Homework Assignment #1

1.1 What are the basic tasks that all software engineering projects must handle?

- **The basic tasks that all software engineering projects must handle include requirements gathering, high-level design, low-level design, development, and testing, deployment, maintenance, and wrap-up (post-mortem).**

1.2 Give a one-sentence description of each of the tasks you listed in Exercise 1.

- **Requirements gathering: entails figuring out what the customer wants and needs.**
- **High-level design: includes information about the project architecture that is broken up into different pieces that handle different areas of functionality.**
- **Low-level design: occurs after creating the high-level design and includes information on how each specific piece of the project should work.**
- **Development: occurs after creating the high-level and low-level, and where the programmers begin writing code.**
- **Testing: programmers should test their code to try to catch and fix as many bugs as they reasonably can**
- **Deployment: this is when you roll out the software you just created to the users**
- **Maintenance: after deployment, users will start using your software and will find bugs that you will need to fix**
- **Wrap-up: this is where you reflect on the project to see what went right and what went wrong.**

2.4 Compare this process to what you can do with GitHub versions. How are the two tools different? How are they the same?

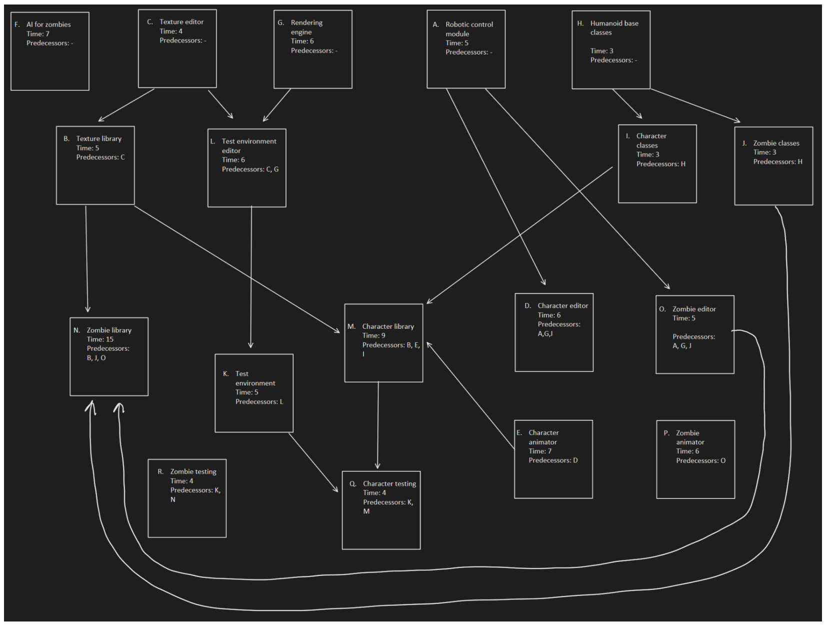
- **Google Docs version history is similar to GitHub's commit history. In both cases, you can access the previous version and revert to it if you want. In GitHub, if you make changes to something but don't commit or push, you can't see those in the commit history. However, in Google Docs, even if you don't save your changes, since it autosaves they appear in the version history.**

2.5 What does JGBE stand for and what does it mean?

- **JGBE stands for "Just Barely Good Enough". It's a philosophy in software engineering that states that one should avoid providing too much documentation since it can waste a lot of time both when you're writing it and when you have to update it. The author states that although it can be a hassle, the programmer should suck it up and write up the documentation anyways to make things clear to those who will read/update code in the future.**

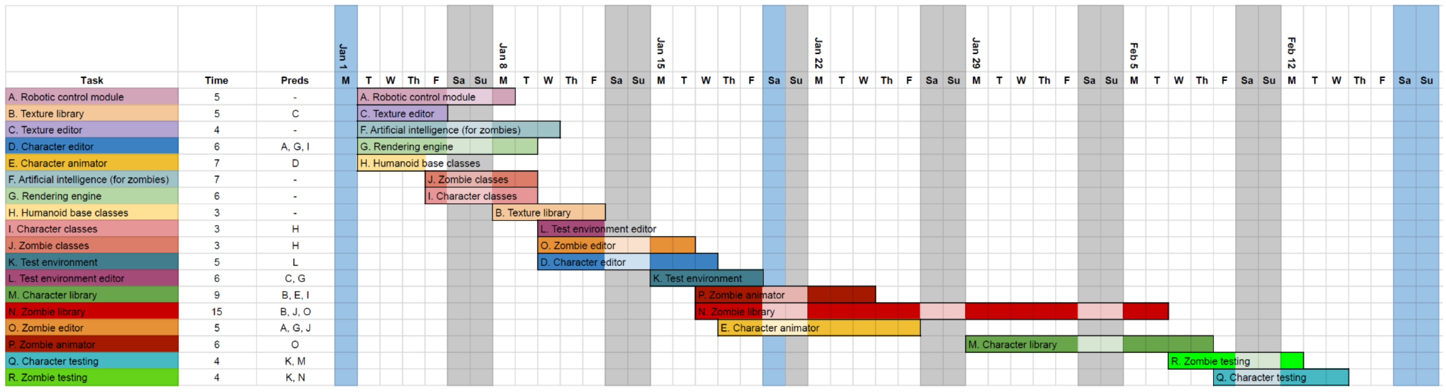
4.2 Use critical path methods to find the total expected time from the project's start for each task's completion. Find the critical path. What are the tasks on the critical path? What is the total expected duration of the project in working days?

- **Here is the PERT chart for this problem (we don't know if we need to do this):**


F. AI for zombies Time: 7 Predecessors: -
C. Texture editor Time: 4 Predecessors: -
G. Rendering engine Time: 6 Predecessors: -
A. Robotic control module Time: 5 Predecessors: -
H. Humanoid base classes Time: 3 Predecessors: -
B. Texture library Time: 5 Predecessors: C
L. Test environment editor Time: 6 Predecessors: C, G
I. Character classes Time: 3 Predecessors: H
J. Zombie classes Time: 3 Predecessors: H
D. Character editor Time: 6 Predecessors: A,G,I
O. Zombie editor Time: 5 Predecessors: A, G, J
N. Zombie library Time: 15 Predecessors: B, J, O
M. Character library Time: 9 Predecessors: B, E, I
K. Test environment Time: 5 Predecessors: L
E. Character animator Time: 7 Predecessors: D
P. Zombie animator Time: 6 Predecessors: O
R. Zombie testing Time: 4 Predecessors: K, N
Q. Character testing Time: 4 Predecessors: K, M

- **The critical path is 32 days, G, D, E, Q**

4.4 Build a Gantt for the network you drew in Exercise 3.



Task	Time	Preds
A. Robotic control module	5	-
B. Texture library	5	C
C. Texture editor	4	-
D. Character editor	6	A, G, I
E. Character animator	7	D
F. Artificial intelligence (for zombies)	7	-
G. Rendering engine	6	-
H. Humanoid base classes	3	-
I. Character classes	3	H
J. Zombie classes	3	H
K. Test environment	5	L
L. Test environment editor	6	C, G
M. Character library	9	B, E, I
N. Zombie library	15	B, J, O
O. Zombie editor	5	A, G, J
P. Zombie animator	6	O
Q. Character testing	4	K, M
R. Zombie testing	4	K, N

4.6 In addition to losing time from vacation and sick leave, projects can suffer from problems that just strike out of nowhere. Sort of a bad version of deus ex machina. For example, senior management could decide to switch your target platform from Windows desktop PCs to the latest smartwatch technology. Or a pandemic, hurricane, trade war, earthquake, alien invasion, and so on could delay the shipment of your new servers. (Not that anything as far-fetched as a pandemic might occur.) Or one of your developers might move to Iceland. How can you handle these sorts of completely unpredictable problems?

- **A couple of ways you can handle this is by expanding the task's time estimate by some amount or adding tasks to represent sick/lost time. Another way to avoid these problems is to just add things like planning for approvals, order lead times, and setup tasks on the schedule**

4.8 What are the two biggest mistakes you can make while tracking tasks?

- **One big mistake is ignoring the problem of fixing the schedule when falling behind on tasks in the hopes of catching up later. The second biggest mistake you can make is to pile extra developers on the task hoping that will reduce the time needed to finish it since it takes time to get people up to speed with the project, and this may lead to further delays.**

5.1 List five characteristics of good requirements.

- **Requirements must be clear, unambiguous, consistent, prioritized, and verifiable.**

5.3 Suppose you want to build a program called TimeShifter to upload and download files at scheduled times while you're on vacation. The following list shows some of the application requirements. For this exercise, list the audience-oriented categories for each requirement. Are there requirements in each category?

User Requirements

- **a. Allow users to monitor uploads/downloads while away from the office.**
- **b. Let the user specify website log-in parameters such as an Internet address, a port, a username, and a password.**
- **d. Let the user select an Internet location, a local file, and a time to perform the upload/download.**
- **e. Let the user schedule uploads/downloads at any time.**
- **l. Let the user empty the log.**
- **n. Let the user view the log reports on a remote device such as a phone.**

Business Requirements

- **o. Send an e-mail to an administrator if an upload/download fails more than its maximum retry number of times.**
- **p. Send a text message to an administrator if an upload/download fails more than its maximum return number of times.**

Function Requirements

- **c. Let the user specify upload/download parameters such a number of retries if there's a problem.**
- **j. Perform schedule uploads/downloads.**
- **k. Keep a log of all attempted uploads/downloads and whether they succeeded.**
- **m. Display reports of upload/download attempts.**

Nonfunctional Requirements

- **f. Allow uploads/downloads to run at any time.**
- **g. Make uploads/downloads transfer at least 8 Mbps.**
- **h. Run uploads/downloads sequentially. Two cannot run at the same time.**

- **i. If an upload/download is scheduled for a time when another is in progress, it waits until the other one finishes.**

  Implementation Requirements

  - **The given list does not include implementation requirements, which specify how the system must be developed, deployed, or maintained after deployment.**

<u>5.9</u> Figure 5-1 [right] shows the design for a simple hangman game that will run on smartphones. When you click the New Game button, the program picks a random mystery word from a large list and starts a new game. Then if you click a letter, either the letter is filled in where it appears in the mystery word, or a new piece of Mr. Bones's skeleton appears. In either case, the letter you clicked is grayed out so that you don't pick it up again. If you guess all the letters in the mystery word, the game displays a message that says, "Congratulations, you won!" If you build Mr. Bones's complete skeleton, a message says, "Sorry, you lost."
Brainstorm this application and see if you can think of ways you might change it. Use the MOSCOW method to prioritize your changes.

- **Must: wrong letters list to make things more clear, add a start screen where the “new game” button will be and replace the one in the game with “play again”, remove the watermark on Mr. Bones**
- **Should: change the mystery word to not look the same as the keyboard, color of the new game button, add an outline to the image that fits the theme**
- **Could: change the layout to fit the screen better, add a more interesting background, sound effects for when you lose and win, animate Mr. Bones**
- **Won’t: the mechanics of the game and the beauty of Mr. Bones**
- **PS: the mystery word in the image is doubloon**