Tyler Edmiston
CMSI 402
BJ Johnson
2/6/19

Week 4 Homework

1.1

The basic tasks that all software engineering projects must handle includes: Requirements Gathering, High-Level Design, Low-Level Design, Development, Testing, Deployment, Maintenance, and the Wrap-up.

1.2

Requirements Gathering is learning what the customers want and need.
High-Level Design is where the beginning decisions are made, including what platform and data design to use.
Low-Level Design is about how the pieces of the project will work, including stuff like initial database designs.
Development is where the programmers get to work and continue refining low-level designs until they are able to implement them with code.
Testing is where the programmers check for bugs in the code and continue to write tests to try to flesh out the program.
Deployment is where the software is distributed to the users. Often it does not go as smoothly as planned, with a variety of possible issues.
Maintenance is repairing and bug fixing the software as users find bugs found in deployment.
Wrap-up is where you perform a post-mortem, evaluating where the project development went right and wrong.

2.4

This assignment is written in a Google Doc. The revision history tool is very powerful, and I didn't realize that docs had this tool similar to Microsoft Word.

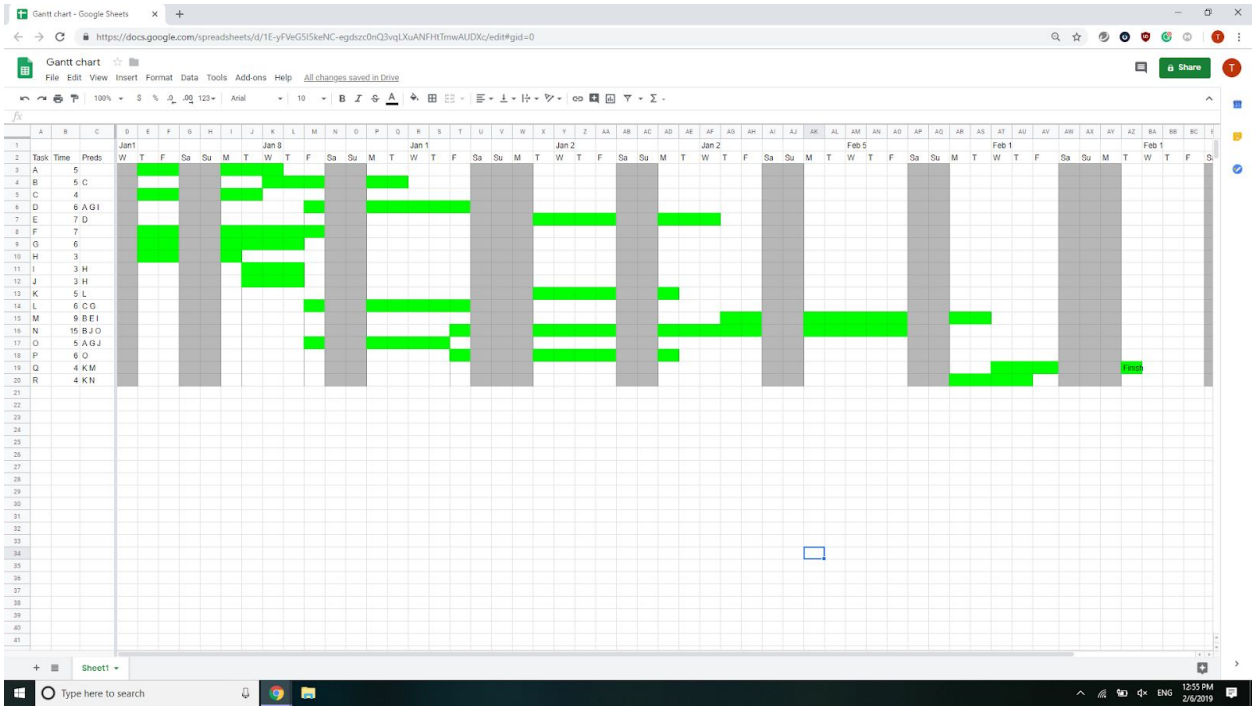
2.5

JBGE stands for "just barely good enough". These are referring code documentation and comments. You should not spend too much time documenting, but you need to do "just barely good enough" so that others, or your future self, can understand.

3.2

The critical paths are Q - M - E - D - G, as well as Q - M - E - D - I - H. Both of these paths have an expected project duration of 33 working days.

3.4



Finishes on February 18th, after working that day.

3.6

One way to prepare for the unexpected is to increase each task by a certain amount. You could add 5% time to each given task, to allow for stuff like server failure or vacation time. This has its drawbacks, as people will use this time up instead of in emergencies, which just extended the timeline by 5%.

3.8

The biggest mistake you can make is to ignore the problem and hope you can make up the time later. Without a reason to believe this, you'll just fall farther behind.
The second is adding more developers to a team when it is already behind schedule. This takes away time from the developers, as they have to train the new teammates.

4.1

The five characteristics of good requirements are: clear, unambiguous, consistent, prioritized, and verifiable.

4.3

None of these are business requirements, as they are not the high-level goals.
The User Requirements only include j, because this is the design of the program for the user.
The Functional Requirements are a, b, c, d, e, i, k, l, m, n, o, p because they are all part of the project's desired capabilities.
The Nonfunctional Requirements are f, g, h since they all are related to the quality of the project's performance.
There are no The Implementation Requirements, as none of these requirements are temporary features to transition to using the new system.

4.9

Must: The game is already functional, so there are no features we absolutely need to include.
Should: Similar to how there are no Must features, since the application is already functional there are no features to be included in should.
Could: A feature to allow users to input a word, then allow to play. This allows two+ players to play hangman in a more traditional sense. Another possible feature is to add a menu when you hit new game, to allow you to select different categories (or all categories which is the default option) to get words from a specific category, like locations or food. Based on the screenshot provided, the application does not use screen space very well. It could make the keyboard larger and placed at the bottom, to allow for fewer misclicks when trying to push the buttons.
Won't: Since we have not talked with the customer, there are no features that we could agree with them upon that are being delayed past release.

Dun goofed photo.