The purpose of this lab is to get familiar with using databases with mobile applications.

You will need the code distributed /discussed in the lecture session in order to do this lab. A copy of this is posted in webcourses under the lab sheet.

|  |  |
| --- | --- |
|  | You will also definitely need to use the error log (called **LogCat)** in Android Studio to diagnose errors for this lab as it is difficult to get your dB programme running perfectly first time.  When your app “stops” when you run it, you have likely hit a run time error. Fatal run time errors in LogCat are show in Red , with a “caused by” statement. Just scroll from the bottom of the log upwards to find the error. |

The purpose of this lab is to get an activity running that creates and posts data into a database called “TaskList”. The database is specified as follows:

One table called **Tasks** containing 4 columns:

* 1. RowID (autoincremented, primary key). Standard in Android is to use \_id as the field name for the primary key.
  2. Task name
  3. Task Description
  4. Complete status – 1 or 0. (SQLite does not have a separate Boolean storage class. Instead, Boolean values are stored as integers 0 (false) and 1 (true).

The set up steps and detailed specification are outlined:

1. **Set up a project in Android Studio** as usual.
2. **Set up a class that handles database operations:** The code sample from class showed an example of this - using the Helper class to create a database, and having the various calls to the database (inserts, updates etc) all in one place. Create a new class based on this example for the TaskList database. Note: This is NOT an activity class.

As part of your code, add methods to the TaskList database class to:

* + - Open or “Get” the database (this is already in the sample database class, but note it – as you will need to call this from your activity )
    - INSERT a row on to the table
    - Select all rows on the table
    - Closes the database

1. **Get the dB created and add data to it:** The class you’ve just coded write in (2) can’t DO anything until you use it from elsewhere. Create an activity class (you’ll have probably already generated on when you created the project) that instantiates your “database “ class from above- and then calls the relevant methods to open/“get” the database, add a row, and close the database . Run the code. To add rows, you can
   1. For simplicity, hardcode the row(s) to be inserted for now OR
   2. You could use proper data entry fields (<EditText>) for the user to enter data into and add rows.
2. **Download the dB your created:** Once you’re done (2), you should have an SQLLite database created, with at least one row one it (if you called the insert properly). To actually look inside your SQLLite database, you can download it from Android Studio If you are using the Android Virtual Device. If you are using your own individual phone[[1]](#footnote-1) or other third party emulators , google the instructions for your particular set up.

**Android AVDs:** locating the dB file:

* 1. Click on Device File Explorer on the bottom right hand side of android studio
  2. Go to “Data” directory/ then again subdirectory “data”.
  3. Find the package your app/code is stored in (i.e. whatever your package declaration is in your code)
  4. Click in to databases
  5. If you dB exists – it’s there as a file called “TaskList” if you named it that.
  6. Click and “save as” to your hard drive.

1. **View the DB:** As discussed in class, you need an SQLlite browser to view the dB file.
   1. There are plenty of free SQLlite browser e.g. <http://sqlitebrowser.org/> - about 6M. Download the browser .. very lightweight … and open your SQLLite database that you exported.
   2. Make sure the row(s) you added are there correctly

----------------------------------------------------------------------------------------------------------  
  
**OPTIONAL - if you have time:**

1. **Add more** : - If you have all of that working – add more rows to your database – and add code to your activity /dBs that retrieves one or more rows and displays them back on the screen. Remember that queries return a Cursor object.

1. (Note: If you run into problems trying to the database from your physical phone, you can also test if the database creation and insert has worked by putting code into your activity that retrieves all the rows and checks the number of items in the Cursor object returned. The method name in the Cursor class is getCount().  
    [↑](#footnote-ref-1)