Problem Set Up

The first thing to do is set up the project-plan. I populated the work estimates with my standard WAG of 8 points per task, which translates to 2 weeks, 64 dev hours, or 80 work hours. I also happen to have my own Jira account for personal projects (I'm a hobbyist game developer for a small team), so the goal was to populate the various tasks and dependencies into Jira and then use Jira to create a Gantt chart and schedule which can be found in Figure 1 of the Appendix. More on that later.

Model Specification

We're starting by using Critical Path Analysis, which is going to provide the earliest possible time a task can be started based on any dependencies being predicted to have been completed. This is going to be useful to show us where tasks with common dependencies can be started in the first optimal distribution of tasks. However, it's not initially going to provide a good understanding of where tasks may be provided extra time for completion. In some cases, specific tasks may have a less stringent scheduling requirement than initially indicated by Critical Path Analysis. In these cases, a task may have more time available to complete it or could be started later than would otherwise be apparent without impacting the predicted end of the schedule.

Programming

Code for this assignment can be found in Cogswell_code.py in the git repo, and the output can be found in output.txt. The results indicate tasks A, B, C, and E can occur relatively early and in parallel with D, and then the remaining tasks of F, G and H all require D to have completed before the final three tasks can be started.

Solution 1: Best Case Scenario

Beginning with the Best Case scenario, that my initial estimations are approximately correct, results in the Gantt chart presented in Figure 1. In it, several different tasks can go over their expected time without causing a delay. For instance, task 'B - Develop Marketing Strategy'

can take until C is complete before it needs to be done, since it's only a prerequisite for E (which in turn is only a prerequisite for F, which also requires all of D to be completed). What this means is that, while A needs to be completed expediently in order to kick off task D, B, C, and E can all be completed in the interim while coders are working on D.

Within D, D5 is the only task which seems to be able to be pushed, since D5 and D6 can both be started as soon as D4 is completed, but D5 is a prereq for D8, which also requires D7 (which in turn requires D6 to be completed.) In other words, D5 needs to be completed in the time both D6 and D7 are to be completed.

This results in a schedule where, at any one time and given minimal schedule drift, only two tasks should need be worked on simultaneously. From a hiring perspective, this would mean any hiring done could either be on a multi-project basis (i.e. Each specialist group could work on this project along with several others, only contributing to each when their specific skillset is required) or you could have a minimal number of skill-generalists who are specialized to the project (i.e. a Project Manager who is also able to contribute to the code base stays on the project through its whole lifespan). In my experience, both likely have their merits, so I'd probably recommend a hybrid in which one person is dedicated to the project while the other, more specialized skillsets could be distributed across multiple projects.

Solution 2: Most Likely Scenario

Since D2 and D3 both have their prerequisite D1 satisfied at the same time and are prerequisites in turn for D4, if anything goes wrong with either of those two engineering tasks, D5 is at risk of being delayed. Similarly, the completion of D8 triggers both F and G, which are in turn prerequisites for the final task, H, and are reliant on all previous tasks to have occurred before they can be begun. For this reason, those two points, during D2 & D3 and during F & G, are the most likely points in which delay will occur, irrespective of estimated complexity of tasks. It seems unlikely that impactful delay will occur with tasks A, B, C, and E since they can occur simultaneously with the various D tasks, and the later D tasks already have leeway incorporated

for them in the Best Case Scenario since D5 can occur any time before both D6 and D7 are finished.

Solution 3: Worst Case Scenario

The worst case scenario is colored by the fact that I am a federal government contractor and earlier worked as a database administrator for a Legal Malpractice Insurance company. I have witnessed first hand remarkably catastrophic failures in planning and execution. In a worst case scenario, only a single employee is able to work on this project. Not only does this preclude the ability to effectively work on multiple portions of the project simultaneously, it also injects risk in the skillset of the singular employee, increasing the likelihood of the individually supported tasks to be delayed in their execution. In this scenario, many of the 16 tasks could take as long as a month where previously they would've been expected to take half as long. In this worst case, the project could easily take over a year where a best case-scenario could be delivered at half the cost in a third of the time.

Overview:

Based on an estimated requirement of two relatively senior engineers over the course of 18 weeks at a rate of approximately 104k per person per year (I'm sure the cost of living in Marlborough, Massachusetts is significantly lower than my wage sample of convenience here in Fairfax County, Virginia and 104k per person rounds out to 2k a week), I'd put the wage costs alone at 36k. To mitigate risk of underbidding the contract and provide for a degree of profit, I'd round that up to 50k with an estimated time of delivery at the end of 6 months from the start date.

Appendix

Figure 1 (Best-case Scenario): Initial chart showing the earliest point at which tasks can start. It might be hard to see, but there's also dependency relationships depicted as links from one

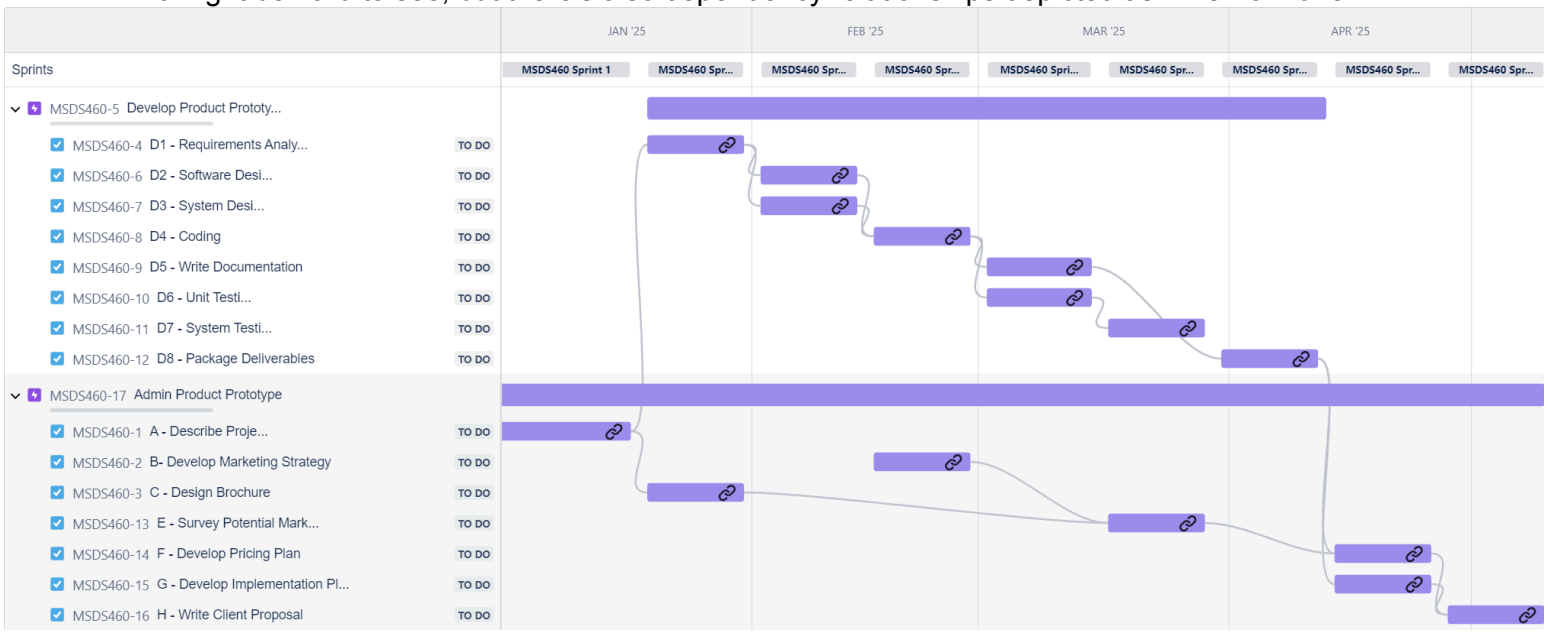

Figure 2 (Most Likely Scenario): This chart shows the tasks with the most simultaneous dependencies starting later than hoped for, pushing the last tasks and the schedule to the right.

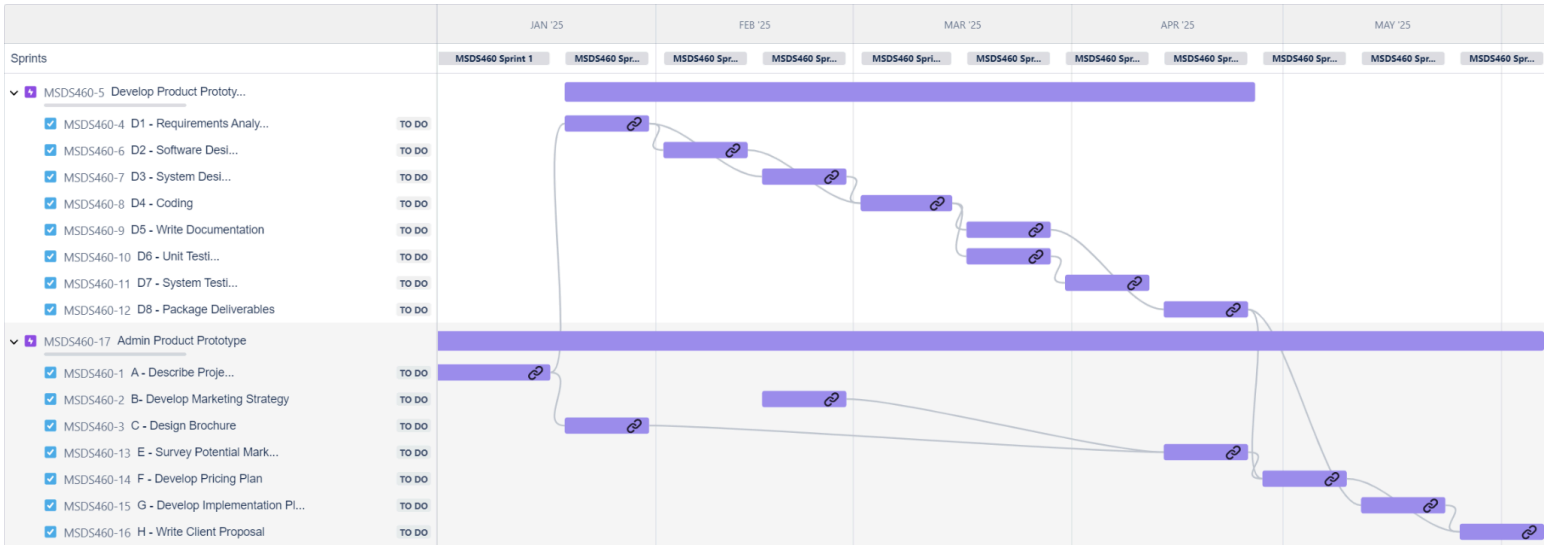

Figure 3 (Worst Case Scenario): This chart shows what happens when only one person can work on the various tasks at a time. Additionally, it's unlikely one person will have all the skills needed to complete the entirety of the project, indicating yet further delay. **Note:** The resultant Gantt chart was intractably illegible in a word document, but can be found in the git repo.