```java
package com.example.costmanager;

import android.annotation.SuppressLint;
import android.webkit.WebView;
import android.webkit.WebViewClient;

/**
 * View class contains only Webview component.
 * for displaying our UI, using only web technologies.
 */
public class View {

    private WebView webView;

    /**
     * Constructor for View
     * @param webView instance of webView
     */
    @SuppressLint("SetJavaScriptEnabled")
    public View(WebView webView) {
        this.webView = webView;
        this.webView.setWebViewClient(new WebViewClient());
        this.webView.getSettings().setJavaScriptEnabled(true);
        this.webView.loadUrl("file:///android_asset/index.html");
    }

    /**
     * Getter for retrieving WebView
```

```
28 component
29       * @return WebView Component, which
   represents UserInterface.
30       */
31      public WebView getWebView() {
32          return webView;
33      }
34 }
```

```java
package com.example.costmanager;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.webkit.WebView;

import com.example.costmanager.Model.DatabaseHelper;
import com.example.costmanager.Model.Model;
import com.example.costmanager.ModelView.ViewModel;

/**
 * This is the main entry for our application.
 * our project is made by MVVM architecture
 * ,in this class we create each instance of MVVM
 * @author Maor Tene
 * @author Nuriel Mavashev
 */
public class MainActivity extends AppCompatActivity {

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

```
24
25          //creating the webview model
26          View view = new View(new WebView(
    this));
27          //creating the model
28          Model model = new Model(new
    DatabaseHelper(this));
29          //creating the ViewModel
30          ViewModel viewModel = new
    ViewModel(view, model);
31          //attaching viewmodel object to
    the web view
32          view.getWebView().
    addJavascriptInterface(viewModel,"
    viewModel");
33          //set the activity content
34          setContentView(view.getWebView
    ());
35      }
36  }
37
```

```java
package com.example.costmanager.Model;

import android.database.Cursor;

import org.json.JSONArray;
import org.json.JSONObject;

import java.util.ArrayList;
import java.util.List;

/**
 * Model class helps to handle the logic of our application
 * and communicate with our database and the ViewModel unit
 */
public class Model implements IModel {

    private DatabaseHelper database;

    /**
     * Constructor for Model
     * @param database our database
     */
    public Model(DatabaseHelper database) {
        this.database = database;
    }

    /**
     * This method adds new item to database
     * @param itemName item's name
```

```
30          * @param category item's category
31          * @param cost item's cost
32          * @param date item's bought date
33          */
34         @Override
35         public void addData(String itemName,
   String category, String cost, String date
   ) {
36             database.insertData(itemName,
   category, cost, date);
37         }
38
39         /**
40          * This method gets all items from
   database
41          * @return Returning JSONObject
   obejct which represents our data
42          */
43         @Override
44         public JSONObject getItems() {
45             JSONObject result = cursorToJson(
   database.getItems());
46             return result;
47         }
48
49         /**
50          * This method converts the Cursor
   object to a JSONObject
51          * Because it's more easier in the
   client side to parse it with build-in
   functions
52          * @param cursor Our Data from
   database represented by cursor instance
```

```
53      * @return Our data from database
   represented by JSONObject instance
54      */
55     private JSONObject cursorToJson(
   Cursor cursor) {
56         JSONObject jsonObject = new
   JSONObject();
57         JSONArray resultSet = new
   JSONArray();
58         cursor.moveToFirst();
59         try{
60             jsonObject.put("myitems",
   resultSet);
61         }
62         catch(Exception e)
63         {
64             e.printStackTrace();
65         }
66         /*
67          * scan columns for each row -
   represented by Cursor object
68          * and construct new item -
   represented by JSONObject
69          */
70         while (cursor.isAfterLast() ==
   false) {
71             int totalColumn = cursor.
   getColumnCount();
72             JSONObject rowObject = new
   JSONObject();
73             for (int i = 0; i <
   totalColumn; i++) {
74                 if (cursor.getColumnName(
```

```
74 i) != null) {
75                     try {
76                         rowObject.put(
   cursor.getColumnName(i),
77                                 cursor.
   getString(i));
78                     } catch (Exception e
   ) {
79                     }
80                 }
81             }
82             resultSet.put(rowObject);
83             cursor.moveToNext();
84         }
85         cursor.close();
86         return jsonObject;
87     }
88
89     /**
90      * This method deletes a specifc row
    from database according to id
91      * @param id id to delete a specific
    row from database
92      */
93     @Override
94     public void deleteData(String id) {
95         database.deleteDataDB(id);
96     }
97 }
98
```

```java
package com.example.costmanager.Model;

import org.json.JSONObject;

/**
 * Represents an interface of our database.
 * gives more accessibility and flexibility to the code
 */
public interface IModel {
    /**
     * This method adds new item to database
     * @param itemName item's name
     * @param category item's category
     * @param cost item's cost
     * @param date item's bought date
     */
    void addData(String itemName,String category,String cost,String date);

    /**
     * This method gets all items from database
     * @return Returning JSONObject obejct which represents our data
     */
    public JSONObject getItems();

    /**
     * This method deletes a specifc row from database according to id
```

```
27          * @param id id to delete a specific
    row from database
28          */
29         public void deleteData(String id);
30 }
```

```java
package com.example.costmanager.Model;


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

import androidx.annotation.Nullable;

import java.util.ArrayList;
import java.util.List;

/**
 * DatabaseHelper class for executing SQL requests
 * and managing a local database.
 */
public class DatabaseHelper extends SQLiteOpenHelper {
    private static final String DATABASE_NAME = "Costmanager.db";
    private static final String ITEM_TABLE = "PERSON_TABLE";
    private static final int DATABASE_VERSION = 1;
    private static final String COL_2 = "ITEM_NAME";
    private static final String COL_3 = "ITEM_CATEGORY";
```

```
25      private static final String COL_4 = "
    ITEM_COST";
26      private static final String COL_5 = "
    ITEM_DATE";
27
28      /**
29       * Constructor for database
30       * @param context database's context
31       */
32      public DatabaseHelper(@Nullable
    Context context) {
33          super(context, DATABASE_NAME,
    null, DATABASE_VERSION);
34      }
35
36      /**
37       * This method create a custom table
    inside the database.
38       * @param db database's instance
39       */
40      @Override
41      public void onCreate(SQLiteDatabase
    db) {
42          String createTableStatement = "
    CREATE TABLE " + ITEM_TABLE + " (ID
    INTEGER PRIMARY KEY AUTOINCREMENT, " +
    COL_2 + " TEXT, " + COL_3 + " TEXT, " +
    COL_4 + " TEXT, " + COL_5 + " TEXT)";
43          db.execSQL(createTableStatement);
44      }
45
46      /**
47       * This method is called when version
```

```
47   of my database has changed.
48        * @param db my database.
49        * @param oldVersion version before
   the upgrade.
50        * @param newVersion version after
   the upgrade.
51        */
52       @Override
53       public void onUpgrade(SQLiteDatabase
   db, int oldVersion, int newVersion) {
54           db.execSQL("DROP TABLE IF EXISTS
    "+ITEM_TABLE);
55           onCreate(db);
56       }
57
58       /**
59        * This method insert an item into
   the database.
60        * @param itemName item's name.
61        * @param category item's category.
62        * @param cost item's cost.
63        * @param date item's bought date.
64        */
65       public void insertData(String
   itemName, String category ,String cost ,
   String date){
66           SQLiteDatabase db = this.
   getWritableDatabase();
67           ContentValues cv = new
   ContentValues();
68           cv.put(COL_2, itemName);
69           cv.put(COL_3, category);
70           cv.put(COL_4, cost);
```

```
71          cv.put(COL_5, date);
72          db.insert(ITEM_TABLE, null, cv);
73      }
74
75      /**
76       * This method gets items from the
   table.
77       * @return returning Cursor value
   when calling query.
78       */
79      public Cursor getItems(){
80          SQLiteDatabase db = this.
   getWritableDatabase();
81          Cursor result = db.rawQuery("
   select * from "+ITEM_TABLE, null);
82          return result;
83      }
84
85      /**
86       * This method for deleting an item.
87       * @param id id to delete a specific
    row from database.
88       */
89      public void deleteDataDB(String id){
90          SQLiteDatabase db = this.
   getWritableDatabase();
91          db.delete(ITEM_TABLE, "ID = ?",
   new String[] {id});
92      }
93  }
94
```

```java
package com.example.costmanager.ModelView;


import android.app.Activity;

import com.example.costmanager.Model.Model;
import com.example.costmanager.View;

import org.json.JSONObject;

import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

/**
 * ViewModel class handles the communication
 * between the WebView and the Model
 * It gets requests from the UI
 * and provides informations from the Model
 */
public class ViewModel extends Activity implements IViewModel {
    private View view;
    private Model model;
    private ExecutorService pool;
    private static final int MaxThreads = 5;

    /**
```

```
27          * Constructor for ViewModel
28          * @param view instance of view
29          * @param model instance of model
30          */
31         public ViewModel(View view, Model
    model) {
32             this.view = view;
33             this.model = model;
34             pool = Executors.
    newFixedThreadPool(MaxThreads);
35         }
36
37
38         /**
39          * This method sends to the client
    all items
40          * from the database for View Report
    page
41          */
42         @android.webkit.JavascriptInterface
43         @Override
44         public void getReport() {
45             pool.submit(new Runnable() {
46                 public void run() {
47                     final JSONObject result
     = model.getItems();
48                     runOnUiThread(new
    Runnable() {
49                         @Override
50                         public void run() {
51                             view.getWebView
    ().evaluateJavascript("displayReport('"+
    result.toString()+"')", null);
```

```
52                      }
53                  });
54              }
55          });
56      }
57
58      /**
59       * This method adds new item to the database.
60       * @param itemName item's name
61       * @param category item's category
62       * @param cost item's cost
63       * @param date item's bought date
64       */
65      @android.webkit.JavascriptInterface
66      @Override
67      public void addItem(final String itemName,final String category,final String cost,final String date) {
68          pool.submit(new Runnable() {
69              public void run() {
70                  model.addData(itemName, category, cost, date);
71                  final JSONObject result = model.getItems();
72                  runOnUiThread(new Runnable() {
73                      @Override
74                      public void run() {
75                          view.getWebView().evaluateJavascript("displayItemsList('" + result.toString() + "')", null);
76                      }
```

```
77                  });
78              }
79          });
80      }
81
82      /**
83       * This method sends to the client
    all items
84       * from the database for Add Item
    page
85       */
86      @android.webkit.JavascriptInterface
87      @Override
88      public void showItems() {
89          final JSONObject result = model.
    getItems();
90          runOnUiThread(new Runnable() {
91              @Override
92              public void run() {
93                  view.getWebView().
    evaluateJavascript("displayItemsList('"+
    result.toString()+"')", null);
94              }
95          });
96      }
97
98      /**
99       * This method delete a item from
    the database
100      * @param id id to delete a specific
     row from database
101      */
102     @android.webkit.JavascriptInterface
```

```
103     @Override
104     public void deleteItemVM(final
    String id) {
105         pool.submit(new Runnable() {
106             public void run() {
107                 model.deleteData(id);
108                 final JSONObject result
     = model.getItems();
109                 runOnUiThread(new
    Runnable() {
110                     @Override
111                     public void run() {
112                         view.getWebView
    ().evaluateJavascript("displayItemsList
    ('"+result.toString()+"')", null);
113                     }
114                 });
115             }
116         });
117     }
118 }
119
```

```java
package com.example.costmanager.ModelView;

/**
 * Represents an interface of our model.
 * gives more accessibility and flexibility to the code
 */
public interface IViewModel {
    /**
     * This method sends to the client all items
     * from the database for View Report page
     */
    public void getReport();
    /**
     * This method adds new item to the database.
     * @param itemName item's name
     * @param category item's category
     * @param cost item's cost
     * @param date item's bought date
     */
    public void addItem(final String itemName,final String category,final String cost,final String date);
    /**
     * This method sends to the client all items
     * from the database for Add Item page
     */
```

```
25      void showItems();
26      /**
27       * This method delete a item from the
    database
28       * @param id id to delete a specific
   row from database
29       */
30      void deleteItemVM(final String id);
31  }
32
```