Written Assignment #1
Anika Bhatnagar
Ria Singh

Problem 1.1, Stephens page 12

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

Requirements gathering, high-level design, low-level design, development, testing, deployment, and maintenance.

Problem 1.2, Stephens page 12

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

Requirements gathering: Collecting and understanding what the users and stakeholders need from the software.

High-level design: Creating an overall architecture or blueprint of the system, defining major components and their interactions.

Low-level design: Detailing the design of individual components, specifying how they will be implemented.

Development: Writing the actual code to build the software according to the design specifications.

Testing: Verifying that the software works as intended and meets all requirements.

Deployment: Releasing the software to users and making it operational in the intended environment.

Maintenance: Updating and improving the software after deployment to fix issues, add features, or adapt to changing requirements.

Problem 2.4, Stephens page 26

Google Docs' version history feature enables users to automatically track changes, name different versions, and view modifications between revisions. Edits are highlighted, indicating both the changes made and the contributors, with an option to revert to earlier versions. In contrast to GitHub, which relies on manual commits, Google Docs automatically saves versions without user intervention. GitHub, on the other hand, provides more sophisticated version control options, such as branching, commit messages, and detailed code comparisons, making it ideally suited for software development. Although both platforms support collaboration and version tracking, Google Docs is optimized for document editing, while GitHub offers more structured version management tailored for coding projects.

Problem 2.5, Stephens page 26

What does JBGE stand for and what does it mean?

Just Barely Good Enough. This means that models and documentation should be just good enough to complete the task needed without any overcomplication and unnecessary work.

Use the following table of data for Exercises 3.2 and 3.4.

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 3.2, Stephens page 51

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?

H, I, D, E M, Q. Humanoid base classes, Character classes, Character editor, Character animator, Character library, Character testing

Expected duration is the sum of the days for the critical path = 39 working days.

Problem 3.4, Stephens page 51

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, 2020, 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

Problem 3.6, Stephens page 51

In addition to losing time from vacation and sick leave, projects can suffer from problems that just strike out of nowhere. Sort of a bad version of *deus ex machina*. For example, senior management could decide to switch your target platform from Windows desktop PSs to the latest smartwatch technology. Or a strike in the Far East could delay the shipment of your new servers. Or one of your developers might move to Iceland. How can you handle these sorts of completely unpredictable problems?

There are several possibilities to handle these unpredictable problems. One is adding buffer time/resources in the project plan to account for instances like this. Similarly, risk management can be assessed during the project planning phase. Having open and regular communication is really important and can help in unexpected situations like these. Ensure that multiple team members are equipped to handle important tasks in case a member has to drop out or can't help for some reason.

Problem 3.8, Stephens page 51

What are the two biggest mistakes you can make while tracking tasks?

1: failing to update the task status regularly

2: focusing solely on task completion without focusing on quality

Problem 4.1, Stephens page 82

List five characteristics of good requirements.

Clear/precise, complete, verifiable, consistent, and feasible.

Problem 4.3, Stephens page 82

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 each category? [If not, state why not…]

End-User Requirements (Usability & Functionality)

- Monitor uploads/downloads remotely.
- Specify website login details (Internet address, port, username, password).
- Set upload/download parameters (e.g., retry attempts).
- Choose an Internet location, local file, and scheduled time.
- Schedule and execute uploads/downloads anytime.
- View log reports remotely (e.g., on a phone).
- Display reports of upload/download attempts.
- Empty the log when needed.

2. System Requirements (Performance & Constraints)

- Ensure transfer speeds of at least 8 Mbps.
- Run uploads/downloads sequentially (no concurrent operations).
- Queue scheduled tasks if another is in progress.

- Execute scheduled uploads/downloads.
- Maintain a log of all upload/download attempts and results.

3. Administrative Requirements (Notifications & Alerts)

- Send an email to an administrator if a transfer fails beyond the retry limit.
- Send a text message to an administrator under the same failure condition.

All major categories are covered, though **security measures** (e.g., encryption, authentication) are not explicitly mentioned and may be worth considering.

Problem 4.9, Stephens page 83-84

Figure 4-1 [right] shows the design for a simple hangman game that will run on smartphones. When you click the New Bame 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 Changes (Critical for functionality and user experience)

1. Mobile Responsiveness: Ensure the game scales well on different screen sizes and resolutions.

2. Error Feedback: Display clear messages for invalid inputs (e.g., non-alphabet characters).
3. Enhanced Word List: Include a large, categorized, and randomized word list for replayability.
4. Accessibility: Add support for text-to-speech or colorblind-friendly themes.

Should-Have Changes (Important but not critical for launch)

1. Multiplayer Mode: Allow two players to compete by guessing words or playing simultaneously.
2. Difficulty Levels: Provide easy, medium, and hard modes with varying word lengths and guess limits.
3. Hint System: Add optional hints to assist players (e.g., category or first letter).

Could-Have Changes (Nice to include for enhanced experience)

1. Leaderboard: Track scores and display the highest-ranking players globally or locally.
2. Customization Options: Let players customize Mr. Bones' appearance or the background theme.
3. Sound Effects: Include sound effects for correct/incorrect guesses and animations for Mr. Bones.
4. Timed Mode: Add a timer to increase challenge levels.

Won't-Have Changes (Out of scope for now)

1. AR/VR Integration: Implementing augmented reality features is too complex for the current scope.
2. Cross-Platform Play: Allowing players to connect across devices would require significant infrastructure changes.