## Part 1: Problem setup

This element included planning and finding the critical path for the recommender system. There are a variety of files in this repository.

In the excel document linked titled project-plan.xlsx, the time estimates, individuals required for the work, and hourly cost per individual was estimated. This was the first step in the planning process. In the document, an x in the role column indicates that that individual is required to work on that task. All developers, the data scientist, and the engineer are needed for all of the development tasks. Additionally, some of the technical members of the team might be called in on implementation details and the pricing model. For example, if the pricing model is based on number of API calls or number of recommendations, the software engineers would need to build a mechanism to measure this in the software.

Hours per task were estimated based on experience and the expected time was calculated using the PERT framework (SOURCE). A better time estimate might be created based on the skill level of engineers and scope of the recommender system. There is always discovered work, and I think understanding where that might come from or the most likely pain points come mostly with additional professional experience. Additionally, geographical location greatly impacts hourly cost for the roles in the project

The directed graph diagram is attached in the pdf. Many of the tasks can be performed in parallel with the blue block of engineering tasks. They are appropriately lined up with the timeline to show which tasks can be done at the same time. For example, A and B could be completed at the same time. B and C can be completed at the same time. While B, E, C and F can not all be performed at the same time, they can all be performed while any D task is being performed.

The next part of the process included executing and running the code. This is in the jupyter notebook in the repository. This calculates the critical path based on three different time estimates. It assumes that the minimum time solution is also the minimum cost solution. The code is adapted from what was provided by Dr. Miller via Canvas. The code also saves the outputs of the time and the critical path to analyze the results in the Gantt Charts. Additionally, I adjust the time constraints into days in the code because that made the most sense in terms of mapping out the timeline for the project.

All three Gantt Charts are in Gantt_charts.xlsx in this repository. In order for these charts to make the most sense, instead of making one period an hour, I made one period 6 hours. I made the assumption that while we hope that we get 8 hours of work done a day, it is unlikely that no one at work eats, uses the bathroom, has a water cooler conversation, or runs into a separate issue.

## Part 2: Model specification.

The goal of the linear programming model is to minimize the total end time of the entire problem. The decision variables are the start and end time of each task. The constraints are as follows:

1. The end time for a task is equal to the start time of the task + time it takes to complete the task
2. The start time for each task is greater than or equal to the end time of all predecessor variables (variables that must be completed first as shown in the directed graph in the diagram in the repository)

The objective function is minimizing the end times for each task.

A note on how the application is formulated: after estimating times for each task for each individual, I divided the number of hours for the task by days and by the individuals working on that task so that the output minimizes days for the team to complete the task. When dividing by the number of people working on the task, I did look at each task and adjust the divisor if adding another individual would not significantly decrease the number of hours for the task for the other collaborator. For example, the project manager is responsible for developing the pricing plan, but some developers might be involved due to any technical needs for the price tracking. I did this to make it more applicable to recommending a timeline to another company because days are more digestible and informative than hours. Because there is no intermediate rounding, this does not impact the final path.

### Part 3:

See the code and implementation details in the file titled critical_path.ipynb.

### Part 4: Solution.

The results from the code are in the python file. The critical path in all three solutions is as follows:

A and B begin at the same time

C and D1 begin

D2 and D3 begin simultaneously

E begins

D4 begins

D5 and D6 begin simultaneously

D7 begins

D8 begins

F and G begin simultaneously

H begins

For the best case, the project would take almost 16 work days. The expected times result in a 24 day timeline and the worst case times lead to a 35 day timeline. These days are only business days and do not include weekends. Additionally, the solutions are not in integers, but I only rounded up because of the context of the problem and it simply doesn't make sense to give a timeline of 23.07 days.

See the three Gantt charts and in the gantt_charts.xlsx file to see in which order the the tasks finish. The gantt charts are rounded up to visually make the chart work. This also makes sense because if a task takes 2.02 days, it would roll over onto the second day. The gantt charts are organized such that 1 period is 1 work day.

## Part 5: Overview

It is estimated that the project will take 5 weeks. It could take up to 7 weeks given discovered work and unforeseen complications. The project will cost $65,000 for development fees and will include 6 months of free maintenance and bug fixes after deployment. I would expect a project prototype to be available between the 2nd and 3rd week of development and a fully tested prototype available by the end of the 4th week of development. If an additional developer was added, the project could be completed a few days earlier. An additional project manager would speed up production by almost a week. These estimates were built by adding up all of the development hours and adding an additional developer. The length of each task completed by the project manager would be halved with an additional project manager, decreasing the length of the total managerial tasks by approximately 8 days.

**Resources:**

Arc.dev. (n.d.). *Freelance developer rates: Backend*. Retrieved October 21, 2024, from https://arc.dev/freelance-developer-rates/back-end

CloudDevs. (n.d.). *Backend developer hourly rates*. Retrieved October 21, 2024, from https://clouddevs.com/backend/hourly-rates/

Fullstack. (2024). *2024 software developer price guide*. Retrieved October 21, 2024, from https://www.fullstack.com/labs/resources/blog/2024-price-guide

Smartworking. (n.d.). *Software developer hourly rates in the USA & UK*. Retrieved October 21, 2024, from https://smartworking.io/blog/software-developer-hourly-rates-in-the-usa-uk/

Wikipedia. (2024). *Program evaluation and review technique*. In Wikipedia. Retrieved October 21, 2024, from https://en.wikipedia.org/wiki/Program_evaluation_and_review_technique