*Peter’s edits in italics*

Waterfall –

First figure out the requirements of the software

*Could be assessing what needs to be in the CURSE database*

Second figure out what you want the software to do, this can be something as simple as doing math equations

*I chose to use Python with the sqlite3 library to make this happen*

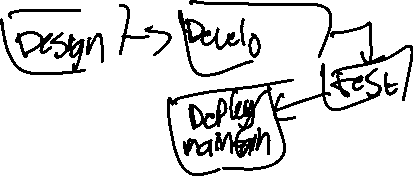
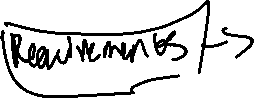
Third begin development for the software, start with the most basic functions first and work from there

*Create framework classes, functions, add SQL, connect both, create UI*

Fourth test your software and fix any bugs you might encounter

*Do unit tests on all of the functions then do a system test on the entire product before releasing.*

Fifth roll out the software as a package or download and continue to maintain any issues that arise



Incremental development –

First figure out the requirements of the software

Second figure out what you need the software to do and its constraints

Third decide on a function to create

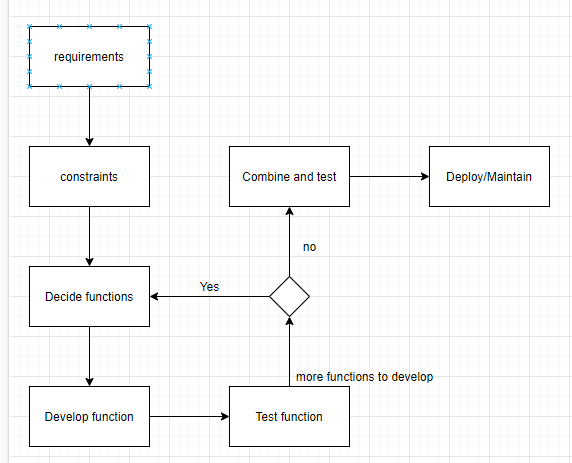
Fourth begin development for the software, creating functions as you go (this can start as basic set/get and evolve to create an entire playlist in spotify)

Fifth test each function before moving on to the next to make sure you won’t encounter any major bugs (repeat steps 3-6 until your software is complete)

Sixth do the final testing of the software with all the functions together

Seventh roll out the software as a package or download and continue to maintain any issues that arise

*What is described is more of a waterfall model. An increment and develop would contain the lifecycle described in steps 1-4, but repeated over and over, adding whatever features the customer might like. In the CURSE database, one example of this could be a cycle where the basic classes and functions’ structures are created and tested before moving on to another cycle, for instance SQL integration.*



Integrate and configure –

First figure out the requirements of the software

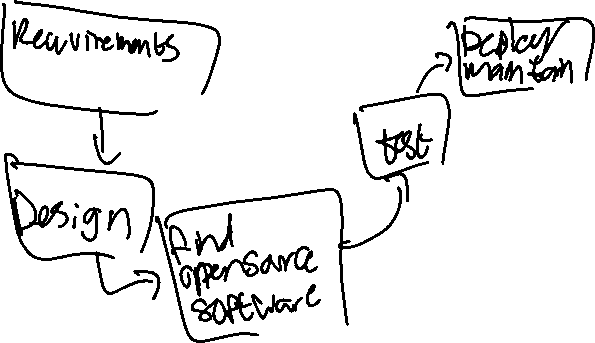
Second figure out what you want the software to do

Third find an open source code that can be built off of allowing you to reuse assets for quicker development

*Does not have to be code, could be an entire external program to use. You could technically find an academic scheduling service and use that to solve the problem.*

Fourth test your software and fix any bugs that are present

Fifth roll out your software as a package or download and continue to maintain any bugs that arise



Sorry for the lack of specificity, I did not have internet access while making this

