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

The basic tasks that are required involve gathering requirements for the software project, designing the high-level design of the project, then breaking apart the high level design into parts that are then further designed (known as low-level design), actually writing the code, testing the code, deploying the code, and then finally maintaining the code.

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

Gathering requirements involve gathering exactly what the customers want and what they need from your software. High level design involves designing the architecture of your software, often abstracting more detailed parts for the sake of making sure the software meets the requirements listed. Low level design is the designing of the individual parts of the software aka the implementation details of the parts. Writing the code is exactly what it sounds like: you take the high and low level designs and then use them to write code that fits those designs. Testing the code involves often gathering testers who did not write the code to try to find bugs/features that were not intentional. During the testing stage is often the case where code needs to be rewritten to fix the bugs that eventually arise. Deploying the code involves rolling out the software to the consumers and usually involves setting up servers/fixing bugs that arise during deployment. Maintaining the code is the last but equally crucial step of making sure your software consistently meets the expectations of the users, that might be from simply fixing bugs that arise or adding improvements/new features that the users may demand.

2.4 Click the versions listed on the right to see what changed between versions. 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?

From Google Docs you can clearly see the changes made from the version history and also you have the option to restore different versions that you have saved in the past. This is very similar to github as they both allow the restoration/retrieval of previous versions. The main difference is their use case, Google Docs is mainly for text documents so their version control is very simplified. However Github is meant more for a general purpose version control for software and thus they have much more features that allow collaboration and also in depth analysis of changes between versions. They however are both meant for version control between text whether that be .doc files or .py/any other coding language files.

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

JBGE stands for just barely good enough and it means that when you are writing code documentation/comments that describe the code that you simply should write the bare minimum. It is a school of thought in software engineering and it is mostly based on the idea that because code is ever changing, writing too much documentation can lead to wasting a lot of time because you'll have to revise/rewrite it when the code eventually changes.

4.2a. Use critical path methods to find the total expected time from the project's start for each task's completion.

The critical path involves finishing Q and includes H, I, A, G, D, E, C, B, M, L, K, Q. Added together this is 3 + 3 + 5 + 6 + 6 + 7 + 4 + 5 + 9 + 6 + 5 + 4 = 63

4.2b. Find the critical path. What are the tasks on the critical path?

H, I, A, G, D, E, C, B, M, L, K, Q. The tasks are Humanoid base classes, Character classes, Robotic control module, Rendering engine, Character editor, Character animator, Texture editor, Texture library, Character library, Test environment editor, Test environment, and Character testing.

4.2c. What is the total expected duration of the project in working days?

Given the critical path we get the total expected duration of 63 days.

4.4 Build a Gantt chart for the critical path you drew in Exercise 2. Start on Wednesday, January 1, 2024, and don't work on weekends or the following holidays.

4072 - Assignment 1 Question 4.4

4.6 In addition to losing time from vacation and sick leave, projects can suffer from problems that just strike out of nowhere, like 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, right?] Or one of your developers might move to Iceland, which is a real nice place to raise your kids up. How can you handle these sorts of completely unpredictable problems?

We can handle these sorts of completely unpredictable problems by expanding each task's time estimate by a percentage amount, which gives breathing room for these delays. Another way to handle this is to add specific tasks to the project to represent lost time that has a high likelihood of happening. For example, already setting sick leave or vacation time as a task in the schedule so that when someone inevitably does take time off, it fits into the plan.

4.8 According to your textbook, what are the two biggest mistakes you can make while tracking tasks?

The biggest mistake you can make is to blindly trust a developer's progress estimate, as a lot of unseen/unpredicted work lies near a deadline. This often leads to a missed deadline or long work hours, which leads to burnout which is not good. The second biggest mistake is to put more people on a task that is running late, because most of the time adding more people does not make a task happen faster.

5.1 List five characteristics of good requirements

1. Clear
2. Unambiguous
3. Consistent
4. Prioritized
5. Verifiable

5.3 For this exercise, list the audience-oriented categories for each requirement. Are there requirements in every category?

- **Business Requirements (B)**: These describe high-level goals that the application must achieve.
    - (a) Allow users to monitor uploads/downloads while away from the office.
- **User Requirements (U)**: These specify the actions that the user must be able to perform.
    - (b) Let the user specify website log-in parameters such as an Internet address, a port, a username, and a password.

    - (c) Let the user specify upload/download parameters such as the number of retries if there's a problem.
    - (d) Let the user select an Internet location, a local file, and a time to perform the upload/download.
    - (l) Let the user empty the log.
    - (m) Display reports of upload/download attempts.
    - (n) Let the user view the log reports on a remote device such as a phone.
    - (o) Send an email 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 retry number of times.
- **Functional Requirements (F)**: These define the application's core functions and behaviors.
    - (b) Let the user specify website log-in parameters such as an Internet address, a port, a username, and a password.
    - (c) Let the user specify upload/download parameters such as the number of retries if there's a problem.
    - (d) Let the user select an Internet location, a local file, and a time to perform the upload/download.
    - (j) Perform scheduled uploads/downloads.
    - (k) Keep a log of all attempted uploads/downloads and whether they succeeded.
    - (l) Let the user empty the log.
    - (m) Display reports of upload/download attempts.
    - (n) Let the user view the log reports on a remote device such as a phone.
    - (o) Send an email 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 retry number of times.
- **Nonfunctional Requirements (N)**: These specify system-wide constraints or qualities, such as performance, reliability, and security.
    - (e) Let the user schedule uploads/downloads at any time.
    - (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 (I)**: These define technical constraints related to the system's development and deployment.
    - **No implementation requirements were specified.** While additional hardware or network upgrades might be required, it's assumed that the necessary infrastructure is already in place

5.9

a. Must Have:
    i. Better Win/Loss reinforcement: If you guess the wrong letter there should be a sound effect to make the player more aware than just graying out the letter. When you win there should be colorful text as well as a winning sound effect.
    ii. Increasing Difficulty: In the current description it doesn't say if the game gets harder after winning, so after the player wins the difficulty should scale higher to increase player engagement and retention.

iii. Score Tracking: Have a way to track the number of guesses per word and how fast a player completes the word to encourage people aiming for faster times.

b. Should Have:
   i. Multiplayer: Having a multiplayer mode would allow players to challenge each other and would increase social interaction.
   ii. Word Categories: Set up different categories/themes of words for people to pick from so it's not just random.
   iii. Leaderboards: To incentivize players to keep playing, set up a leaderboard system for everyone worldwide to compete against each other.

c. Could Have:
   i. Customization/Item Shop: Allow players to customize Mr.Bones by purchasing cosmetics from an item shop.
   ii. Daily Challenges: Players can earn in game currency for completing dailies keeping players engaged.
   iii. Achievements: With achievements it will give players a challenge to try to 100% the game and provide them with an exclusive reward.

d. Won’t Have Now:
   i. Voice-Over: Voicelines for Mr bones if the player wins or loses.
   ii. Multi Platform: Currently only available for PC, but maybe for consoles later.
   iii. Custom Wordlists: Allows players to make and share their own word lists.