Sam R., CJ P., Brady A.

HW #1

1. **What are the basic tasks that all software engineering projects must handle?**
    a. Requirements Gathering, High-level Design, Low-level Design, Development, Testing, Deployment, Maintenance, and Wrap-up
2. **Give a one sentence description of each of the tasks you listed in Exercise 1.**
    a. Requirements Gathering - Determining customers' wants and needs
    b. High-level Design - Breaking the projecting into large chunks and major functionality
    c. Low-level Design - Breaking High-Level parts into smaller, more achievable parts
    d. Development - Carrying out work to complete requirements
    e. Testing - Effectively testing code to ensure it has minimal bugs
    f. Deployment - Rolling out the software for real-life testing
    g. Maintenance - Fixing Bugs
    h. Wrap-up - Post-Mortem evaluation
3. **Document what you've noticed about the information you see, and how the differences between versions are displayed; Compare this process to what you can do with GitHub versions. How are the two tools different? How are they the same?**
    a. Google Docs has a great system for version control. As you go, you can name and save different versions of your document that can easily be referenced and recovered. They are all easily accessed and can be seamlessly compared to each other.
    b. This is very similar to version control in GitHub, where you can also easily access older versions of projects to compare and revert any changes. The also both allow streamlined change tracking. Google Docs saves changes automatically though, while GitHub requires manual commits. GitHub is also different in how it allows multiple files and functions like branching and merging.
4. **What does JBGE stand for and what does it mean?**
    a. "Just Barely Good Enough"
    b. "The idea is that if you provide too much documentation, you end up wasting a lot of time updating it as you make changes to the code."
5. **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?**
    a. Critical Path: A>C>G>H>B>L>J>O>K>N>R
    b. Tasks: A. Robotic control module; C. Texture editor; G. Rendering engine; L. Test environment editor; B. Texture library; H. Humanoid base classes; J. Zombie classes; O. Zombie editor; K. Test environment; N. Zombie library; R. Zombie testing
    c. Total time duration: 30 Days (*Breakdown shown in Gantt Chart*)
6. **Build a Gantt chart for the network you drew in Exercise 3. [Yes, I know, you weren't assigned that one — however, when you do Exercise 2 you should have**

enough information for this one.] Start on Wednesday, January 1, 2024, and don't work on weekends or the following holidays:

a. Work is in “HW #1 - Gantt chart.pdf”

7. **How can you handle these sorts of completely unpredictable problems?**
    a. One option is to expand the estimated time of each task by some small amount just in case. Another option is to add specific tasks that represent lost time. Lastly, it is essential to carefully plan for approvals, setup, and order lead times.
8. **What are the two biggest mistakes you can make while tracking tasks?**
    a. 1. Ignoring problems that come up and assuming that you will find time for it later.
    b. 2. Piling extra developers on a task and expecting that it will reduce the time it takes to complete it.
9. **List five characteristics of good requirements.**
    a. Clear, Unambiguous, Consistent, Prioritized, and Verifiable
10. **For this exercise, list the audience-oriented categories for each requirement. Are there requirements in each category? [If not, state why not…]**
    a. Allow users to monitor uploads/downloads while away from the office.
        i. Functional
    b. Let the user specify website log-in parameters such as an Internet address, a port, a username, and a password.
        i. Functional
    c. Let the user specify upload/download parameters such a number of retries if there's a problem.
        i. Functional
    d. Let the user select an Internet location, a local file, and a time to perform the upload/download.
        i. Functional
    e. Let the user schedule uploads/downloads at any time.
        i. Functional
    f. Allow uploads/downloads to run at any time.
        i. Non-Functional
    g. Make uploads/downloads transfer at least 8 Mbps.
        i. Non-Functional
    h. Run uploads/downloads sequentially. Two cannot run at the same time.
        i. Non-Functional
    i. If an upload/download is scheduled for a time when another is in progress, it waits until the other one finishes.
        i. Non-Functional
    j. Perform schedule uploads/downloads.
        i. Non-Functional
    k. Keep a log of all attempted uploads/downloads and whether they succeeded.
        i. Functional
    l. Let the user empty the log.
        i. Functional
    m. Display reports of upload/download attempts.

        i. Functional
    n. Let the user view the log reports on a remote device such as a phone.
        i. Functional
    o. Send an email to an administrator if an upload/download fails more than its maximum retry number of times.
        i. Functional
    p. Send a text message to an administrator if an upload/download fails more than its maximum retry number of times.
        i. Functional
11. **Brainstorm this application and see if you can think of ways you might change it. Use the MOSCOW method to prioritize your changes.**
    a. Must - Allowing the user to choose a theme/category before starting the game, Data tracking to show how many wins and losses the player has
    b. Should - An option to choose and change the difficulty of the word given, create an animation for the skeleton
    c. Could - Functionality to guess letters with your voice instead of typing, Better visual effect and feedback when correctly getting the word or a letter
    d. Won't - Create the game in other languages, Multiplayer mode where players compete to get the word in the fewest guesses