Problem 1.1

The basic tasks are requirements gathering, high-level design, low-level design, development, testing, deployment, and maintenance.

Problem 1.2

Requirements gathering is figuring out what the customer wants and needs. High-level design is deciding the overall structure and platform of the system. Low-level design is detailing how each individual piece will work. Development is writing the actual code. Testing is finding and fixing bugs. Deployment is releasing the software to users. Maintenance is fixing bugs and adding features after release.

Problem 2.4

I created two named versions in Google Docs using File > Version History > Name Current Version. When viewing version history, the right panel lists versions with timestamps and editor names. Changes are highlighted inline, additions in green, and deletions in strikethrough red. Compared to GitHub both tools track changes over time and allow reverting to previous versions. GitHub requires manual commits with messages and shows line diffs, making it better suited for code.

Problem 2.5

JBGE stands for just barely good enough. It means writing only the minimum documentation or comments needed to understand the code, avoiding unnecessary overhead. The risk is developers take it too far and write too little, making the code hard to maintain.

Problem 4.2

Earliest finish times:

A = 5, B = 9, C = 4, D = 12, E = 19, F = 7, G = 6, H = 3, I = 6, J = 6, K = 17, L = 12, M = 28, N = 26, O = 11, P = 17, Q = 32, R = 30

Critical path: H → I → D → E → M → Q (also G → D → E → M → Q converges at D)

Total project duration: 32 working days

Problem 4.4

Starting January 2 (Jan 1 is a holiday), skipping weekends and all listed holidays:

H (Humanoid base classes, 3 days): Jan 2 – Jan 4 I (Character classes, 3 days): Jan 5 – Jan 9 G (Rendering engine, 6 days, parallel): Jan 2 – Jan 9 A (Robotic control module, 5 days, parallel): Jan 2 – Jan 8 D (Character editor, 6 days): Jan 10 – Jan 17 E (Character animator, 7 days): Jan 18 – Jan 26 M (Character library, 9 days): Jan 29 – Feb 8 Q (Character testing, 4 days): Feb 9 – Feb 15

Project completes: Thursday, February 15, 2024 (Feb 14 excluded as Valentine's Day holiday)

Problem 4.6

You can't prevent unpredictable problems but you can plan for them by building buffer time into the schedule from the start. This means padding task estimates, leaving slack between dependent tasks, and never scheduling developers at 100% capacity so there is room to absorb surprises without collapsing the entire timeline.

Problem 4.8

The two biggest mistakes are not tracking tasks at all, which means you lose visibility into whether the project is on schedule, and micromanaging so heavily that you waste more time on status updates and meetings than on actual work.

Problem 5.1

Good requirements must be clear, unambiguous, consistent, prioritized, and verifiable.

Problem 5.3

a — User requirement b — User requirement c — User requirement d — User requirement e — User requirement f — Functional requirement g — Nonfunctional requirement (performance) h — Functional requirement i — Functional requirement j — Functional requirement k — Functional requirement l — User requirement m — User requirement n — User requirement o — Functional requirement p — Functional requirement

There are no business requirements because none describe a high-level organizational goal behind building the system. There are no implementation requirements because there is no existing system being replaced and no infrastructure transition needed.

Problem 5.9

Must: Show remaining wrong guesses allowed so the player knows how close they are to losing. Fix any crash if the word list is empty.

Should: Add difficulty levels using easy vs. hard words. Track win/ loss record across sessions to give players a reason to continue playing.

Could: Add a system that reveals one letter at a cost. Add word categories so players can choose a topic. Add simple sound effects on correct and wrong guesses.

Won't: Online leaderboard and social sharing