Problem 1.1, Stephens page 13

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

The basic tasks that all software projects must handle are Requirements Gathering, High Level design, Low Level design, Development, Testing, Deployment, Maintenance, and Wrap-up.

Problem 1.2, Stephens page 13

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

Requirements Gathering: The process of identifying the specific needs of stakeholders for a project.

High-Level Design: Creating an overall architecture for the system, defining major components and their interactions.

Low-Level Design: Detailed design work that specifies the internal workings of the components.

Development: The phase where the coding of the system takes place.

Testing: Evaluating the system for bugs, errors, and performance issues to ensure it meets the defined requirements.

Deployment: The process of releasing the system to users.

Maintenance: Ongoing support and updates after deployment to fix issues.

Wrap-up: The final phase where project documentation is completed, lessons are learned, and the project is concluded.

Problem 2.4, Stephens page 27

Like Microsoft Word, Google Docs [sic] provides some simple change tracking tools. Go to http://www.google.com/docs/about/ to learn more and sign up [if you do not have an account already]. Then create a document, open the File menu's Version History submenu, select Name Current Version, and name the file 'Version 1'. Make some changes and repeat the preceding steps to name the revised document 'Version 2'. Now open the File menu's Version History submenu again but this time select See Version History. 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.

On the right side, the versions are displayed with a timestamp for each edit. Google Docs records the person who made the changes with the corresponding timestamp. When you click on a version, any changes between versions are highlighted. For each version, you can see what was added or removed. This helps track edits made to the document over time. Finally, you can also restore previous versions if needed by selecting a specific version and clicking "Restore this version."

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

Problem 2.5, Stephens page 27

What does JBGE stand for and what does it mean?

"just barely good enough" – it means that sometimes developers spend too much time and detail on documentation, so rather than doing that, we should keep the documentation simple and straightforward.

Data for Problems 4.2 and 4.4

Table 4.2 [below] summarizes some of the classes and modules you might need (and their unreasonably optimistic expected times) to develop players and zombies for the game. (The program would also need lots of other pieces not listed here to handle other parts of the game.)

Use the following table of data for Exercises 4.2 and 4.4.

Task	Time (Days)	Predecessors
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

Problem 4.2, Stephens page 78

1. Use `critical path methods` to find the total expected time from the project's start for each task's completion.
2. Find the critical path. What are the tasks on the critical path?

   The critical path is G, D, E, M, and Q

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

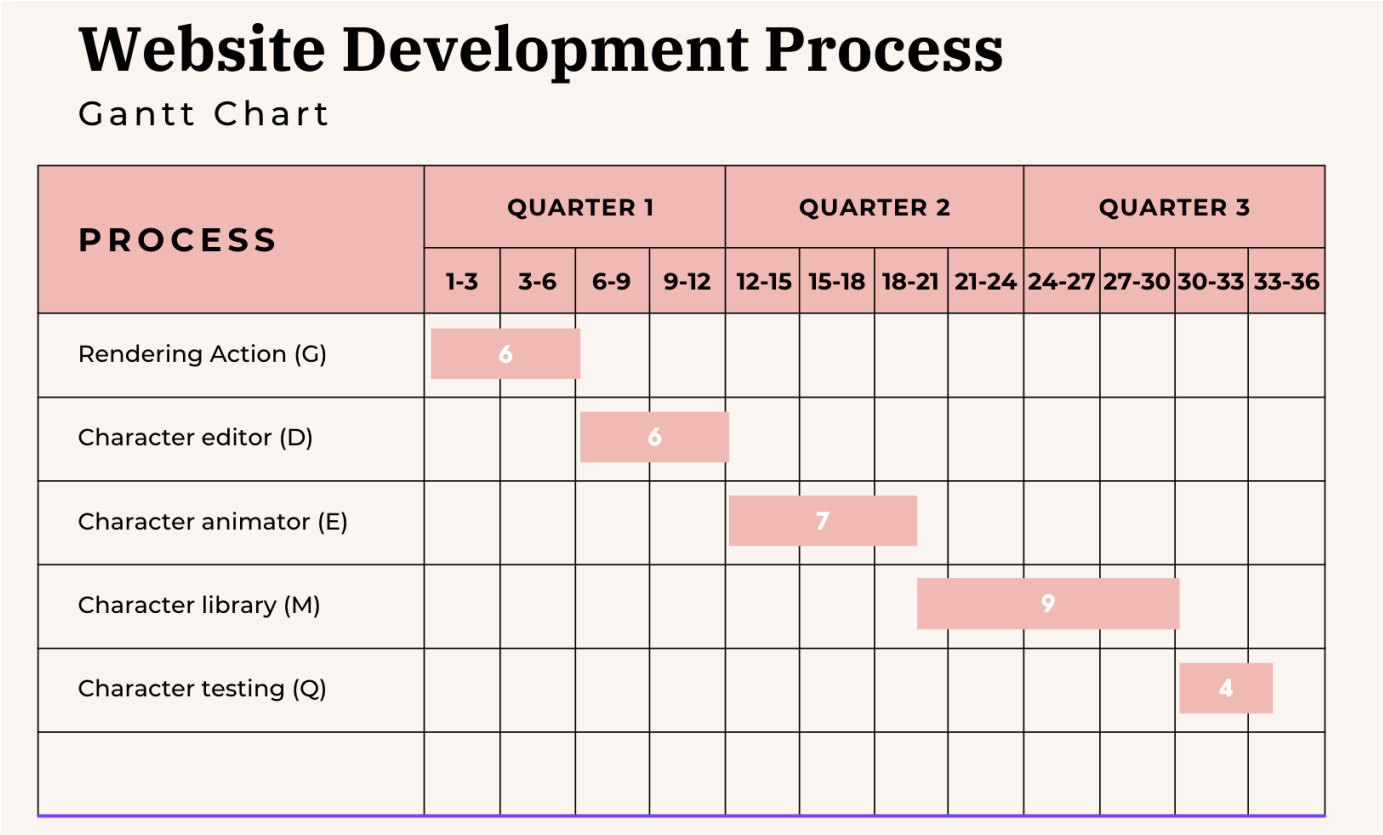
   The critical path time is 32 working days from adding up the values in the critical path.

Problem 4.4, Stephens page 78

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:

Holiday	Date
New Year's Day	January 1
Martin Luther King Day	January 20
President's Day	February 17
St. Valentine's Day	February 14

Alien Overloard Appreciation Day	March 26
Income Tax Day	April 15



Problem 4.6, Stephens page 79

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?

Unpredictable problems can disrupt the smooth progression of any project. For instance, unforeseen global events like a pandemic, hurricane, or trade war could delay crucial deliveries, such as servers. Project managers can mitigate their impact by adopting flexible planning strategies. Building contingency plans, having backup resources or talent, and maintaining open communication with stakeholders can help manage the effects of these unexpected disruptions.

Problem 4.8, Stephens page 79

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

The biggest mistake you can make is ignoring a problem and hoping you'll catch up later. The second biggest mistake is adding more developers to the task and thinking it will help finish it faster.

Problem 5.1, Stephens page 114

List five characteristics of good requirements.

Clear, unambiguous,Consistent, Prioritized, Verifiable

Problem 5.3, Stephens page 114

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 applications 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 a 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.
- 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 whan another is in progress, it waits until the other one finishes.
- j. Perform schedule uploads/downloads.
- k. Keep a log of all attempted uploads/downloads and whether the succeeded.
- l. Let the user empty the log.
- m. Display reports of upoad/download attempts.
- n. Let the user view the log reports on a remote device such as a phone.
- 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 it's maximum retury umber of times.

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

Business :

- Enable users to monitor upload and download processes while away from the office.

User :

- Allow users to define website login parameters, including the internet address, port, username, and password.
- Permit users to specify upload and download settings, such as the number of retry attempts in case of an error.
- Allow users to select an internet location, a local file, and a scheduled time for uploading or downloading.

- Let the user empty the log.
- Display reports of upload/download attempts.
- Let the user view the log reports on a remote device such as a phone.
- Send an e-mail to an administrator if an upload/download fails more than its maximum retry number of times
- Send a text message to an administrator if an upload/download fails more than its maximum retry number of times.

Functional:

- Allow users to define website login parameters, including the internet address, port, username, and password.
- Permit users to specify upload and download settings, such as the number of retry attempts in case of an error.
- Allow users to select an internet location, a local file, and a scheduled time for uploading or downloading.
- Perform scheduled uploads/downloads.
- Keep a log of all attempted uploads/downloads and whether they succeeded.
- Let the user empty the log.
- Display reports of upload/download attempts.
- Let the user view the log reports on a remote device such as a phone.
- Send an e-mail to an administrator if an upload/download fails more than its maximum retry number of times.
- Send a text message to an administrator if an upload/download fails more than its maximum retry number of times.

Nonfunctional:

- Provide users the ability to schedule uploads/downloads at any time.
- Enable uploads/downloads to occur at any time without restrictions.
- Ensure uploads/downloads transfer at a minimum speed of 8 Mbps.
- Ensure uploads/downloads occur sequentially, with no simultaneous transfers.
- If an upload/download is scheduled for a time when another is in progress, the new task waits until the other one finishes.

No, implementation requirements are empty. This is because there is no mention of any additional hardware needed to complete these aforementioned steps.

Problem 5.9, Stephens page 115

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 again. If you guess all the letters in the mystery word, the game displays a message that says, "Contratulations, 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 have
Game Reset Functionality: Add a "Restart" or "Reset Game" button
High Score Tracking: Track high scores for the number of correct guesses and the number of games won locally.
Interactive Design: Ensure the game is fully responsive on desktop and mobile.

Should have

Leaderboard: Display a leaderboard showing the best scores.
Progressive Difficulty: Gradually increase the difficulty level as the user wins more games

Could have

Multiplayer: Allow users to play in a head-to-head multiplayer mode.
Word Categories: Provide word categories (e.g., Animals, Sports, Movies, etc.),
Wont have

Advanced AI Opponents: Introducing an AI that can "choose" words intelligently or try to guess the player's word
Full Story Mode: Adding a narrative or story behind the hangman game is not necessary for this simple application.