Owen Dewing
CMSI 4072
Homework 1

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

1.2 Give a one-sentence description of each of the tasks you listed for exercise 1
1. Requirements Gathering - Finding out what the customer wants and needs.
2. High-level design - Describing the applications major areas/features.
3. Low-level design - How these major features function and how the developers will implement them.
4. Development - Writing code to create the application.
5. Testing - Trying to detect any problems or bugs in your application.
6. Deployment - Roll out the application to users.
7. Maintenance - Fixing bugs as users find them.
8. Wrap-up - Evaluate the project and decide what went well and what went wrong.

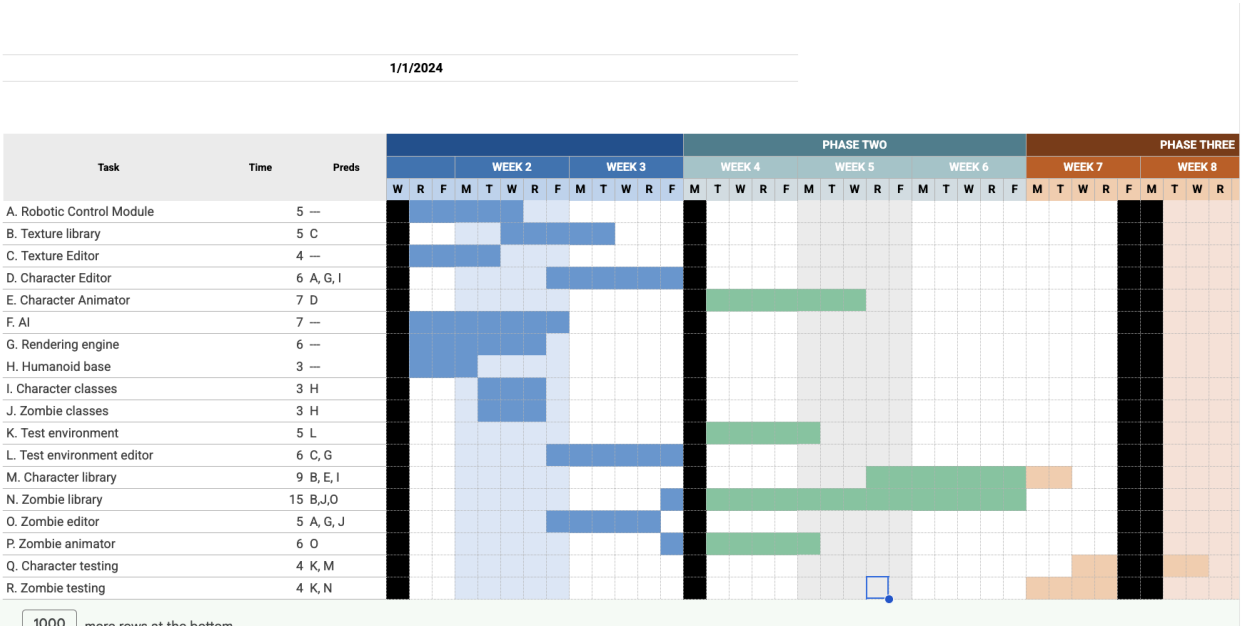
2.4
Google docs observations:
- Changes between versions are highlighted, with text additions appearing in a different color and deletions indicated with strikethroughs.
- Saves by timestamp
- Saves automatically
- Users can restore previous versions

Comparison with Github:
- Does not save automatically, requires manual commits
- Similar to google docs in which it shows which lines of code have changed after a commit
- Supports branches for parallel development (docs does not)
- Github can also revert commits the same way docs can restore previous versions

2.5 What does JBGE stand for and what does it mean?
JBGE stands for “just barely good enough”. This means that you shouldn’t provide more documentation/comments about your code than needed.

4.2/4.4
Total expected time from start->end: 32 days.
Tasks on path: G, D, E, M, and Q

Weekends are already excluded in this chart



4.6
Treat deus ex machina problems like sick leave. You can add extra tasks at the end of your schedule to try and anticipate these problems. Insert the time into the schedule if a problem does occur.

4.8 Two biggest mistakes while tracking tasks?

1. Not taking action when a task slips.
2. Piling more people on a task and assuming that will cut time.

5.1 5 characteristics of good requirements:

1. clear
2. unambiguous
3. consistent
4. prioritized
5. verifiable

5.3
B=Business, U=User, F=Functional, N=Nonfunctional, I=Implementation

a. Allow users to monitor uploads/downloads while away from the office (B)
b. Let the user specify website log-in parameters such as an Internet address, a port, a username, and a password. (U,F)

c. Let the user specify upload/download parameters such as number of retries if there's a problem. (U,F).
d. Let the user select an Internet location, a local file, and a time to perform the upload/download. (U,F)
e. Let the user schedule uploads/downloads at any time. (N)
f. Allow uploads/downloads to run at any time. (N)
g. Make uploads/downloads transfer at least 8 Mbps. (N)
h. Run uploads/downloads sequentially. Two cannot run at the same time. (N)
i. If an upload/download is scheduled for a time when another is in progress, the new task waits until the other one finishes. (N)
j. Perform schedules uploads/downloads. (F)
k. Keep a log of all attempted uploads/downloads and whether they succeeded. (F)
l. Let the user empty the log. (U,F)
m. Display reports of upload/download attempts. (U,F)
n. Let the user view the log reports on a remote device such as a phone. (U,F)
o. Send an email to an administrator if an upload/download fails more than its maximum retry number of times. (U,F)
p. Send a text message to an administrator if an upload/download fails more than its maximum retry number of times. (U,F)

5.9

Must haves: Is app free? Is it paid? Does it show ads?
Should-have: difficulty feature? Hint system? light/dark mode?
Could-have: multiplayer, customization, leaderboard
Wont-have: 3d graphics, time limits, AI implementations