Room Space Planner

Android Application

- **A manifest for the projects, all sub-modules and libraries:**
  - Repository resides at: https://github.com/dopeamine/RoomSpaceSaver
  - No external libraries or sub-modules have been used in the project. All functionality implemented has used standard Android and Java API.
  - ***The file "AndroidManifest.xml" located in ./app/src/main. This manifest file provides essential information (application specific information) about your app to the Android system, which the system must have before it can run any of the app's code.***
  - ***The following classes have been implemented: (./app/src/main/java/com/example/shant/roomspacesaver)***
    1. User.java
    2. Room.java
    3. Furniture.java
    4. Settings.java
    5. MainActivity.java
    6. RoomsActivity.java
    7. AddRoomDialogFragment.java
    8. RoomsListCursorAdapter.java
    9. EditRoomActivity.java
    10. AddFurnitureDialogFragment.java
    11. RectArea.java
    12. RectsDrawingView.java
    13. DBHelper.java
  - ***The following display layouts (XML files) have been implemented: (./app/src/main/res/layout)***
    1. activity_main.xml
    2. activity_rooms.xml
    3. dialog_add_room.xml
    4. room_list_item.xml
    5. activity_edit_room.xml
    6. dialog_add_furniture.xml
  - Some of the important standard Android API classes have been used to implement the application:
    - Java framework collections: Contains the collections framework, legacy collection classes, event model, date and time facilities, internationalization, and miscellaneous utility classes (a string tokenizer, a random-number generator, and a bit array).

    1. java.util.ArrayList;
    2. java.util.Arrays;
    3. java.util.HashSet;
    4. java.util.Random;

- View: Provides classes that expose basic user interface classes that handle screen layout and interaction with the user.
    1. android.view.LayoutInflater;
    2. android.view.View;
    3. android.view.ViewGroup;
    4. android.view.MotionEvent;

- Content management: Contains classes for accessing and publishing data on a device
    1. android.content.DialogInterface;
    2. android.content.ContentValues;
    3. android.content.Context;
    4. android.content.Intent;

- Graphics library: Provides low level graphics tools such as canvases, color filters, points, and rectangles that let you handle drawing to the screen directly.
    1. android.graphics.Bitmap;
    2. android.graphics.BitmapFactory;
    3. android.graphics.Canvas;
    4. android.graphics.Color;
    5. android.graphics.Paint;
    6. android.graphics.Rect;

- Widgets (Display elements):The widget package contains (mostly visual) UI elements to use on your Application screen. You can also design your own.
    1. android.widget.EditText;
    2. android.widget.Toast;
    3. android.widget.Button;
    4. android.widget.RelativeLayout;
    5. android.widget.LinearLayout;
    6. android.widget.AdapterView;
    7. android.widget.ArrayAdapter;
    8. android.widget.CursorAdapter;
    9. android.widget.ListView;
    10. android.widget.TextView;

- Utilities: Provides common utility methods such as date/time manipulation, base64 encoders and decoders, string and number conversion methods, and XML utilities.
    1. android.util.Log;
    2. android.util.AttributeSet;
    3. android.util.DisplayMetrics;
    4. android.util.SparseArray;

- Database: Contains classes to explore data returned through a content provider. All databases are stored on the device in /data/data/<package_name>/databases
    1. android.database.Cursor;
    2. android.database.sqlite.SQLiteDatabase;
    3. android.database.sqlite.SQLiteOpenHelper;
    4. android.database.sqlite.*;

- Other extras:
    1. java.lang.reflect.Array;
    2. android.app.Dialog;
    3. android.app.DialogFragment;
    4. android.support.v7.app.AlertDialog;
    5. android.support.v7.app.AppCompatActivity;
    6. android.os.Bundle;
    7. android.text.InputType;
    8. android.icu.text.StringPrepParseException;
    9. android.text.LoginFilter;

- **Descriptions of the projects, all sub-modules and libraries:**
    - ***The file "AndroidManifest.xml" located in ./app/src/main. This manifest file provides essential information (application specific information) about your app to the Android system, which the system must have before it can run any of the app's code.***
    - ***The following classes have been implemented: (./app/src/main/java/com/example/shant/roomspacesaver)***
        1. User.java, Room.java, Furniture.java, Settings.java: Application specific classes. Define blueprints of classes and their member variables.
        2. MainActivity.java: The Login user interface presented when the application starts. This activity loads the layout from **activity_main.xml** file. User can Login or signup here.
        3. RoomsActivity.java: After a user succesfully logs in, he is presented with this user interface. If the user is logging in the first time, he will only see a 'Add Room' button. He can choose to add a new room or edit existing rooms

presented in the list. The layout for this UI is loaded from the **activity_rooms.xml** file.

4. AddRoomDialogFragment.java: This class inherits from DialogFragment class and implements a custom dialog box for getting room details when creating a new room. The custom layout is loaded from the file **dialog_add_room.xml**
5. RoomsListCursorAdapter.java: This implements a custom cursor adapter that acts as bridge between the list of rooms view and the database query result of the user's rooms.
6. EditRoomActivity.java: When a user clicks on a particular room from the list of rooms with an intent to edit the virtual room. This load the UI from the file **activity_edit_room.xml**. If the user is editing the room for the first time, he will only see a 'Add Furniture' button and an empty virtual room. He can choose to add a new furniture or edit room layout if furnitures have already been added. The layout for this UI is loaded from the **activity_rooms.xml** file. If furnitures are already present the virtual room is populated with colored rectangles (representing furniture) from the database.
7. AddFurnitureDialogFragment.java: This class inherits from DialogFragment class and implements a custom dialog box for getting furniture details when adding a new piece of furniture. The custom layout is loaded from the file **dialog_add_furniture.xml**
8. RectArea.java: Defines the blueprint of the rectangular are to be drawn on the virtual room representation.
9. RectsDrawingView.java: Custom view responsible for drawing the rectangles on the screen.
10. DBHelper.java: This class extends the SQLiteOpenHelper class and implements methods to create a database name "appData.db" and tables (users,rooms,furnitures). It also implements methods required for functionality such as login, registration, adding rooms, adding furniture, etc.

- **Instructions to install, configure, and run the programs:**
  1. **Requirements:**
     a. Android Studio
     b. Android SDK with API 26 (i.e Marshmallow) installed
     c. Configured Emulator device or Android device
     d. Sqlite browser
  2. **Installation instruction:**
     a. Extract the project zip file
     b. Open Android Studio
     c. File -> Open Project
     d. Browse to extracted directory and open the project
  3. **Build instructions:**
     a. To build the application goto Run -> Run 'app'

   b. You will be asked to select the device to build the 'app' to, select the intended device.
   c. If all goes well, you will be presented with the Login screen.
   d. If you are a new user, click the 'Signup' link to register as a new user.
   e. Once registered you can login with your credentials and use the application.
4. **Debug instructions:**
   a. In order to check if values are getting read from/written to the database:
      i. You can use the dbpull.bat file to pull the sqlite database file named 'appData.db' from the device.
      ii. This file can then be open in sqlite browser and the database can be checked.
      iii. You can also enter new records from the sqlite browser, write changes to the database file and use the dbpush.bat script to push the updated database file to the device.

- **Known bugs:**
  1. Multiple users with the same username can be registered (not implemented code to check if username exists)
  2. On room creation the Rooms List is not automatically updated. Need to restart the application to see the changes
  3. On opening room to edit the layout, existing furnitures are displayed at respective positions, but if you press back and open the room again, furnitures are redrawn. (i.e if there are 2 furnitures you can now see 4 furniture pieces)
  4. Existing furnitures drawn sometimes have wrong dimensions.
  5. Avoiding furnitures from overlapping or moving them outside the walls is not perfectly implemented yet.

- **Program files/Scripts:**

  Two small scripts have been used to observe the database from the device:
  - **Dbpull.bat** - This file is used to pull the database file 'appData.db' from the device
  - **Dbpush.bat** - This file is used to push the database file 'appData.db' to the device
  - **NOTE**:
    - These scripts work only for "the only running emulator". So make sure you have only one emulator running. These scripts will not work for physical devices.

- **Program Executables, if any:**
  1. For convenience, a compiled .apk file is provided in the root directory of the project.
  2. This apk file can directly be copied to the phone and installed.
