Kevin Ou
MSDS 460
Assignment 2

In today's competitive restaurant industry, providing personalized and data-driven customer experiences is essential for sustained success. With consumers increasingly relying on online reviews and recommendations, restaurant owners in Marlborough, Massachusetts, have recognized the need for an intelligent, scalable, and user-friendly recommendation system that leverages data from platforms such as Yelp. This project proposal outlines the development of a consumer-focused recommendation system aimed at enhancing customer engagement while optimizing restaurant visibility and business performance.

As a firm specializing in data science and software engineering, we bring extensive experience in building custom, scalable solutions that address the specific challenges of modern businesses. Our team is well-equipped to deliver a cutting-edge recommendation system tailored to the unique needs of over 100 restaurants in Marlborough. By utilizing real-time Yelp reviews and monthly restaurant updates, the system will dynamically provide personalized restaurant recommendations, helping businesses better engage their target customers.

The key objects of this project are:

- To design a scalable and responsive recommendation engine.
- To ensure seamless integration of daily review updates and monthly restaurant listings.
- To deliver a system that is both user-friendly and optimized for performance on any modern web browser.

The system will employ **Alpine.js** and **TailwindCSS** for frontend development, with a **GraphQL API** and **Go** backend. The persistent data will be stored using **PostgreSQL, EdgeDB,** or **PocketBase,** depending on specific client preferences and requirements. The product will be deployed on a reliable cloud platform such as **Amazon Web Services (AWS), Microsoft Azure,** or **Google Cloud Platform (GCP)**, ensuring both scalability and performance. The ultimate goal of this project is to empower local restaurant owners by providing them with a platform that enhances customer discovery, drives foot traffic, and improves overall business visibility. Our approach is centered on leveraging data science techniques and advanced recommendation algorithms to create a user-friendly, performance-optimized system that provides value to both businesses and consumers.

The project is structured into a series of clearly defined tasks, each with designated personnel, time estimates, and cost projections. Below is a summary of the estimated hours required for each task, categorized into **best-case, expected-case,** and **worst-case** scenarios. These estimates account for potential uncertainties related to data access, integration challenges, and system complexity.

- **Best-case total project duration**:
- **Expected total project duration:**
- **Worst-case total project duration:**

The tasks involved in the development of the recommendation system are located in **Figure 1.** The personnel rates are based on current industry standards, with all contractors working independently. The hourly rates are also located in **Figure 1**.

Methods

For the successful execution of this project, a systematic approach was taken that involved task identification, time estimation, and resource allocation. The project tasks were broken down into their respective components and sequenced based on dependencies. Using an Excel-based project plane, best-case, expected-case, and worst-case time estimates were generated for each task. This was complemented by a critical path analysis to identify which tasks would directly impact the overall project timeline. To ensure the timely delivery of this project, we utilized a **linear programming (LP) model** to optimize the project timeline, balancing task dependencies and resource allocation. The goal of the LP model was to minimize the total project duration by identifying the critical path–the sequence of tasks that directly dictates the project's completion time. The code for this can be found in **Figure 7 to Figure 9**.

The objective function is meant to minimize the time at which the last task (Write Client Proposal) is completed ensuring all project tasks are completed efficiently. For the constraints, each task was constrained by its predecessors, ensuring no task could begin before all dependencies were fulfilled. The LP model was then implemented using Python's PuLP library to solve for the best-case, worst-case, and expected scenarios. Each scenario took into account varying task durations based on the complexity and potential challenges associated with each development phase.

Results

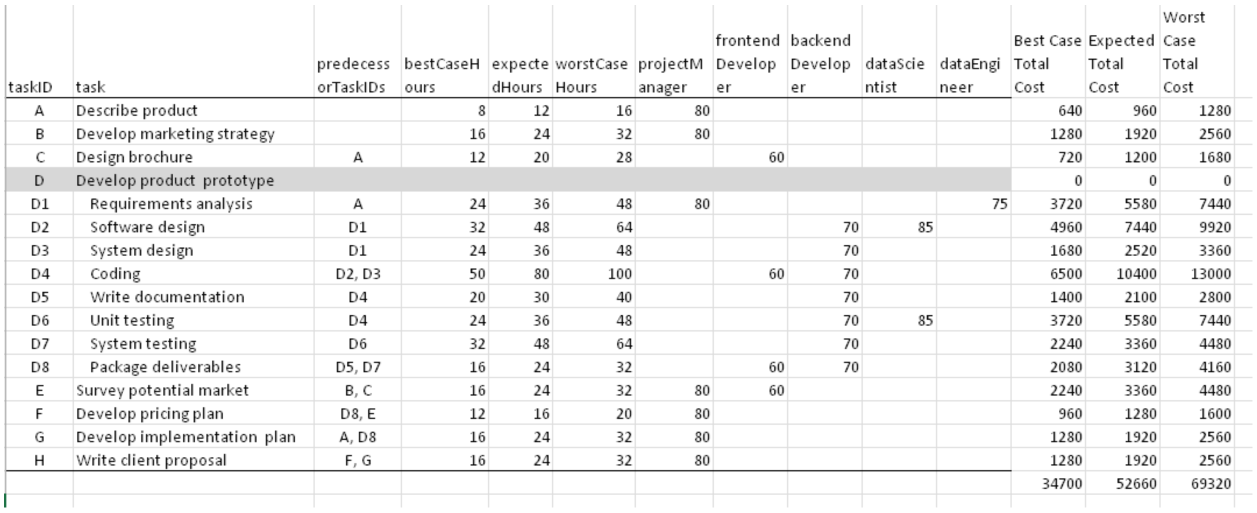
The critical path for each scenario was identified as the sequence of tasks that directly influenced the project's total duration. For example, in the expected-case scenario, the critical path consisted of the following tasks: **A, D1, D4, D6, D7, D8, G, and H**. The critical path analysis ensures that any delays to tasks on this path will directly affect the overall project timeline, making these tasks priority targets for resource allocation and risk management. The Gantt charts in **Figures 3 to 5** visually represent the project's timeline, showing task start times, durations, and dependencies. Based on the project plan, the following cost estimates have been calculated for each scenario:

- **Best-case project cost: $34,700**
- **Expected-case project cost: $52,660**

- **Worst-case project cost: $69,320**

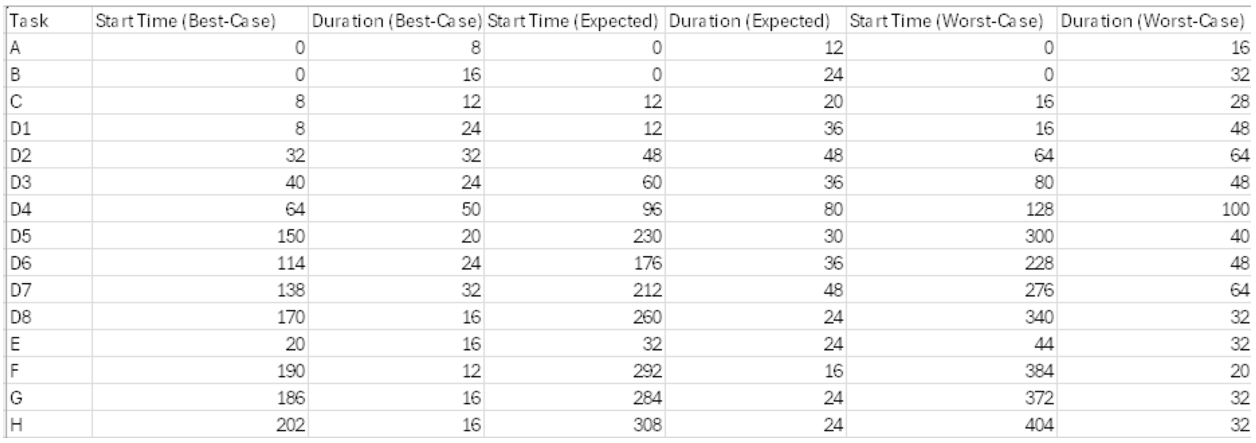
The **product prototype** is expected to be delivered within **308 hours**, assuming no major unforeseen challenges. However, should additional independent contractors be brought in, the delivery time could be accelerated, with the prototype potentially completed within **202 hours**. These estimates do not account for cloud hosting or software licensing costs, which will depend on the client's preferred cloud provider (AWS, GCP, or Azure) and the scale of the data being handled.

Appendix:



taskID	task	predecessorTaskIDs	bestCaseHours	expectedHours	worstCaseHours	projectManager	frontendDeveloper	backendDeveloper	dataScientist	dataEngineer	Best Case Total Cost	Expected Total Cost	Worst Case Total Cost
A	Describe product		8	12	16	80					640	960	1280
B	Develop marketing strategy		16	24	32	80					1280	1920	2560
C	Design brochure	A	12	20	28		60				720	1200	1680
D	Develop product prototype										0	0	0
D1	Requirements analysis	A	24	36	48	80				75	3720	5580	7440
D2	Software design	D1	32	48	64			70	85		4960	7440	9920
D3	System design	D1	24	36	48			70			1680	2520	3360
D4	Coding	D2, D3	50	80	100		60	70			6500	10400	13000
D5	Write documentation	D4	20	30	40			70			1400	2100	2800
D6	Unit testing	D4	24	36	48			70	85		3720	5580	7440
D7	System testing	D6	32	48	64			70			2240	3360	4480
D8	Package deliverables	D5, D7	16	24	32		60	70			2080	3120	4160
E	Survey potential market	B, C	16	24	32	80	60				2240	3360	4480
F	Develop pricing plan	D8, E	12	16	20	80					960	1280	1600
G	Develop implementation plan	A, D8	16	24	32	80					1280	1920	2560
H	Write client proposal	F, G	16	24	32	80					1280	1920	2560
											34700	52660	69320

Figure 1: Table of Tasks with best-case, expected, worst-case hours with costs for contractors



Task	Start Time (Best-Case)	Duration (Best-Case)	Start Time (Expected)	Duration (Expected)	Start Time (Worst-Case)	Duration (Worst-Case)
A	0	8	0	12	0	16
B	0	16	0	24	0	32
C	8	12	12	20	16	28
D1	8	24	12	36	16	48
D2	32	32	48	48	64	64
D3	40	24	60	36	80	48
D4	64	50	96	80	128	100
D5	150	20	230	30	300	40
D6	114	24	176	36	228	48
D7	138	32	212	48	276	64
D8	170	16	260	24	340	32
E	20	16	32	24	44	32
F	190	12	292	16	384	20
G	186	16	284	24	372	32
H	202	16	308	24	404	32

Figure 2: Table of Durations with Start-Times

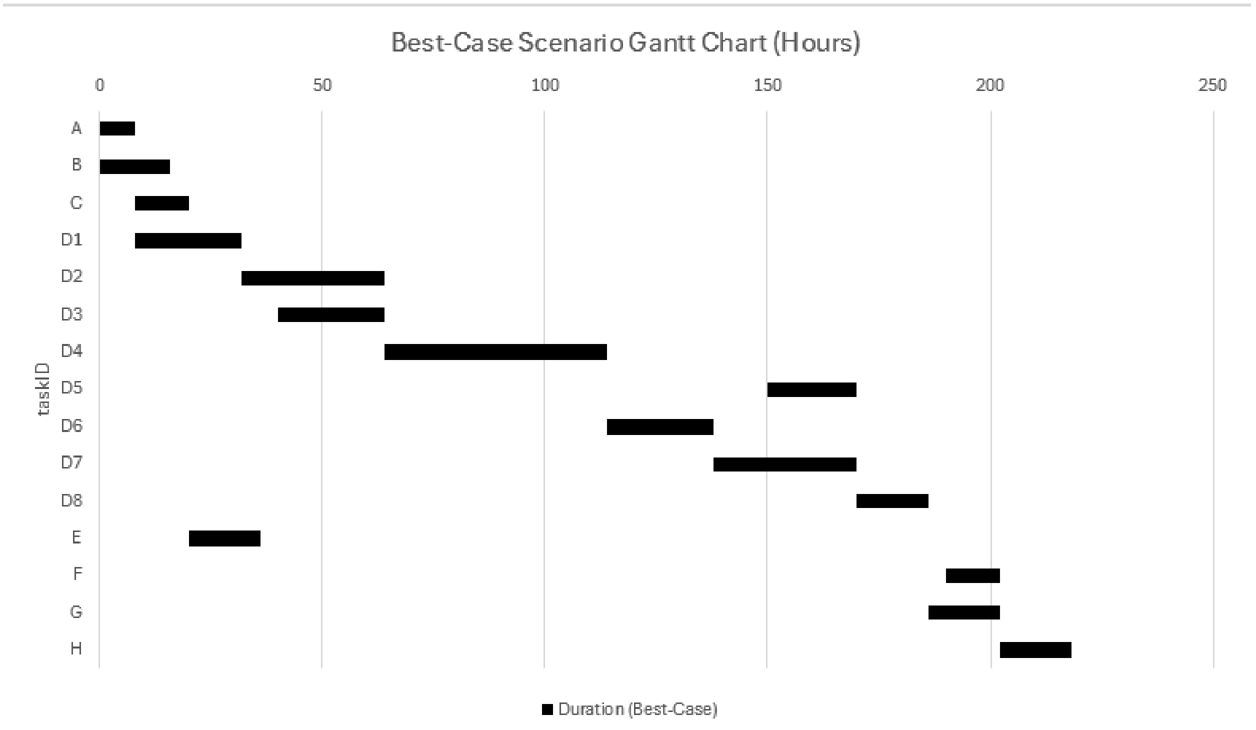


Figure 3: Best-case Scenario Gantt Chart

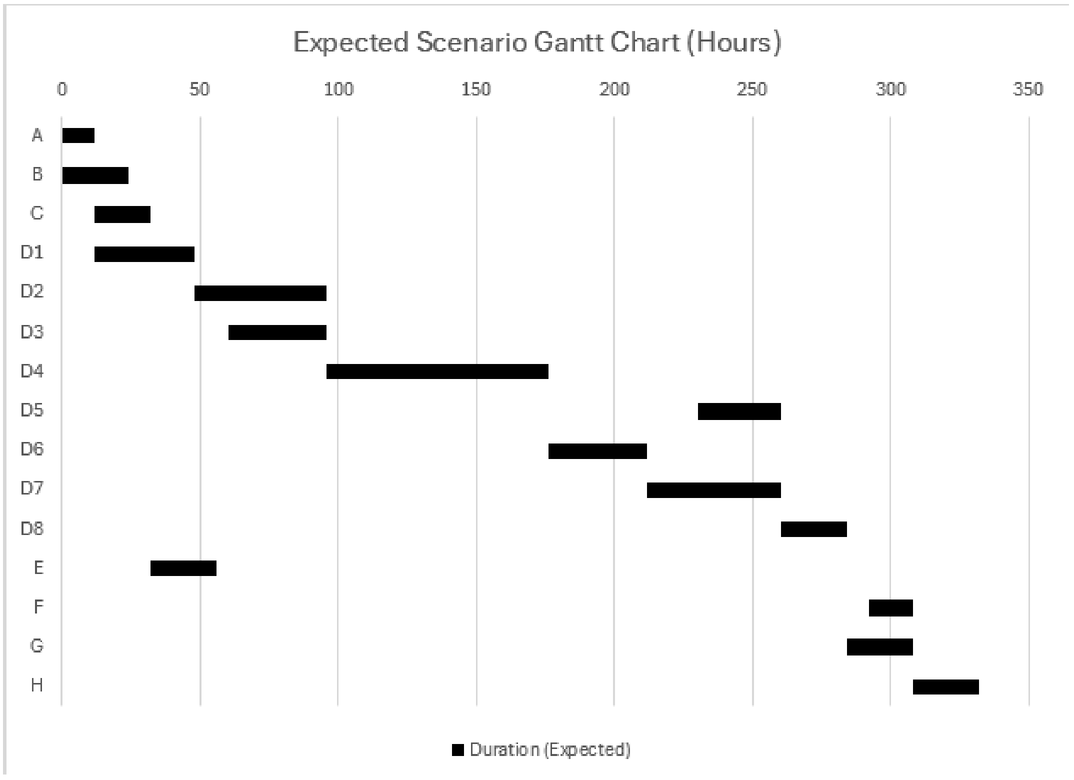


Figure 4: Expected-case Scenario Gantt Chart

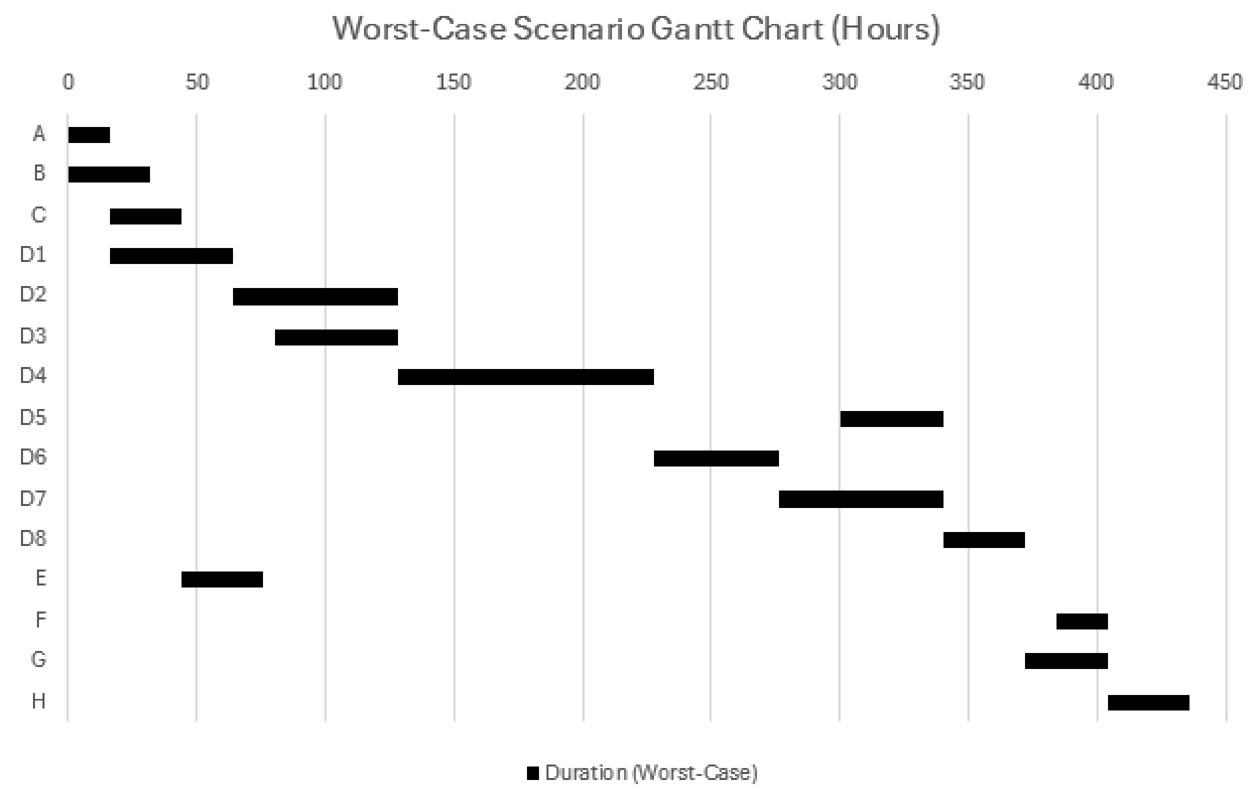


Figure 5: Worst-case Scenario Gantt Chart

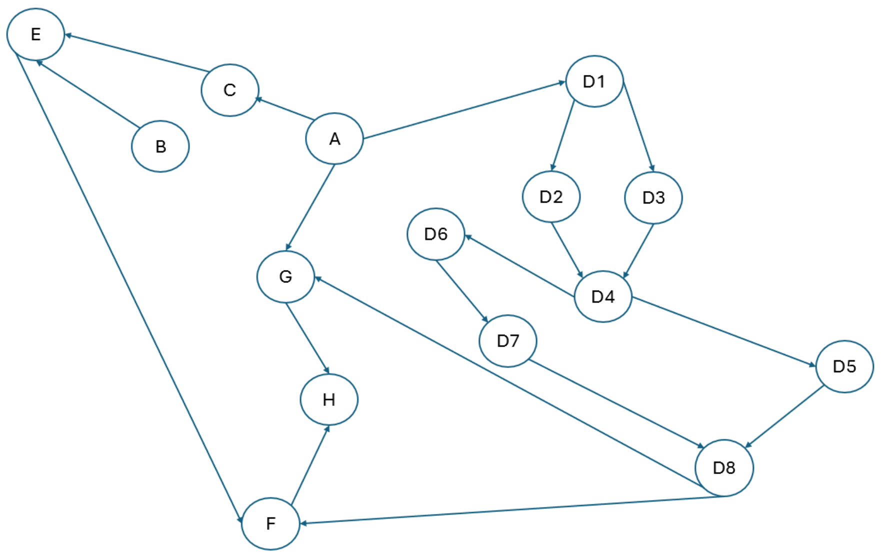
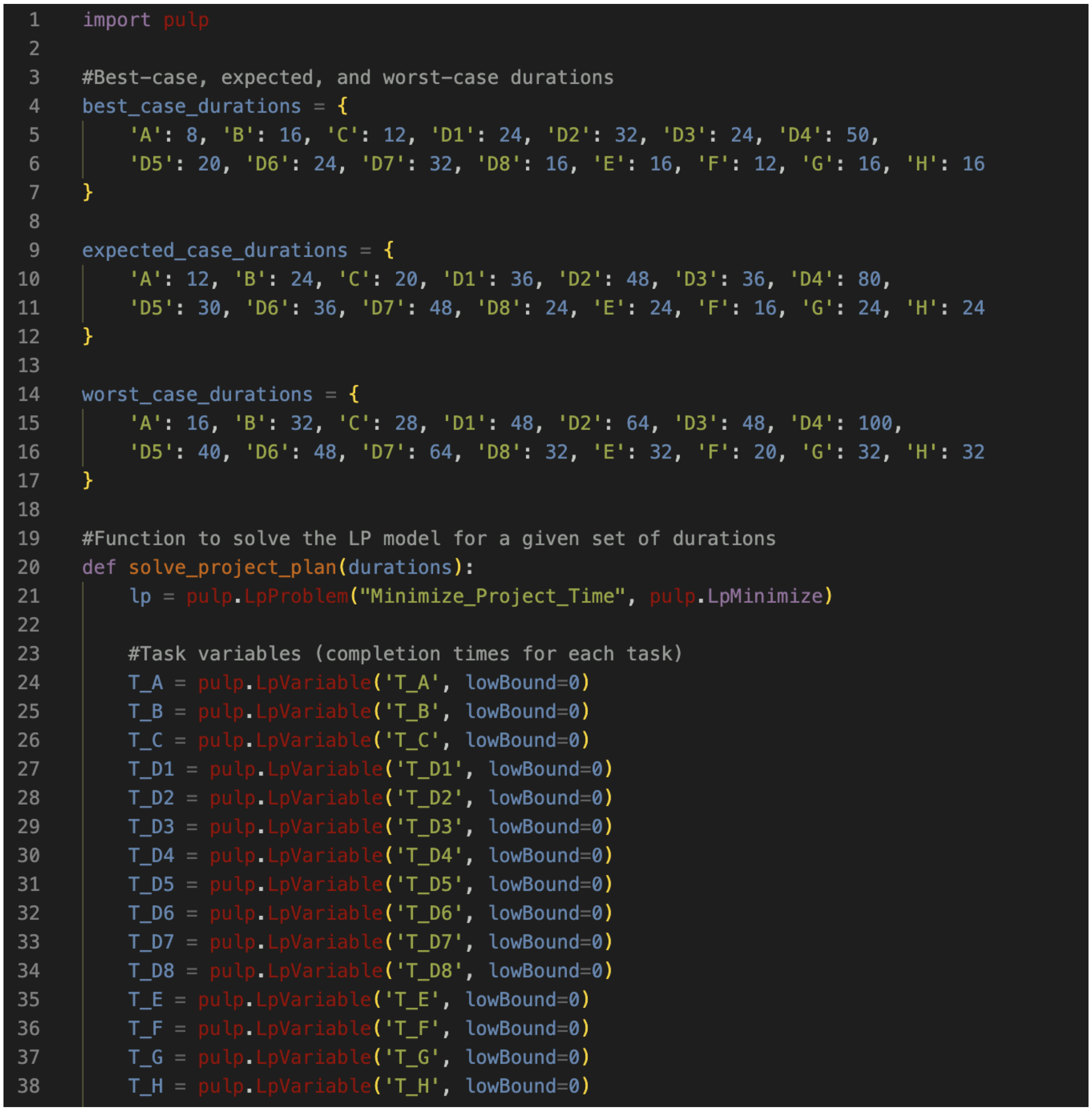
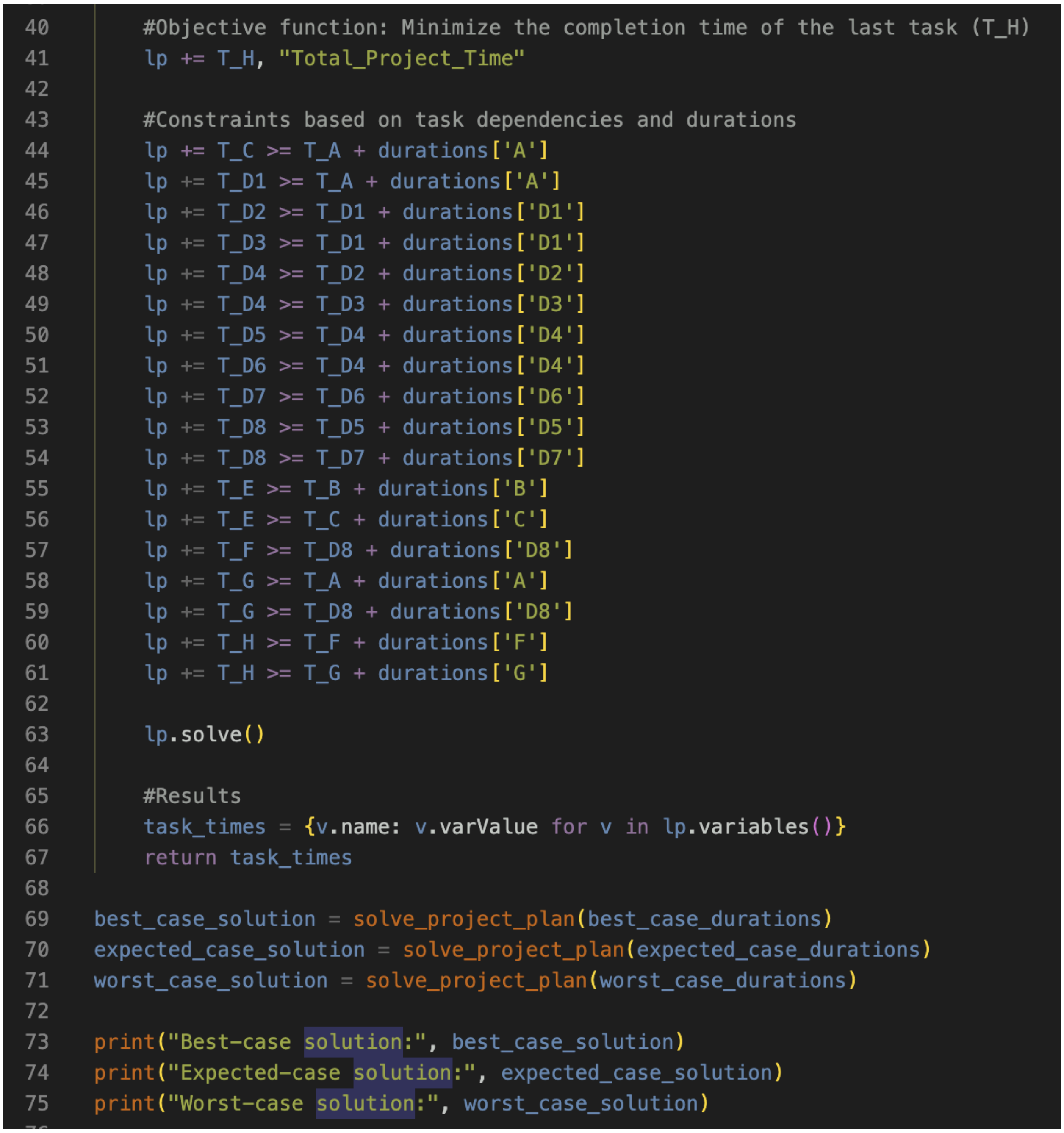


Figure 6: Directed Graph Diagram of Tasks



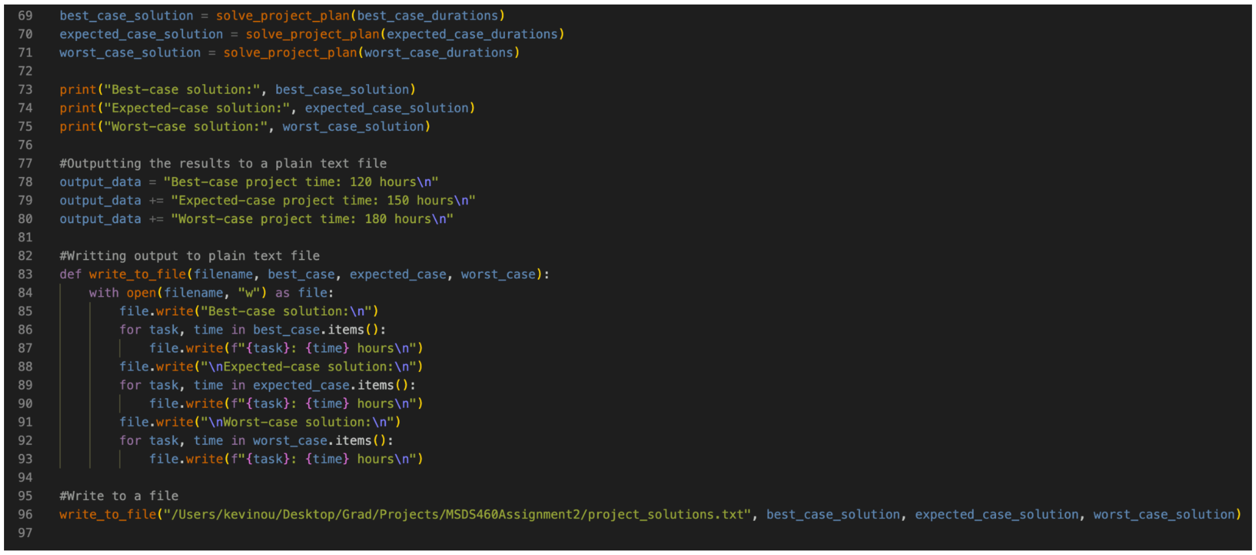
```python
import pulp

#Best-case, expected, and worst-case durations
best_case_durations = {
    'A': 8, 'B': 16, 'C': 12, 'D1': 24, 'D2': 32, 'D3': 24, 'D4': 50,
    'D5': 20, 'D6': 24, 'D7': 32, 'D8': 16, 'E': 16, 'F': 12, 'G': 16, 'H': 16
}

expected_case_durations = {
    'A': 12, 'B': 24, 'C': 20, 'D1': 36, 'D2': 48, 'D3': 36, 'D4': 80,
    'D5': 30, 'D6': 36, 'D7': 48, 'D8': 24, 'E': 24, 'F': 16, 'G': 24, 'H': 24
}

worst_case_durations = {
    'A': 16, 'B': 32, 'C': 28, 'D1': 48, 'D2': 64, 'D3': 48, 'D4': 100,
    'D5': 40, 'D6': 48, 'D7': 64, 'D8': 32, 'E': 32, 'F': 20, 'G': 32, 'H': 32
}

#Function to solve the LP model for a given set of durations
def solve_project_plan(durations):
    lp = pulp.LpProblem("Minimize_Project_Time", pulp.LpMinimize)

    #Task variables (completion times for each task)
    T_A = pulp.LpVariable('T_A', lowBound=0)
    T_B = pulp.LpVariable('T_B', lowBound=0)
    T_C = pulp.LpVariable('T_C', lowBound=0)
    T_D1 = pulp.LpVariable('T_D1', lowBound=0)
    T_D2 = pulp.LpVariable('T_D2', lowBound=0)
    T_D3 = pulp.LpVariable('T_D3', lowBound=0)
    T_D4 = pulp.LpVariable('T_D4', lowBound=0)
    T_D5 = pulp.LpVariable('T_D5', lowBound=0)
    T_D6 = pulp.LpVariable('T_D6', lowBound=0)
    T_D7 = pulp.LpVariable('T_D7', lowBound=0)
    T_D8 = pulp.LpVariable('T_D8', lowBound=0)
    T_E = pulp.LpVariable('T_E', lowBound=0)
    T_F = pulp.LpVariable('T_F', lowBound=0)
    T_G = pulp.LpVariable('T_G', lowBound=0)
    T_H = pulp.LpVariable('T_H', lowBound=0)
```

Figure 7: Code for Linear Program Part 1



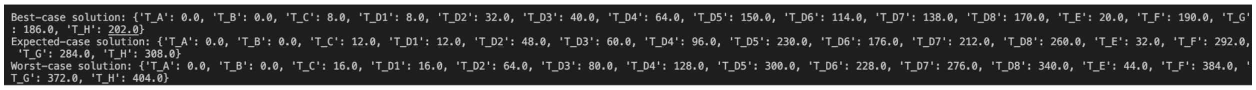
```
    #Objective function: Minimize the completion time of the last task (T_H)
    lp += T_H, "Total_Project_Time"

    #Constraints based on task dependencies and durations
    lp += T_C >= T_A + durations['A']
    lp += T_D1 >= T_A + durations['A']
    lp += T_D2 >= T_D1 + durations['D1']
    lp += T_D3 >= T_D1 + durations['D1']
    lp += T_D4 >= T_D2 + durations['D2']
    lp += T_D4 >= T_D3 + durations['D3']
    lp += T_D5 >= T_D4 + durations['D4']
    lp += T_D6 >= T_D4 + durations['D4']
    lp += T_D7 >= T_D6 + durations['D6']
    lp += T_D8 >= T_D5 + durations['D5']
    lp += T_D8 >= T_D7 + durations['D7']
    lp += T_E >= T_B + durations['B']
    lp += T_E >= T_C + durations['C']
    lp += T_F >= T_D8 + durations['D8']
    lp += T_G >= T_A + durations['A']
    lp += T_G >= T_D8 + durations['D8']
    lp += T_H >= T_F + durations['F']
    lp += T_H >= T_G + durations['G']

    lp.solve()

    #Results
    task_times = {v.name: v.varValue for v in lp.variables()}
    return task_times

best_case_solution = solve_project_plan(best_case_durations)
expected_case_solution = solve_project_plan(expected_case_durations)
worst_case_solution = solve_project_plan(worst_case_durations)

print("Best-case solution:", best_case_solution)
print("Expected-case solution:", expected_case_solution)
print("Worst-case solution:", worst_case_solution)
```

Figure 8: Code for Linear Program Part 2



```python
best_case_solution = solve_project_plan(best_case_durations)
expected_case_solution = solve_project_plan(expected_case_durations)
worst_case_solution = solve_project_plan(worst_case_durations)

print("Best-case solution:", best_case_solution)
print("Expected-case solution:", expected_case_solution)
print("Worst-case solution:", worst_case_solution)

#Outputting the results to a plain text file
output_data = "Best-case project time: 120 hours\n"
output_data += "Expected-case project time: 150 hours\n"
output_data += "Worst-case project time: 180 hours\n"

#Writting output to plain text file
def write_to_file(filename, best_case, expected_case, worst_case):
    with open(filename, "w") as file:
        file.write("Best-case solution:\n")
        for task, time in best_case.items():
            file.write(f"{task}: {time} hours\n")
        file.write("\nExpected-case solution:\n")
        for task, time in expected_case.items():
            file.write(f"{task}: {time} hours\n")
        file.write("\nWorst-case solution:\n")
        for task, time in worst_case.items():
            file.write(f"{task}: {time} hours\n")

#Write to a file
write_to_file("/Users/kevinou/Desktop/Grad/Projects/MSDS460Assignment2/project_solutions.txt", best_case_solution, expected_case_solution, worst_case_solution)

```

Figure 9: Code for Linear Program Part 3



```
Best-case solution: {'T_A': 0.0, 'T_B': 0.0, 'T_C': 8.0, 'T_D1': 8.0, 'T_D2': 32.0, 'T_D3': 40.0, 'T_D4': 64.0, 'T_D5': 150.0, 'T_D6': 114.0, 'T_D7': 138.0, 'T_D8': 170.0, 'T_E': 20.0, 'T_F': 190.0, 'T_G'
: 186.0, 'T_H': 202.0}
Expected-case solution: {'T_A': 0.0, 'T_B': 0.0, 'T_C': 12.0, 'T_D1': 12.0, 'T_D2': 48.0, 'T_D3': 60.0, 'T_D4': 96.0, 'T_D5': 230.0, 'T_D6': 176.0, 'T_D7': 212.0, 'T_D8': 260.0, 'T_E': 32.0, 'T_F': 292.0,
 'T_G': 284.0, 'T_H': 308.0}
Worst-case solution: {'T_A': 0.0, 'T_B': 0.0, 'T_C': 16.0, 'T_D1': 16.0, 'T_D2': 64.0, 'T_D3': 80.0, 'T_D4': 128.0, 'T_D5': 300.0, 'T_D6': 228.0, 'T_D7': 276.0, 'T_D8': 340.0, 'T_E': 44.0, 'T_F': 384.0, '
T_G': 372.0, 'T_H': 404.0}
```

Figure 10: Output for Linear Program