Aidan Esposito

Professor Johnson

CMSI 4072

7 February 2025

Homework 1

1.1 The basic tasks that all software projects must handle include requirements gathering, high-level design, low-level design, development, testing, deployment, maintenance, and wrap-up.

1.2 Requirements gathering is the concept of finding out customer wants and needs and forming them into a requirements document which can be referred to by developers and the customer throughout the course of the project. High-level design is decision making based upon what platform to use, what data to use, and what interfaces to use while also having an idea of what the project architecture and interactions should look like. Low-level design includes information on how each part of the project should work and should give the developer an idea of how the interactions between each piece should work. Development involves turning the high-level and low-level designs into code and trying to limit bugs to the best of their abilities. Testing is writing tests and aiming to fix as many bugs as possible in the code either as a group or solo to aim for an acceptably low rate of bugs in general. Deployment involves the concept of launching the program and running through obstacles such as user training and bug fixes that may come with it. Maintenance is enhancing and bug fixing on the application that has been launched and aiming to improve the experience for the customers even after deployment. Finally, wrap-up is

performing a final post-mortem on the project analyzing what went well and what went bad to prepare for other projects in the future.

2.4 When comparing two different versions of a Google doc in version history, notable changes are listed between the two documents. In the second document, each bit of new addition is highlighted in a different color associated with a user account. Each user also has a timestamp associated with them to allow for change times to be listed on the document. The new document is also listed as the "current" document while the previous one is listed by the specific date which can be restored with a simple button click. Version history also allows you to look at both versions at once to compare the changes in great detail. Finally, the amount of edits done in the new version compared to the old version is listed on the top of the page.

2.5 The acronym JGBE stands for "just barely good enough" and specifically highlights the fact that if you spend a majority of your time on documentation you can waste time updating it when making code changes. It is a good idea to make changes to documentation when ideas are in mind instead of ignoring it even if it is a large amount because it could help understand tough code segments in the future.

4.2

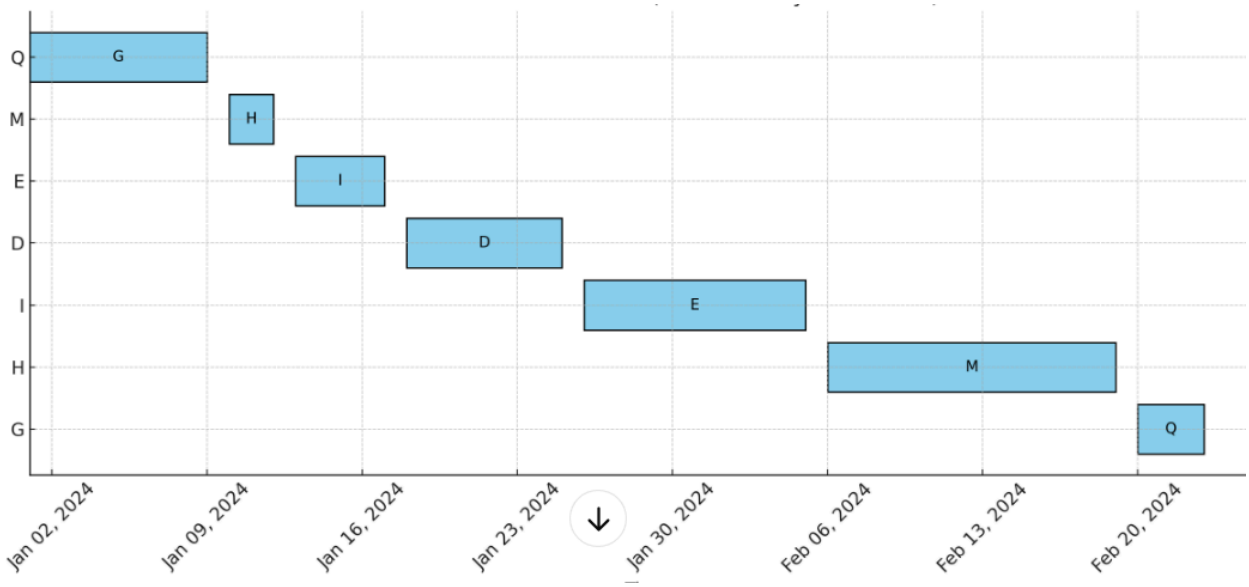
a. To start this problem, I had to find the earliest and latest start and finish times via a diagram creator to help. The earliest and latest start times are added to the duration to find the finish times. The latest finish times are found by doing the opposite and

subtracting the latest start time. We subtract the latest times by the earliest times to find an overall critical path and total time which will be listed in the next few problems.

b. The critical path for this problem is "**G → H → I → D → E → M → Q**." The tasks on the path include: Rendering engine, humanoid base classes, character classes, character editor, character animations, character library, and finally character testing.

c. The total expected duration in working days is 32 days.

4.4 The Gantt chart for the previous problem is as follows:



4.6 To handle these sorts of situations involving completely unprecedented problems, you must try to anticipate any risks with risk management that could hit you throughout the

development process. There needs to be plans in place to handle these risks. If possible, you should also have extra time set aside to deal with any issues that may arise to allow the project to still be completed in a certain time frame. You must also keep this information consistent with other developers and customers in order to have the ability to counter these problems in the first place.

4.8 The two biggest mistakes you can make while tracking tasks includes making estimates for development allowing tasks to fall behind deadlines without a set buffer to counter extra problems that could arise as well as the mistake of adding extra developers to a task and assuming that will fix a problem by itself and not possibly add more issues.

5.1 The 5 characteristics of good requirements include being clear, being unambiguous, being consistent, being prioritized, and being verifiable.

5.3

a. User Requirements, Functional Requirements, Nonfunctional Requirements, Implementation Requirements.
b. User Requirements, Functional Requirements, Nonfunctional Requirements, Implementation Requirements.
c. User Requirements, Functional Requirements, Nonfunctional Requirements, Implementation Requirements.

d. User Requirements, Functional Requirements, Nonfunctional Requirements, Implementation Requirements.
e. User Requirements, Functional Requirements, Nonfunctional Requirements, Implementation Requirements.
f. Functional Requirements, Nonfunctional Requirements, Implementation Requirements.
g. Functional Requirements, Implementation Requirements.
h. Functional Requirements, Nonfunctional Requirements, Implementation Requirements.
i. Functional Requirements, Nonfunctional Requirements, Implementation Requirements.
j. Functional Requirements, Nonfunctional Requirements, Implementation Requirements.
k. Nonfunctional Requirements, Implementation Requirements.
l. User Requirements, Functional Requirements, Nonfunctional Requirements, Implementation Requirements.
m. Functional Requirements, Nonfunctional Requirements, Implementation Requirements.
n. Functional Requirements, Nonfunctional Requirements, Implementation Requirements.
o. Business Requirements, Functional Requirements, Nonfunctional Requirements, Implementation Requirements.
p. Business Requirements, Nonfunctional Requirements, Implementation Requirements.

5.9 The Mr. Bones application can be changed in a few ways with ideas involving the use of the MOSCOW method. Some ideas that are already there in the Must category involve those such as having random word selection, having the skeleton update with each word, and having a

win or loss condition for the game. For the Should section, the game could have possible ideas such as a timer to make the game more intense, hints to allow for easier word selection, or a possible score system for the game. The Could section may involve ideas such as a multiplayer mode for the game or leaderboards for the games internal systems. Finally, the Won't section could involve ideas such as in-app purchases or adding a storyline to the game itself.