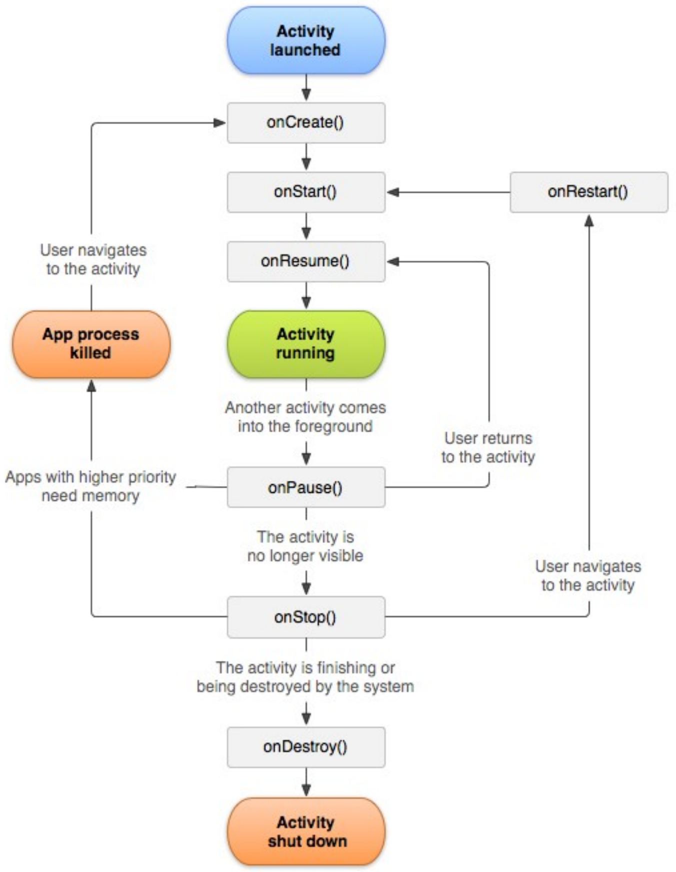
Assignment 1: Write an Android app that collects accelerometer data (while the application is in the foreground) into a buffer and then calculate the energy value after a set interval, writing it to a local SQLite database. The app should have buttons to display the most recent 60 entries and to send the local data to the remote database.

Part 0: Install Android Studio, open the project files, and make sure you understand the provided code for the given files. Files of interest to you are listed below.

App/src/main/res/layout/main_activity.xml – This is the design layout for the user interface. Note that buttons may have an onClick defined here. onClicks can also be set programmatically via the Java side. You will not need to modify this file, but it can be a fun exercise if you are interested in layouts.

App/src/main/java/MainActivity.java – This is the class that defines what the activity can do. In the Activity Control Methods section, make sure you understand what is being done and why. If you are not yet familiar with the life cycle of an activity, the image below may help. Note that you will have to implement toggleAccelClicked(). You should also make note of what happens in the other methods, as you will have to implement some of the functions that these methods call.


Activity launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

App/src/main/java/DataAccumulator.java – This class can receive new accelerometer values and stores them into volatile memory in the form of a linked list. When it notices that TIME_INTERVAL has passed, it will calculate the energy of all of the accelerometer values and return the single energy. Note that you will have to implement the energy calculation.

App/src/main/java/EnergyReading.java – This class represents an energy value at a time. The class creates a DATETIME string for later storage in a SQL database. Note that this will not need to be modified and an instance of this class should be returned from what you implement in the DataAccumulator.

App/src/main/java/SQLInterface.java – This class provides nice methods for classes not involved with the SQL database to easily make calls to modify or read the database. Note you will not need to modify this file.

App/src/main/java/EnergyDBHelper.java – This class implements all of the direct queries to the database. Note that the database structure and creation of the database are implemented for you and you should not need to change anything under the header DATABASE STRUCTURE. You should, however, use the constants defined in the inner class EnergyEntry. You will need to write all of the methods under the header INTERRACTION METHODS. Pay attention to the docstrings as there are hints in there.

App/src/main/java/Replicator.java – This class sends chunks of the local database to the back end for permanent storage and analysis. Note you will be asked to implement the actual post request to your own cgi script on Murphy. Also note that it is already implemented in a way that multiple requests to replicate will be ignored until the first request is completed.

SQLGateway.py – This is a cgi script that receives the POST data from the replicator and stores it in the MySQL database on Murphy. When SQLGateway receives a GET request, it will display the most recent 60 entries in the database. Use a web browser to check that your data is being received. If you send data and your web page does not change upon refresh, try using curl in the format below:

```
curl -X POST http://murphy.wot.eecs.northwestern.edu/~<netID>/SQLGateway.py --data
“param1=123”
```

The response from the curl should have info about any errors you may run into on the gateway side. For documentation on curl, reference die.net.

Part 1: Implement toggelAccelClicked() in MainActivity.java to switch accelerometer streaming on and off. Try logging the accelerometer values in onSensorChanged() at the bottom of the file to make sure that your implementation works properly.

Hint: look at the activity control methods.

Part 2: Implement the sum of squared differences algorithm in addEvent() in DataAccumulator.java to return a new EnergyReading.

Part 3: Implement the queries in EnergyDBHelper.java. Be sure to use the constant names in the inner class EnergyEntry.

Hint: There are some docstring links to tutorials using SQLite.

Part 4: Implement the http post section of the Replicator.java class. Be sure to use the url to your own cgi script on Murphy. Be sure to delete the local data entries that you successfully backed up to avoid running out of space on the phone. The given cgi script only takes one datapoint at a time, but if you want to add some challenge try modifying the cgi to take data in batches.

Note: the parameter key “mac” is misleading. Please send the data stored in the variable 'id' with the mac parameter. For other parameter keys, you should read the SQLGateway.py and understand what it does.

Hint: If you are not getting the data to show up on your browser, try logging the request string in AndroidStudio and the use the curl command with that request string to see what problems you are running into. Some common problems are misspelled parameter keys and improper encoding. More hints are in the docstrings.