12/8/2021

Food Ordering

Logical Specification Document

CIS 212 Project

Professor Jack N. Donato

Team Members: Justyce Countryman, Izabelle Liendecker, and Tatiana Njamen Sayep

1. System Description

A. Possible Alternative Configurations/Solutions - In addition to the primary and recommended system functionalities, the program could also produce a weekly sales report, display tip percentages on the bill, ask for the desired size of applicable food items, and present descriptions of each food item.

B. System Capabilities - The program will first obtain the menu items and prices from the menu file with the proper menu ID. Once this procedure is successful, the program will then ask the user if they want to add a valid food item to a customer order, remove a valid food item from a customer order, modify the menu, or quit and display the bill. Based on the user choice, the program will then request the user to continuously input food items and quantities to either add or delete within the current customer order until the user chooses to quit, in which a bill will be displayed containing the ordered item names, quantities, single quantity prices, and totals. The calculated subtotal, tax, and total due will be included at the end of each bill. Secondly, suppose the user decides to alter the menu. In that case, the program will continuously ask the user if he or she wants to add, remove, or change a food item and its information until the user chooses to quit and return to the previous menu. The system will append the updated food information given by user input to the menu file.

2. System Requirements

A: Functional Requirements - This system will allow restaurant servers to enter food orders and produce customer bills. Other alternative functional requirements may include permitting authorized workers to add, delete, and change food menu information within the system, including food item names and prices, producing a weekly sales report, displaying tip percentages on the bill, asking for the desired size of applicable food items, and presenting descriptions of each food item and their prices.

B: Nonfunctional Requirements - The system will display output in a presentable format that is easily readable for all users. Each time the system needs to get information from the user or provide information to the user, the system will feel usable to users to accomplish the primary functional requirements within the food ordering system. Additionally, when users wish to alter their menus, it is beneficial to display the current menu price of the searched menu item for user convenience.

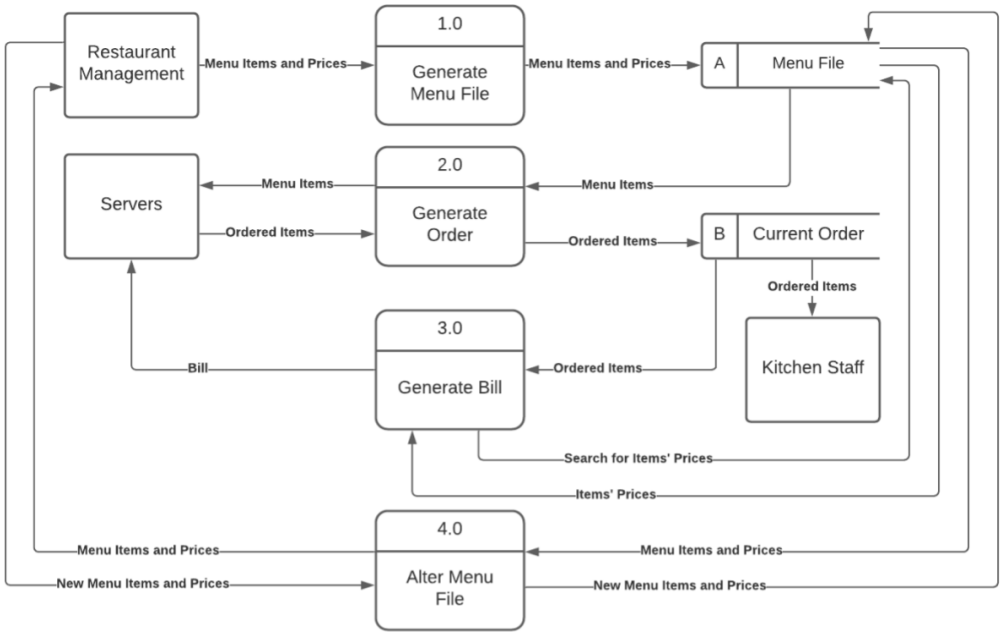
3. Business Benefits

- Faster restaurant service at a rate of approximately 15%
- Growth in food ordering sales at a rate of roughly 15%

- Approximately a 25% decrease in complications with food item prices and customer bill totals
- A rise in improved customer satisfaction and trust by about 40%
- Around a 35% probability for customers to come back regularly
- An estimated 23% boost in revenues
- Roughly 8% fewer employees needed for business stability
- Around an 8% decrease in labor costs
- Easier to review sales income for the company through a sales report

4. Complete Set of Data Flow Diagrams

The data for the food ordering system will initially start with restaurant management providing the menu items and prices so that the user can generate a menu file with a unique menu ID. Then, after validating the menu items and expenses, the food information will be kept in a file. With the menu information stored properly, users can generate an order once the servers receive the valid menu items and enter the applicable ordered items and quantities that each party of one or more customers want. After this, the kitchen staff will prepare the food for each customer order. Additionally, the program will validate the ordered items and place the requested food items in the current customer order. Then, the program will use the ordered items to help generate a bill that the program will display for the servers to give to the customers. When developing a bill, the system will use the menu file to search for the proper item IDs to represent the ordered items. Once found, their menu item names and prices will be sent back to the bill and displayed to the servers once calculations are completed to determine the price of each ordered menu item based on the quantity of each ordered item, subtotal, tax, and total. If any menu information needs to be modified, restaurant management will be given the current menu items and prices, which will also undergo validation before being displayed. Moreover, restaurant management may give the new menu items and prices, altering the menu file. After verification, the program will appropriately store this menu information in the menu file for future use.



5. Data Dictionary

A. Data Elements -

Element Name	Element Length	Description	Data Type	Element Format	Validation
Party_ID	8	Unique ID numbers will be used to represent each party of one or more customers to help with organizing bills.	Integer	Ex: 22222222	Each Party_ID must be unique to every party of one or more customers.
Server_Name	Depends on the number of characters in the names of each server in the restaurant.	The name of a server in the food ordering restaurant.	String	Ex: Steve Jobs	There must be at least one Server_Names but no more than two thousand.

Server_ID	8	Unique ID numbers will be used to represent each server to keep track of the current servers who work at the restaurant. These IDs may also help with knowing which servers are serving what parties.	Integer	Ex: 33333333	Each Server_ID must be unique to each server within the food ordering place using the system. There must be at least one Server_ID but no more than two thousand.
Kitchen_Staff_Name	Depends on the number of characters in the names of each kitchen staff member in the restaurant.	The name of a kitchen staff member in the food ordering restaurant.	String	Ex: Bill Gates	There must be at least one Kitchen_Staff_Names but no more than two thousand.
Kitchen_Staff_ID	8	Unique ID numbers will be used to represent each kitchen staff member who works at the restaurant. These IDs may also help with knowing which kitchen staff members are in charge of preparing what ordered menu items.	Integer	Ex: 44444444	Each Kitchen_Staff_ID must be unique to each kitchen staff member within the food ordering place using the system. There must be at least one Kitchen_Staff_ID but no more than two thousand.
Menu_ID	8	Unique ID numbers will be used to represent each menu file within each food ordering system. As a result, it may be possible for a restaurant to implement several Menu_IDs within a single food ordering system.	Integer	Ex: 55555555	Each Menu_ID must be unique to each menu file within the food ordering place using the system.

Manager_ Name	Depends on the number of characters in the names of each manager in the restaurant.	The names of each manager in the food ordering restaurant.	String	Ex: Jeffery Bezos	There must be at least one Manager_Name but no more than two thousand.
Manager_I D	8	Unique ID numbers will be used to represent each manager in restaurant management who works at the restaurant using the system.	Integer	Ex: 66666666	Each Manager_ID must be unique to each manager within the food ordering place using the system. There must be at least one Manager_ID but no more than two thousand.
Order_ID	8	Unique ID numbers will be used to represent each current order within the restaurant using the system.	Integer	Ex: 77777777	Each Order_ID must be unique to each order within the food ordering place using the system.
Bill_ID	8	Unique ID numbers will be used to represent each displayed bill within the restaurant using the system.	Integer	Ex: 88888888	Each Bill_ID must be unique to each order within the food ordering place using the system.
menu-items	Depends on the number of characters to describe each menu item.	The names of each food item in the menu file.	String	Ex: Hamburger	Must have at least one menu-item but no more than two thousand.
menu-prices	At least one real digit number with two real digit numbers after decimal.	The prices of each food item in the menu file.	Double	Ex: 1.99 12.50 108.25	Must have at least one menu-price but no more than two thousand.

				…	
userChoice	1	User input that determines whether to add or remove a food item and its quantity to the bill, modify the menu, or display the bill.	Char	Ex: ‘A,’ ‘R,’ ‘M,’ ‘D,’ ‘I,’ or ‘Q’	Users may only input ‘A,’ ‘R,’ ‘M,’ ‘D,’ ‘I,’ or ‘Q.’ All other characters will ask for the userChoice again.
ordered-item	Depends on the number of characters to describe each menu item.	The food items to add to the customer bill. May repeat to add multiple food items.	String	Ex: Hamburger	Must enter valid food items. Capitalization will be ignored.
item-ID	8	Unique ID numbers will be assigned to each food item to help search for their prices.	Integer	Ex: 11111111	item-IDs must be unique to every food item.
item-quantity	At least one real digit number.	One numerical quantity ordered by the customer of a specific menu item.	Integer	Ex: Hamburger 2	No negative item-quantities will be allowed to be inputted.
quantity-total	At least one real digit number.	The total quantity ordered by the customer of a specific menu item.	Integer	Ex: Hamburger 2+3 = 5	Quantity-totals that are zero or negative will not take part in the calculations performed before displaying the bill.

item-totals	At least one real digit number with two real digits after decimal.	The total price of each individual ordered-item multiplied by their individual item-quantities.	Double	Ex: Cheeseburger 3.25 * 2 = 6.50	N/A
subtotal	At least one real digit number with 2 digits after decimal.	The total price of all ordered-items added together without tax.	Double	Ex: 2.50 36.00 418.25 …	N/A
total-tax	At least one real digit number with 2 digits after decimal.	The total tax price by multiplying the subtotal by 0.08.	Double	Ex: 2.50 36.00 418.25 …	N/A
total	At least one real digit number with 2 digits after decimal.	The total price of all ordered-items added together with tax.	Double	Ex: 2.50 36.00 418.25 …	N/A

B. Data Structures -

Data Structure Name	Definition
menu-file	{item-ID + menu-items + menu-prices}

server-database-table	{Server_Name + Server_ID}
kitchen-staff-database-table	{Kitchen_Staff_Name + Kitchen_Staff_ID}
manager-database-table	{Manager_Name + Manager_ID}
items-to-be-made	{ordered-item}
subtotal	{menu-prices + item-quantity}
total-tax	subtotal
total	total-tax + subtotal
restaurant-management	{item-ID + menu-items + menu-prices}
current-order	{ordered-item + item-quantity}

servers	{menu-items + ordered-item + item-quantity + menu-prices}
kitchen-staff	{ordered-item}
bill	{ordered-item + item-quantity + menu-prices}

C. Data Flows -

Data Flow Name	Data Flow Source	Data Flow Destination
menu-items	restaurant-management	menu-file
menu-prices	restaurant-management	menu-file
ordered-item	servers	bill
items-to-be-made	servers	kitchen-staff
item-quantity	servers	bill

D. Data Stores -

Data Store Name	Description	File Type	Maximum Size	Growth Estimate

menu-file	Contains the item-IDs, menu items, and menu prices to be processed provided by restaurant management.	File	2000 entries	Depends on how often the restaurant adds more entries to their menu. Approximately ten to twenty new entries per year.
server-database-table	Contains the Server_Names and their unique Server_IDs within a food ordering restaurant.	Database Table	2000 entries	Depends on how often and how many servers join or leave the restaurant staff within a particular food ordering restaurant. Approximately fifty to one hundred entries per year.
kitchen-staff-database-table	Contains the Kitchen_Staff_Names and their unique Kitchen_Staff_IDs within a food ordering restaurant.	Database Table	2000 entries	Depends on how often and how many kitchen staff members join or leave the restaurant staff within a particular food ordering restaurant. Approximately fifty to seventy-five entries per year.
manager-database-table	Contains the Manager_Names and their unique Manager_IDs within a food ordering restaurant.	Database Table	2000 entries	Depends on how often and how many managers join or leave the restaurant staff within a particular food ordering restaurant. Approximately twenty to thirty entries per year.

E. Process Descriptions -

Process Name	Process Boundaries	Outputs	Inputs	Activities

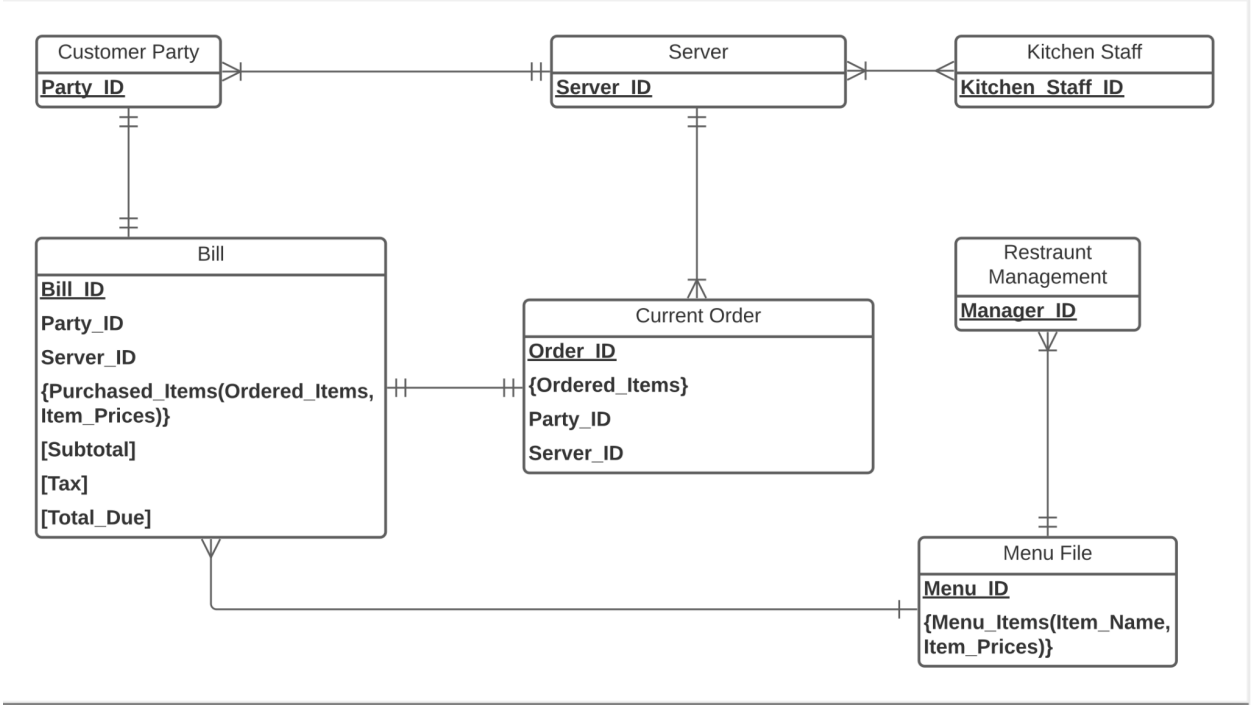
get-menu-information-from-file	Starts at system initiation. Once completed, the next process, get-choice, will initiate.	item-IDs, menu-items, and menu-prices	menu-file with entries containing one or more item-IDs, menu-items, and menu-prices	For each food entry, the process will get the current item-ID, menu-item, and menu-price.
get-choice	Starts after menu information is received or when the program could not find the ordered-item after searching through all menu-items. Once completed, either the get-ordered-items or calculate-item-totals process will initiate.	get-ordered-items or display-bill	userChoice	While the userChoice is not ‘A’ and not ‘R’ and not ‘M’ and not ‘I’ and not ‘Q,’ get the userChoice from the user.
get-ordered-items	Starts if userChoice is ‘A’ or ‘R,’ ‘M’, or ‘D.’ Once completed, the next process, search-for-ordered-item, will initiate.	search-for-ordered-item	ordered-item	Get the ordered-item from the user.
search-for-ordered-item	Starts after ordered-item is received. Once completed, the system will either continue to the next process, get-item-quantity, or return to get-choice.	get-item-quantity or get-choice	item-IDs and menu-items	Search the menu-items for ordered-item.

get-item-quantity	Starts if the userChoice is ‘A’ or ‘R’ and the program found the ordered-item after searching through all menu-items. Once completed, either the add-quantity-total or remove-quantity-total process will initiate. However, if the item-quantity is not valid, the program will ask for the item-quantity again.	update-quantity-totals	item-quantity	Get the valid item-quantity from the user.
add-quantity-total	Starts after item-quantity is received and userChoice is ‘A.’ Once completed, the program will return to get-choice.	Current quantity-total	item-quantity	Set the current quantity-total to current quantity-total + item-quantity.
remove-quantity-total	Starts after item-quantity is received and userChoice is ‘R.’ Once completed, the program will return to get-choice.	Current quantity-total	item-quantity	Set the current quantity-total to current quantity-total - item-quantity.
add-item-to-menu-file	Starts when userChoice is ‘I.’ Once completed, the program will return to get-choice.	Updated menu-file	New item-ID, new menu-item, and new menu-price	Get new item-ID from user, get new menu-item from user, and get new menu-price from user and append all the inputted information to the menu file.

modify-item	Starts if the userChoice is 'M' and the program found the ordered-item after searching through all menu-items. Once completed, the program will return to get-choice.	Updated menu-file	Corresponding item-ID, corresponding menu-item, and corresponding menu-price	Get corresponding item-ID from user, get corresponding menu-item from user, and get corresponding menu-price from user.
delete-item	Starts if the userChoice is 'D' and the program found the ordered-item after searching through all menu-items. Once completed, the program will return to get-choice.	Updated menu-file	Corresponding item-ID, corresponding menu-item, and corresponding menu-price	Delete corresponding item-ID, delete corresponding menu-item, and delete corresponding menu-price.
calculate-item-totals	Starts when userChoice is 'Q.' Once completed, the next process, calculate-subtotal, will initiate.	Current item-total	Current menu-item, current menu-price, and current item-quantity	For each menu-item, set current item-total to current menu-price * current item-quantity.
calculate-subtotal	Starts after all item-totals are received. Once completed, the next process, calculate-tax, will initiate.	subtotal	Current item-total	Add all item-totals together to get the subtotal.
calculate-tax	Starts after subtotal is calculated. Once completed, the next process, calculate-total, will initiate.	tax	subtotal	Set tax to subtotal * 0.08.

calculate-total	Starts after tax is calculated. Once completed, the next process, display-bill, will initiate.	total	subtotal and tax	Set total to subtotal + tax.
display-bill	Starts after total is calculated. Once finished, a complete and accurate bill will be displayed based on ordered-items. After the bill is displayed, the program will end.	bill	Current menu-item, current menu-price, current item-quantity, current item-total, subtotal, tax, and total	For each menu-item, display current menu-item, current menu-price, current item-quantity, and current item-total if current item-quantity is greater than 0. Then, display subtotal, tax, and total.

6. Entity Relationship Diagram



7. Reassessed Feasibility

A. Economic Analysis - Assigning and paying a qualified programmer to design, code, and test the system costs around $1,400. This cost could also increase if additional system debugging, adjustments, and maintenance is needed to achieve stakeholder satisfaction. Moreover, there are training costs that could vary based on the number of employees who must use the system and their levels of technological expertise. One-time hardware costs could also be apparent since the system needs to be implemented into a specific environment to ensure stakeholders can benefit from the system. Cash registers are a primary example of mandatory hardware for this situation and cost approximately $1,200 each. In terms of financial benefits, this system should speed up the food ordering process and may increase the flow of customers by about 8%. As a result, revenue from food ordering sales could rise by 15%. In addition, the system should also make customer food orders more accurate, which will help customers feel more content about coming back to the restaurant due to satisfaction and trustworthiness. The system could also reduce the number of employees needed by 8%, saving the restaurant from inessential labor costs by 8% in the process. The recommended budget for this project would be about $5,000.

B. Technical Analysis - The system will involve a language program that carries out several food orderings tasks. The difficulty of designing the system code to establish these procedures ranges from mildly to moderately complex. The system code is expected to be completed, tested, debugged, and finalized within three months of beginning the project. The program should be able to run on most forms of computers. One primary technical risk for this project may include dealing with hardware that is not typically standard with the process of food ordering, which could cause the system to fail while attempting to perform mandatory food ordering objectives.

C. Operational Analysis - This program will make it easier and faster for servers to input orders and calculate bills accurately. Advantageous solutions to the issues of falling behind on customer food orders, misreading handwritten food orders, inaccurately inputting food item prices, and losing customers because of other forms of restaurant inconveniences are possible through the usage of this system. This system would increase by 15% the quality of service at the restaurant, which provides an increase of 15% of revenue and establishes efficient business flow. There is also a risk of a slowdown of 10% of the quality of the service if the servers are not well trained to use the system, which may result in a 10% overall revenue decrease.

D. Legal and Contractual Analysis - Legal issues may be possible if there are conflicts in the food ordering system that causes one or more customers to pay more or less than what they were supposedly charged. In the development of the program, the hired programmer should not work more than the regular eight hours unless the programmer agrees to work overtime. On another point, once a system is complete for one restaurant, creating a new food ordering system for another food service will have to be redone from scratch to minimize copyright risks. A complete contract should be established between the restaurant manager and the programmer.

E. Political Analysis - Stakeholders should be in support of this program because it should cause an increase in revenue. Stronger accuracy and faster food ordering speeds will make customers likely to tip more and come back frequently. This system could also result in exceptional customer reviews and recommendations for the restaurant. The outcomes of this project would help employees and customers feel more content. However, stakeholders may criticize that the food ordering system may not cover any advanced food ordering methods, like automatically modifying, adding, or deleting food items based on specific restaurant circumstances rather than manual user input. This concern may arise if restaurants promote discounts regularly, remove food items that are rarely purchased by customers, or add a variety of new food selections to their menus.

F. Schedule, Timeline, and Resource Analysis - Delays towards the expected project completion time could be possible if the team members do not have the skills needed to create the system at or before the desired completion time. For instance, designing, testing, and debugging the system code may take longer than expected if the programmer either runs into errors or realizes the code is more complex than anticipated. Not only that, requesting meetings with the stakeholders may not always be planned out properly, especially if it takes more time than estimated to get dates and times for the meetings, they get rescheduled, or the project team needs more time to prepare. Every primary task in the project is likely, but not guaranteed, to be completed within the specified expected times as indicated in the preliminary project schedule. The estimated expected time to complete the project is 6 months.

8. Redefined Project Schedule

A. Work Breakdown Structure -

Step A: Analysis of all primary system requirements and essential steps

Step B: Create system code

Step C: Perform system testing and debugging

Step D: Perform system finalization with project team

Step E: Establish system report to present to target stakeholders

Step F: Review system report with project team

Step G: Request and hold stakeholder meeting for report presentation and system alpha testing

Step H: Make system adjustments based on meeting

Step I: Finalize system adjustments with project team

Step J: Update system report based on meeting and system adjustments

Step K: Review updated system report with project team

Step L: Request and hold second stakeholder meeting for updated report presentation and system beta testing

Step M: Make additional system and report adjustments based on second meeting

Step N: Finalize additional adjustments with project team

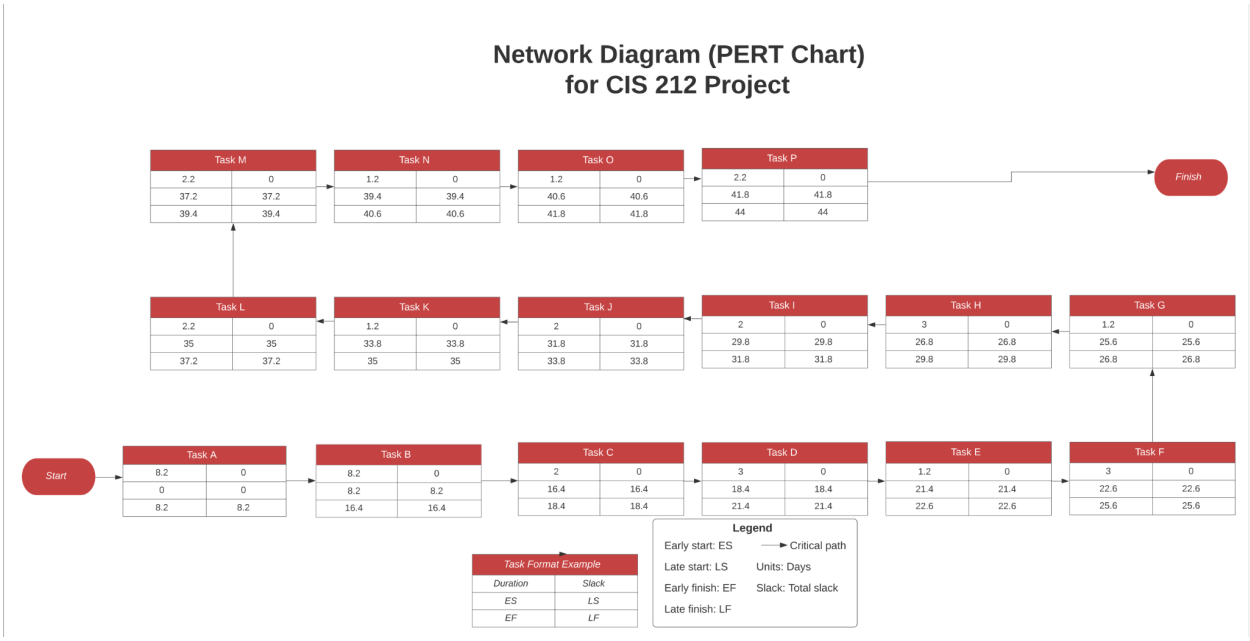
Step O: Verify with stakeholders that system meets all requirements

Step P: Export system to satisfied stakeholders

B. Preliminary Schedule for Project -

Step	Prerequisite Steps	Optimistic Time	Realistic Time	Pessimistic Time	Expected Time
A	-	7 Days	7 Days	14 Days	8.2 Days
B	A	7 Days	7 Days	14 Days	8.2 Days
C	B	1 Day	2 Days	3 Days	2 Days
D	C	2 Days	3 Days	4 Days	3 Days
E	D	1 Day	1 Day	2 Days	1.2 Days
F	E	2 Days	3 Days	4 Days	3 Days
G	F	1 Day	1 Day	2 Days	1.2 Days
H	G	2 Days	3 Days	4 Days	3 Days
I	H	1 Days	2 Days	3 Days	2 Days
J	I	1 Days	2 Days	3 Days	2 Days
K	J	1 Day	1 Day	2 Days	1.2 Days

L	K	2 Days	2 Days	3 Days	2.2 Days
M	L	2 Days	2 Days	3 Days	2.2 Days
N	M	1 Day	1 Day	2 Days	1.2 Days
O	N	1 Day	1 Day	2 Days	1.2 Days
P	O	2 Days	2 Days	3 Days	2.2 Days



9. Testing Plans

A. System Testing - System testing will initiate once a rough draft of the code is completed and will reoccur every time after program adjustments are required and implemented. The project team will meet through a Zoom meeting to review the code and run the program to see if all essential tasks are executable. This form of testing will ensure that the menu-driven program will accept the menu information from the menu file given by restaurant management and successfully get and perform the user choices continuously until chosen to quit, which will display the bill. The team will test all forms of capitalization so that the program does not focus on if characters are uppercase or lowercase. Additionally, the project team will verify that the program utilizes the correct menu

information from the menu file after searching for the desired menu item. There will also be validation that confirms users can input no negative integers while adding or removing ordered items and quantities. If the food item is not in the file, the program should return to the primary user choices without changing the order. The project team will assess calculations to confirm that the subtotal is determined precisely every time by adding all item-totals together. The program should obtain the item totals by taking the currently ordered items and multiplying their respective menu prices by their respective item quantities. Tax will be 8% of the subtotal, and the total will be the subtotal plus the total. Lastly, the project team will create a suitable format to view displayed information. If all these functionalities are tested for errors or inaccuracies and compiled clean and correct, the project team may move on to system finalization. Otherwise, the team will continue to achieve total system stability until there is no probability for the user to run into bugs. If the team cannot fix an error due to its solution not being found by any team members, assistance from the sponsor, Professor Jack N. Donato, may be considered a last resort. In this situation, a request for a private meeting between the project team and Professor Donato will initiate in-person or on Zoom to determine possible solutions or adjustments if the error does not have a proper fix method.

B. Integration Testing - Integration testing will initiate after system testing, debugging, and finalization and will commence during each meeting with interested stakeholders. During the first part of integration testing, or alpha testing, all stakeholders will try the system and see if it will fit all their needs for their food ordering services. The progress of integration testing is likely to be similar in comparison to system testing, except that the program is now from the stakeholders' perspective. As a result, they will also ensure that all menu-driven options in the program are executable consistently without errors or incorrect information. Suppose stakeholders notice and reveal any concerns with the program. In that case, the project team will consider and attempt to resolve all issues while making system adjustments based on the two or more primary stakeholder meetings. All adjustments should be complete and ready for all stakeholders to try out during beta testing, which is the second part of integration testing. The project team will likely bring the system to the location where the system will have the most use, primarily restaurants and other areas where customers may order food. The program will need to have implementation capabilities to function in a computer system that is easily accessible to all system users, preferably a cash register. If the program has complications with being integrated into the target system, the project team may request assistance from stakeholders who know the system well. Suppose this plan does not solve the issue. In that case, it may be helpful to contact the company that manufactured the cash register or another system and receive customer support to determine compatibility requirements. All project team members will consider all given information from customer support. If the current program is not supported, the project team must comprehend how to make the program compatible. Once system integration is successful, the stakeholders and system users may execute all tasks from the menu to see if all functions work similarly to how they did from the previous meeting and if previously mentioned program problems are now nonexistent. Suppose there are any noticeable errors or concerns with how the program operates. In that case, the project team will take the issues into account again during the second round of system adjustments. After completing any additional

adjustments, the cycle of testing and modifying will continue until stakeholders are satisfied. Once the system is ready, the project manager will appropriately export the working system code to the food ordering service.