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

All software engineering projects must handle gathering requirements, high level design, low level design, development, testing, deployment, and maintenance.

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

Gathering requirements involves identifying what a piece of software needs to accomplish based on customer wants and needs. High level design involves defining the overall architecture and major components of the system. Low level design involves specifying the internal details of the components. Development involves writing the code based on the design specifications. Testing involves verifying that the software functions as it should and eliminating bugs before deployment. Deployment involves releasing the software to users, which might involve installation or hosting. Maintenance involves continuously resolving bugs, improving performance and adding new features.

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

Google Docs' file version history is different from Github versions in that it tracks changes to documents in real time as opposed to managing versions of software source code that are committed at the developer's discretion. They are similar in that they both maintain a history of changes made to a file over time, as well as the ability to revert to previous versions.

4. What does JBGE stand for and what does it mean?

JBGE stands for just barely good enough, and refers to pieces of code that are barely sufficient to fulfil their intended purpose without unnecessary complexity.

5.
    a. Use critical path methods to find the total expected time from the project's start for each task's completion.

A. Robotic control module: 5 days
B. Texture library: 9 days
C. Texture editor: 4 days
D. Character editor: 12 days
E. Character animator 19 days
F. Artificial intelligence (for zombies): 7 days
G. Rendering engine: 6 days
H. Humanoid base classes: 3 days
I. Character classes: 6 days
J. Zombie classes: 6 days
K. Test environment: 17 days
L. Test environment editor: 12 days
M. Character library: 28 days
N. Zombie library: 34 days
O. Zombie editor: 11 days
P. Zombie animator: 17 days
Q. Character testing: 32 days
R. Zombie testing: 38 days

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

Task G, then task L, then task K, then task N, then task R.

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

38 days

6. 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:

Task	Start	End
A	1/2	1/8
B	1/9	1/15
C	1/2	1/5
D	1/9	1/16
E	1/17	1/27
F	1/2	1/10
G	1/2	1/9
H	1/2	1/6
I	1/7	1/10
J	1/7	1/10
K	1/29	2/4
L	1/9	1/16
M	1/28	2/8
N	2/9	2/28
O	1/9	1/15
P	1/16	1/23
Q	2/9	2/14
R	3/3	3/6

7. How can you handle these sorts of completely unpredictable problems?

In order to handle completely unpredictable problems, one could set up backup plans, and adopt agile methodologies in order to dedicate the next sprint towards what needs to be fixed.

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

The two biggest mistakes you can make while tracking tasks are ignoring a slipping task and hoping to make up the time later, and adding more developers to a late task assuming that it will speed up completion.

9. List five characteristics of good requirements.

Good requirements should be clear/unambiguous, complete, verifiable, feasible, and traceable.

10. For this exercise, list the audience-oriented categories for each requirement. Are there requirements in every category? [If not, state why not…]

Business Requirements: a. Allow users to monitor uploads/downloads while away from the office, e. Let the user schedule uploads/downloads at any time

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, 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, n. Let the user view the log reports on a remote device such as a phone, l. Let the user empty the log.

Functional Requirements: 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, e. Let the user schedule uploads/downloads at any time, f. Allow uploads/downloads to run at any time, 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, j. Perform scheduled uploads/downloads, k. Keep a log of all attempted uploads/downloads and whether they succeeded, m. Display reports of upload/download attempts.

Nonfunctional Requirements: g. Make uploads/downloads transfer at least 8 Mbps, 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 retry number of times.

Implementation Requirements: None

There are requirements for almost every category excluding the implementation requirements. If TimeShifter was replacing a previously implemented system, implementation requirements would be necessary.

11. Brainstorm this application and see if you can think of ways you might change it. Use the MOSCOW method to prioritize your changes.

Based on the MOSCOW method, I will break down the changes I thought about into categories: must change, should change, could change, and won't change.

Must change: n/a. The game seems to function as intended, and the core functionality of the game of hangman is there. I don't feel that there are any imperative changes that need to be made.

Should change: One feature that should be implemented is a word category selector, which allows the player to select a theme for the word selected, which can allow for a more engaging and slightly easier game.

Could change: One change that could be made is an improvement to the UI, whether that be better animations of Mr. Bones' limbs or the additions of sound effects.

Won't change: The Mr. Bones theme seems to be the key differentiating factor between this game and traditional hangman, so I believe the theme should not change.