Code Document

for

Project: ManageMe (Group 2)

An All-in-One Self-Management System

Prepared by
Ankur Sharma (2015CS50278)
Lovish Madaan (2015CS50286)
Sudeep Agrawal (2015CS50295)
Siddharth Khera (2015MT60567)

Supervised by
Prof. S.C. Gupta

COL 740 - Software Engineering
Indian Institute of Technology, Delhi
Hauz Khas, New Delhi

Contents

Chapter 1

Java files

We haven't mentioned all the imports for these Java files for the sake of brevity of the document. Only important activities have been shown here. Unimportant files haven't been shown for maintaining conciseness. Similar modules for Expenses/Calories have been removed too.

1.1 ApplicationController

Listing 1.1: ApplicationController.java

```
1  public class ApplicationController extends Application {
2      @Override
3      protected void attachBaseContext(Context base) {
4          super.attachBaseContext(base);
5
6          // some of your own operations before content provider will launch
7          ExpenseDbHelper.DATABASE_NAME = "expense_tracer.db";
8          CalorieDbHelper.DATABASE_NAME = "calorie_tracer.db";
9      }
10 }
```

1.2 Activities

1.2.1 MainActivity.java

Listing 1.2: MainActivity.java

```
1
2  public class MainActivity extends Activity {
3      public static int STATE = 0;
4      public static String USERID;
5      public static String USERNAME;
6
7
8      @Override
9      protected void onCreate(Bundle savedInstanceState) {
10         super.onCreate(savedInstanceState);
11         setContentView(R.layout.activity_main);
12
13         Button expenseButton = (Button) findViewById(R.id.expense_button);
14         Button calorieButton = (Button) findViewById(R.id.calorie_button);
15         Button noteButton = (Button) findViewById(R.id.note_button);
16         Button logoutButton = (Button) findViewById(R.id.logout_button);
17         Button exportButton = (Button) findViewById(R.id.export_button);
18         TextView text = (TextView)findViewById(R.id.userNameId);
19
20         Intent intent = getIntent();
21         USERNAME = intent.getStringExtra("user-name");
22         USERID = intent.getStringExtra("user-id");
23         text.setText("Welcome " + USERNAME + "!");
24         if(USERID != null)
25             Log.d("USERID", USERID);
26         else
27             Log.d("USERID", "NULLLLLLLLL");
28         expenseButton.setOnClickListener(new View.OnClickListener()
29         {    public void onClick(View v)
30         {
31             Intent intent = new Intent(MainActivity.this, ExpenseActivity.class);
32             startActivity(intent);
33             finish();
34         }
35         });
36
37         calorieButton.setOnClickListener(new View.OnClickListener()
38         {    public void onClick(View v)
39         {
40             Intent intent = new Intent(MainActivity.this, CalorieActivity.class);
41             startActivity(intent);
42             finish();
43         }
44         });
45
46         logoutButton.setOnClickListener(new View.OnClickListener()
47         {    public void onClick(View v)
48         {
49             LoginActivity.resetSharedPreference();
50             Intent intent = new Intent(MainActivity.this, LoginActivity.class);
51             startActivity(intent);
52             finish();
53         }
54         });
```

```
55
56          noteButton.setOnClickListener(new View.OnClickListener()
57          {    public void onClick(View v)
58          {
59              Intent intent = new Intent(MainActivity.this, com.example.manageme.notes.MainActivity.class);
60              startActivity(intent);
61              finish();
62          }
63          });
64
65          exportButton.setOnClickListener(new View.OnClickListener()
66          {    public void onClick(View v)
67          {
68              isStoragePermissionGranted();
69              File direct = new File(Environment.getExternalStorageDirectory() + "/ManageMe");
70              if(!direct.exists())
71              {
72                  direct.mkdir();
73              }
74              exportExpenses();
75              exportCalories();
76              exportNotes();
77              Toast.makeText(getApplicationContext(), "Data downloaded!", Toast.LENGTH_SHORT).show();
78          }
79          });
80
81          MainActivity.STATE = 0;
82      }
83  }
```

1.2.2 ExportActivty.java

Listing 1.3: ExportActivity.java

```
    //exporting database
    private void exportCalories() {
        ArrayList<TableData> aList = getCalorieValues("calories", "calorieCategories", USERID);
        String fileName = "/ManageMe/Calories.csv";
        String filePath = Environment.getExternalStorageDirectory() + File.separator + fileName;
        File f = new File(filePath);
        CSVWriter writer;

        // File exist
        try {
            if(f.exists()&&!f.isDirectory())
            {
                FileWriter mFileWriter = new FileWriter(filePath, false);
                writer = new CSVWriter(mFileWriter, ',');
            }
            else
            {
                writer = new CSVWriter(new FileWriter(filePath), ',');
            }

            String heading[] = {"Calories (Cal)", "Date (mm/dd/YY)", "Category"};
            writer.writeNext(heading);
            for(TableData entry: aList) {
                String data[] = {entry.mColumn1, entry.mColumn2, entry.mColumn3};
                writer.writeNext(data);
            }

            writer.close();
        } catch (IOException e) {
            e.printStackTrace();
        }

    }

    public boolean isStoragePermissionGranted() {
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
            if (checkSelfPermission(android.Manifest.permission.WRITE_EXTERNAL_STORAGE)
                    == PackageManager.PERMISSION_GRANTED) {
                Log.v(TAG,"Permission is granted");
                return true;
            } else {

                Log.v(TAG,"Permission is revoked");
                ActivityCompat.requestPermissions(this, new
                        String[]{Manifest.permission.WRITE_EXTERNAL_STORAGE}, 1);
                return false;
            }
        }
        else { //permission is automatically granted on sdk<23 upon installation
            Log.v(TAG,"Permission is granted");
            return true;
        }
    }

    private void exportExpenses() {
        ArrayList<TableData> aList = getExpenseValues("expenses", "categories", USERID);
        String fileName = "/ManageMe/Expenses.csv";
```

```
        String filePath = Environment.getExternalStorageDirectory() + File.separator + fileName;
        File f = new File(filePath);
        CSVWriter writer;

        // File exist
        try {
            if(f.exists()&&!f.isDirectory())
            {
                FileWriter mFileWriter = new FileWriter(filePath, false);
                writer = new CSVWriter(mFileWriter, ',');
            }
            else
            {
                writer = new CSVWriter(new FileWriter(filePath), ',');
            }

            String heading[] = {"Amount (INR)", "Date (mm/dd/YY)", "Category"};
            writer.writeNext(heading);
            for(TableData entry: aList) {
                String data[] = {entry.mColumn1, entry.mColumn2, entry.mColumn3};
                writer.writeNext(data);
            }

            writer.close();
        } catch (IOException e) {
            e.printStackTrace();
        }

//       Toast.makeText(this, "Expenses Data Exported!", Toast.LENGTH_SHORT).show();
    }

    public ArrayList<TableData> getExpenseValues(String table, String catTable, String userid) {
        SQLiteDatabase db = new ExpenseDbHelper(this).getReadableDatabase();
        Cursor cursor = db.rawQuery("SELECT * FROM " + table + " WHERE user_id=" + userid, null);
        Cursor cursor2 = db.rawQuery("SELECT * FROM " + catTable, null);
        ArrayList<TableData> resultList = new ArrayList();
        ArrayList<String> categories = new ArrayList<String>();

        if(cursor2.moveToFirst()) {
            do {
                String columnName = cursor2.getString(cursor2.getColumnIndex("name"));
                categories.add(columnName);
            } while (cursor2.moveToNext());
        }

        if(cursor.moveToFirst()) {
            do {
                TableData row = new TableData();
                row.mColumn1 = cursor.getString(cursor.getColumnIndex("value"));
                row.mColumn2 = cursor.getString(cursor.getColumnIndex("date"));
                String catIndex = cursor.getString(cursor.getColumnIndex("category_id"));
                row.mColumn3 = categories.get(Integer.parseInt(catIndex) - 1);
                resultList.add(row);
            } while (cursor.moveToNext());
        }

        cursor.close();
        cursor2.close();
        db.close();
        return resultList;
    }
```

```
118
119     public ArrayList<TableData> getCalorieValues(String table, String catTable, String userid) {
120         SQLiteDatabase db = new CalorieDbHelper(this).getReadableDatabase();
121         Cursor cursor = db.rawQuery("SELECT * FROM " + table + " WHERE user_id=" + userid, null);
122         Cursor cursor2 = db.rawQuery("SELECT * FROM " + catTable, null);
123         ArrayList<TableData> resultList = new ArrayList();
124         ArrayList<String> categories = new ArrayList<String>();
125         ArrayList<String> values = new ArrayList<String>();
126
127         if(cursor2.moveToFirst()) {
128             do {
129                 String columnName = cursor2.getString(cursor2.getColumnIndex("name"));
130                 String calorie = cursor2.getString(cursor2.getColumnIndex("calories"));
131                 categories.add(columnName);
132                 values.add(calorie);
133             } while (cursor2.moveToNext());
134         }
135
136         if(cursor.moveToFirst()) {
137             do {
138                 TableData row = new TableData();
139                 row.mColumn2 = cursor.getString(cursor.getColumnIndex("date"));
140                 String catIndex = cursor.getString(cursor.getColumnIndex("category_id"));
141                 row.mColumn3 = categories.get(Integer.parseInt(catIndex) - 1);
142                 row.mColumn1 = values.get(Integer.parseInt(catIndex) - 1);
143                 resultList.add(row);
144             } while (cursor.moveToNext());
145         }
146
147         cursor.close();
148         cursor2.close();
149         db.close();
150         return resultList;
151     }
152
153     public String readFile(String fileName) throws IOException {
154         BufferedReader reader = new BufferedReader(new FileReader(fileName));
155         StringBuilder stringBuilder = new StringBuilder();
156         String line = null;
157         String ls = System.getProperty("line.separator");
158         while ((line = reader.readLine()) != null) {
159             stringBuilder.append(line);
160             stringBuilder.append(ls);
161         }
162         // delete the last new line separator
163         stringBuilder.deleteCharAt(stringBuilder.length() - 1);
164         reader.close();
165
166         String content = stringBuilder.toString();
167         return content;
168     }
169
170     public void exportNotes() {
171         String currentDBPath = "/data/data/com.example.manageme/files/notes_" + USERID + ".json";
172         String fileName = "/ManageMe/Notes.csv";
173         String filePath = Environment.getExternalStorageDirectory() + File.separator + fileName;
174         try {
175             String content = readFile(currentDBPath);
176
177             JFlat flatMe = new JFlat(content);
178
```

```
            //get the 2D representation of JSON document
            flatMe.json2Sheet().headerSeparator("_").getJsonAsSheet();

            //write the 2D representation in csv format
            flatMe.write2csv(filePath);

        } catch (IOException e) {
            e.printStackTrace();
        } catch (Exception e) {
            e.printStackTrace();
        }
        return;
    }
```

1.2.3 BaseFragmentActivity.java

Listing 1.4: BaseFragmentActivity.java

```
1
2  public abstract class BaseFragmentActivity extends AppCompatActivity {
3      protected Toolbar mToolbar;
4
5      @LayoutRes
6      protected int getLayoutResId() {
7          return R.layout.activity_base;
8      }
9
10     @Override
11     protected void onCreate(@Nullable Bundle savedInstanceState) {
12         super.onCreate(savedInstanceState);
13         setContentView(getLayoutResId());
14
15         // Set a Toolbar to replace the ActionBar
16         mToolbar = (Toolbar) findViewById(R.id.toolbar);
17         setSupportActionBar(mToolbar);
18     }
19
20     protected void insertFragment(Fragment fragment) {
21         // Insert the fragment by replacing any existing fragment
22         FragmentManager fragmentManager = getSupportFragmentManager();
23         fragmentManager.beginTransaction()
24                 .replace(R.id.content_frame, fragment)
25                 .commit();
26     }
27 }
```

1.2.4 SettingsActivity.java

Listing 1.5: SettingsActivity.java

```
1
2  public class SettingsActivity extends BaseFragmentActivity {
3
4      @Override
5      protected void onCreate(@Nullable Bundle savedInstanceState) {
6          super.onCreate(savedInstanceState);
7
8          insertFragment(new SettingsFragment());
9
10         setTitle(R.string.nav_settings);
11
12         setupActionBar();
13     }
14
15     private void setupActionBar() {
16         ActionBar actionBar = getSupportActionBar();
17         if (actionBar != null) {
18             // Show the Up button in the action bar (toolbar).
19             actionBar.setDisplayHomeAsUpEnabled(true);
20         }
21     }
22 }
```

1.2.5 ExpenseActivity.java

Listing 1.6: ExpenseActivity.java

```
1
2  public class ExpenseActivity extends BaseFragmentActivity {
3      private DrawerLayout mDrawerLayout;
4      private NavigationView mNavDrawer;
5      private ActionBarDrawerToggle mDrawerToggle;
6
7      @Override
8      @LayoutRes
9      protected int getLayoutResId() {
10         return R.layout.expense_main;
11     }
12
13     @Override
14     protected void onCreate(Bundle savedInstanceState) {
15         super.onCreate(savedInstanceState);
16
17         mDrawerLayout = (DrawerLayout) findViewById(R.id.drawer_layout);
18         mNavDrawer = (NavigationView) findViewById(R.id.nav_drawer);
19         mDrawerToggle = setupDrawerToggle();
20
21         Intent intent = getIntent();
22
23         // Tie DrawerLayout events to the ActionBarToggle
24         mDrawerLayout.addDrawerListener(mDrawerToggle);
25
26         // Setup drawer view
27         setupDrawerContent(mNavDrawer);
28
29         // Select TodayFragment on app start by default
30         loadTodayFragment();
31
32         // Setup the main home button
33         Button homeButton = (Button) findViewById(R.id.home_button);
34         homeButton.setOnClickListener(new View.OnClickListener()
35         {    public void onClick(View v)
36         {
37             Intent intent = new Intent(ExpenseActivity.this, MainActivity.class);
38             intent.putExtra("user-name", MainActivity.USERNAME);
39             intent.putExtra("user-id", MainActivity.USERID);
40             startActivity(intent);
41             finish();
42         }
43         });
44
45         MainActivity.STATE = 1;
46     }
47
48
49     @Override
50     protected void onPause() {
51         super.onPause();
52         closeNavigationDrawer();
53     }
54
55     @Override
56     public boolean onOptionsItemSelected(MenuItem item) {
57         // 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(ExpenseActivity.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));
    }
}
```

1.2.6 ExpenseEditActivity.java

Listing 1.7: ExpenseEditActivity.java

```
public class ExpenseEditActivity 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 ExpenseEditFragment());
        setupActionBar();
    }

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

1.3 Adapters

1.3.1 SimpleAdapter.java

Listing 1.8: SimpleAdapter.java

```
public class SimpleExpenseAdapter extends CursorAdapter {
    private String mCurrency;

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

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

    // The newView method is used to inflate a new view and return it
    @Override
    public View newView(Context context, Cursor cursor, ViewGroup parent) {
        return LayoutInflater.from(context).inflate(R.layout.expense_list_item, parent, false);
    }

    // The bindView method is used to bind all data to a given view
    @Override
    public void bindView(View view, Context context, Cursor cursor) {
        // Find fields to populate in inflated template
        TextView tvExpenseCurrency = null;
        TextView tvExpenseValue = (TextView) view.findViewById(R.id.expense_value_text_view);
        if(MainActivity.STATE == 1)
            tvExpenseCurrency = (TextView) view.findViewById(R.id.expense_currency_text_view);
        TextView tvExpenseCatName = (TextView) view.findViewById(R.id.expense_category_name_text_view);

        // Extract values from cursor
        float expValue = 0.0f;
        String categoryName = "";
        if(MainActivity.STATE == 1) {
//          Toast.makeText(context, "1111111",
//                  Toast.LENGTH_SHORT).show();
            expValue = cursor.getFloat(cursor.getColumnIndexOrThrow(ExpensesContract.Expenses.VALUE));
            categoryName =
                cursor.getString(cursor.getColumnIndexOrThrow(ExpensesContract.Categories.NAME));
        } else if(MainActivity.STATE == 2){
//          Toast.makeText(context, "222222",
//                  Toast.LENGTH_SHORT).show();
            expValue =
                cursor.getFloat(cursor.getColumnIndexOrThrow(CaloriesContract.CalorieCategories.VALUE));
            categoryName =
                cursor.getString(cursor.getColumnIndexOrThrow(CaloriesContract.CalorieCategories.NAME));
        }

        // Populate views with extracted values
        if(MainActivity.STATE == 1)
            tvExpenseValue.setText(Utils.formatToCurrency(expValue));
        else
            tvExpenseValue.setText(Integer.toString((int)expValue));

        tvExpenseCatName.setText(categoryName);
        if(MainActivity.STATE == 1)
```

```
52              tvExpenseCurrency.setText(mCurrency);
53      }
54  }
```

1.3.2 SectionAdapter.java

Listing 1.9: SectionAdapter.java

```
1
2  public class SectionExpenseAdapter extends SectionCursorAdapter {
3      private String mCurrency;
4
5      public SectionExpenseAdapter(Context context) {
6          super(context, null, 0);
7      }
8
9      public void setCurrency(String currency) {
10         mCurrency = currency;
11         notifyDataSetChanged();
12     }
13
14     @Override
15     protected Object getSectionFromCursor(Cursor cursor) {
16         String dateStr = "";
17         if(MainActivity.STATE == 1) {
18             dateStr = cursor.getString(cursor.getColumnIndexOrThrow(ExpensesContract.Expenses.DATE));
19         } else if(MainActivity.STATE == 2) {
20             dateStr = cursor.getString(cursor.getColumnIndexOrThrow(CaloriesContract.Calories.DATE));
21         }
22         return Utils.getSystemFormatDateString(mContext, dateStr);
23     }
24
25     @Override
26     protected View newSectionView(Context context, Object item, ViewGroup parent) {
27         return getLayoutInflater().inflate(R.layout.expense_report_section_header, parent, false);
28     }
29
30     @Override
31     protected void bindSectionView(View convertView, Context context, int position, Object item) {
32         ((TextView) convertView).setText((String) item);
33     }
34
35     @Override
36     protected View newItemView(Context context, Cursor cursor, ViewGroup parent) {
37         return getLayoutInflater().inflate(R.layout.expense_list_item, parent, false);
38     }
39
40     @Override
41     protected void bindItemView(View convertView, Context context, Cursor cursor) {
42         // Find fields to populate in inflated template
43         TextView tvExpenseValue = (TextView) convertView.findViewById(R.id.expense_value_text_view);
44         TextView tvExpenseCurrency = (TextView) convertView.findViewById(R.id.expense_currency_text_view);
45         TextView tvExpenseCatName = (TextView)
               convertView.findViewById(R.id.expense_category_name_text_view);
46
47         // Extract values from cursor
48         float expValue = 0.0f;
49         String categoryName = "";
50         if(MainActivity.STATE == 1) {
51             expValue = cursor.getFloat(cursor.getColumnIndexOrThrow(ExpensesContract.Expenses.VALUE));
52             categoryName =
                   cursor.getString(cursor.getColumnIndexOrThrow(ExpensesContract.Categories.NAME));
53         } else if(MainActivity.STATE == 2){
54             expValue =
                   cursor.getFloat(cursor.getColumnIndexOrThrow(CaloriesContract.CalorieCategories.VALUE));
```

```
            categoryName =
                cursor.getString(cursor.getColumnIndexOrThrow(CaloriesContract.CalorieCategories.NAME));
        }

        if(MainActivity.STATE == 1)
            tvExpenseValue.setText(Utils.formatToCurrency(expValue));
        else
            tvExpenseValue.setText(Integer.toString((int)expValue));
        // Populate views with extracted values
        tvExpenseCatName.setText(categoryName);
        if(MainActivity.STATE == 1)
            tvExpenseCurrency.setText(mCurrency);
    }
}
```

1.4 Contract

1.4.1 CalorieContract.java

Listing 1.10: CalorieContract.java

```
1  public final class CaloriesContract {
2      /**
3       * The authority for the calories provider
4       */
5      public static final String AUTHORITY = "com.example.manageme.provider2";
6      /**
7       * The content:// style URI for calories provider
8       */
9      public static final Uri AUTHORITY_URI = Uri.parse("content://" + AUTHORITY);
10
11     /**
12      * The contract class cannot be instantiated
13      */
14     private CaloriesContract(){}
15
16     public static class CalorieCategories implements BaseColumns, CalorieCategoriesColumns {
17         /**
18          * This utility class cannot be instantiated
19          */
20         private CalorieCategories() {}
21
22         /**
23          * The content:// style URI for this table
24          */
25         public static final Uri CONTENT_URI = Uri.withAppendedPath(AUTHORITY_URI, "calorieCategories");
26
27         /**
28          * The MIME type of {@link #CONTENT_URI} providing a directory of categories.
29          */
30         public static final String CONTENT_TYPE =
31                 "vnd.android.cursor.dir/vnd.ematiyuk.ManageMe.provider2.calorie_category";
32
33         /**
34          * The MIME type of a {@link #CONTENT_URI} sub-directory of a single category.
35          */
36         public static final String CONTENT_ITEM_TYPE =
37                 "vnd.android.cursor.item/vnd.ematiyuk.ManageMe.provider2.calorie_category";
38
39         /**
40          * Sort by ascending order of _id column (the order as items were added).
41          */
42         public static final String DEFAULT_SORT_ORDER = _ID + " ASC";
43     }
44
45
46     public static class Calories implements BaseColumns, CaloriesColumns {
47         private Calories() {}
48
49         public static final Uri CONTENT_URI = Uri.withAppendedPath(AUTHORITY_URI, "calories");
50
51         /**
52          * The MIME type of {@link #CONTENT_URI} providing a directory of calories.
53          */
54         public static final String CONTENT_TYPE =
```

```
            "vnd.android.cursor.dir/vnd.ematiyuk.ManageMe.provider2.calories";

    /**
     * The MIME type of a {@link #CONTENT_URI} sub-directory of a single calorie.
     */
    public static final String CONTENT_ITEM_TYPE =
            "vnd.android.cursor.item/vnd.ematiyuk.ManageMe.provider2.calories";

    /**
     * Sort by descending order of date (the most recent items are at the end).
     */
    public static final String DEFAULT_SORT_ORDER = DATE + " ASC";

    /**
     * calorie sum value column name to return for joined tables
     */
    public static final String VALUES_SUM = "values_sum";
}

public static class CaloriesWithCategories implements BaseColumns {
    /**
     * This utility class cannot be instantiated.
     */
    private CaloriesWithCategories() {}

    /**
     * The content:// style URI for this table.
     */
    public static final Uri CONTENT_URI = Uri.withAppendedPath(AUTHORITY_URI,
        "caloriesWithCategories");

    /**
     * The MIME type of {@link #CONTENT_URI} providing a directory of calories with categories.
     */
    public static final String CONTENT_TYPE =
            "vnd.android.cursor.dir/vnd.ematiyuk.ManageMe.provider2.calories_with_category";

    /**
     * The content:// style URI for this joined table to filter items by a specific date.
     */
    public static final Uri DATE_CONTENT_URI = Uri.withAppendedPath(CONTENT_URI, "date");

    /**
     * The content:// style URI for this joined table to filter items by a specific date range.
     */
    public static final Uri DATE_RANGE_CONTENT_URI = Uri.withAppendedPath(CONTENT_URI, "dateRange");

    /**
     * The content:// style URI for getting sum of calorie values
     * for this joined table by "date" filter.
     */
    public static final Uri SUM_DATE_CONTENT_URI = Uri.withAppendedPath(DATE_CONTENT_URI, "sum");

    /*
     * The content:// style URI for getting sum of calorie values
     * for this joined table by "date range" filter.
     */
    public static final Uri SUM_DATE_RANGE_CONTENT_URI =
            Uri.withAppendedPath(DATE_RANGE_CONTENT_URI, "sum");
}
```

```
115     protected interface CalorieCategoriesColumns {
116         String NAME = "name";
117         String VALUE = "calories";
118     }
119 
120     protected interface CaloriesColumns {
121         String DATE = "date";
122         String CALORIE_CATEGORY_ID = "category_id";
123         String USER_ID ="user_id";
124     }
125 }
```

1.5 Fragments

1.5.1 TodayCalFragment.java

Listing 1.11: TodayCalFragment.java

```
public class TodayCalFragment extends Fragment implements LoaderManager.LoaderCallbacks<Cursor> {
    private static final int SUM_LOADER_ID = 0;
    private static final int LIST_LOADER_ID = 1;

    private ListView mExpensesView;
    private View mProgressBar;
    private SimpleExpenseAdapter mAdapter;
    private TextView mTotalExpSumTextView;
    private TextView mTotalExpCurrencyTextView;

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

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

        mExpensesView = (ListView) rootView.findViewById(R.id.calories_list_view);
        mProgressBar = rootView.findViewById(R.id.expenses_progress_bar);
        mTotalExpSumTextView = (TextView) rootView.findViewById(R.id.total_calorie_consumed_sum_text_view);
        mTotalExpCurrencyTextView = (TextView)
             rootView.findViewById(R.id.total_calorie_consumed_text_view);

        mExpensesView.setEmptyView(rootView.findViewById(R.id.calories_empty_list_view));
        mExpensesView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
                prepareExpenseToEdit(id);
            }
        });

        rootView.findViewById(R.id.add_calorie_button_if_empty_list).setOnClickListener(new
             View.OnClickListener() {
            @Override
            public void onClick(View view) {
                prepareExpenseToCreate();
            }
        });
        mTotalExpSumTextView.setText(Integer.toString(0));

        registerForContextMenu(mExpensesView);

        return rootView;
    }

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

```

```
        // Set default values for preferences (settings) on startup
        PreferenceManager.setDefaultValues(getActivity(), R.xml.cal_preferences, false);

        mAdapter = new SimpleExpenseAdapter(getActivity());
        mExpensesView.setAdapter(mAdapter);

        // Initialize the CursorLoaders
        getLoaderManager().initLoader(SUM_LOADER_ID, null, this);
        getLoaderManager().initLoader(LIST_LOADER_ID, null, this);

    }

    @Override
    public void onResume() {
        super.onResume();
        reloadExpenseData();
        reloadSharedPreferences();
    }

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

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case R.id.new_expense_menu_item:
                prepareExpenseToCreate();
                return true;
            default:
                return super.onOptionsItemSelected(item);
        }
    }

    @Override
    public void onCreateContextMenu(ContextMenu menu, View v, ContextMenu.ContextMenuInfo menuInfo) {
        super.onCreateContextMenu(menu, v, menuInfo);
        getActivity().getMenuInflater().inflate(R.menu.expense_list_item_context, menu);

    }

    @Override
    public boolean onContextItemSelected(MenuItem item) {
        AdapterView.AdapterContextMenuInfo info = (AdapterView.AdapterContextMenuInfo) item.getMenuInfo();
        switch (item.getItemId()) {
            case R.id.delete_expense_menu_item:
                deleteExpense(info.id);
                return true;
            default:
                return super.onContextItemSelected(item);
        }
    }

    @Override
    public Loader<Cursor> onCreateLoader(int id, Bundle args) {
        Uri uri = null;
        switch (id) {
            case SUM_LOADER_ID:
                uri = CaloriesWithCategories.SUM_DATE_CONTENT_URI;
```

```
                break;
            case LIST_LOADER_ID:
                uri = CaloriesWithCategories.DATE_CONTENT_URI;
                break;
        }

        // Retrieve today's date string
        String today = Utils.getDateString(new Date());
        String[] selectionArgs = {MainActivity.USERID, today };

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

    @Override
    public void onLoadFinished(Loader<Cursor> loader, Cursor data) {
        switch (loader.getId()){
            case SUM_LOADER_ID:
                int valueSumIndex = data.getColumnIndex(Calories.VALUES_SUM);
                data.moveToFirst();
                float valueSum = data.getFloat(valueSumIndex);
                mTotalExpSumTextView.setText(Integer.toString((int)valueSum));
                break;

            case LIST_LOADER_ID:
                // Hide the progress bar
                mProgressBar.setVisibility(View.GONE);

                mAdapter.swapCursor(data);
                break;
        }

    }

    @Override
    public void onLoaderReset(Loader<Cursor> loader) {
        switch (loader.getId()) {
            case SUM_LOADER_ID:
                mTotalExpSumTextView.setText(Integer.toString(0));
                break;
            case LIST_LOADER_ID:
                mAdapter.swapCursor(null);
                break;
        }
    }


    private void reloadSharedPreferences() {
        SharedPreferences sharedPref = PreferenceManager.getDefaultSharedPreferences(getActivity());
        String prefCurrency = sharedPref.getString(SettingsCalFragment.KEY_PREF_CALORIE, "");

        mTotalExpCurrencyTextView.setText(prefCurrency);
        mAdapter.setCurrency(prefCurrency);
    }
```

```
    private void reloadExpenseData() {
        // Show the progress bar
        mProgressBar.setVisibility(View.VISIBLE);
        // Reload data by restarting the cursor loaders
        getLoaderManager().restartLoader(LIST_LOADER_ID, null, this);
        getLoaderManager().restartLoader(SUM_LOADER_ID, null, this);
    }

    private int deleteSingleExpense(long expenseId) {
        Uri uri = ContentUris.withAppendedId(Calories.CONTENT_URI, expenseId);

        // Defines a variable to contain the number of rows deleted
        int rowsDeleted;

        // Deletes the expense that matches the selection criteria
        rowsDeleted = getActivity().getContentResolver().delete(
                uri,       // the URI of the row to delete
                null,      // where clause
                null       // where args
        );

        showStatusMessage(getResources().getString(R.string.calories_deleted));
        reloadExpenseData();

        return rowsDeleted;
    }

    private void deleteExpense(final long expenseId) {
        new AlertDialog.Builder(getActivity())
                .setTitle(R.string.delete_expense)
                .setMessage(R.string.delete_exp_dialog_msg)
                .setNeutralButton(android.R.string.cancel, null)
                .setPositiveButton(R.string.delete_string, new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialogInterface, int i) {
                        deleteSingleExpense(expenseId);
                    }
                })
                .show();
    }

    private void showStatusMessage(CharSequence text) {
        Toast.makeText(getActivity(), text, Toast.LENGTH_SHORT).show();
    }

    private void prepareExpenseToCreate() {
        startActivity(new Intent(getActivity(), CalorieEditActivity.class));
    }

    private void prepareExpenseToEdit(long id) {
        Intent intent = new Intent(getActivity(), CalorieEditActivity.class);
        intent.putExtra(CalorieEditFragment.EXTRA_EDIT_EXPENSE, id);
        startActivity(intent);
    }

}
```

1.6 Login

1.6.1 LoginActivity.java

Listing 1.12: LoginActivity.java

```
public class LoginActivity extends AppCompatActivity {

    //Declaration EditTexts
    EditText editTextEmail;
    EditText editTextPassword;
    static SharedPreferences sp;

    //Declaration TextInputLayout
    TextInputLayout textInputLayoutEmail;
    TextInputLayout textInputLayoutPassword;

    //Declaration Button
    Button buttonLogin;

    //Declaration SqliteHelper
    SqliteHelper sqliteHelper;



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

        sp = getSharedPreferences("login",MODE_PRIVATE);

        if(sp.getBoolean("logged",false)){
            goToMainActivity(sp.getString("username", ""),
                    sp.getString("userid", ""));
        }

        sqliteHelper = new SqliteHelper(this);

        initCreateAccountTextView();
        initViews();

        //set click event of login button
        buttonLogin.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {

                //Check user input is correct or not
                if (validate()) {

                    //Get values from EditText fields
                    String Email = editTextEmail.getText().toString();
                    String Password = editTextPassword.getText().toString();

                    //Authenticate user
                    User currentUser = sqliteHelper.Authenticate(new User(null, null, null, Email,
                        Password));

                    //Check Authentication is successful or not
                    if (currentUser != null) {
```

```
                    Snackbar.make(buttonLogin, "Successfully Logged in!", Snackbar.LENGTH_LONG).show();
                    goToMainActivity(currentUser.name, currentUser.id);
                    sp.edit().putBoolean("logged",true).apply();
                    sp.edit().putString("username",currentUser.name).apply();
                    sp.edit().putString("userid",currentUser.id).apply();


                    //User Logged in Successfully Launch You home screen activity
//                    Intent intent=new Intent(LoginActivity.this, MainActivity.class);
//                    intent.putExtra("user-name", currentUser.name);
//                    intent.putExtra("user-id", currentUser.id);
//                    startActivity(intent);
//                    finish();
                } else {

                    //User Logged in Failed
                    Snackbar.make(buttonLogin, "Failed to log in , please try again",
                        Snackbar.LENGTH_LONG).show();

                }
            }
        }
    });


}


private void goToMainActivity(String name, String id) {
    Intent intent = new Intent(LoginActivity.this, MainActivity.class);
    intent.putExtra("user-name", name);
    intent.putExtra("user-id", id);
    startActivity(intent);
    finish();
}

//this method used to set Create account TextView text and click event( maltipal colors
// for TextView yet not supported in Xml so i have done it programmatically)
private void initCreateAccountTextView() {
    TextView textViewCreateAccount = (TextView) findViewById(R.id.textViewCreateAccount);
    textViewCreateAccount.setText(fromHtml("<font color='#000000'>No account yet? </font><font
        color='#3F51B5'>Create one</font>"));
    textViewCreateAccount.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            Intent intent = new Intent(LoginActivity.this, RegisterActivity.class);
            startActivity(intent);
        }
    });
}

//this method is used to connect XML views to its Objects
private void initViews() {
    editTextEmail = (EditText) findViewById(R.id.editTextEmail);
    editTextPassword = (EditText) findViewById(R.id.editTextPassword);
    textInputLayoutEmail = (TextInputLayout) findViewById(R.id.textInputLayoutEmail);
    textInputLayoutPassword = (TextInputLayout) findViewById(R.id.textInputLayoutPassword);
    buttonLogin = (Button) findViewById(R.id.buttonLogin);

}

```

```
    //This method is for handling fromHtml method deprecation
    @SuppressWarnings("deprecation")
    public static Spanned fromHtml(String html) {
        Spanned result;
        if (android.os.Build.VERSION.SDK_INT >= 24) {
            result = Html.fromHtml(html);
        } else {
            result = Html.fromHtml(html);
        }
        return result;
    }

    public static void resetSharedPreference()
    {
        sp.edit().putBoolean("logged",false).apply();
    }

    //This method is used to validate input given by user
    public boolean validate() {
        boolean valid = false;

        //Get values from EditText fields
        String Email = editTextEmail.getText().toString();
        String Password = editTextPassword.getText().toString();

        //Handling validation for Email field
        if (!android.util.Patterns.EMAIL_ADDRESS.matcher(Email).matches()) {
            valid = false;
            textInputLayoutEmail.setError("Please enter valid email!");
        } else {
            valid = true;
            textInputLayoutEmail.setError(null);
        }

        //Handling validation for Password field
        if (Password.isEmpty()) {
            valid = false;
            textInputLayoutPassword.setError("Please enter valid password!");
        } else {
            if (Password.length() > 5) {
                valid = true;
                textInputLayoutPassword.setError(null);
            } else {
                valid = false;
                textInputLayoutPassword.setError("Password is too short!");
            }
        }

        return valid;
    }


}
```

1.6.2 RegisterActivity

Listing 1.13: RegisterActivity

```
public class RegisterActivity extends AppCompatActivity {

    //Declaration EditTexts
    EditText editTextName;
    EditText editTextUserName;
    EditText editTextEmail;
    EditText editTextPassword;

    //Declaration TextInputLayout
    TextInputLayout textInputLayoutUserName;
    TextInputLayout textInputLayoutEmail;
    TextInputLayout textInputLayoutPassword;

    //Declaration Button
    Button buttonRegister;

    //Declaration SqliteHelper
    SqliteHelper sqliteHelper;

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_register);
        sqliteHelper = new SqliteHelper(this);
        initTextViewLogin();
        initViews();
        buttonRegister.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                if (validate()) {
                    String name = editTextName.getText().toString();
                    String UserName = editTextUserName.getText().toString();
                    String Email = editTextEmail.getText().toString();
                    String Password = editTextPassword.getText().toString();

                    //Check in the database is there any user associated with this email
                    if (!sqliteHelper.isEmailExists(Email)) {
                        if(!sqliteHelper.isUsernameExists(UserName)) {
                            //Email does not exist now add new user to database
                            sqliteHelper.addUser(new User(null, name, UserName, Email, Password));
                            Snackbar.make(buttonRegister, "User created successfully! Please Login ",
                                    Snackbar.LENGTH_LONG).show();
                            new Handler().postDelayed(new Runnable() {
                                @Override
                                public void run() {
                                    finish();
                                }
                            }, Snackbar.LENGTH_LONG);
                            Toast.makeText(RegisterActivity.this, "User successfully registered!",
                                    Toast.LENGTH_SHORT).show();
                        } else {
                            //Username exists with username input provided so show error user already exist
                            Snackbar.make(buttonRegister, "User already exists with same username ",
                                    Snackbar.LENGTH_LONG).show();
                        }
                    } else {
                        //Email exists with email input provided so show error user already exist
```

```
                    Snackbar.make(buttonRegister, "User already exists with same email ",
                        Snackbar.LENGTH_LONG).show();
                }


            }
        }
    });
}

//this method used to set Login TextView click event
private void initTextViewLogin() {
    TextView textViewLogin = (TextView) findViewById(R.id.textViewLogin);
    textViewLogin.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            finish();
        }
    });
}

//this method is used to connect XML views to its Objects
private void initViews() {
    editTextEmail = (EditText) findViewById(R.id.editTextEmail);
    editTextPassword = (EditText) findViewById(R.id.editTextPassword);
    editTextUserName = (EditText) findViewById(R.id.editTextUserName);
    editTextName = (EditText) findViewById(R.id.editTextName);
    textInputLayoutEmail = (TextInputLayout) findViewById(R.id.textInputLayoutEmail);
    textInputLayoutPassword = (TextInputLayout) findViewById(R.id.textInputLayoutPassword);
    textInputLayoutUserName = (TextInputLayout) findViewById(R.id.textInputLayoutUserName);
    buttonRegister = (Button) findViewById(R.id.buttonRegister);

}

//This method is used to validate input given by user
public boolean validate() {
    boolean valid = false;

    //Get values from EditText fields
    String UserName = editTextUserName.getText().toString();
    String Email = editTextEmail.getText().toString();
    String Password = editTextPassword.getText().toString();

    //Handling validation for UserName field
    if (UserName.isEmpty()) {
        valid = false;
        textInputLayoutUserName.setError("Please enter valid username!");
    } else {
        if (UserName.length() > 5) {
            valid = true;
            textInputLayoutUserName.setError(null);
        } else {
            valid = false;
            textInputLayoutUserName.setError("Username is to short!");
        }
    }

    //Handling validation for Email field
    if (!android.util.Patterns.EMAIL_ADDRESS.matcher(Email).matches()) {
        valid = false;
        textInputLayoutEmail.setError("Please enter valid email!");
```

```
        } else {
            valid = true;
            textInputLayoutEmail.setError(null);
        }

        //Handling validation for Password field
        if (Password.isEmpty()) {
            valid = false;
            textInputLayoutPassword.setError("Please enter valid password!");
        } else {
            if (Password.length() > 5) {
                valid = true;
                textInputLayoutPassword.setError(null);
            } else {
                valid = false;
                textInputLayoutPassword.setError("Password is too short!");
            }
        }


        return valid;
    }
}
```

1.6.3 User.java

Listing 1.14: User.java

```
public class User {
    public String id;
    public String userName;
    public String name;
    public String email;
    public String password;

    public User(String id, String name, String userName, String email, String password) {
        this.id = id;
        this.name = name;
        this.userName = userName;
        this.email = email;
        this.password = password;
    }

}
```

1.7 Notes

1.7.1 MainActivity.java

Listing 1.15: MainActivity.java

```
@TargetApi(Build.VERSION_CODES.HONEYCOMB)
public class MainActivity extends AppCompatActivity implements AdapterView.OnItemClickListener,
        Toolbar.OnMenuItemClickListener, AbsListView.MultiChoiceModeListener,
        SearchView.OnQueryTextListener {

    private static File localPath, backupPath;

    // Layout components
    private static ListView listView;
    private ImageButton newNote;
    private TextView noNotes;
    private Toolbar toolbar;
    private MenuItem searchMenu;

    private static JSONArray notes; // Main notes array
    private static NoteAdapter adapter; // Custom ListView notes adapter

    // Array of selected positions for deletion
    public static ArrayList<Integer> checkedArray = new ArrayList<Integer>();
    public static boolean deleteActive = false; // True if delete mode is active, false otherwise

    // For disabling long clicks, favourite clicks and modifying the item click pattern
    public static boolean searchActive = false;
    private ArrayList<Integer> realIndexesOfSearchResults; // To keep track of real indexes in searched
        notes

    private int lastFirstVisibleItem = -1; // Last first item seen in list view scroll changed
    private float newNoteButtonBaseYCoordinate; // Base Y coordinate of newNote button

    private AlertDialog backupCheckDialog, backupOKDialog, restoreCheckDialog, restoreFailedDialog;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Intent intent = getIntent();
//        final String userName = intent.getStringExtra("user-name");
//        final String userId = intent.getStringExtra("user-id");

        // set the notes json file
        DataUtils.NOTES_FILE_NAME = "notes_" + com.example.manageme.activities.MainActivity.USERID +
            ".json";

        // Initialize local file path and backup file path
        localPath = new File(getFilesDir() + "/" + NOTES_FILE_NAME);

        File backupFolder = new File(Environment.getExternalStorageDirectory() +
                BACKUP_FOLDER_PATH);

        if (isExternalStorageReadable() && isExternalStorageWritable() && !backupFolder.exists())
            backupFolder.mkdir();

        backupPath = new File(backupFolder, BACKUP_FILE_NAME);

```

```
        // Android version >= 18 -> set orientation userPortrait
        if (Build.VERSION.SDK_INT >= 18)
            setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_USER_PORTRAIT);

        // Android version < 18 -> set orientation sensorPortrait
        else
            setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_SENSOR_PORTRAIT);

        // Init notes array
        notes = new JSONArray();

        // Retrieve from local path
        JSONArray tempNotes = retrieveData(localPath);

        // If not null -> equal main notes to retrieved notes
        if (tempNotes != null)
            notes = tempNotes;

        setContentView(R.layout.notes_activity_main);

        // Setup the main home button
        Button homeButton = (Button) findViewById(R.id.home_button);
        homeButton.setOnClickListener(new View.OnClickListener()
        {    public void onClick(View v)
        {
            Intent intent = new Intent(MainActivity.this,
                com.example.manageme.activities.MainActivity.class);
            intent.putExtra("user-name", com.example.manageme.activities.MainActivity.USERNAME);
            intent.putExtra("user-id", com.example.manageme.activities.MainActivity.USERID);
            startActivity(intent);
            finish();
        }
        });

        // Init layout components
        toolbar = (Toolbar)findViewById(R.id.toolbarMain);
        listView = (ListView)findViewById(R.id.listView);
        newNote = (ImageButton)findViewById(R.id.newNote);
        noNotes = (TextView)findViewById(R.id.noNotes);

        if (toolbar != null)
            initToolbar();

        newNoteButtonBaseYCoordinate = newNote.getY();

        // Initialize NoteAdapter with notes array
        adapter = new NoteAdapter(getApplicationContext(), notes);
        listView.setAdapter(adapter);

        // Set item click, multi choice and scroll listeners
        listView.setOnItemClickListener(this);
        listView.setChoiceMode(ListView.CHOICE_MODE_MULTIPLE_MODAL);
        listView.setMultiChoiceModeListener(this);
        listView.setOnScrollListener(new AbsListView.OnScrollListener() {
            @Override
            public void onScrollStateChanged(AbsListView view, int scrollState) {
                // If last first visible item not initialized -> set to current first
                if (lastFirstVisibleItem == -1)
                    lastFirstVisibleItem = view.getFirstVisiblePosition();

                // If scrolled up -> hide newNote button
```

```
            if (view.getFirstVisiblePosition() > lastFirstVisibleItem)
                newNoteButtonVisibility(false);

            // If scrolled down and delete/search not active -> show newNote button
            else if (view.getFirstVisiblePosition() < lastFirstVisibleItem &&
                   !deleteActive && !searchActive) {

                newNoteButtonVisibility(true);
            }

            // Set last first visible item to current
            lastFirstVisibleItem = view.getFirstVisiblePosition();
        }

        @Override
        public void onScroll(AbsListView view, int firstVisibleItem, int visibleItemCount,
                             int totalItemCount) {}
    });


    // If newNote button clicked -> Start EditActivity intent with NEW_NOTE_REQUEST as request
    newNote.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            Intent intent = new Intent(MainActivity.this, EditActivity.class);
            intent.addFlags(Intent.FLAG_ACTIVITY_NO_ANIMATION);
            intent.putExtra(NOTE_REQUEST_CODE, NEW_NOTE_REQUEST);

            startActivityForResult(intent, NEW_NOTE_REQUEST);
        }
    });

    // If no notes -> show 'Press + to add new note' text, invisible otherwise
    if (notes.length() == 0)
        noNotes.setVisibility(View.VISIBLE);

    else
        noNotes.setVisibility(View.INVISIBLE);

    initDialogs(this);
}


/**
 * Initialize toolbar with required components such as
 * - title, menu/OnMenuItemClickListener and searchView -
 */
protected void initToolbar() {
    toolbar.setTitle(R.string.app_name);

    // Inflate menu_main to be displayed in the toolbar
    toolbar.inflateMenu(R.menu.menu_main);

    // Set an OnMenuItemClickListener to handle menu item clicks
    toolbar.setOnMenuItemClickListener(this);

    Menu menu = toolbar.getMenu();

    if (menu != null) {
        // Get 'Search' menu item
        searchMenu = menu.findItem(R.id.action_search);
```

```
174
175             if (searchMenu != null) {
176                 // If the item menu not null -> get it's support action view
177                 SearchView searchView = (SearchView) MenuItemCompat.getActionView(searchMenu);
178
179                 if (searchView != null) {
180                     // If searchView not null -> set query hint and open/query/close listeners
181                     searchView.setQueryHint(getString(R.string.action_search));
182                     searchView.setOnQueryTextListener(this);
183
184                     MenuItemCompat.setOnActionExpandListener(searchMenu,
185                             new MenuItemCompat.OnActionExpandListener() {
186
187                         @Override
188                         public boolean onMenuItemActionExpand(MenuItem item) {
189                             searchActive = true;
190                             newNoteButtonVisibility(false);
191                             // Disable long-click on listView to prevent deletion
192                             listView.setLongClickable(false);
193
194                             // Init realIndexes array
195                             realIndexesOfSearchResults = new ArrayList<Integer>();
196                             for (int i = 0; i < notes.length(); i++)
197                                 realIndexesOfSearchResults.add(i);
198
199                             adapter.notifyDataSetChanged();
200
201                             return true;
202                         }
203
204                         @Override
205                         public boolean onMenuItemActionCollapse(MenuItem item) {
206                             searchEnded();
207                             return true;
208                         }
209                     });
210                 }
211             }
212         }
213     }
214
215
216     /**
217      * Implementation of AlertDialogs such as
218      * - backupCheckDialog, backupOKDialog, restoreCheckDialog, restoreFailedDialog -
219      * @param context The Activity context of the dialogs; in this case MainActivity context
220      */
221     protected void initDialogs(Context context) {
222         /*
223          * Backup check dialog
224          *  If not sure -> dismiss
225          *  If yes -> check if notes length > 0
226          *    If yes -> save current notes to backup file in backupPath
227          */
228         backupCheckDialog = new AlertDialog.Builder(context)
229                 .setTitle(R.string.action_backup)
230                 .setMessage(R.string.dialog_check_backup_if_sure)
231                 .setPositiveButton(R.string.yes_button, new DialogInterface.OnClickListener() {
232                     @Override
233                     public void onClick(DialogInterface dialog, int which) {
234                         // If note array not empty -> continue
```

```
                if (notes.length() > 0) {
                    boolean backupSuccessful = saveData(backupPath, notes);

                    if (backupSuccessful)
                        showBackupSuccessfulDialog();

                    else {
                        Toast toast = Toast.makeText(getApplicationContext(),
                               getResources().getString(R.string.toast_backup_failed),
                               Toast.LENGTH_SHORT);
                        toast.show();
                    }
                }

                // If notes array is empty -> toast backup no notes found
                else {
                    Toast toast = Toast.makeText(getApplicationContext(),
                           getResources().getString(R.string.toast_backup_no_notes),
                           Toast.LENGTH_SHORT);
                    toast.show();
                }
            }
        })
        .setNegativeButton(R.string.no_button, new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                dialog.dismiss();
            }
        })
        .create();


    // Dialog to display backup was successfully created in backupPath
    backupOKDialog = new AlertDialog.Builder(context)
        .setTitle(R.string.dialog_backup_created_title)
        .setMessage(getString(R.string.dialog_backup_created) + " "
               + backupPath.getAbsolutePath())
        .setNeutralButton(android.R.string.ok, new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                dialog.dismiss();
            }
        })
        .create();


    /*
     * Restore check dialog
     *  If not sure -> dismiss
     *  If yes -> check if backup notes exists
     *    If not -> display restore failed dialog
     *    If yes -> retrieve notes from backup file and store into local file
     */
    restoreCheckDialog = new AlertDialog.Builder(context)
        .setTitle(R.string.action_restore)
        .setMessage(R.string.dialog_check_restore_if_sure)
        .setPositiveButton(R.string.yes_button, new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                JSONArray tempNotes = retrieveData(backupPath);

```

```
                    // If backup file exists -> copy backup notes to local file
                    if (tempNotes != null) {
                        boolean restoreSuccessful = saveData(localPath, tempNotes);

                        if (restoreSuccessful) {
                            notes = tempNotes;

                            adapter = new NoteAdapter(getApplicationContext(), notes);
                            listView.setAdapter(adapter);

                            Toast toast = Toast.makeText(getApplicationContext(),
                                   getResources().getString(R.string.toast_restore_successful),
                                   Toast.LENGTH_SHORT);
                            toast.show();

                            // If no notes -> show 'Press + to add new note' text, invisible otherwise
                            if (notes.length() == 0)
                                noNotes.setVisibility(View.VISIBLE);

                            else
                                noNotes.setVisibility(View.INVISIBLE);
                        }

                        // If restore unsuccessful -> toast restore unsuccessful
                        else {
                            Toast toast = Toast.makeText(getApplicationContext(),
                                   getResources().getString(R.string.toast_restore_unsuccessful),
                                   Toast.LENGTH_SHORT);
                            toast.show();
                        }
                    }

                    // If backup file doesn't exist -> show restore failed dialog
                    else
                        showRestoreFailedDialog();
                }
            })
            .setNegativeButton(R.string.no_button, new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialog, int which) {
                    dialog.dismiss();
                }
            })
            .create();


    // Dialog to display restore failed when no backup file found
    restoreFailedDialog = new AlertDialog.Builder(context)
            .setTitle(R.string.dialog_restore_failed_title)
            .setMessage(R.string.dialog_restore_failed)
            .setNeutralButton(android.R.string.ok, new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialog, int which) {
                    dialog.dismiss();
                }
            })
            .create();
}

// Method to dismiss backup check and show backup successful dialog
protected void showBackupSuccessfulDialog() {
```

```
357         backupCheckDialog.dismiss();
358         backupOKDialog.show();
359     }
360 
361     // Method to dismiss restore check and show restore failed dialog
362     protected void showRestoreFailedDialog() {
363         restoreCheckDialog.dismiss();
364         restoreFailedDialog.show();
365     }
366 
367 
368     /**
369      * If item clicked in list view -> Start EditActivity intent with position as requestCode
370      */
371     @Override
372     public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
373         Intent intent = new Intent(this, EditActivity.class);
374         intent.addFlags(Intent.FLAG_ACTIVITY_NO_ANIMATION);
375 
376         // If search is active -> use position from realIndexesOfSearchResults for EditActivity
377         if (searchActive) {
378             int newPosition = realIndexesOfSearchResults.get(position);
379 
380             try {
381                 // Package selected note content and send to EditActivity
382                 intent.putExtra(NOTE_TITLE, notes.getJSONObject(newPosition).getString(NOTE_TITLE));
383                 intent.putExtra(NOTE_BODY, notes.getJSONObject(newPosition).getString(NOTE_BODY));
384                 intent.putExtra(NOTE_COLOUR, notes.getJSONObject(newPosition).getString(NOTE_COLOUR));
385                 intent.putExtra(NOTE_FONT_SIZE, notes.getJSONObject(newPosition).getInt(NOTE_FONT_SIZE));
386 
387                 if (notes.getJSONObject(newPosition).has(NOTE_HIDE_BODY)) {
388                     intent.putExtra(NOTE_HIDE_BODY,
389                             notes.getJSONObject(newPosition).getBoolean(NOTE_HIDE_BODY));
390                 }
391 
392                 else
393                     intent.putExtra(NOTE_HIDE_BODY, false);
394 
395             } catch (JSONException e) {
396                 e.printStackTrace();
397             }
398 
399             intent.putExtra(NOTE_REQUEST_CODE, newPosition);
400             startActivityForResult(intent, newPosition);
401         }
402 
403         // If search is not active -> use normal position for EditActivity
404         else {
405             try {
406                 // Package selected note content and send to EditActivity
407                 intent.putExtra(NOTE_TITLE, notes.getJSONObject(position).getString(NOTE_TITLE));
408                 intent.putExtra(NOTE_BODY, notes.getJSONObject(position).getString(NOTE_BODY));
409                 intent.putExtra(NOTE_COLOUR, notes.getJSONObject(position).getString(NOTE_COLOUR));
410                 intent.putExtra(NOTE_FONT_SIZE, notes.getJSONObject(position).getInt(NOTE_FONT_SIZE));
411 
412                 if (notes.getJSONObject(position).has(NOTE_HIDE_BODY)) {
413                     intent.putExtra(NOTE_HIDE_BODY,
414                             notes.getJSONObject(position).getBoolean(NOTE_HIDE_BODY));
415                 }
416 
417                 else
```

```
                intent.putExtra(NOTE_HIDE_BODY, false);

            } catch (JSONException e) {
                e.printStackTrace();
            }

            intent.putExtra(NOTE_REQUEST_CODE, position);
            startActivityForResult(intent, position);
        }
    }


    /**
     * Item clicked in Toolbar menu callback method
     * @param menuItem Item clicked
     * @return true if click detected and logic finished, false otherwise
     */
    @Override
    public boolean onMenuItemClick(MenuItem menuItem) {
        int id = menuItem.getItemId();
        return false;
    }


    /**
     * During multi-choice menu_delete selection mode, callback method if items checked changed
     * @param mode ActionMode of selection
     * @param position Position checked
     * @param id ID of item, if exists
     * @param checked true if checked, false otherwise
     */
    @Override
    public void onItemCheckedStateChanged(ActionMode mode, int position, long id, boolean checked) {
        // If item checked -> add to array
        if (checked)
            checkedArray.add(position);

        // If item unchecked
        else {
            int index = -1;

            // Loop through array and find index of item unchecked
            for (int i = 0; i < checkedArray.size(); i++) {
                if (position == checkedArray.get(i)) {
                    index = i;
                    break;
                }
            }

            // If index was found -> remove the item
            if (index != -1)
                checkedArray.remove(index);
        }

        // Set Toolbar title to 'x Selected'
        mode.setTitle(checkedArray.size() + " " + getString(R.string.action_delete_selected_number));
        adapter.notifyDataSetChanged();
    }

    /**
     * Callback method when 'Delete' icon pressed
```

```
     * @param mode ActionMode of selection
     * @param item MenuItem clicked, in our case just action_delete
     * @return true if clicked, false otherwise
     */
    @Override
    public boolean onActionItemClicked(final ActionMode mode, MenuItem item) {
        if (item.getItemId() == R.id.action_delete) {
            new AlertDialog.Builder(this)
                    .setMessage(R.string.dialog_delete)
                    .setPositiveButton(android.R.string.ok, new DialogInterface.OnClickListener() {
                        @Override
                        public void onClick(DialogInterface dialog, int which) {
                            // Pass notes and checked items for deletion array to 'deleteNotes'
                            notes = deleteNotes(notes, checkedArray);

                            // Create and set new adapter with new notes array
                            adapter = new NoteAdapter(getApplicationContext(), notes);
                            listView.setAdapter(adapter);

                            // Attempt to save notes to local file
                            Boolean saveSuccessful = saveData(localPath, notes);

                            // If save successful -> toast successfully deleted
                            if (saveSuccessful) {
                                Toast toast = Toast.makeText(getApplicationContext(),
                                        getResources().getString(R.string.toast_deleted),
                                        Toast.LENGTH_SHORT);
                                toast.show();
                            }

                            // Smooth scroll to top
                            listView.post(new Runnable() {
                                public void run() {
                                    listView.smoothScrollToPosition(0);
                                }
                            });

                            // If no notes -> show 'Press + to add new note' text, invisible otherwise
                            if (notes.length() == 0)
                                noNotes.setVisibility(View.VISIBLE);

                            else
                                noNotes.setVisibility(View.INVISIBLE);

                            mode.finish();
                        }
                    })
                    .setNegativeButton(android.R.string.cancel, new DialogInterface.OnClickListener() {
                        @Override
                        public void onClick(DialogInterface dialog, int which) {
                            dialog.dismiss();
                        }
                    })
                    .show();

            return true;
        }

        return false;
    }

```

```
    // Long click detected on ListView item -> start selection ActionMode (delete mode)
    @Override
    public boolean onCreateActionMode(ActionMode mode, Menu menu) {
        mode.getMenuInflater().inflate(R.menu.menu_delete, menu); // Inflate 'menu_delete' menu
        deleteActive = true; // Set deleteActive to true as we entered delete mode
        newNoteButtonVisibility(false); // Hide newNote button
        adapter.notifyDataSetChanged(); // Notify adapter to hide favourite buttons

        return true;
    }

    // Selection ActionMode finished (delete mode ended)
    @Override
    public void onDestroyActionMode(ActionMode mode) {
        checkedArray = new ArrayList<Integer>(); // Reset checkedArray
        deleteActive = false; // Set deleteActive to false as we finished delete mode
        newNoteButtonVisibility(true); // Show newNote button
        adapter.notifyDataSetChanged(); // Notify adapter to show favourite buttons
    }

    @Override
    public boolean onPrepareActionMode(ActionMode mode, Menu menu) {
        return false;
    }


    /**
     * Method to show and hide the newNote button
     * @param isVisible true to show button, false to hide
     */
    @TargetApi(Build.VERSION_CODES.ICE_CREAM_SANDWICH)
    protected void newNoteButtonVisibility(boolean isVisible) {
        if (isVisible) {
            newNote.animate().cancel();
            newNote.animate().translationY(newNoteButtonBaseYCoordinate);
        } else {
            newNote.animate().cancel();
            newNote.animate().translationY(newNoteButtonBaseYCoordinate + 500);
        }
    }


    /**
     * Callback method for 'searchView' menu item widget text change
     * @param s String which changed
     * @return true if text changed and logic finished, false otherwise
     */
    @Override
    public boolean onQueryTextChange(String s) {
        s = s.toLowerCase(); // Turn string into lowercase

        // If query text length longer than 0
        if (s.length() > 0) {
            // Create new JSONArray and reset realIndexes array
            JSONArray notesFound = new JSONArray();
            realIndexesOfSearchResults = new ArrayList<Integer>();

            // Loop through main notes list
            for (int i = 0; i < notes.length(); i++) {
                JSONObject note = null;
```

```
601                 // Get note at position i
602                 try {
603                     note = notes.getJSONObject(i);
604 
605                 } catch (JSONException e) {
606                     e.printStackTrace();
607                 }
608 
609                 // If note not null and title/body contain query text
610                 // -> Put in new notes array and add i to realIndexes array
611                 if (note != null) {
612                     try {
613                         if (note.getString(NOTE_TITLE).toLowerCase().contains(s) ||
614                             note.getString(NOTE_BODY).toLowerCase().contains(s)) {
615 
616                             notesFound.put(note);
617                             realIndexesOfSearchResults.add(i);
618                         }
619 
620                     } catch (JSONException e) {
621                         e.printStackTrace();
622                     }
623                 }
624             }
625 
626             // Create and set adapter with notesFound to refresh ListView
627             NoteAdapter searchAdapter = new NoteAdapter(getApplicationContext(), notesFound);
628             listView.setAdapter(searchAdapter);
629         }
630 
631         // If query text length is 0 -> re-init realIndexes array (0 to length) and reset adapter
632         else {
633             realIndexesOfSearchResults = new ArrayList<Integer>();
634             for (int i = 0; i < notes.length(); i++)
635                 realIndexesOfSearchResults.add(i);
636 
637             adapter = new NoteAdapter(getApplicationContext(), notes);
638             listView.setAdapter(adapter);
639         }
640 
641         return false;
642     }
643 
644     @Override
645     public boolean onQueryTextSubmit(String s) {
646         return false;
647     }
648 
649 
650     /**
651      * When search mode is finished
652      * Collapse searchView widget, searchActive to false, reset adapter, enable listView long clicks
653      * and show newNote button
654      */
655     protected void searchEnded() {
656         searchActive = false;
657         adapter = new NoteAdapter(getApplicationContext(), notes);
658         listView.setAdapter(adapter);
659         listView.setLongClickable(true);
660         newNoteButtonVisibility(true);
661     }
```

```
662
663
664     /**
665      * Callback method when EditActivity finished adding new note or editing existing note
666      * @param requestCode requestCode for intent sent, in our case either NEW_NOTE_REQUEST or position
667      * @param resultCode resultCode from activity, either RESULT_OK or RESULT_CANCELED
668      * @param data Data bundle passed back from EditActivity
669      */
670     @Override
671     protected void onActivityResult(int requestCode, int resultCode, Intent data) {
672         if (resultCode == RESULT_OK) {
673             // If search was active -> call 'searchEnded' method
674             if (searchActive && searchMenu != null)
675                 searchMenu.collapseActionView();
676
677             // Get extras
678             Bundle mBundle = null;
679             if (data != null)
680                 mBundle = data.getExtras();
681
682             if (mBundle != null) {
683                 // If new note was saved
684                 if (requestCode == NEW_NOTE_REQUEST) {
685                     JSONObject newNoteObject = null;
686
687                     try {
688                         // Add new note to array
689                         newNoteObject = new JSONObject();
690                         newNoteObject.put(NOTE_TITLE, mBundle.getString(NOTE_TITLE));
691                         newNoteObject.put(NOTE_BODY, mBundle.getString(NOTE_BODY));
692                         newNoteObject.put(NOTE_COLOUR, mBundle.getString(NOTE_COLOUR));
693                         newNoteObject.put(NOTE_FAVOURED, false);
694                         newNoteObject.put(NOTE_FONT_SIZE, mBundle.getInt(NOTE_FONT_SIZE));
695                         newNoteObject.put(NOTE_HIDE_BODY, mBundle.getBoolean(NOTE_HIDE_BODY));
696
697                         notes.put(newNoteObject);
698
699                     } catch (JSONException e) {
700                         e.printStackTrace();
701                     }
702
703                     // If newNoteObject not null -> save notes array to local file and notify adapter
704                     if (newNoteObject != null) {
705                         adapter.notifyDataSetChanged();
706
707                         Boolean saveSuccessful = saveData(localPath, notes);
708
709                         if (saveSuccessful) {
710                             Toast toast = Toast.makeText(getApplicationContext(),
711                                     getResources().getString(R.string.toast_new_note),
712                                     Toast.LENGTH_SHORT);
713                             toast.show();
714                         }
715
716                         // If no notes -> show 'Press + to add new note' text, invisible otherwise
717                         if (notes.length() == 0)
718                             noNotes.setVisibility(View.VISIBLE);
719
720                         else
721                             noNotes.setVisibility(View.INVISIBLE);
722                     }
```

```
723                 }
724
725                 // If existing note was updated (saved)
726                 else {
727                     JSONObject newNoteObject = null;
728
729                     try {
730                         // Update array item with new note data
731                         newNoteObject = notes.getJSONObject(requestCode);
732                         newNoteObject.put(NOTE_TITLE, mBundle.getString(NOTE_TITLE));
733                         newNoteObject.put(NOTE_BODY, mBundle.getString(NOTE_BODY));
734                         newNoteObject.put(NOTE_COLOUR, mBundle.getString(NOTE_COLOUR));
735                         newNoteObject.put(NOTE_FONT_SIZE, mBundle.getInt(NOTE_FONT_SIZE));
736                         newNoteObject.put(NOTE_HIDE_BODY, mBundle.getBoolean(NOTE_HIDE_BODY));
737
738                         // Update note at position 'requestCode'
739                         notes.put(requestCode, newNoteObject);
740
741                     } catch (JSONException e) {
742                         e.printStackTrace();
743                     }
744
745                     // If newNoteObject not null -> save notes array to local file and notify adapter
746                     if (newNoteObject != null) {
747                         adapter.notifyDataSetChanged();
748
749                         Boolean saveSuccessful = saveData(localPath, notes);
750
751                         if (saveSuccessful) {
752                             Toast toast = Toast.makeText(getApplicationContext(),
753                                     getResources().getString(R.string.toast_note_saved),
754                                     Toast.LENGTH_SHORT);
755                             toast.show();
756                         }
757                     }
758                 }
759             }
760         }
761
762
763         else if (resultCode == RESULT_CANCELED) {
764             Bundle mBundle = null;
765
766             // If data is not null, has "request" extra and is new note -> get extras to bundle
767             if (data != null && data.hasExtra("request") && requestCode == NEW_NOTE_REQUEST) {
768                 mBundle = data.getExtras();
769
770                 // If new note discarded -> toast empty note discarded
771                 if (mBundle != null && mBundle.getString("request").equals("discard")) {
772                     Toast toast = Toast.makeText(getApplicationContext(),
773                             getResources().getString(R.string.toast_empty_note_discarded),
774                             Toast.LENGTH_SHORT);
775                     toast.show();
776                 }
777             }
778         }
779
780         super.onActivityResult(requestCode, resultCode, data);
781     }
782
783
```

```
784     /**
785      * Favourite or un-favourite the note at position
786      * @param context application context
787      * @param favourite true to favourite, false to un-favourite
788      * @param position position of note
789      */
790     public static void setFavourite(Context context, boolean favourite, int position) {
791         JSONObject newFavourite = null;
792 
793         // Get note at position and store in newFavourite
794         try {
795             newFavourite = notes.getJSONObject(position);
796 
797         } catch (JSONException e) {
798             e.printStackTrace();
799         }
800 
801         if (newFavourite != null) {
802             if (favourite) {
803                 // Set favoured to true
804                 try {
805                     newFavourite.put(NOTE_FAVOURED, true);
806 
807                 } catch (JSONException e) {
808                     e.printStackTrace();
809                 }
810 
811                 // If favoured note is not at position 0
812                 // Sort notes array so favoured note is first
813                 if (position > 0) {
814                     JSONArray newArray = new JSONArray();
815 
816                     try {
817                         newArray.put(0, newFavourite);
818 
819                     } catch (JSONException e) {
820                         e.printStackTrace();
821                     }
822 
823                     // Copy contents to new sorted array without favoured element
824                     for (int i = 0; i < notes.length(); i++) {
825                         if (i != position) {
826                             try {
827                                 newArray.put(notes.get(i));
828 
829                             } catch (JSONException e) {
830                                 e.printStackTrace();
831                             }
832                         }
833                     }
834 
835                     // Equal main notes array with new sorted array and reset adapter
836                     notes = newArray;
837                     adapter = new NoteAdapter(context, notes);
838                     listView.setAdapter(adapter);
839 
840                     // Smooth scroll to top
841                     listView.post(new Runnable() {
842                         public void run() {
843                             listView.smoothScrollToPosition(0);
844                         }
```

```
                });
            }

            // If favoured note was first -> just update object in notes array and notify adapter
            else {
                try {
                    notes.put(position, newFavourite);

                } catch (JSONException e) {
                    e.printStackTrace();
                }

                adapter.notifyDataSetChanged();
            }
        }

        // If note not favourite -> set favoured to false and notify adapter
        else {
            try {
                newFavourite.put(NOTE_FAVOURED, false);
                notes.put(position, newFavourite);

            } catch (JSONException e) {
                e.printStackTrace();
            }

            adapter.notifyDataSetChanged();
        }

        // Save notes to local file
        saveData(localPath, notes);
    }
}


/**
 * If back button pressed while search is active -> collapse view and end search mode
 */
@TargetApi(Build.VERSION_CODES.ICE_CREAM_SANDWICH)
@Override
public void onBackPressed() {
    if (searchActive && searchMenu != null) {
        searchMenu.collapseActionView();
        return;
    }

    super.onBackPressed();
}


/**
 * Orientation changed callback method
 * If orientation changed -> If any AlertDialog is showing, dismiss it to prevent WindowLeaks
 * @param newConfig New Configuration passed by system
 */
@Override
public void onConfigurationChanged(Configuration newConfig) {
    if (backupCheckDialog != null && backupCheckDialog.isShowing())
        backupCheckDialog.dismiss();

    if (backupOKDialog != null && backupOKDialog.isShowing())
```

```
906             backupOKDialog.dismiss();
907 
908         if (restoreCheckDialog != null && restoreCheckDialog.isShowing())
909             restoreCheckDialog.dismiss();
910 
911         if (restoreFailedDialog != null && restoreFailedDialog.isShowing())
912             restoreFailedDialog.dismiss();
913 
914         super.onConfigurationChanged(newConfig);
915     }
916 
917 
918     // Static method to return File at localPath
919     public static File getLocalPath() {
920         return localPath;
921     }
922 
923     // Static method to return File at backupPath
924     public static File getBackupPath() {
925         return backupPath;
926     }
927 }
```

1.7.2 NoteAdapter.java

Listing 1.16: NoteAdapter.java

```
class NoteAdapter extends BaseAdapter implements ListAdapter {
    private Context context;
    private JSONArray adapterData;
    private LayoutInflater inflater;

    /**
     * Adapter constructor -> Sets class variables
     * @param context application context
     * @param adapterData JSONArray of notes
     */
    NoteAdapter(Context context, JSONArray adapterData) {
        this.context = context;
        this.adapterData = adapterData;
        this.inflater = (LayoutInflater) context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
    }

    // Return number of notes
    @Override
    public int getCount() {
        if (this.adapterData != null)
            return this.adapterData.length();

        else
            return 0;
    }

    // Return note at position
    @Override
    public JSONObject getItem(int position) {
        if (this.adapterData != null)
            return this.adapterData.optJSONObject(position);

        else
            return null;
    }

    @Override
    public long getItemId(int position) {
        return 0;
    }


    // View inflater
    @TargetApi(Build.VERSION_CODES.JELLY_BEAN)
    @Override
    public View getView(final int position, View convertView, ViewGroup parent) {
        // Inflate custom note view if null
        if (convertView == null)
            convertView = this.inflater.inflate(R.layout.notes_list_view_note, parent, false);

        // Initialize layout items
        RelativeLayout relativeLayout = (RelativeLayout) convertView.findViewById(R.id.relativeLayout);
        LayerDrawable roundedCard = (LayerDrawable)
             context.getResources().getDrawable(R.drawable.rounded_card);
        TextView titleView = (TextView) convertView.findViewById(R.id.titleView);
        TextView bodyView = (TextView) convertView.findViewById(R.id.bodyView);
        ImageButton favourite = (ImageButton) convertView.findViewById(R.id.favourite);
```

```
57
58          // Get Note object at position
59          JSONObject noteObject = getItem(position);
60
61          if (noteObject != null) {
62              // If noteObject not empty -> initialize variables
63              String title = context.getString(R.string.note_title);
64              String body = context.getString(R.string.note_body);
65              String colour = String.valueOf(context.getResources().getColor(R.color.white));
66              int fontSize = 18;
67              Boolean hideBody = false;
68              Boolean favoured = false;
69
70              try {
71                  // Get noteObject data and store in variables
72                  title = noteObject.getString(NOTE_TITLE);
73                  body = noteObject.getString(NOTE_BODY);
74                  colour = noteObject.getString(NOTE_COLOUR);
75
76                  if (noteObject.has(NOTE_FONT_SIZE))
77                      fontSize = noteObject.getInt(NOTE_FONT_SIZE);
78
79                  if (noteObject.has(NOTE_HIDE_BODY))
80                      hideBody = noteObject.getBoolean(NOTE_HIDE_BODY);
81
82                  favoured = noteObject.getBoolean(NOTE_FAVOURED);
83
84              } catch (JSONException e) {
85                  e.printStackTrace();
86              }
87
88              // Set favourite image resource
89              if (favoured)
90                  favourite.setImageResource(R.drawable.ic_fav);
91
92              else
93                  favourite.setImageResource(R.drawable.ic_unfav);
94
95
96              // If search or delete modes are active -> hide favourite button; Show otherwise
97              if (searchActive || deleteActive)
98                  favourite.setVisibility(View.INVISIBLE);
99
100             else
101                 favourite.setVisibility(View.VISIBLE);
102
103
104             titleView.setText(title);
105
106             // If hidBody is true -> hide body of note
107             if (hideBody)
108                 bodyView.setVisibility(View.GONE);
109
110             // Else -> set visible note body, text to normal and set text size to 'fontSize' as sp
111             else {
112                 bodyView.setVisibility(View.VISIBLE);
113                 bodyView.setText(body);
114                 bodyView.setTextSize(TypedValue.COMPLEX_UNIT_SP, fontSize);
115             }
116
117             // If current note is selected for deletion -> highlight
```

```
            if (checkedArray.contains(position)) {
                ((GradientDrawable) roundedCard.findDrawableByLayerId(R.id.card))
                        .setColor(context.getResources().getColor(R.color.theme_primary));
            }

            // If current note is not selected -> set background colour to normal
            else {
                ((GradientDrawable) roundedCard.findDrawableByLayerId(R.id.card))
                        .setColor(Color.parseColor(colour));
            }

            // Set note background style to rounded card
            relativeLayout.setBackground(roundedCard);

            final Boolean finalFavoured = favoured;
            favourite.setOnClickListener(new View.OnClickListener() {
                // If favourite button was clicked -> change that note to favourite or un-favourite
                @Override
                public void onClick(View v) {
                    setFavourite(context, !finalFavoured, position);
                }
            });
        }

        return convertView;
    }
}
```

1.8 Providers

1.8.1 CalorieProvider.java

Listing 1.17: CalorieProvider.java

```
public class CaloriesProvider extends ContentProvider {
    public static final int CALORIES = 10;
    public static final int CALORIES_ID = 11;

    public static final int CALORIE_CATEGORIES = 20;
    public static final int CALORIE_CATEGORIES_ID = 21;

    public static final int CALORIES_WITH_CATEGORIES = 30;
    public static final int CALORIES_WITH_CATEGORIES_ID = 31;

    public static final int CALORIES_WITH_CATEGORIES_DATE = 32;
    public static final int CALORIES_WITH_CATEGORIES_DATE_RANGE = 33;
    public static final int CALORIES_WITH_CATEGORIES_SUM_DATE = 34;
    public static final int CALORIES_WITH_CATEGORIES_SUM_DATE_RANGE = 35;

    private SQLiteOpenHelper mDbHelper;
    private SQLiteDatabase mDatabase;

    private static final UriMatcher sUriMatcher = new UriMatcher(UriMatcher.NO_MATCH);

    static {
        sUriMatcher.addURI(CaloriesContract.AUTHORITY, "calories", CALORIES);
        sUriMatcher.addURI(CaloriesContract.AUTHORITY, "calories/#", CALORIES_ID);
        sUriMatcher.addURI(CaloriesContract.AUTHORITY, "calorieCategories", CALORIE_CATEGORIES);
        sUriMatcher.addURI(CaloriesContract.AUTHORITY, "calorieCategories/#", CALORIE_CATEGORIES_ID);
        sUriMatcher.addURI(CaloriesContract.AUTHORITY, "caloriesWithCategories",
                CALORIES_WITH_CATEGORIES);
        sUriMatcher.addURI(CaloriesContract.AUTHORITY, "caloriesWithCategories/#",
                CALORIES_WITH_CATEGORIES_ID);
        sUriMatcher.addURI(CaloriesContract.AUTHORITY, "caloriesWithCategories/date",
                CALORIES_WITH_CATEGORIES_DATE);
        sUriMatcher.addURI(CaloriesContract.AUTHORITY, "caloriesWithCategories/dateRange",
                CALORIES_WITH_CATEGORIES_DATE_RANGE);
        sUriMatcher.addURI(CaloriesContract.AUTHORITY, "caloriesWithCategories/date/sum",
                CALORIES_WITH_CATEGORIES_SUM_DATE);
        sUriMatcher.addURI(CaloriesContract.AUTHORITY, "caloriesWithCategories/dateRange/sum",
                CALORIES_WITH_CATEGORIES_SUM_DATE_RANGE);
    }

    /*
     * SELECT expenses._id, expenses.value, categories.name, expenses.date
     * FROM expenses JOIN categories
     * ON expenses.category_id = categories._id
     */
    private static final String BASE_SELECT_JOIN_EXPENSES_CATEGORIES_QUERY =
            "SELECT " + CALORIES_TABLE_NAME + "." + Calories._ID + ", " +
                    CALORIES_TABLE_NAME + "." + Calories.CALORIE_CATEGORY_ID + ", " +
                    CALORIES_CATEGORIES_TABLE_NAME + "." + CalorieCategories.NAME + ", " +
                    CALORIES_CATEGORIES_TABLE_NAME + "." + CalorieCategories.VALUE + ", " +
                    CALORIES_TABLE_NAME + "." + Calories.DATE + " FROM " +
                    CALORIES_TABLE_NAME + " JOIN " + CALORIES_CATEGORIES_TABLE_NAME + " ON " +
                    CALORIES_TABLE_NAME + "." + Calories.CALORIE_CATEGORY_ID + " = " +
                    CALORIES_CATEGORIES_TABLE_NAME + "." + CalorieCategories._ID;

```

```
55      /**
56       * <p>
57       * Initializes the provider.
58       * </p>
59       *
60       * <i>Note</i>: provider is not created until a
61       * {@link android.content.ContentResolver ContentResolver} object tries to access it.
62       *
63       * @return <code>true</code> if the provider was successfully loaded, <code>false</code> otherwise
64       */
65      @Override
66      public boolean onCreate() {
67          mDbHelper = new CalorieDbHelper(getContext());
68          return true;
69      }
70
71      @Override
72      public Cursor query(Uri uri, String[] projection, String selection, String[] selectionArgs, String
            sortOrder) {
73          Cursor cursor;
74          String table;
75          String rawQuery;
76          mDatabase = mDbHelper.getReadableDatabase();
77          String sqlquery = "SELECT TABLE_NAME\n" +
78                  "FROM INFORMATION_SCHEMA.TABLES\n" +
79                  "WHERE TABLE_TYPE = 'BASE TABLE' AND TABLE_CATALOG='calorie_tracer.db'";
80          Log.d("createable", String.valueOf(sUriMatcher.match(uri)));
81
82          switch (sUriMatcher.match(uri)) {
83              // The incoming URI is for all of categories
84              case CALORIE_CATEGORIES:
85                  table = CALORIES_CATEGORIES_TABLE_NAME;
86                  selectionArgs = null;
87                  sortOrder = (sortOrder == null || sortOrder.isEmpty())
88                          ? CalorieCategories.DEFAULT_SORT_ORDER
89                          : sortOrder;
90                  break;
91  //              "user_profile_number = ? AND pkg_name = ? ", new String[]{"2", "abc"}
92              // The incoming URI is for a single row from categories
93              case CALORIE_CATEGORIES_ID:
94                  table = CALORIES_CATEGORIES_TABLE_NAME;
95                  // Defines selection criteria for the row to query
96                  selection = CalorieCategories._ID + " = ? " ;
97                  selectionArgs = new String[]{ uri.getLastPathSegment() };
98                  break;
99
100             // The incoming URI is for all of expenses
101             case CALORIES:
102                 Log.d("create", "Calories");
103                 table = CALORIES_TABLE_NAME;
104                 selection = Calories.USER_ID + " = ? " ;
105                 selectionArgs = new String[]{selectionArgs[0]};
106                 sortOrder = (sortOrder == null || sortOrder.isEmpty())
107                         ? Calories.DEFAULT_SORT_ORDER
108                         : sortOrder;
109                 break;
110
111             // The incoming URI is for a single row from expenses
112             case CALORIES_ID:
113                 Log.d("create", "CaloriesId");
114                 table = CALORIES_TABLE_NAME;
```

```
            // Defines selection criteria for the row to query
            selection = Calories.USER_ID + " = ? AND " + Calories._ID + " = ?";
            selectionArgs = new String[]{selectionArgs[0], uri.getLastPathSegment() };
            break;

        // The incoming URI is for all expenses with categories
        case CALORIES_WITH_CATEGORIES:
            /*
             * SELECT expenses._id, expenses.value, categories.name, expenses.date
             * FROM expenses JOIN categories
             * ON expenses.category_id = categories._id
             */
            rawQuery =
                   BASE_SELECT_JOIN_EXPENSES_CATEGORIES_QUERY + " WHERE " +
                           CALORIES_TABLE_NAME + "." + Calories.USER_ID + " = ?";

            return mDatabase.rawQuery(rawQuery, selectionArgs);

        // The incoming URI is for all expenses with categories
        case CALORIES_WITH_CATEGORIES_ID:
            /*
             * SELECT expenses._id, expenses.value, categories.name, expenses.date
             * FROM expenses JOIN categories
             * ON expenses.category_id = categories._id
             */
            rawQuery =
                   BASE_SELECT_JOIN_EXPENSES_CATEGORIES_QUERY + " WHERE " +
                           CALORIES_TABLE_NAME + "." + Calories.USER_ID + " = ? AND " +
                       CALORIES_TABLE_NAME + "." + Calories._ID + " = ?";

            selectionArgs = new String[]{ selectionArgs[0], uri.getLastPathSegment() };
            return mDatabase.rawQuery(rawQuery, selectionArgs);

        // The incoming URI is for the expenses with categories for a specific date
        case CALORIES_WITH_CATEGORIES_DATE:
            /*
             * SELECT expenses._id, expenses.value, categories.name, expenses.date
             * FROM expenses JOIN categories
             * ON expenses.category_id = categories._id
             * WHERE expense.date = ?
             */
            rawQuery =
                   BASE_SELECT_JOIN_EXPENSES_CATEGORIES_QUERY + " WHERE " +
                           CALORIES_TABLE_NAME + "." + Calories.USER_ID + " = ? AND " +
                           CALORIES_TABLE_NAME + "." + Calories.DATE + " = ?";
            if(selectionArgs != null) {
                Log.d("selection", String.valueOf(selectionArgs.length));

                for(int i = 0;i < selectionArgs.length;i++){
                    if(selectionArgs[i] != null)
                        Log.d("selection", String.valueOf(i) +" " +selectionArgs[i]);
                }
            }

            return mDatabase.rawQuery(rawQuery, selectionArgs);

        // The incoming URI is for the expense values sum for a specific date range
        case CALORIES_WITH_CATEGORIES_SUM_DATE:
            /*
             * SELECT SUM(expenses.value) as values_sum
             * FROM expenses WHERE expenses.date = ?
```

```
                 */
                /*
                SELECT a.id, a.name, a.num, b.date, b.roll
                FROM a
                INNER JOIN b ON a.id=b.id;
                */
                rawQuery =
                        "SELECT SUM(" + CALORIES_CATEGORIES_TABLE_NAME + "." + CalorieCategories.VALUE + ")
                            as " +
                                Calories.VALUES_SUM + " FROM " + CALORIES_CATEGORIES_TABLE_NAME +
                                " INNER JOIN " + CALORIES_TABLE_NAME +" ON " +
                                CALORIES_TABLE_NAME + "." + Calories.CALORIE_CATEGORY_ID + " = " +
                                CALORIES_CATEGORIES_TABLE_NAME + "." + CalorieCategories._ID + " WHERE " +
                                CALORIES_TABLE_NAME + "." + Calories.USER_ID + " = ? AND " +
                                CALORIES_TABLE_NAME + "." + Calories.DATE + " = ?";
                if(selectionArgs != null) {
                    Log.d("selection", String.valueOf(selectionArgs.length));

                    for(int i = 0;i < selectionArgs.length;i++){
                        if(selectionArgs[i] != null)
                            Log.d("selection", String.valueOf(i) +" " +selectionArgs[i]);
                    }
                }

                return mDatabase.rawQuery(rawQuery, selectionArgs);

            // The incoming URI is for the calories with categories for a specific date range
            case CALORIES_WITH_CATEGORIES_DATE_RANGE:
                /*
                 * SELECT expenses._id, expenses.value, categories.name, expenses.date
                 * FROM expenses JOIN categories
                 * ON expenses.category_id = categories._id
                 * WHERE expense.date BETWEEN ? AND ?
                 */
                rawQuery =
                        BASE_SELECT_JOIN_EXPENSES_CATEGORIES_QUERY + " WHERE " +
                                CALORIES_TABLE_NAME + "." + Calories.USER_ID + " = ? AND " +
                                CALORIES_TABLE_NAME + "." + Calories.DATE + " BETWEEN ? AND ?";

                return mDatabase.rawQuery(rawQuery, selectionArgs);

            // The incoming URI is for the expense values sum for a specific date range
            case CALORIES_WITH_CATEGORIES_SUM_DATE_RANGE:
                /*
                 * SELECT SUM(expenses.value) as values_sum
                 * FROM expenses WHERE expense.date BETWEEN ? AND ?
                 */
                rawQuery =
                        "SELECT SUM(" + CALORIES_CATEGORIES_TABLE_NAME + "." + CalorieCategories.VALUE + ")
                            as " +
                                Calories.VALUES_SUM + " FROM " + CALORIES_CATEGORIES_TABLE_NAME +
                                " INNER JOIN " + CALORIES_TABLE_NAME +" ON " +
                                CALORIES_TABLE_NAME + "." + Calories.CALORIE_CATEGORY_ID + " = " +
                                CALORIES_CATEGORIES_TABLE_NAME + "." + CalorieCategories._ID + " WHERE " +
                                CALORIES_TABLE_NAME + "." + Calories.USER_ID + " = ? AND " +
                                CALORIES_TABLE_NAME + "." + Calories.DATE + " BETWEEN ? AND ?";

                return mDatabase.rawQuery(rawQuery, selectionArgs);

            default:
                throw new IllegalArgumentException("Unknown Uri provided.");
```

```
        }


        if (selection != null)
            Log.d("selection", selection);
        if(selectionArgs != null) {
            for(int i = 0;i < selectionArgs.length;i++)
                Log.d("selection", selectionArgs[i]);
        }

        cursor = mDatabase.query(
                table,
                projection,
                selection,
                selectionArgs,
                null,
                null,
                sortOrder
        );

        return cursor;
    }

    @Override
    public Uri insert(Uri uri, ContentValues values) {
        String table;
        Uri contentUri;
        switch (sUriMatcher.match(uri)) {
            // The incoming URI is for all of categories
            case CALORIE_CATEGORIES:
                table = CALORIES_CATEGORIES_TABLE_NAME;
                contentUri = CalorieCategories.CONTENT_URI;
                break;
            // The incoming URI is for all of expenses
            case CALORIES:
                table = CALORIES_TABLE_NAME;
                contentUri = Calories.CONTENT_URI;
                break;
            // The incoming URI is for a single row from categories
            case CALORIE_CATEGORIES_ID:
                // The incoming URI is for a single row from expenses
            case CALORIES_ID:
                throw new UnsupportedOperationException("Inserting rows with specified IDs is forbidden.");
            case CALORIES_WITH_CATEGORIES:
            case CALORIES_WITH_CATEGORIES_DATE:
            case CALORIES_WITH_CATEGORIES_DATE_RANGE:
            case CALORIES_WITH_CATEGORIES_SUM_DATE:
            case CALORIES_WITH_CATEGORIES_SUM_DATE_RANGE:
                throw new UnsupportedOperationException("Modifying joined results is forbidden.");
            default:
                throw new IllegalArgumentException("Unknown Uri provided.");
        }

        mDatabase = mDbHelper.getWritableDatabase();

        long newRowID = mDatabase.insert(
                table,
                null,
                values
        );

```

```
296         Uri newItemUri = ContentUris.withAppendedId(contentUri, newRowID);
297 
298         return (newRowID < 1) ? null : newItemUri;
299     }
300 
301     @Override
302     public int delete(Uri uri, String selection, String[] selectionArgs) {
303         String table;
304         switch (sUriMatcher.match(uri)) {
305             // The incoming URI is for a single row from categories
306             case CALORIE_CATEGORIES_ID:
307                 table = CALORIES_CATEGORIES_TABLE_NAME;
308                 // Defines selection criteria for the row to delete
309                 selection = CalorieCategories._ID + " = ?";
310                 selectionArgs = new String[]{ uri.getLastPathSegment() };
311                 break;
312             // The incoming URI is for all of expenses
313             case CALORIES:
314                 table = CALORIES_TABLE_NAME;
315                 break;
316             // The incoming URI is for a single row from expenses
317             case CALORIES_ID:
318                 table = CALORIES_TABLE_NAME;
319                 // Defines selection criteria for the row to delete
320                 selection = Calories._ID + " = ?";
321                 selectionArgs = new String[]{ uri.getLastPathSegment() };
322                 break;
323             // The incoming URI is for all of categories
324             case CALORIE_CATEGORIES:
325                 throw new UnsupportedOperationException("Removing multiple rows from the table is
                        forbidden.");
326             case CALORIES_WITH_CATEGORIES:
327             case CALORIES_WITH_CATEGORIES_DATE:
328             case CALORIES_WITH_CATEGORIES_DATE_RANGE:
329             case CALORIES_WITH_CATEGORIES_SUM_DATE:
330             case CALORIES_WITH_CATEGORIES_SUM_DATE_RANGE:
331                 throw new UnsupportedOperationException("Modifying joined results is forbidden.");
332             default:
333                 throw new IllegalArgumentException("Unknown Uri provided.");
334         }
335 
336         mDatabase = mDbHelper.getWritableDatabase();
337 
338         return mDatabase.delete(
339                 table,
340                 selection,
341                 selectionArgs
342         );
343     }
344 
345     @Override
346     public int update(Uri uri, ContentValues values, String selection, String[] selectionArgs) {
347         String table;
348         switch (sUriMatcher.match(uri)) {
349             // The incoming URI is for a single row from categories
350             case CALORIE_CATEGORIES_ID:
351                 table = CALORIES_CATEGORIES_TABLE_NAME;
352                 // Defines selection criteria for the row to delete
353                 selection = CalorieCategories._ID + " = ?";
354                 selectionArgs = new String[]{ uri.getLastPathSegment() };
355                 break;
```

```
            // The incoming URI is for a single row from expenses
            case CALORIES_ID:
                table = CALORIES_TABLE_NAME;
                // Defines selection criteria for the row to delete
                selection = Calories._ID + " = ?";
                selectionArgs = new String[]{ uri.getLastPathSegment() };
                break;
            // The incoming URI is for all of categories
            case CALORIE_CATEGORIES:
                // The incoming URI is for all of expenses
            case CALORIES:
                throw new UnsupportedOperationException("Updating multiple table rows is forbidden.");
            case CALORIES_WITH_CATEGORIES:
            case CALORIES_WITH_CATEGORIES_DATE:
            case CALORIES_WITH_CATEGORIES_DATE_RANGE:
            case CALORIES_WITH_CATEGORIES_SUM_DATE:
            case CALORIES_WITH_CATEGORIES_SUM_DATE_RANGE:
                throw new UnsupportedOperationException("Modifying joined results is forbidden.");
            default:
                throw new IllegalArgumentException("Unknown Uri provided.");
        }

        mDatabase = mDbHelper.getWritableDatabase();

        return mDatabase.update(
                table,
                values,
                selection,
                selectionArgs
        );
    }

    @Override
    public String getType(Uri uri) {
        final int match = sUriMatcher.match(uri);
        switch (match) {
            case CALORIE_CATEGORIES:
                return CalorieCategories.CONTENT_TYPE;
            case CALORIE_CATEGORIES_ID:
                return CalorieCategories.CONTENT_ITEM_TYPE;
            case CALORIES:
                return Calories.CONTENT_TYPE;
            case CALORIES_ID:
                return Calories.CONTENT_ITEM_TYPE;
            case CALORIES_WITH_CATEGORIES:
            case CALORIES_WITH_CATEGORIES_DATE:
            case CALORIES_WITH_CATEGORIES_DATE_RANGE:
            case CALORIES_WITH_CATEGORIES_SUM_DATE:
            case CALORIES_WITH_CATEGORIES_SUM_DATE_RANGE:
                return CaloriesWithCategories.CONTENT_TYPE;
            default:
                return null;
        }
    }
}
```

1.9 Utils

1.9.1 Utils.java

Listing 1.18: Utils.java

```java
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);
    }
}
```

Chapter 2

Resource files

These include several layout, drawable, strings, values, etc files and cannot be all shown here.

Listing 2.1: activity_main.xml

```
1
2  <?xml version="1.0" encoding="utf-8"?>
3  <ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
4      xmlns:app="http://schemas.android.com/apk/res-auto"
5      android:layout_width="match_parent"
6      android:layout_height="match_parent"
7      android:background="@color/colorWhite"
8      android:fitsSystemWindows="true">
9
10     <LinearLayout
11         android:layout_width="match_parent"
12         android:layout_height="wrap_content"
13         android:layout_gravity="center_vertical"
14         android:layout_marginLeft="16dp"
15         android:layout_marginRight="16dp"
16         android:orientation="vertical">
17
18         <ImageView
19             android:id="@+id/imageView"
20             android:layout_width="wrap_content"
21             android:layout_height="wrap_content"
22             android:layout_gravity="center_horizontal"
23             android:layout_margin="16dp"
24             android:src="@drawable/logo" />
25
26         <TextView
27             android:text="Hi there!"
28             android:id="@+id/userNameId"
29             android:layout_width="wrap_content"
30             android:layout_height="wrap_content"
31             android:textSize="25sp"
32             android:layout_gravity="center_horizontal"
33             android:fontFamily="sans-serif-light"
34             android:textColor="@android:color/black"
35             android:background="#ffffff"
36             android:padding="20dp"/>
37
38         <Button
39             android:id="@+id/expense_button"
```

```
            android:layout_width="389dp"
            android:layout_height="wrap_content"
            android:layout_gravity="center_horizontal"
            android:layout_marginLeft="0dp"
            android:layout_marginTop="16dp"
            android:layout_weight="1"
            android:background="@color/colorPrimary"
            android:text="Manage your Expenses"
            android:textColor="@android:color/white" />

        <Button
            android:id="@+id/calorie_button"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:textColor="@android:color/white"
            android:layout_gravity="center_horizontal"
            android:layout_marginTop="16dp"
            android:layout_weight="1"
            android:background="@color/colorPrimary"
            android:text="Check your calories intake" />

        <Button
            android:id="@+id/note_button"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_gravity="center_horizontal"
            android:layout_marginTop="16dp"
            android:layout_weight="1"
            android:background="@color/colorPrimary"
            android:text="Take Notes"
            android:textColor="@android:color/white" />

        <Button
            android:id="@+id/export_button"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_gravity="center_horizontal"
            android:layout_marginTop="16dp"
            android:layout_weight="1"
            android:background="@color/colorPrimary"
            android:text="Export my data"
            android:textColor="@android:color/white" />

        <Button
            android:id="@+id/logout_button"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_gravity="center_horizontal"
            android:layout_marginTop="128dp"
            android:layout_weight="1"
            android:background="@color/colorPrimary"
            android:text="Logout"
            android:textColor="@android:color/white" />


    </LinearLayout>
</ScrollView>
```