Senior Project Assignment 1

Problem 1.1

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

1. Requirements Gathering
2. High Level Design
3. Low Level Design
4. Development
5. Testing
6. Deployment
7. Maintenance
8. Wrap up

Problem 1.2

Give a one sentence description of all the tasks you listed above

1. Requirements Gathering: find out what the customer needs are.
2. High Level Design: Make decisions about the platform, interfaces, and what data design to use
3. Low Level Design: Decide how each piece of the project will work
4. Development: Implement the agreed solution to the customer's needs
5. Testing: ensure that the project works as expected and intended
6. Doplymenet: release the product for the customers to use
7. Maintenance: Fix any bugs that arise as customers use product and add necessary features
8. Wrap up: Evaluate the project and determine what went right and what went wrong

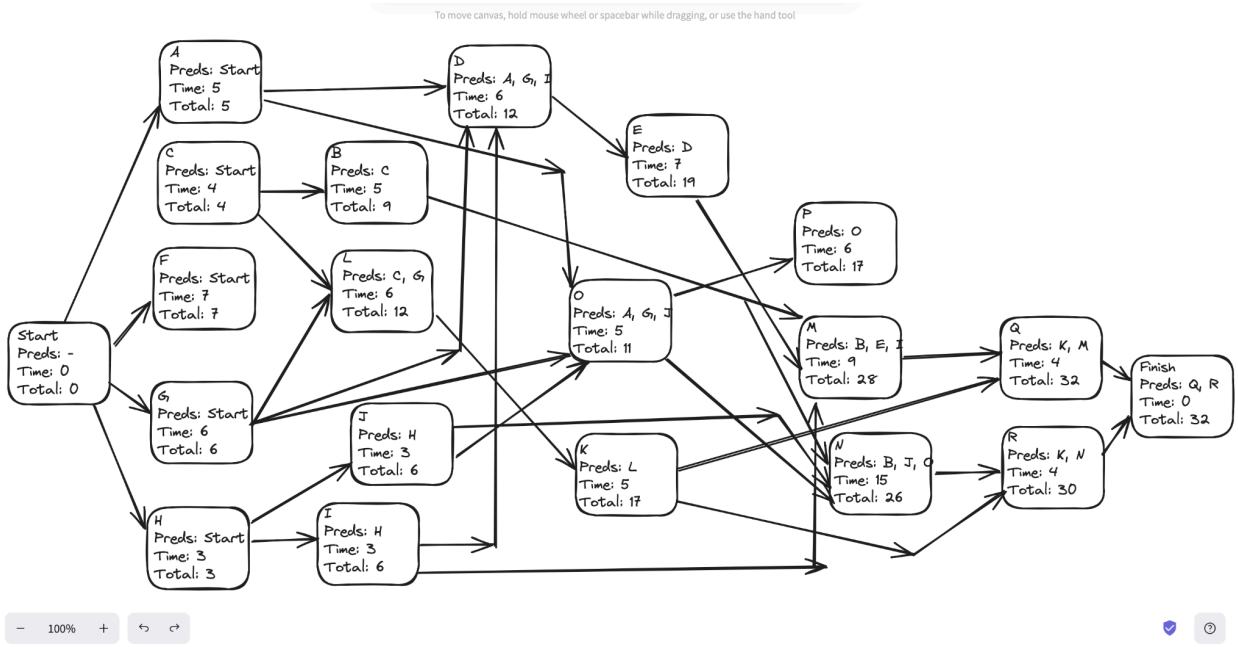
Problem 2.4

Things that were deleted between versions are crossed out. Things that were inserted into newer versions are highlighted green.

Problem 2.5

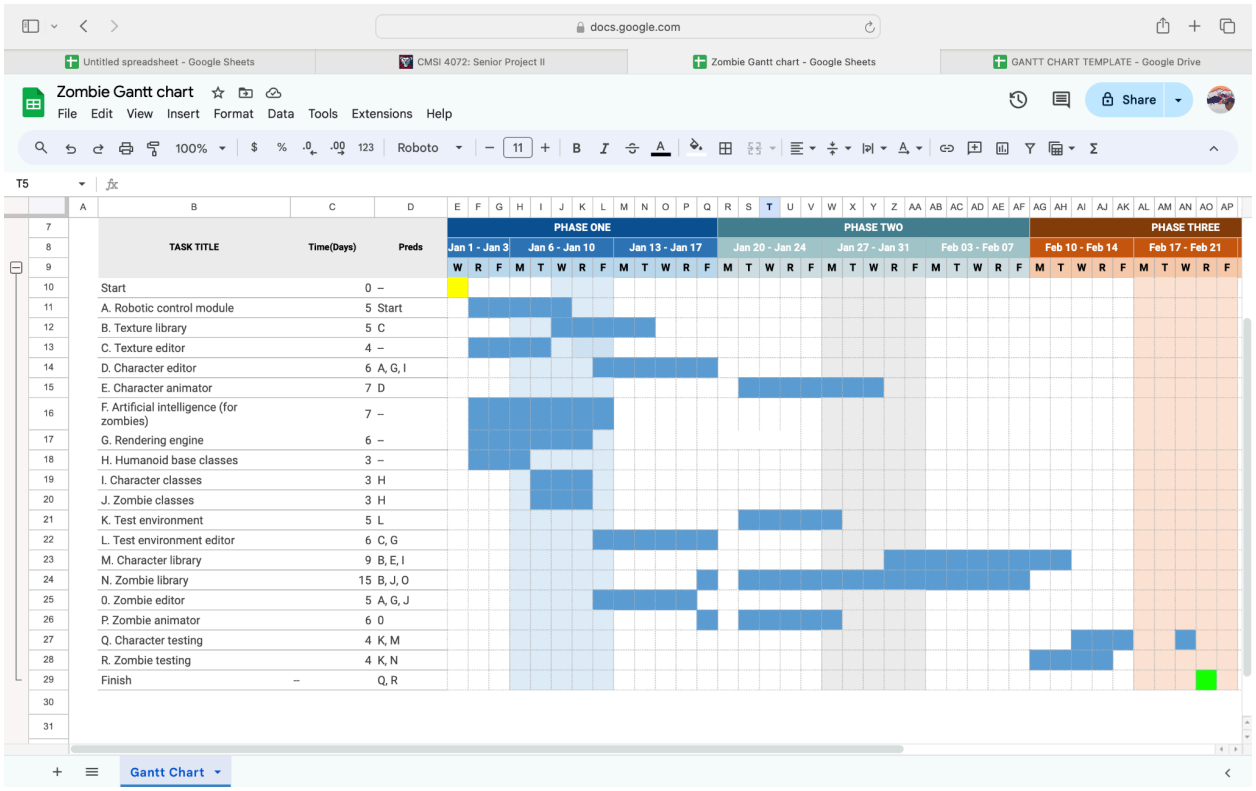
JBGE stands for **J**ust **B**arely **G**ood **E**nough. It means writing documentation and comments in code that are just okay but not giving sufficient details about what the code actually does.

Problem 4.2



The critical path is G->D->E->M->Q. The tasks on the critical path are tasks A, D, E, M, and Q. The total expected time is 32 days.

Problem 4.4



TASK TITLE	Time(Days)	Preds
Start	0	–
A. Robotic control module	5	Start
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
0. Zombie editor	5	A, G, J
P. Zombie animator	6	0
Q. Character testing	4	K, M
R. Zombie testing	4	K, N
Finish	–	Q, R

Problem 4.6

There are various ways to do this.

1. Add extra time to each task's estimated time
2. Increase the project's estimated timeline by an amount
3. Add specific tasks to the project timeline, like "vacation days," "travel days," or "sick leave."

Problem 4.8

1. Ignore the problem and assume you can make up time
2. Pile up extra developers on the task and assume they can do the task faster

Problem 5.1

Clear (easy to understand), unambiguous, consistent, prioritized, and verifiable.

Problem 5.3

a. Business Requirement
b. User Requirement and Functional Requirement
c. User Requirement and Functional Requirement
d. User Requirement and Functional Requirement
e. Nonfunctional Requirement
f. Nonfunctional Requirement
g. Nonfunctional Requirement
h. Nonfunctional Requirement
i. Nonfunctional Requirement
j. Functional Requirement
k. User Requirement and Functional Requirement
l. User Requirement and Functional Requirement
m. User Requirement and Functional Requirement
n. User Requirement and Functional Requirement
o. User Requirement and Functional Requirement
p. User Requirement and Functional Requirement

Every category is used once except for implementation requirements. The app is able to manage uploads and downloads just fine and none of the requirements are suggesting temporary features that will be discarded later.

Problem 5.9

The letters MSCW correspond to their meaning in the acronym MOSCOW

Advertising (M)—A phone application typically costs money to install or display advertising. The application could be modified to display advertising.

Scoring (S)—Right now you either win or lose. The program could be changed to calculate a score.

Score keeping (S)—If the program calculates scores, it could keep track of them so that the user can try to beat the previous best score.

Animated win (C)—When the user wins, the program could display an animation.

Word difficulty tracking (W)—The program could track the number of incorrect guesses for each word to determine its difficulty. It would periodically report values to a central database for distribution during later updates.