Carter Esparza

Assignment 1

1.1

Software engineering projects must handle several essential tasks. First, they perform requirements analysis and specification in which the needs and constraints of stakeholders are gathered and documented. Next, project planning and management is carried out to define schedules, allocate resources, and manage risks. The design phase follows, which is divided into high-level design (architectural structure) and low-level design (detailed module interfaces and data flows). Then, implementation or coding transforms design documents into executable code. After coding, systematic testing and verification ensure that all components work as intended and meet the specified requirements. Once testing is complete, deployment and delivery transition the software into the production environment. Finally, maintenance and evolution address bug fixes, enhancements, and adaptation to future needs over the lifecycle of the software.

1.2.

- Requirements Analysis and Specification: This task involves engaging with stakeholders to identify, document, and validate exactly what the software must achieve.
- Project Planning and Management: In this phase, the project is scheduled and resources are allocated while risks are managed to ensure a smooth development process.
- System and Software Design: Designers create both high-level and low-level architectural plans that define how the software will meet its requirements.
- Implementation (Coding): Developers translate the designs into functional, working code by writing and integrating source code.

- Testing and Verification: The software is rigorously tested in both isolated units and as an integrated whole to ensure it behaves as expected.
- Deployment and Delivery: The finished software is installed into a production environment and made available to its end users.
- Maintenance and Evolution: The software is continuously monitored, updated, and enhanced over its lifetime to adapt to user needs and fix any issues.

2.4

When using Google Docs, version history is maintained automatically, making changes immediately visible through color-coded differences that show all modifications. In contrast, GitHub versioning is commit-based; developers explicitly create commits with messages that detail the changes, and the system supports branching and merging, which enables parallel development. Although both systems record the evolution of the work, Google Docs is optimized for real-time document collaboration, while GitHub is designed for detailed, distributed source code management.

2.5

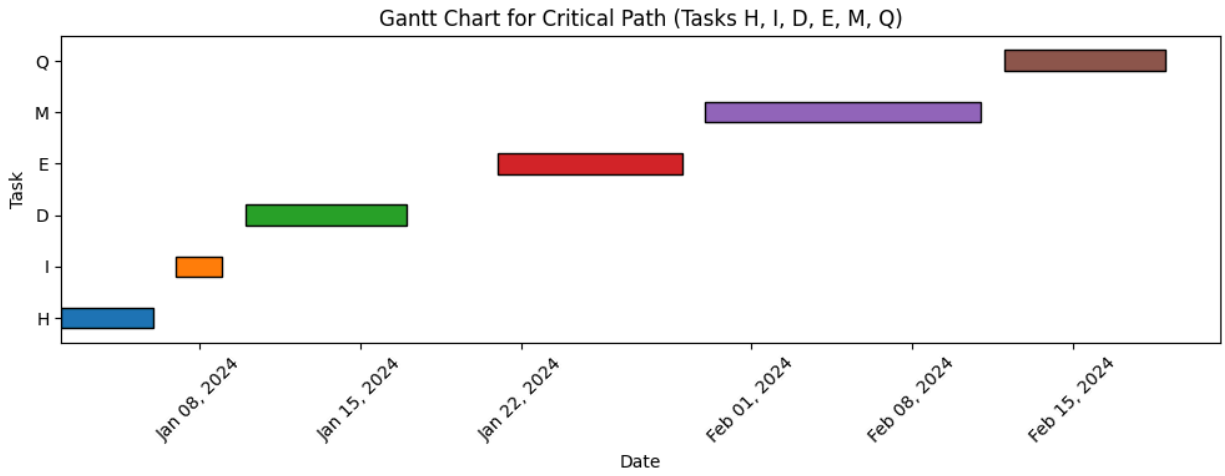
JBGE stands for “Just Barely Good Enough,” which is the idea of delivering a solution that fully meets the necessary requirements without adding unnecessary complexity or features, ensuring that development is efficient and practical.

4.2

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

In the project, the earliest finish times for the tasks are determined by considering tasks that have no predecessors and then calculating the finish times for those with dependencies. After computing the finish times for all tasks, it was determined that the task with the latest finish time was Task Q, at 32 working days. The critical path, which is the sequence of tasks with no slack and the longest duration, is identified as follows: Task H (3 days) → Task I (3 days) → Task D (6 days) → Task E (7 days) → Task M (9 days) → Task Q (4 days). Therefore, the critical path is H, I, D, E, M, and Q, and the total expected duration of the project is 32 working days.

4.4



4.6

To address unpredictable issues such as sudden changes in target platforms, natural disasters, or unexpected personnel shifts, project teams should incorporate contingency buffers in the schedule and budget. They should also engage in thorough risk management by assessing potential risks early and developing contingency plans. Agile or iterative development processes can help accommodate changes incrementally, and continuous, transparent communication among team members and stakeholders will ensure that emerging issues are promptly managed.

4.8

According to the textbook, the two biggest mistakes when tracking tasks are failing to update task progress regularly, which leads to a misalignment between planned and actual work, and overcomplicating the tracking process, which can obscure true progress and slow down decision-making.

5.1

Good requirements should be clear and unambiguous, meaning that every stakeholder understands them in the same way. They must be complete, including all necessary details to

describe the system's functionality. Requirements also need to be consistent, with no conflicting information. They should be testable, so that criteria or tests can be developed to verify that they have been met. Finally, requirements must be feasible in that they can be realistically implemented given the current technological, budgetary, and time constraints.

5.3

For the TimeShifter application, the requirements can be organized into several audience-oriented categories:

- End-User Requirements: These include features that allow users to monitor file transfers remotely, specify login parameters, configure upload/download settings, choose file locations and times, schedule transfers, and view logs on remote devices.
- System or Functional Requirements: These are the functional aspects of the application that mandate how transfers run (e.g., transfers run at any time, execute sequentially, and are queued if overlapping).
- Performance Requirements: One specific requirement is that uploads/downloads must transfer at least 8 Mbps, ensuring adequate performance.
- Administrative and Reporting Requirements: This category covers requirements for maintaining and clearing logs, generating reports, and notifying administrators via email or text in case of repeated failures.

5.9

Using the MOSCOW prioritization technique, the improvements for the Mr. Bones Hangman game can be categorized as follows:

- Must-Haves: The game needs to have a refreshed UI, instructions for the user for the game, and ensuring state manages the key inputs for past guesses, and ensuring wins and loss messages are enhanced and immersive.
- Should-Haves: The game should offer enhanced visual and audio feedback on correct or incorrect guesses and could include multiple difficulty levels to cater to a wider audience. Moreover, saving and resuming game progress is a valuable feature for mobile users.
- Could-Haves: Potential additions include multiplayer capabilities, customizable visual themes, or adaptive hints that provide assistance after several wrong guesses.
- Won’t-Haves (for now): The design will not include overly complex game mechanics or fully integrated social media features until the core gameplay is stable and refined.