**.\121`**

ITMD 455/555 *Intelligent Device Applications* Lab 7

#### SQLite Books Database (part 2)- 50 points

**Introduction**. This lab will have you continue your work from lab 6. This time around you’ll present the Books database to the user. You will also be adding in a column in code to your database as well that will allow for book ratings to be possible. Also included in this lab is the use of a FF plug-in called **SQLite Manager** to allow for updates to your table as well as views of your table, etc. Finally a custom adapter will be created in your code to allow for a listview and objects such as a rating bar to display data to the user in the form of text and graphics.

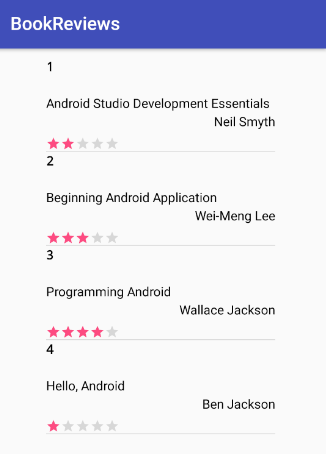
**Objective:** For this lab you will continue working with the books database with some slight modifications to the database involving adding in a rating column to the table. You will populate the new

column entry manually in SQLite Manager which is included as a plug-in thru FireFox. Then you will show off your database via a custom list adapter, which will allow you to include both

text and some graphics to the screen.

Activities of your app will be further looked at in LogCat. An initial sample output of your app

is shown below:



**STEP 1 Open up your last project (BookReviews) you created for lab 6**

Open up your Book.java file and include the *following* additions:

Declare another class variable called rating as follows:

**private** String rating;

Add in another getter/setter your class as follows:

**public** **void** setRating(String rating) { **this**.rating = rating; }

**public** String getRating() { **return** rating; }

Adjust your toString()method at the bottom to include your rating to show up in your

LogCat file as follows:

@Override

**public** String toString() {

**return** "Book [id=" + id + ", title=" + title + ", author=" + author

+ **", rating=" + rating** + "]";

}

**STEP 2 Setup layouts for a custom view effect.**

Create an xml file called itemlistrow and enter the following code:

<?xml version=*"1.0"* encoding=*"utf-8"*?>

<LinearLayout xmlns:android=*"http://schemas.android.com/apk/res/android"*

android:layout\_width=*"fill\_parent"*

android:layout\_height=*"fill\_parent"*

android:orientation=*"vertical"* >

<TextView android:textColor=*"#000"*

android:id=*"@+id/\_id"*

android:layout\_width=*"fill\_parent"*

android:layout\_height=*"wrap\_content"*

android:text=*"id"* android:textStyle=*"bold"*

android:gravity=*"left"*

android:layout\_weight=*"1"*

android:typeface=*"monospace"*

android:height=*"40sp"* />

<TextView android:textColor=*"#000"*

android:id=*"@+id/title"*

android:layout\_width=*"fill\_parent"*

android:layout\_height=*"wrap\_content"*

android:text=*"title"*

android:layout\_weight=*"1"*

android:height=*"20sp"* />

<TextView android:layout\_height=*"wrap\_content"*

android:layout\_width=*"fill\_parent"*

android:layout\_weight=*"1"*

android:textColor=*"#000"*

android:gravity=*"right"*

android:id=*"@+id/author"*

android:text=*"author"*

android:height=*"20sp"* />

<RatingBar

android:id=*"@+id/rating"*

style=*"?android:attr/ratingBarStyleSmall"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:paddingTop=*"6dip"*

android:stepSize=*"0.25"*

android:numStars=*"5"*

/>

</LinearLayout>

Modify your activity\_main xml file to only include the following markup:

<LinearLayout xmlns:android=*"http://schemas.android.com/apk/res/android"*

xmlns:tools=*"http://schemas.android.com/tools"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"match\_parent"*

android:baselineAligned=*"false"*

android:orientation=*"vertical"* >

<ListView

android:id=*"@+id/list"*

android:layout\_width=*"250dp"*

android:layout\_marginTop=*"10dp"*

android:layout\_marginBottom=*"10dp"*

android:layout\_height=*"fill\_parent"*

android:layout\_gravity=*"center"* >

</ListView>

</LinearLayout>

These layouts will help you provide data to the screen in the form of a listview plus accompanying textviews and a rating bar, which will help serve up your record data to the screen.

**STEP 3 Create a class in your package called ListAdapter.**

This class will allow for a custom view to be set up to bind text data and a rating bar to the screen via two layouts and a ListAdapter.

Enter code for your new adapter class as follows:

**public** **class** ListAdapter **extends** ArrayAdapter<Book> {

**private** List<Book> items;

**public** ListAdapter(Context context, **int** textViewResourceId) {

**super**(context, textViewResourceId);

}

**public** ListAdapter(Context context, **int** resource, List<Book> items) {

**super**(context, resource, items);

**this**.items = items;

}

@Override

**public** View getView(**int** position, View convertView, ViewGroup parent) {

View v = convertView;

**if** (v == **null**) {

LayoutInflater vi;

vi = LayoutInflater.*from*(getContext());

v = vi.inflate(R.layout.*itemlistrow*, **null**);

}

Book p = getItem(position);

**if** (p != **null**) {

TextView tt = (TextView) v.findViewById(R.id.*\_id*);

TextView tt1 = (TextView) v.findViewById(R.id.*title*);

TextView tt3 = (TextView) v.findViewById(R.id.*author*);

RatingBar rb = (RatingBar) v.findViewById(R.id.*rating*);

**if** (tt != **null**) {

tt.setText("" + p.getId());

}

**if** (tt1 != **null**) {

tt1.setText(p.getTitle());

}

**if** (tt3 != **null**) {

tt3.setText(p.getAuthor());

}

**if** (rb != **null**) {

**float** rating = Float.*parseFloat*(p.getRating());

rb.setRating(rating);

}

}

**return** v;

}

}

Enter imports for your new adapter class as follows:

**import** java.util.List;

**import** android.content.Context;

**import** android.view.LayoutInflater;

**import** android.view.View;

**import** android.view.ViewGroup;

**import** android.widget.ArrayAdapter;

**import** android.widget.RatingBar;

**import** android.widget.TextView;

What this code roughly does is retrieve the **ids** defined in your itemlistrow layout and allows for textviews and a rating bar to take on record values displayable in a custom view till there are no more records to be read via your buddy, the ‘Book’ class.

**STEP 4 Modify your SqlHelper class to include additional rating column information**

In yourgetAllBooks() function add in the following inside your **do** loop to set the ratings

values

book.setRating(cursor.getString(3));

In youronUpgrade(...) function add in the following query statement at

the bottom of your method after your **this**.onCreate(db); line, to allow for an additional

column to be added into your database as follows:

String upgradeQuery = "ALTER TABLE books ADD COLUMN rating TEXT";

**if** (oldVersion == 1 && newVersion == 2)

db.execSQL(upgradeQuery);

**Be *weary* at this point as only when there is a *change* in the database version will the onUpgrade method fire and therefore perform the** ALTER TABLE **command. NOTICE something else VERY, VERY important, the database versions in the if statement above namely**

**if** (oldVersion == 1 && newVersion == 2)

**Note you would have to adjust these version values in your condition statement when it comes time to testing your app where you will have to include the versions correctly based on what you have in your code at any point in time, reflecting your CURRENT (oldVersion) number and your NEW (newVersion) number when it’s time to update your dbase when you run your app next!**

More on this logic to come.

Note we *could* have just added in a rating column to the table build, drop the existing table then add in records from scratch here, but then that would be no fun, so tighten your seat belts a bit as you will see some workarounds are needed to accomplish this simple task of altering the table for now and not rebuilding things, but everything comes with a price as you will see.

Further note that alter commands are limited in SQLite to just really altering the column as demonstrated above. If you were to delete a column then you would really need to rebuild table design & logic from scratch as mentioned which in the real world could be even more of a pain then the workaround suggestions that will be shown later, so everything is a tradeoff, you just have to weigh the odds and know your options!!!

**STEP 5 Open up your MainActivity.java file and add in the following:**

In your class code after the line

db.getAllBooks();

include the following code (and make sure to add in needed imports):

ListView listContent = (ListView) findViewById(R.id.*list*);

list = **new** ArrayList<Book>();

list=db.getAllBooks();

//get data from the table by the ListAdapter

ListAdapter customAdapter = **new** ListAdapter(**this**, R.layout.*itemlistrow*,list);

listContent.setAdapter(customAdapter);

I would also comment out the db.deleteBook(list.get(0)); line as you have to allow for all books to be displayed and resident in the database **Books** table.

That’s basically it for now. Here you added in code to call the ListAdapter constructor you created which will help inflate the custom view you defined consisting of the itemlistrow layout. By default your activity\_main layout inflates at start up which defined your listview and thus the custom layout intertwines with that layout creating a view override which shows itemlistrow objects to be displayed in a row by row fashion within the listview. How cool is that?

**STEP 6 Run your code, …twice!**

Now for some tweaks. You need to run your code twice (in a moment), to allow for your updated query (the ALTER TABLE command) to execute and another run for the actual display to trigger. You see as you are adding a column, namely rating, to your table, when your app runs it won’t see anything but null values for each record concerning your rating column. That *does* present a two-fold problem for emulator displaying purposes, as well as an app that will crash on your initial run, as you will see by checking over your Logcat output.

Okay- unhook any commented code if you have any, in MainActivity that blocks out your **addBook()** statements as you will be rebuilding from scratch, your dbase for this exercise.

Next go to your **SQLHelper** file and change your **current** database version to the next number. For example if you have your **private** **static** **final** **int** *DATABASE\_VERSION* = 8; you would change it to 9. Likewise in your **if** (oldVersion == 1 && newVersion == 2) line in your onUpgrade(...) method, change the values to correctly reflect your old and new versions. Example (oldVersion == 8 && newVersion == 9)

Okay, run your app. It will crash. But that’s okay. Notice your LogCat file. It should show 4 records were created with the rating field reflecting null values as shown on a partial LogCat snapshot below:

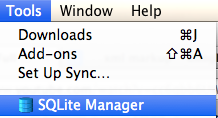


Now to rectify this we will now add in values via the SQLite Manager to the rating column so nulls will not exist the next time you run your app. Of course you could of added in rating values

in your add method in MainActivity or ran update queries to add in values to your rating column in code but that would be no fun and it wouldn’t give you a chance to work the SQLite Manager!

**STEP 7 Download SQLite Manager.**

Go to your FireFox and google SQLite Manager plug-in for FireFox. After you install your plug in you can open it in FireFox from your Tools menu (Windows users, press your alt key if you don’t see a menu) as shown below.



Go ahead and open it. Keep the window open and now your will execute some commands from the command line interface to pull your database from your package, update it in SQLite Manager then import the updates it back into your package.

The basic process for all OSes will be to yank the database file (BookDB) from your package using the **adb** interactive shell, to your sdcard where you can gain access to the file, edit it. then push the changes back into your package.

Mac people

Open up your terminal app. Find where you installed android, i.e., under the folder android-sdks. You can always for sure check the SDK Manager from the AS menu, your Finder program or use the find command from the command line and type

iMacs-iMac:~ imac$ **find ~ -iname android-sdks**

My folder was found directly in my root as a user. Yours could be within your Library folder.

Last login: Sun Apr 2 08:07:40 on console

iMacs-iMac:~ imac$ find ~ -iname android-sdks

**/Users/imac/android-sdks**

Change directory to android-sdks/platform-tools at your prompt where you have the sdk loaded

iMacs-iMac:~ imac$ **cd android-sdks/platform-tools**

Evoke your adb shell with the command

iMacs-iMac:platform-tools imac$ **./adb shell**

You should now be at the shell prompt. Issue these commands to pull from your package. Make sure you are using YOUR PACKAGE NAME when referencing it below. Mypackage name for example is com.example.mypackage.bookreviews. Enter commands (shown in **bold**) one line at a time and press enter every time a command finishes.

generic\_x86:/ $ **run-as com.example.mypackage.bookreviews**

generic\_x86:/data/data/com.example.mypackage.bookreviews $ **cp databases/BookDB /sdcard/**

**Make sure to include a space in your copy command to include the destination location**

Continue with the following commands:

generic\_x86:/data/data/com.example.mypackage.bookreviews $ **exit**

generic\_x86:/ $ **exit**

iMacs-iMac:platform-tools imac$ .**/adb pull /sdcard/BookDB .**

/sdcard/BookDB: 1 file pulled. 15.0 MB/s (20480 bytes in 0.001s)

Make sure to include the period or dot ( . ) at the end of your last command to copy the file in.

Your file should now be resident in your platform-tools folder ready to edit! Type in **ls –l** at your prompt to see if the file was pulled in and note the byte size as well! Example follows:

Macs-iMac:platform-tools imac$ **ls -l**

total 10976

**-rw-r--r-- 1 imac staff 20480 Apr 2 10:33 BookDB**

-rw-r--r-- 1 imac staff 253025 Mar 18 23:31 NOTICE.txt

-rwxr-xr-x 1 imac staff 2326236 Mar 18 23:31 adb

Windows people

Open up your command prompt. Find where you installed android sdk by going to SDK Manager which will show the exact path. Most likely it is in your users folder under AppData

Local\Android\Sdk path. You can for sure always check the SDK Manager from the AS menu, or Windows Explorer to try out a search for the sdk folder.

Once you know your path, from your command prompt window type in the following commands depicted in red to evoke adb:

C:\Users\window>cd \users\window\AppData\Local\Android\sdk\platform-tools

You may be in whatever default path in DOS but the cd (change directory) command to your desired path should work.

From your new path execute the following commands shown in red one line at a time and press enter each time to commit your command:

C:\Users\window\AppData\Local\Android\sdk\platform-tools>adb shell

This should bring you into adb’s interactive shell as shown below.

generic\_x86:/ $

In your shell execute the following:  
generic\_x86:/ $ run-as com.example.mypackage.bookreviews

generic\_x86:/data/data/com.example.mypackage.bookreviews $ cp databases/BookDB /sdcard/

**Make sure to include a space in your copy command to include the destination location**

Continue with the following commands:

generic\_x86:/data/data/com.example.mypackage.bookreviews $ exit

generic\_x86:/ $ exit

iMacs-iMac:platform-tools imac$ adb pull /sdcard/BookDB .

/sdcard/BookDB: 1 file pulled. 4.3 MB/s (20480 bytes in 0.005s)

Make sure to include the period or dot ( . ) at the end of your last command to copy the file in.

Your file should now be resident in your platform-tools folder ready to edit! Type in dir at your prompt to see if the file was pulled in and note the byte size as well! Example follows:

C:\Users\window\AppData\Local\Android\sdk\platform-tools>dir

Volume in drive C has no label.

Volume Serial Number is DC02-1CBC

Directory of C:\Users\window\AppData\Local\Android\sdk\platform-tools

04/02/2017 09:05 PM <DIR> .

04/02/2017 09:05 PM <DIR> ..

04/02/2017 06:53 PM 1,543,168 adb.exe

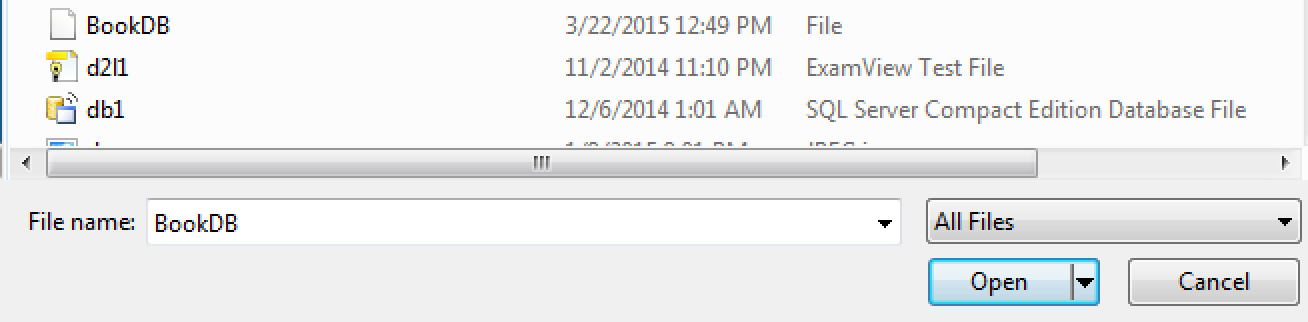
04/02/2017 06:53 PM 97,792 AdbWinApi.dll

04/02/2017 06:53 PM 62,976 AdbWinUsbApi.dll

04/02/2017 06:53 PM <DIR> api

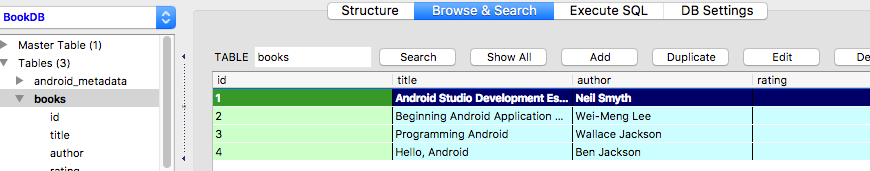
04/02/2017 09:05 PM 20,480 BookDB

Next in SQLite Manager, open up the **BookDB** file from your platform-tools folder (choose All files for visibility of the database file) as shown below.

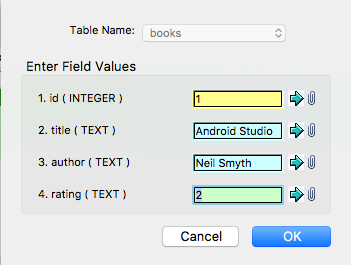


Double click on the **books** table visible under **Tables** on the left hand column of the SQLite Manager (snapshot follows), click on the **Browse & Search** tab under the visible tabs at the top center part of your window and edit your rating column to include the following values listed in the paragraph that follows.

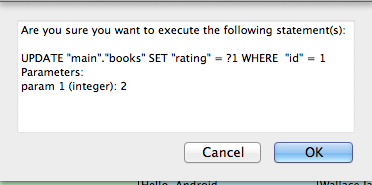
**Snapshot of SQLite Manager follows with the books table opened up:**



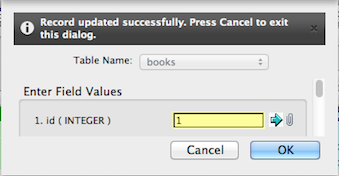
Notice your rating column have null values, so time to edit them. Double click on the first rating column for record 1 to edit it. The following pop up window appears (shown next) allowing you to edit the record. Add in the value of **2** for the rating field. Press OK when complete.



After you click OK a confirmation window pops up. Press OK again to commit.



Now it tells you to press Cancel to exit the dialog box upon a successful update (shown next). Press Cancel at this time.



Do the same routine for the next three records. For record 2 add a column rating of **3**, for record 3 a rating value of **4** and for record 4 a rating of **1**.

Take a snapshot of your SQLite Manager to show your completed table entries.

Okay your database table has been automatically updated in the spot where you saved it to locally. Close out of SQLite Manager.

Now you need to take your *updated* database file and place it back into your package. Commands will be like sort of a role reversal.

Mac superstars:

From your command prompt in your platform-tools directory again perform the following commands (press enter for every line entered):

iMacs-iMac:platform-tools imac$ **./adb push BookDB /sdcard/**

iMacs-iMac:platform-tools imac$ **./adb shell**

generic\_x86:/ $ **run-as com.example.mypackage.bookreviews**

generic\_x86:/data/data/com.example.mypackage.bookreviews $ **cp /sdcard/BookDB databases/**

To exit your adb shell type in at your command prompt **exit** for two times to exit back into your local folder and your good to go!

Windows hackers:

From your command prompt in your platform-tools directory again perform the following commands (press enter for every line entered):

C:\Users\window\AppData\Local\Android\sdk\platform-tools>adb push BookDB /sdcard/

C:\Users\window\AppData\Local\Android\sdk\platform-tools>adb shell

generic\_x86:/ $ run-as com.example.mypackage.bookreviews

generic\_x86:/data/data/com.example.mypackage.bookreviews $ cp /sdcard/BookDB databases/

To exit your adb shell type in at your command prompt exit for two times to exit back into your local folder and your good to go!

**STEP 8 Running your code again.**

Now before running your app again one tweak is needed. Go back to your MainActivity file and comment *out* your **addBook()** statements as you don’t need to add in any more records on subsequent runs. Now rerun your app and you should now see a display on your emulator screen and in your Logcat display with all the database records include title, author, id and a rating (now with no nulls) for each record! Notice no more warnings in your Logcat file! Sweet!

Take a snapshot to show your completed interface showing the listview in your emulator (should look something similar to opening snapshot on page 1). Also snapshot your logcat displaying some of your ratings now filled up for your records. Make sure your name is included in logcat in your snapshot

Example display follows…



****

**STEP 9 For the Grads! Take a break at this point, (I know I need one) if you haven’t to finish modifying your app with images!**

Include an image of each textbook listed in your database (grab a reasonably sized image (could be a facsimile of the text for a particular record) by googling its title) and add it to your display in place of the id that shows in the listview. Include edit points where applicable. Include images in an appropriate res/drawable folder and adjust your applicable layout xml file to reflect an imageview as well as adjust your code in any appropriate java files you deem necessary (namely **ListAdapter.java** for one) to allow for images to be fed into your display. If you feel you have to add in a column to your database table to store the image title for each record that may not be a bad idea, that way when you cycle thru your custom view to build an appropriate view. It may then seem logical to point to each database table record

to fetch the image name and thus have it rendered appropriately into your custom view.

**Example :** p.getImageName() //get image name from table column

Coding logic furthers here as an example:

ImageView icon=(ImageView)row.findViewById(R.id.icon);

**if** (p.getImageName().equals(“androidStudioimg”){

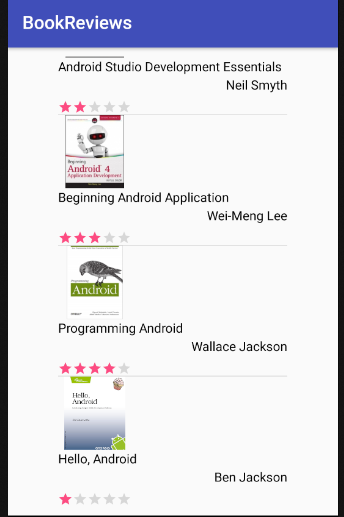
icon.setImageResource(R.drawable.androidStudioimg);

}

//etc...

Take a snapshot to show your completed interface showing the listview now including your textbook images in your emulator.

Sample end result:



**All: Include all your program files in Word as well as all your snapshots labeled appropriately. Also zip your package files and send your results into BB.**

Best of luck. Lab 8, the *finale* lab ( a shorter lab ☺) will follow to end the final part of

SQLite and Android! For that lab you will use Android’s spinner “dropdown” control to allow for

easy searches into your BookDB database file! ::Very cool!::