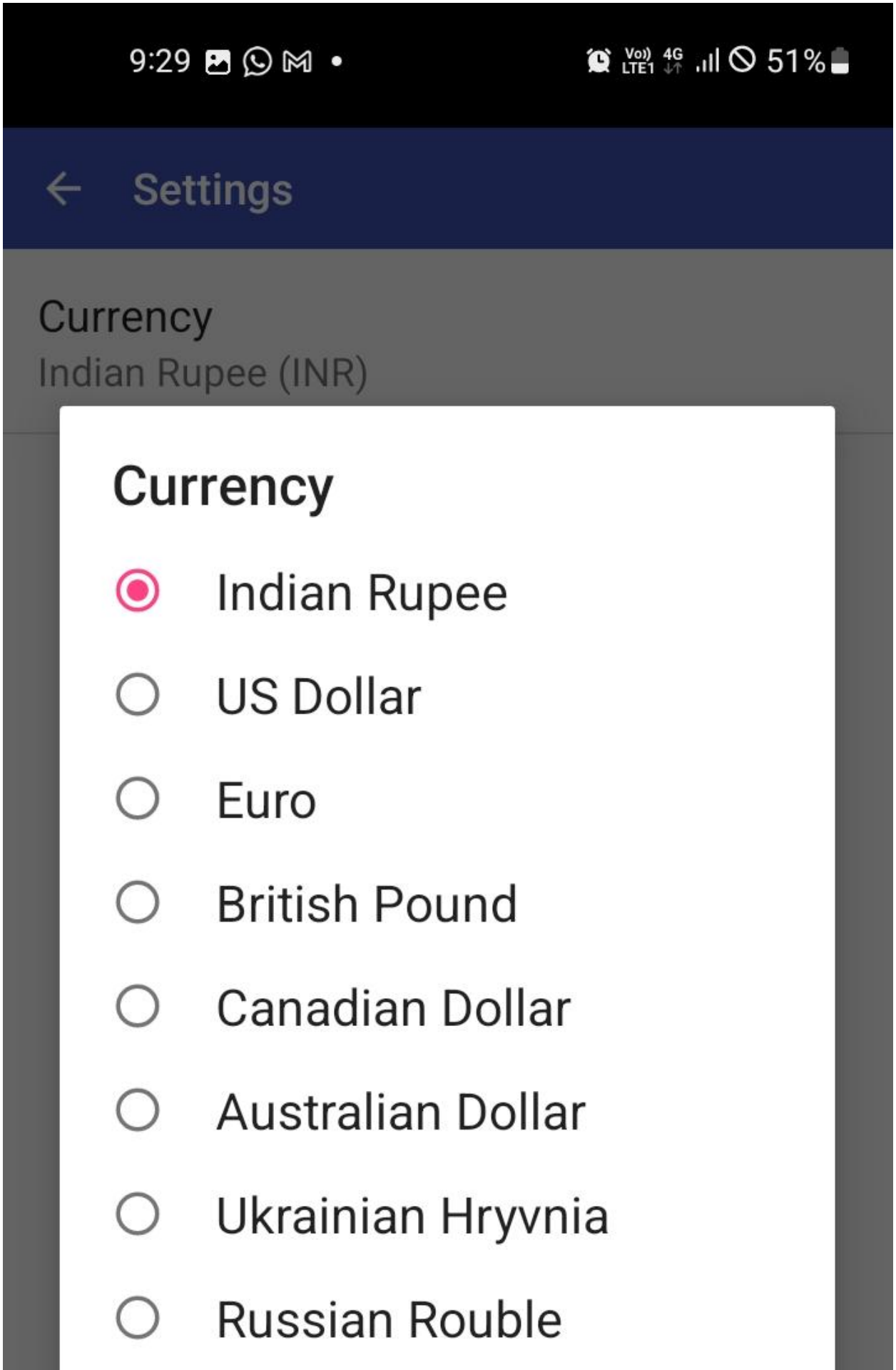
4,789.00 INR
Hygiene

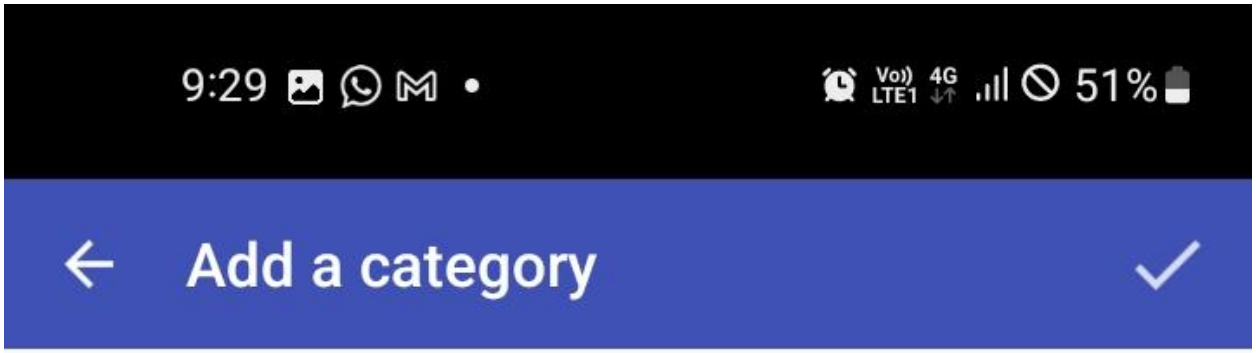

9:28
52%
New expense
0
Category
General

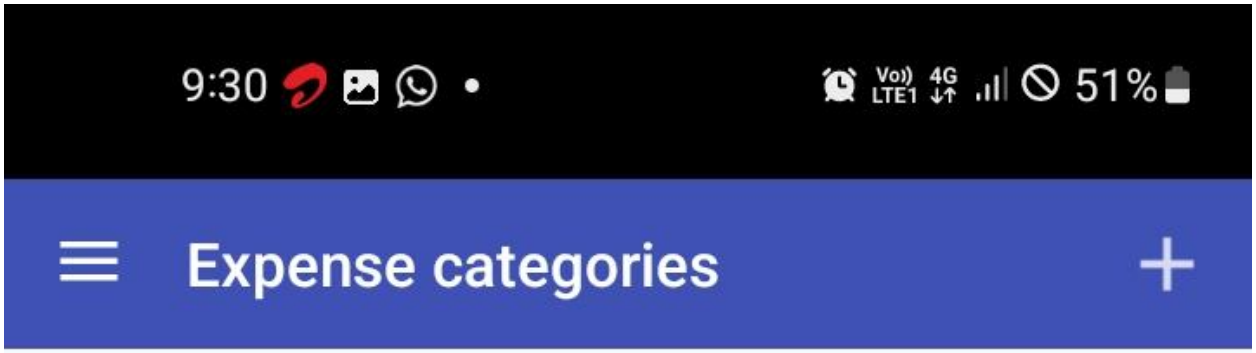

9:28
52%
New expense
120
Category
Health



- General
- Clothes
- Entertainment
- works
- Health
- Household
- Hygiene
- Pets
- Presents


9:29
51%
Settings
Currency
Indian Rupee (INR)
Currency
Indian Rupee
US Dollar
Euro
British Pound
Canadian Dollar
Australian Dollar
Ukrainian Hryvnia
Russian Rouble


9:29
51%
Add a category
Category name
sample category



- General
- Clothes
- Entertainment
- works
- Health
- Household
- Hygiene
- Pets
- Presents

Best Practices:

- The code written to develop the application is concise and easily understandable.
- Each intent layout is provided with its own java file.
- The attributes of the database are clearly defined.
- The colour template used is in such a way that it becomes easier to use the application.
- Proper naming convention is used to define variables and functions.

Learning Outcomes:

- We learnt how to use android studio to create applications.
- We learnt how to create a database to store and display the information accordingly.
- We learnt how to use different concepts involved in android development to build the application.
- We learnt how to handle different layouts to provide a seamless experience.
- We learnt how to design the different pages using the underlying xml file.