#1.1

The basic tasks that all software engineering projects must handle are:

a. Requirements Gathering
b. High-Level Design
c. Low-Level Design
d. Development
e. Testing
f. Deployment
g. Maintenance
h. Wrap-up

#1.2

The following list gives a one sentence description of each of the tasks listed for Exercise 1.

a. Requirements Gathering—Learn the customer's wants and needs.
b. High-Level Design—Describe the major pieces of the application and how they interact.
c. Low-Level Design—Provide more detail about how to build the pieces of the application so that the programmers can actually implement them.
d. Development—Write code to implement the application.
e. Testing—Use the application under different circumstances to try to detect any flaws or bugs.
f. Deployment—Roll out the application to the users.
g. Maintenance—Implement bug fixes, additions, enhancements, and future versions of the program.
h. Wrap-up—Evaluate the project's history to determine what went right and what went wrong so that you can repeat the good things and avoid the bad things in future projects.

#2.4

- Something we noticed is that it's fairly easy to tell which version we are looking at given the names we provided for the document. It's also easy to see which parts were added to the document and any parts that were changed/deleted with parts being added having highlighted colors while parts we deleted got ~~strikethrough.~~ There was also an option to restore some of the previous versions if we didn't like our current iteration of the document which is pretty neat.

#2.5

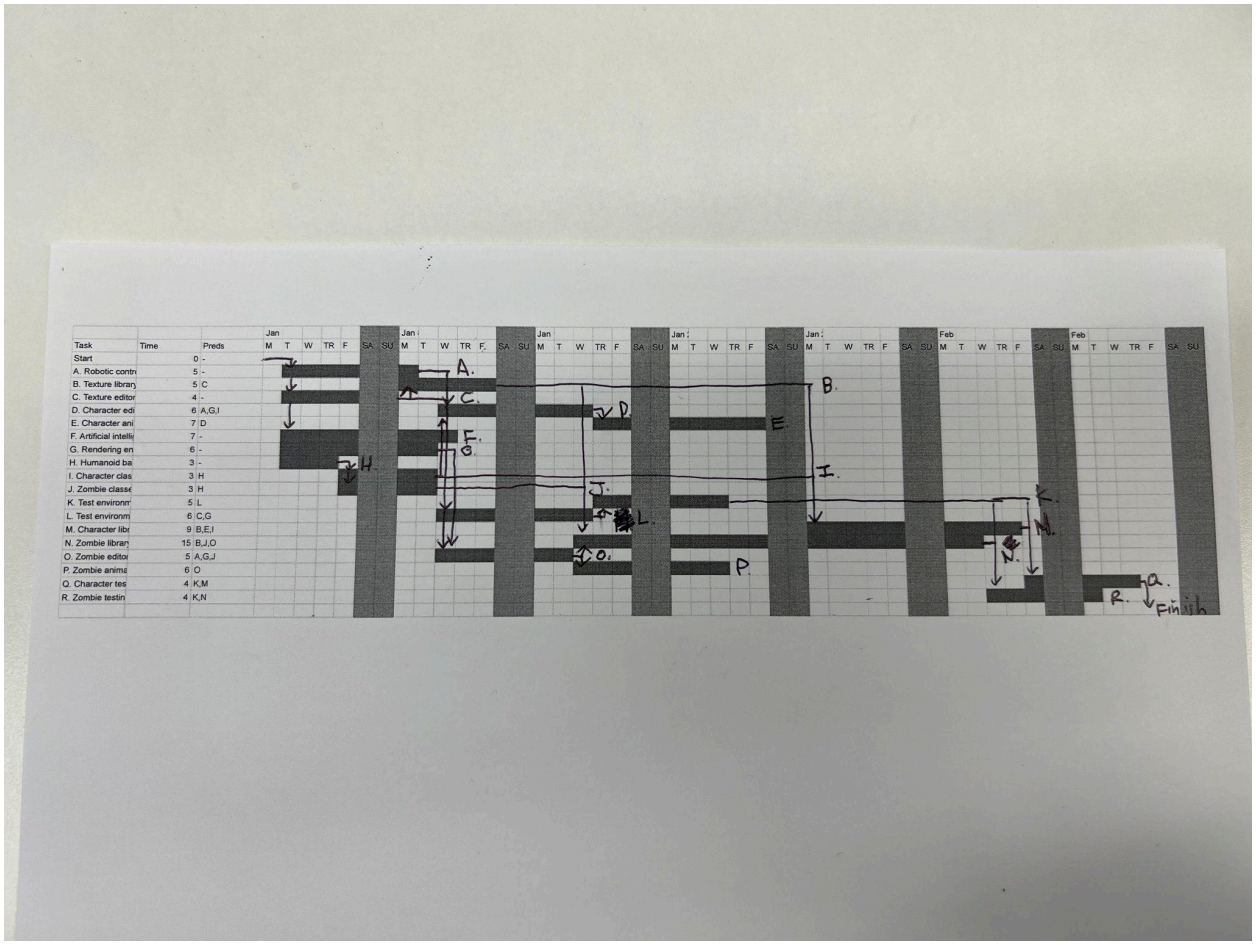
- JBGE stands for "Just Barely Good Enough." It means we should provide only the absolute minimum number of comments necessary for the code that we are writing.

#4.2

- GONR
    - G. Rendering engine
    - O. Zombie editor
    - N. Zombie library

    - R. Zombie testing
- 30 days

#4.4



#4.6

- One of the ways to handle these unpredictable problems is to expand the time on each task. For example, if a task originally takes 2 weeks to complete, maybe give an extra few days or 1 week max just to account for unacceptable things that could come up (sickness, vacation, etc…) Another way is to just add tasks at the end of the schedule to handle unexpected problems.

#4.8

- The first biggest mistake when it comes to tracking tasks is making sure that when an objective appears to be delayed, don't just wait and hope it resolves itself in time, take action if necessary.
- The second biggest mistake is trying to add more people onto an assignment and thinking it will shorten the time needed to complete that task. The only way this works is if the people joining in on the same assignment have had experience or know-how in that subject.

#5.1

- The five characteristics of good requirements are clear (easy to understand), unambiguous (clear on priority), consistent (no contradictions), prioritized (necessity first), and verifiable (testing).

#5.3

- The audience-oriented categories for each requirement are (B = Business, U = User, F = Functional, N = Nonfunctional, I = Implementation).
    a. Allow users to monitor uploads/downloads while away from the office. → **B**
    b. Let the user specify website log-in parameters such as an Internet address, a port, a username, and a password. → **U, F**
    c. Let the user specify upload/download parameters such as number of retries if there's a problem. → **U, F**
    d. Let the user select an Internet location, a local file, and a time to perform the upload/download. → **U, F**
    e. Let the user schedule uploads/downloads at any time. → **N**
    f. Allow uploads/downloads to run at any time. → **N**
    g. Make uploads/downloads should transfer at least 8 Mbps. → **N**
    h. Run uploads/downloads sequentially. Two cannot run at the same time. → **N**
    i. If an upload/download is scheduled for a time when another is in progress, the new task waits until the other one finishes. → **N**
    j. Perform scheduled uploads/downloads. → **F**
    k. Keep a log of all attempted uploads/downloads and whether they succeeded.→ **F**
    l. Let the user empty the log. → **U, F**
    m. Display reports of upload/download attempts. → **U, F**
    n. Let the user view the log reports on a remote device such as a phone. → **U, F**
    o. Send an e-mail to an administrator if an upload/download fails more than its maximum retry number of times. → **U, F**
    p. Send a text message to an administrator if an upload/download fails more than its maximum retry number of times. → **U, F**
- The only requirement that we are missing is implementation and that's probably because we don't need to buy any new hardware or network bandwidth since the jobs above are mainly software focused and maybe the hardware has already been taken care of in the background.

#5.9

- We can add a points or score feature to highlight how many letters and words the user gets correct → **S (Should)**
- Leaderboard system, the user can look at past high scores to try and beat the current highest score → **S (Should)**

- Users should be allowed to guess the whole word and not just fill in letters one by one → **S (Should)**
- Users can choose different characters like an animal or person instead of the skeleton before they start the game to add more variety → **C (Could)**
- Different levels of difficulty from easy to hardest or maybe as the user continues to correctly guess the word, it's gets harder → **C (Could)**
- Maybe a hint button if the player is stuck on a word → **C (Could)**
- Maybe add a multiplayer mode so users can compete with one another → **C (Could)**
- Sound effects → just to add some sounds when a user gets a letter right or wrong → **W (Won't)**
- Adding time limits or a countdown while the user tries to guess the word → **W (Won't)**
- Adding some animations to the drawing or selection of letters → **W (Won't)**