**Planning Document**

* **Introduction** - the introduction should briefly describe the objectives of the project and set out any constraints of the project

OBJECTIVE: The objective of the project is to create a windows form application that allows the user to enter the item, item description, order quantity, and unit price. The application will then display a column of the estimated cost(order quantity \* unit price). At the bottom, it should display the total estimated cost.

CONSTRAINTS:

\*\*LINKS TO REQUIREMENTS VIA NUMBER

1. The total estimated cost of the whole “material list”
2. Total cost for each row or “item”
3. Clear button
4. Save Button
5. Calculate button
6. Text boxes for item description, item, quantity, budget, and unit price.
7. Drop down list for the category
8. Ensure a grid view (DataGridView)
9. Pie chart showing all categories and how much they cost compared to the total
10. Budget option per category and total
11. If the budget is exceeded, highlight a specific category that exceeds the budget or is most expensive.
12. Allow the user to create a new category and save it to the category drop down list.

REQUIREMENTS

1. Adds up all the totals for each row and displays the correct total at the bottom.
2. Calculates the order quantity times the unit price and display the correct total at the far right of each row.
3. The clear button will clear time entire table and begin a new materials list.
4. Save button will save the information to a file
5. Calculate button will display the total estimated cost, budget’s exceeded, and pie chart.
6. Text boxes will allow the user to enter type letters and numbers into the item description and item. However, it will only allow the user to enter numbers into quantity, unit price, and budget. If the user tries to type letters into the text boxes, it will inform the user that letters are not allowed via a prompt.
7. The drop-down list will be enabled on the far left of the grid and will allow the user to select from a variety of categories successfully.
8. Our entire application will be built in a grid in a similar manner to how excel’s grid looks.
9. The pie chart will use the total of each category and divide it by the total estimated cost determining the percent sliver of each slice on the pie graph.
10. Allow the user to input a budget for each category and then a budget for overall and if the user exceeds the budget when they hit calculate, it will highlight the category that exceeds the budget in the red.
11. When the user hits the calculate button, it will highlight any category that exceeds their budget or which category that is most expensive in red.
12. There will be a textbox that will allow the user to add a new category to the drop list if they don’t like the current categories given.

* **Project organization** - list the team members and the roles of the members of the team.

Timothy McMasters: Developer - Appendix 1, 4, 7, 10,

Yihong Yang: Developer - Appendix 3, 5, 8,11,

Zain Rife: Developer - Appendix 2, 6, 9,12,

* **Hardware and software requirements** of the software and also the development environment (language, etc.)

Video Studio 2019, C# Window Form

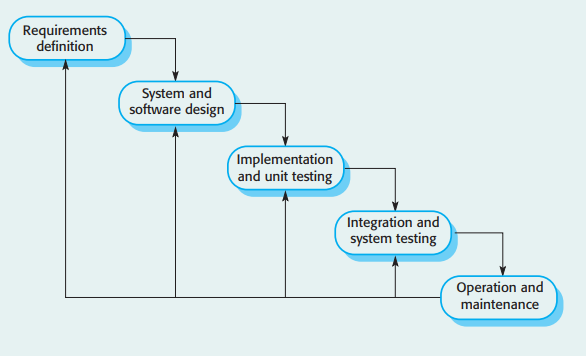
GitHub, Discord, Google Docs, Trello

* **Work breakdown** - how the project is broken into activities.

ACTIVITIES:

1. Create a method that takes the entire total column and adds up every total in the total column.
2. Test the total addition method
3. Create a method that displays the correct total at the bottom of the total column after clicking the calculate button.
4. Test if it displays correctly and the correct information
5. Create a method that takes the input unit price times the input order quantity and displays the multiplied total on the far right(total column) in each row correctly.
6. Test that the math is done correctly
7. Test that it displays correctly for every new row/new item.
8. Create a method that when the user clicks the clear button, it will clear the entire materials list back to the default starting point, thus creating a new materials list.
9. Test that the entire materials list is set back to default whether or not certain components are filled in. It should still be set to default.
10. Create a method that, when the user clicks the save button, will save the entire material’s list to a txt file or other file. It should allow the user to open the txt file or other file to view the previous material’s list.
11. Test that the materials list is saved correctly and can be imported correctly into the program.
12. Create a method that, when the user clicks the calculate button, will display the total estimated cost, budgets exceeded, and display a pie chart below the material’s list that displays which categories cost the most.
13. Test that calculates the button displays the total estimated cost correctly.
14. Test that the calculate button will display/highlight the budgets exceeded or most expensive categories.
15. Test that the pie chart is generated correctly below the materials list.
16. Create a method that will allow the user to enter strings into the item description and item boxes and allow the user to enter an integer into quantity, unit price, and budget boxes. If the user enters the wrong data type, then it will inform the user via prompt.
17. Test that user is entering a string into the item description and item text boxes
18. Test that the user is only entering integers into quantity, unit price, and budget boxes.
19. Test if the program prompts the user to enter that they are entering the wrong data type correctly.
20. Create a method that allows the user to select a category, in the first column, from a drop down list.
21. Test if the drop-down box works as expected/allows the user to select from it and if the program can store the selected data from the drop down list.
22. Create a method that will allow the entire application to be built in a grid format similar to the way excel works.
23. Test that the grid functionality works as intended//allows the user to enter data into the application.
24. Create a method that will generate an accurate pie graph showing slivers of each category. (categoryTotal/totalEstimatedCost) \* 100 = percentageOfPieGraph.
25. Test that the method generates the pie graph correctly. (HARD)
26. Create a method that will allow a user to input an integer budget into each category and then a budget for the total estimated cost. If the user exceeds a budget, it will highlight the budget that was exceeded in red. If there are
27. Test that the text boxes will allow the user to enter the budgets and that the data is stored in the program.
28. Test that when the user hits the calculate button, it highlights any budgets that may be exceeded.
29. Create a method that will allow the user to create a new category and add it to the drop down box categories list. A textbox will allow the user to add a new category to the drop list if they don’t like the current categories given.
30. Test that the category is stored and added to the category’s drop down list.
31. Put it all together.
32. Test final outcome.

* **Process Flow Diagrams** - [Sample Process Flow Diagram](https://www.tutorialspoint.com/sdlc/images/sdlc_stages.jpg)
* [Links to an external site.](https://www.tutorialspoint.com/sdlc/images/sdlc_stages.jpg)



* **Project schedule** - showing dependencies between activities

February 23 - Appendix 1-2-3

February 25 - Appendix 4-5-6

March 2 - Appendix 7-8-9

March 4 - Appendix 10-11-12

March 9- Testing/Refactoring

March 11- Testing/Refactoring

* **Monitoring and reporting mechanisms**

Trello - Monitor the group project process

Regular meeting - Discuss and report the project Thur 2:15 PM, Sat 8:00 PM

Discord - Contact each other. Check every day.

* **Appendix** - The appendix should include a summary of the activities identified in this project plan. This summary should include the following information in either a table or excel spreadsheet form:
  + Number identifying the activity
  + Short description of the activity
  + Estimation of the time it will take to complete the activity
  + List of any activities that the current activity relies upon (dependencies)

The appendix will be based on the activity numbers above and the dependencies will be everything above the activity that you are looking at. So, this is essentially in the order that everything will be done in.

1. **22.** Create a method that will allow the entire application to be built in a grid format similar to the way excel works. **23.** Test that the grid functionality works as intended//allows the user to enter data into the application. 15 minutes
2. **16.** Create a method that will allow the user to enter strings into the item description and item boxes and allow the user to enter an integer into quantity, unit price, and budget boxes. If the user enters the wrong data type, then it will inform the user via prompt. **17.** Test that user is entering a string into the item description and item text boxes **18.**Test that the user is only entering integers into quantity, unit price, and budget boxes. **19.**Test if the program prompts the user to enter that they are entering the wrong data type correctly. 35 Minutes
3. **5.** Create a method that takes the input unit price times the input order quantity and displays the multiplied total on the far right(total column) in each row correctly. **6.** Test that the math is done correctly **7.**Test that it displays correctly for every new row/new item. 30 minutes
4. **20.** Create a method that allows the user to select a category, in the first column, from a drop down list.**21.** Test if the drop-down box works as expected/allows the user to select from it and if the program can store the selected data from the drop-down list. 30 minutes
5. **24.** Create a method that will generate an accurate pie graph showing slivers of each category. (categoryTotal/totalEstimatedCost) \* 100 = percentageOfPieGraph.**25.** Test that the method generates the pie graph correctly. (HARD) 50 minutes.
6. **29.** Create a method that will allow the user to create a new category and add it to the drop down box categories list. A textbox will allow the user to add a new category to the drop list if they don’t like the current categories given.**30.** Test that the category is stored and added to the category’s drop down list. 40 minutes
7. **26.** Create a method that will allow a user to input an integer budget into each category and then a budget for the total estimated cost. If the user exceeds a budget, it will highlight the budget that was exceeded in red. **27.** Test that the text boxes will allow the user to enter the budgets and that the data is stored in the program. 40 minutes
8. **1.** Create a method that takes the entire total column and adds up every total in the total column. **2.**Test the total addition method 25 minutes
9. **12.** Create a method that, when the user clicks the calculate button, will display the total estimated cost, budgets exceeded, and display a pie chart below the material’s list that displays which categories cost the most. **13.**Test that calculates the button displays the total estimated cost correctly. **14.**Test that the calculate button will display/highlight the budgets exceeded or most expensive categories. **15.**Test that the pie chart is generated correctly below the materials list. **28.**Test that when the user hits the calculate button, it highlights any budgets that may be exceeded. **3.** Create a method that displays the correct total at the bottom of the total column after clicking the calculate button. **4.** Test if it displays correctly and the correct information 50 minutes
10. **8.** Create a method that when the user clicks the clear button, it will clear the entire materials list back to the default starting point, thus creating a new materials list. **9.**Test that the entire materials list is set back to default whether or not certain components are filled in. It should still be set to default. 35 minutes
11. **10.** Create a method that, when the user clicks the save button, will save the entire material’s list to a txt file or other file. It should allow the user to open the txt file or other file to view the previous material’s list. **11.**Test that the materials list is saved correctly and can be imported correctly into the program. 40 minutes.
12. **31.** Put it all together. **32.** Test final outcome. 30 minutes