In starting a data science and software engineering firm, I received a request from a group of restaurant owners in Marlborough, Massachusetts. They are requesting that I provide them with a proposal for a new software product including a best-case scenario, target scenario, and worst-case scenario for time to build and costs. This project will be building a consumer recommendation system for more than one hundred restaurants in the area consisting of Yelp reviews through Yelp's GraphQL API. The recommendation system will be implemented using Alpine.js, Tailwind, Python as the analytical backend, and PostgreSQL. It should be accessible from any web browser and hosted on a major cloud platform, in our case Microsoft Azure. Such a large project will require contributions from a front-end developer, a back-end developer, a data scientist, a data engineer and a project manager. Having worked at a company where three people have tackled all of these aspects of a project, we are going to limit each role to a single person and try to tackle this endeavor as quickly as possible.

In setting up the problem, we created an excel sheet that broke down all the tasks into fifteen components (Appendix I). This sheet provides information on the best-case scenarios, worst-case scenarios, and expected scenarios for hours and which employees will need to work on each component. After analyzing the individual components, components that required prior components to be completed were marked so we could adequately build a network graph (Appendix II). The network graph helped us visualize which tasks could be performed in parallel to tackle the project most efficiently. We were able to negotiate a flat rate of $75/hour for each worker due to the lack of benefits provided and the specialization that each task required.

Through the excel spreadsheet and the network graphs, we were able to surmise that many tasks were able to be run in parallel without straining our resources. For example, in developing the product prototype, we split up the tasks to ensure that System Design and

Software Design could be performed in parallel, Unit testing and System Testing could be run in parallel with Writing the Documentation, then allowing everyone to work on Packaging the Deliverables together.

We built a linear programming model using Python and PuLP to determine the best-case scenario, expected scenario, and worst-case scenario for total hours worked (Appendix III). The outcomes of each of these models in total time, broken down by task, was calculated by the software (Appendix IV). This model, coupled with the excel table for total hours worked per employee allowed each of the scenarios to be displayed in tabular form for easier viewing (Appendix V).

Based on the results of the linear programming model, there is a wide range of time dedication needed and cost to building the requested software product. In the best-case scenario, 49 hours are needed from start to finish, requiring less than 7 business days in total. The total cost of the expenditure is $15,000 in total labor, not including storage space or website hosting money, both of which will require a monthly cost likely less than $100 a month based on current cost estimates. The expected cost, based on the expected number of hours for each task brings a total of $35,100, not including monthly hosting. This expected case will be completed in 114 hours for a total of just under 15 business days. The worst-case scenario estimates lead to a cost of $75,000, not including monthly hosting. This worst-case scenario will be completed in 244 hours, or just under 31 business days.

It’s unlikely that any of these estimates will pan out exactly, so the result of building this software product will be somewhere between $15,000 and $75,000 in total costs, with an added expenditure of around $100 per month to host the service. While it pains us to provide such a wide range of initial cost expenditures, providing an honest and upfront range is something we

pride ourselves on as a company and we would be happy to discuss any issues you may have with the pricing. With development, issues arise that can be unforeseen and while we hope to provide a service in the least amount of time possible, we strive to be honest and upfront with our estimates to our customers.

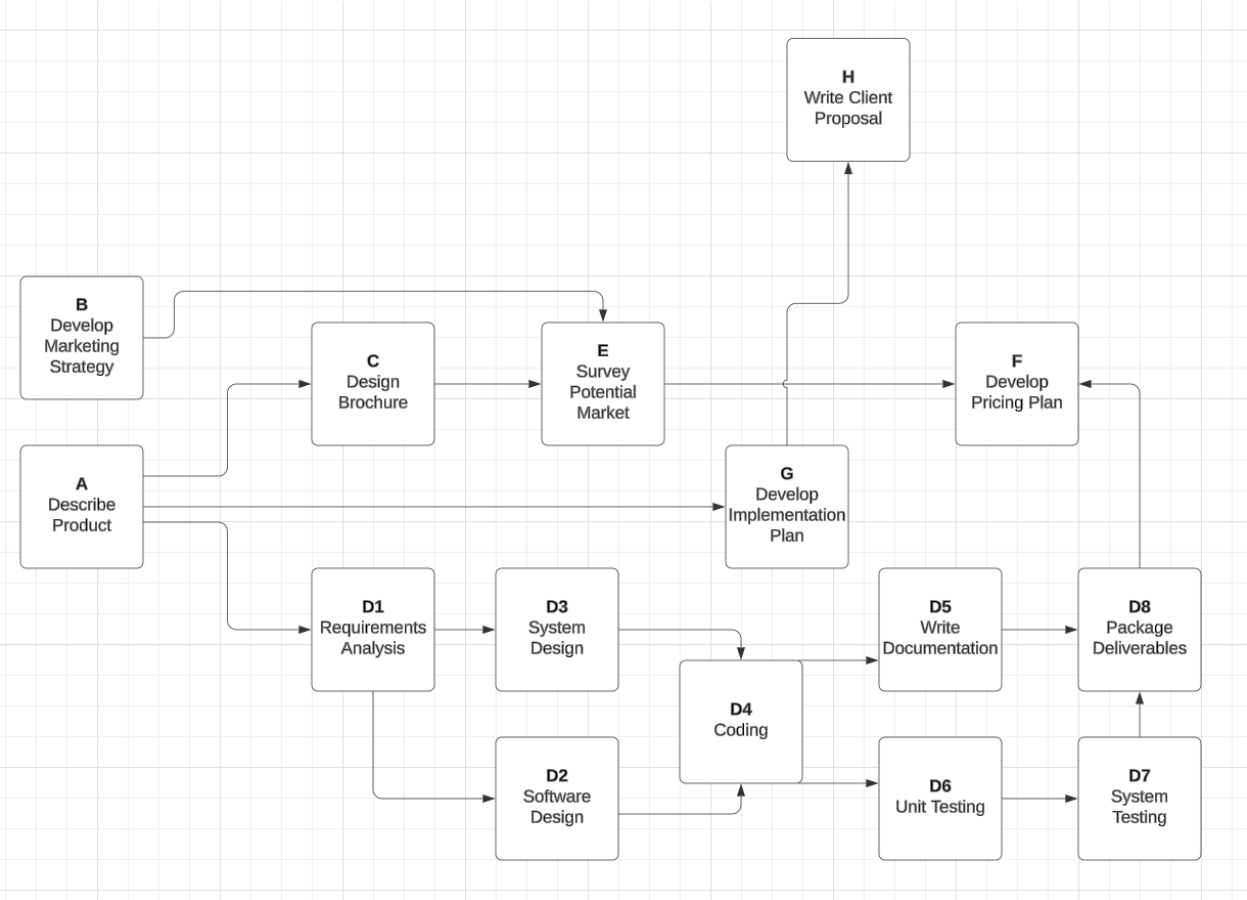
Given all of the information that was presented in the project proposal, I would offer two solutions to building the requested software. I would offer a best estimate plus twenty percent, totaling $42,120 for the entire project or else I would suggest that they pay for whatever it takes in our hourly rates, somewhere between $15,000 and $75,000. Given that reputation plays a big part in getting clients, I would feel comfortable betting that we could provide the service at less than the cost of $42,120 if they chose our hourly rates, but also understand that the client may need to be able to budget out projects and a range of $60,000 might be tougher to swallow than a guaranteed payment of $42,120. Having experience in much of the leg work that makes up the project, I understand the amount of work needed and the issues that can arise when developing such projects, so either choice by the client would be understandable.

If additional independent contractors were added to the mix, more capital would be needed to cover the expenditures but we could release the product more efficiently. If time is the biggest concern for the client and they were willing to allow more developers for the product design, I would feel comfortable guaranteeing the project in two weeks. The best-case scenario would have us delivering the product within seven business days, the expected scenario would have us delivering the product within three business weeks, and the worst case scenario would have us delivering the product within 31 business days or just over six business weeks. Given the need for a faster timeline, I am confident that we could do the entire project within three weeks time if we were given the proper capital to ensure maximum efficiency.

APPENDIX I: Project Plan Table

taskID	task	predecessorTaskIDs	bestCaseHours	expectedHours	worstCaseHours	projectManager	frontendDeveloper	backendDeveloper	dataScientist	dataEngineer
A	Describe product		2	4	8	1				
B	Develop marketing strategy		4	8	16	1	1			
C	Design brochure	A	4	8	16	1	1			
D	Develop product prototype									
D1	Requirements analysis	A	2	4	6		1	1	1	1
D2	Software design	D1	4	8	16		1	1		
D3	System design	D1	4	8	16	1			1	1
D4	Coding	D2, D3	16	40	80		1	1	1	1
D5	Write documentation	D4	8	20	40		1	1		
D6	Unit testing	D4	4	12	40		1	1	1	1
D7	System testing	D6	4	12	40				1	1
D8	Package deliverables	D5, D7	2	4	8		1	1	1	1
E	Survey potential market	B, C	8	20	40	1				
F	Develop pricing plan	D8, E	2	4	16	1				
G	Develop implementation plan	A, D8	8	16	24	1			1	1
H	Write client proposal	F, G	8	16	24	1				

APPENDIX II: Network Model



APPENDIX III: Linear Programming Model

```
from pulp import *

scenarios = {0: "Best Case Scenario", 1: "Expected Scenario", 2: "Worst Case
Scenario"}


for scenario in range(0, 3):
    # Create a dictionary of the activities and their durations
    activities = {
        "DescribeProduct": [2, 4, 8],
        "DevelopMarketingStrategy": [4, 8, 16],
        "DesignBrochure": [4, 8, 16],
        "RequirementsAnalysis": [2, 4, 6],
        "SoftwareDesign": [4, 8, 16],
        "SystemDesign": [4, 8, 16],
        "Coding": [16, 40, 80],
        "WriteDocumentation": [8, 20, 40],
        "UnitTesting": [4, 12, 40],
        "SystemTesting": [4, 12, 40],
        "PackageDeliverables": [2, 4, 8],
        "SurveyPotentialMarket": [8, 20, 40],
        "DevelopPricingPlan": [2, 4, 16],
        "DevelopImplementationPlan": [8, 16, 24],
        "WriteClientProposal": [8, 16, 24],
    }

    # Create a list of the activities
    activities_list = list(activities.keys())

    # Create a dictionary of the activity precedences
    precedences = {
        "DescribeProduct": [],
        "DevelopMarketingStrategy": [],
        "DesignBrochure": ["DescribeProduct"],
        "RequirementsAnalysis": ["DescribeProduct"],
        "SoftwareDesign": ["RequirementsAnalysis"],
        "SystemDesign": ["RequirementsAnalysis"],
        "Coding": ["SoftwareDesign", "SystemDesign"],
        "WriteDocumentation": ["Coding"],
        "UnitTesting": ["Coding"],
        "SystemTesting": ["UnitTesting"],
        "PackageDeliverables": ["WriteDocumentation", "SystemTesting"],
        "SurveyPotentialMarket": ["DevelopMarketingStrategy", "DesignBrochure"],
```

```
        "DevelopPricingPlan": ["PackageDeliverables", "SurveyPotentialMarket"],
        "DevelopImplementationPlan": ["DescribeProduct", "PackageDeliverables"],
        "WriteClientProposal": ["DevelopPricingPlan",
"DevelopImplementationPlan"],
    }

    # Create the LP problem
    prob = LpProblem("Critical Path", LpMinimize)

    # Create the LP variables
    start_times = {
        activity: LpVariable(f"start_{activity}", 0, None)
        for activity in activities_list
    }
    end_times = {
        activity: LpVariable(f"end_{activity}", 0, None) for activity in
activities_list
    }

    # Add the constraints
    for activity in activities_list:
        prob += (
            end_times[activity]
            == start_times[activity] + activities[activity][scenario],
            f"{activity}_duration",
        )
        for predecessor in precedences[activity]:
            prob += (
                start_times[activity] >= end_times[predecessor],
                f"{activity}_predecessor_{predecessor}",
            )

    # Set the objective function
    prob += (
        lpSum([end_times[activity] for activity in activities_list]),
        "minimize_end_times",
    )

    # Solve the LP problem
    status = prob.solve()

    # Print the results
    print(f"Critical Path time for {scenarios[scenario]}:")
    for activity in activities_list:
        if value(start_times[activity]) == 0:
```

```
            print(f"{activity} starts at time 0")
        if value(end_times[activity]) == max(
            [value(end_times[activity]) for activity in activities_list]
        ):
            print(f"{activity} ends at {value(end_times[activity])} hours in
duration")

    # Print solution
    print("\nSolution variable values:")
    for var in prob.variables():
        if var.name != "_dummy":
            print(var.name, "=", var.varValue)
```

APPENDIX IV: Linear Programming Model Output

```
C:\Users\micha\anaconda3\envs\MSDS460\Lib\site-packages\pulp\pulp.py:1316: UserWarning:
Spaces are not permitted in the name. Converted to '_'
  warnings.warn("Spaces are not permitted in the name. Converted to '_'")
Welcome to the CBC MILP Solver
Version: 2.10.3
Build Date: Dec 15 2019

command line - C:\Users\micha\anaconda3\envs\MSDS460\Lib\site-
packages\pulp\solverdir\cbc\win\64\cbc.exe
C:\Users\micha\AppData\Local\Temp\f62eee1f64a64e4682353dccafd0234c-pulp.mps -
timeMode elapsed -branch -printingOptions all -solution
C:\Users\micha\AppData\Local\Temp\f62eee1f64a64e4682353dccafd0234c-pulp.sol (default
strategy 1)
At line 2 NAME          MODEL
At line 3 ROWS
At line 39 COLUMNS
At line 123 RHS
At line 158 BOUNDS
At line 159 ENDATA
Problem MODEL has 34 rows, 30 columns and 68 elements
Coin0008I MODEL read with 0 errors
Option for timeMode changed from cpu to elapsed
Presolve 0 (-34) rows, 0 (-30) columns and 0 (-68) elements
Empty problem - 0 rows, 0 columns and 0 elements
Optimal - objective value 324
After Postsolve, objective 324, infeasibilities - dual 0 (0), primal 0 (0)
Optimal objective 324 - 0 iterations time 0.002, Presolve 0.00
Option for printingOptions changed from normal to all
Total time (CPU seconds):       0.01   (Wallclock seconds):       0.01

Critical Path time for Best Case Scenario:
DescribeProduct starts at time 0
DevelopMarketingStrategy starts at time 0
WriteClientProposal ends at 50.0 hours in duration

Solution variable values:
end_Coding = 24.0
end_DescribeProduct = 2.0
end_DesignBrochure = 6.0
end_DevelopImplementationPlan = 42.0
end_DevelopMarketingStrategy = 4.0
end_DevelopPricingPlan = 36.0
end_PackageDeliverables = 34.0
end_RequirementsAnalysis = 4.0
end_SoftwareDesign = 8.0
```

end_SurveyPotentialMarket = 14.0
end_SystemDesign = 8.0
end_SystemTesting = 32.0
end_UnitTesting = 28.0
end_WriteClientProposal = 50.0
end_WriteDocumentation = 32.0
start_Coding = 8.0
start_DescribeProduct = 0.0
start_DesignBrochure = 2.0
start_DevelopImplementationPlan = 34.0
start_DevelopMarketingStrategy = 0.0
start_DevelopPricingPlan = 34.0
start_PackageDeliverables = 32.0
start_RequirementsAnalysis = 2.0
start_SoftwareDesign = 4.0
start_SurveyPotentialMarket = 6.0
start_SystemDesign = 4.0
start_SystemTesting = 28.0
start_UnitTesting = 24.0
start_WriteClientProposal = 42.0
start_WriteDocumentation = 24.0
Welcome to the CBC MILP Solver
Version: 2.10.3
Build Date: Dec 15 2019

command line - C:\Users\micha\anaconda3\envs\MSDS460\Lib\site-packages\pulp\solverdir\cbc\win\64\cbc.exe C:\Users\micha\AppData\Local\Temp\ac9d2567e7584ac68988448e475d6ed3-pulp.mps -timeMode elapsed -branch -printingOptions all -solution C:\Users\micha\AppData\Local\Temp\ac9d2567e7584ac68988448e475d6ed3-pulp.sol (default strategy 1)
At line 2 NAME MODEL
At line 3 ROWS
At line 39 COLUMNS
At line 123 RHS
At line 158 BOUNDS
At line 159 ENDATA
Problem MODEL has 34 rows, 30 columns and 68 elements
Coin0008I MODEL read with 0 errors
Option for timeMode changed from cpu to elapsed
Presolve 0 (-34) rows, 0 (-30) columns and 0 (-68) elements
Empty problem - 0 rows, 0 columns and 0 elements
Optimal - objective value 764
After Postsolve, objective 764, infeasibilities - dual 0 (0), primal 0 (0)
Optimal objective 764 - 0 iterations time 0.002, Presolve 0.00
Option for printingOptions changed from normal to all

Total time (CPU seconds): 0.01 (Wallclock seconds): 0.01

Critical Path time for Expected Scenario:
DescribeProduct starts at time 0
DevelopMarketingStrategy starts at time 0
WriteClientProposal ends at 116.0 hours in duration

Solution variable values:
end_Coding = 56.0
end_DescribeProduct = 4.0
end_DesignBrochure = 12.0
end_DevelopImplementationPlan = 100.0
end_DevelopMarketingStrategy = 8.0
end_DevelopPricingPlan = 88.0
end_PackageDeliverables = 84.0
end_RequirementsAnalysis = 8.0
end_SoftwareDesign = 16.0
end_SurveyPotentialMarket = 32.0
end_SystemDesign = 16.0
end_SystemTesting = 80.0
end_UnitTesting = 68.0
end_WriteClientProposal = 116.0
end_WriteDocumentation = 76.0
start_Coding = 16.0
start_DescribeProduct = 0.0
start_DesignBrochure = 4.0
start_DevelopImplementationPlan = 84.0
start_DevelopMarketingStrategy = 0.0
start_DevelopPricingPlan = 84.0
start_PackageDeliverables = 80.0
start_RequirementsAnalysis = 4.0
start_SoftwareDesign = 8.0
start_SurveyPotentialMarket = 12.0
start_SystemDesign = 8.0
start_SystemTesting = 68.0
start_UnitTesting = 56.0
start_WriteClientProposal = 100.0
start_WriteDocumentation = 56.0
Welcome to the CBC MILP Solver
Version: 2.10.3
Build Date: Dec 15 2019

command line - C:\Users\micha\anaconda3\envs\MSDS460\Lib\site-packages\pulp\solverdir\cbc\win\64\cbc.exe C:\Users\micha\AppData\Local\Temp\46ffa2dd12074dd5a0638e06ed843ca0-pulp.mps -timeMode elapsed -branch -printingOptions all -solution

C:\Users\micha\AppData\Local\Temp\46ffa2dd12074dd5a0638e06ed843ca0-pulp.sol (default strategy 1)
At line 2 NAME MODEL
At line 3 ROWS
At line 39 COLUMNS
At line 123 RHS
At line 158 BOUNDS
At line 159 ENDATA
Problem MODEL has 34 rows, 30 columns and 68 elements
Coin0008I MODEL read with 0 errors
Option for timeMode changed from cpu to elapsed
Presolve 0 (-34) rows, 0 (-30) columns and 0 (-68) elements
Empty problem - 0 rows, 0 columns and 0 elements
Optimal - objective value 1666
After Postsolve, objective 1666, infeasibilities - dual 0 (0), primal 0 (0)
Optimal objective 1666 - 0 iterations time 0.002, Presolve 0.00
Option for printingOptions changed from normal to all
Total time (CPU seconds): 0.01 (Wallclock seconds): 0.01

Critical Path time for Worst Case Scenario:
DescribeProduct starts at time 0
DevelopMarketingStrategy starts at time 0
WriteClientProposal ends at 246.0 hours in duration

Solution variable values:
end_Coding = 110.0
end_DescribeProduct = 8.0
end_DesignBrochure = 24.0
end_DevelopImplementationPlan = 222.0
end_DevelopMarketingStrategy = 16.0
end_DevelopPricingPlan = 214.0
end_PackageDeliverables = 198.0
end_RequirementsAnalysis = 14.0
end_SoftwareDesign = 30.0
end_SurveyPotentialMarket = 64.0
end_SystemDesign = 30.0
end_SystemTesting = 190.0
end_UnitTesting = 150.0
end_WriteClientProposal = 246.0
end_WriteDocumentation = 150.0
start_Coding = 30.0
start_DescribeProduct = 0.0
start_DesignBrochure = 8.0
start_DevelopImplementationPlan = 198.0
start_DevelopMarketingStrategy = 0.0
start_DevelopPricingPlan = 198.0

start_PackageDeliverables = 190.0
start_RequirementsAnalysis = 8.0
start_SoftwareDesign = 14.0
start_SurveyPotentialMarket = 24.0
start_SystemDesign = 14.0
start_SystemTesting = 150.0
start_UnitTesting = 110.0
start_WriteClientProposal = 222.0
start_WriteDocumentation = 110.0

APPENDIX V: Tabular Results

	Best Case Start Time (hrs)	Best Case End Time (hrs)	Expected Start Time (hrs)	Expected End Time (hrs)	Worst Case Start Time (hrs)	Worst Case End Time (hrs)	Best Case Cost	Expected Cost	Worst Case Cost
Describe product	0	2	0	4	0	8	$150	$300	$600
Develop marketing strategy	0	4	0	8	0	16	$600	$1,200	$2,400
Design brochure	2	6	4	12	8	24	$600	$1,200	$2,400
Develop product prototype									
Requirements analysis	2	4	4	8	8	14	$600	$1,200	$1,800
Software design	4	8	8	16	12	28	$600	$1,200	$2,400
System design	4	8	8	16	12	28	$900	$1,800	$3,600
Coding	8	24	16	56	30	110	$4,800	$12,000	$24,000
Write documentation	24	32	56	76	110	150	$1,200	$3,000	$6,000
Unit testing	24	28	56	68	110	150	$1,200	$3,600	$12,000
System testing	28	32	68	80	150	190	$600	$1,800	$6,000
Package deliverables	32	34	80	84	190	198	$600	$1,200	$2,400
Survey potential market	6	14	12	32	24	64	$600	$1,500	$3,000
Develop pricing plan	34	36	84	88	198	214	$150	$300	$1,200
Develop implementation plan	34	42	84	100	198	222	$1,800	$3,600	$5,400
Write client proposal	42	50	100	116	222	246	$600	$1,200	$1,800
TOTALS		**49**		**114**		**244**	**$15,000**	**$35,100**	**$75,000**

APPENDIX VI: Gantt Chart