# EEC 521 – Process Documentation Questions

(a)This includes all your ​architectural design, prototype, scenarios, user stories, etc ​that generated during your project.

All of these files are in the design folder of the gihub. We uploaded our arch design, user stories, and tech stack planning docs.

<https://github.com/kevinlrak/csu-eec521/wiki#design>

<https://github.com/kevinlrak/csu-eec521/raw/master/design/UserStories.docx>

(b)This includes all your test cases (specify the location in your source code);detailed explanation of your test cases within your source code (i.e. comments)

We have no formal test cases. Did testing as we developed, but did not start from unit tests and code to their passing.

(c)Test cases​ for one unit and test cases for the main features.

See above.

(d)Explanation that ​your software has a great quality​ - Did you design enoughtest cases including stress test? Did you use design patterns to make yoursoftware easy to extend and maintain?

<https://github.com/kevinlrak/csu-eec521/wiki#code-quality>

(e)Scan or take a photo of all ​your notes (including handwritten notes)​ and put in the wiki to show your process.

<https://github.com/kevinlrak/csu-eec521/wiki#design>

(f)Document all the ​failures ​that ever happened during your project and how you overcame it.

Many many difficulties. We had a lot of trouble with the raspberry pi DHT library. The C library is no longer supported (author was sick of people harassing him or using it “stupidly”). This complicated our encapsulation process. After getting that working, we discovered the library did not support cross-compiling and specifically said not to statically link. The only other library we could find that was pre-written is from Adafruit (pi manufacturer) and is a python only library. Moving in this direction broke all the C++ encapsulation code. We tried utilizing a boost library wherein a C++ program can interact seamlessly with a python script. Unfortunately the adafruit library did not play nicely with Boost due to the cross-compilation and not being written for any non-pi hardware (actually has a check and denies if not raspi). Eventually the decision was made to run a python script natively on the pi and treat the entire RSM as the encapsulation. This is part of why we abandoned the message handler and went straight to the SQL server. As hardware changes and libraries update, the RSM itself can be re-written to account without impacting the database design or GUI.

(g)Your ​project progress ​in Gantt Chart format.

<https://raw.githubusercontent.com/kevinlrak/csu-eec521/master/design/DRESS_GANTT.png>

(h)The ​cost ​of your project in dollar amount (for computer hardware, and softwarepurchase/subscription); The ​person hour cost​ - including estimated personhours at the beginning of the project and actual person hour spent.

Pi Zero $10 x 3 = $30

Pi 3B $35 x 1 = $35

DHT11 $2 x 4 = $8

TOTAL $73 in hardware

Cost Estimation (experience-based)  
  
    Kevin estimates based on similar projects in the past that the RSM will take 10 days, the DPS will take 2 days, and the GUI will take 2 days. Total is 14 person-days, where a day is 8 hours. Therefore 112 person-hours.  
  
Cost Estimation (algorithm-based)  
  
    Using Post-architecture model:  
    PM = A\*(Size^B)\*M  
        A=function points + external I/O + user interactions + database tables = 3+2+2+1=8  
        Size = KSLOC from table lookup (QSM 2014) given 3 function points = 0.384  
        B = (2.5+0.0+0.0+1.0+2.0)/100+1.01 = 1.065  
        M = 1  
       = 8\*(0.384^1.065)\*1 = 2.89 Person-Months