Tim Hoyt
IS 520
Final Project Write-up

Executive Summary

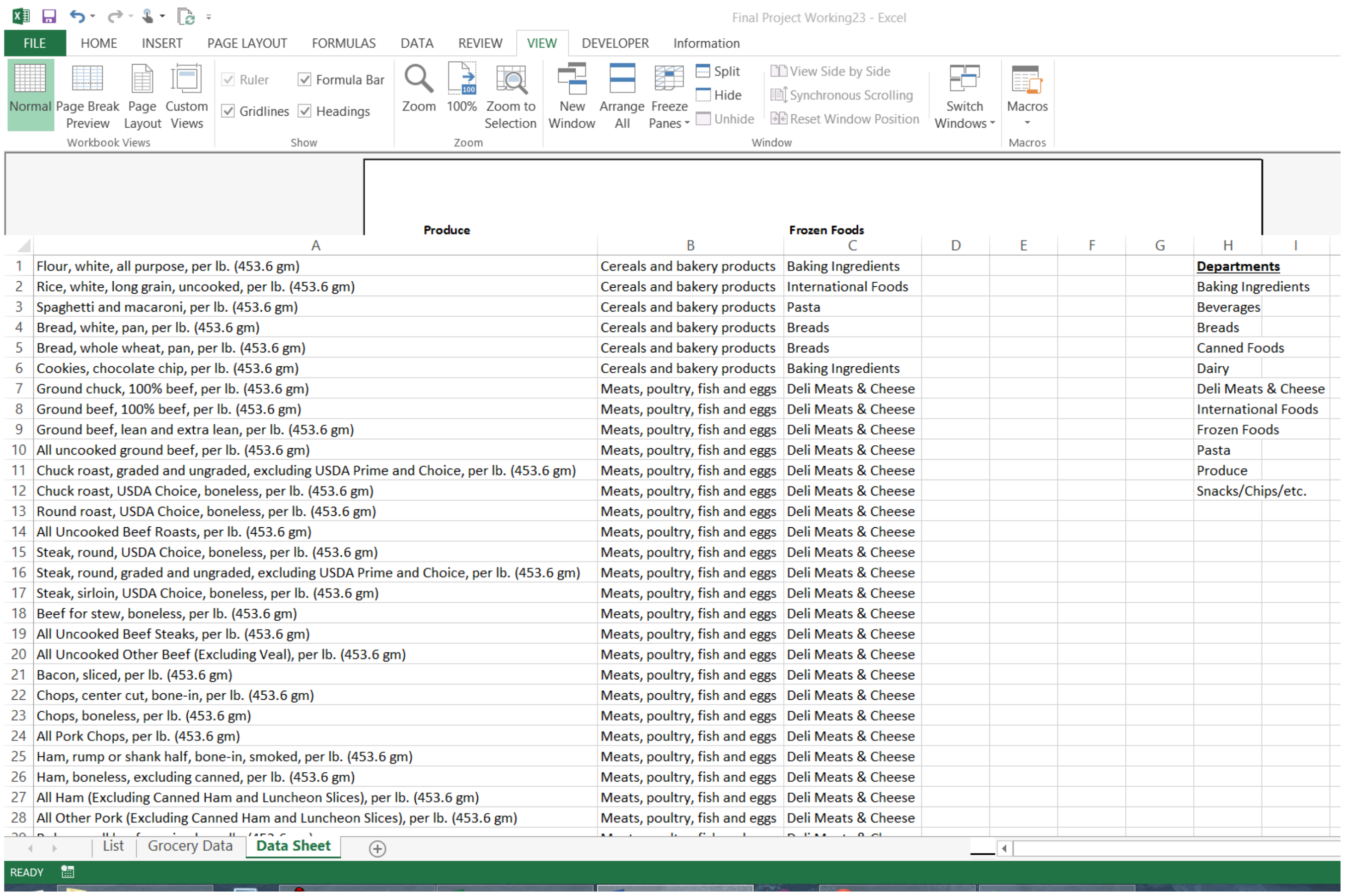
The business that I did my project for is homemaking. While this may seem like a joke to some, homemaking is an intense, very involved job that is stressful, exhausting, and often frustrating. Anything that helps simplify or consolidate some of the many responsibilities of a stay-at-home parent can make a huge difference to that parent. In this case, the specific process that I wanted to expedite is that of grocery shopping and creating a grocery list. It is easy to become overwhelmed when there is a list scribbled quickly onto a post-it note and you can't remember where everything is in the store or whether you've already gotten that item.

This problem is the basis for my project. My project is based heavily on the use of userforms. Clicking a button opens a form that allows the user to select a grocery store item, enter the quantity desired, and then the background programming sorts the item by its department in the store. The departments shown on the spreadsheet are also sorted by location in the store (the specific store is Walmart, since that is the store we commonly shop at, and it is the most uniform grocery store across the country). Another button creates a web query that pulls data from the Bureau of Labor Statistics (BLS). The website includes a list of over 60 grocery items, as well as the national average price for each item, including historical data. Though this will not give the exact amount relative to our local Walmart grocery store, it provides an approximation that can be added to a total. Thus, my wife can have a general idea for how much she can expect to spend on groceries in that trip.

60 is a big number, but it is definitely not an exhaustive list of all grocery items. So, I created a feature (using more userforms) that allows the user to add items and determine where they are in the store. It also allows the user to select and change the department in the store, in which that item is located.

Implementation Documentation

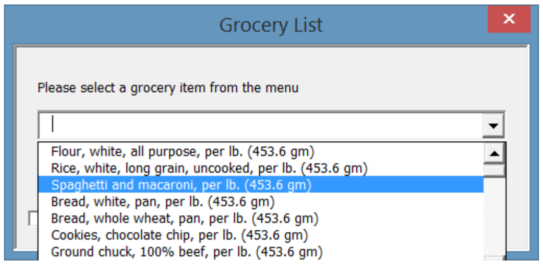
- To start this project off, I simply recorded myself creating a web query, going to the website I had already found, and selecting the table with the grocery item information. I then printed the information on a tab ("Grocery Data") that I will hide when my wife is using the program, but left open for the purpose of grading the project. I included a button with which the user can refresh the data, in case prices have been updated on the website.
- The website had each product categorized by type of food, but it was not set up according Walmart's organization, so I had to run a Do Loop in order to pull the grocery item from the Grocery Data tab and reprint it on a new tab ("Data Sheet"). Once on the Data Sheet tab, I had to individually reassign Walmart's departments to each item by hand.



	A	B	C	D	E	F	G	H	I
1	Flour, white, all purpose, per lb. (453.6 gm)	Cereals and bakery products	Baking Ingredients					**Departments**	
2	Rice, white, long grain, uncooked, per lb. (453.6 gm)	Cereals and bakery products	International Foods					Baking Ingredients	
3	Spaghetti and macaroni, per lb. (453.6 gm)	Cereals and bakery products	Pasta					Beverages	
4	Bread, white, pan, per lb. (453.6 gm)	Cereals and bakery products	Breads					Breads	
5	Bread, whole wheat, pan, per lb. (453.6 gm)	Cereals and bakery products	Breads					Canned Foods	
6	Cookies, chocolate chip, per lb. (453.6 gm)	Cereals and bakery products	Baking Ingredients					Dairy	
7	Ground chuck, 100% beef, per lb. (453.6 gm)	Meats, poultry, fish and eggs	Deli Meats & Cheese					Deli Meats & Cheese	
8	Ground beef, 100% beef, per lb. (453.6 gm)	Meats, poultry, fish and eggs	Deli Meats & Cheese					International Foods	
9	Ground beef, lean and extra lean, per lb. (453.6 gm)	Meats, poultry, fish and eggs	Deli Meats & Cheese					Frozen Foods	
10	All uncooked ground beef, per lb. (453.6 gm)	Meats, poultry, fish and eggs	Deli Meats & Cheese					Pasta	
11	Chuck roast, graded and ungraded, excluding USDA Prime and Choice, per lb. (453.6 gm)	Meats, poultry, fish and eggs	Deli Meats & Cheese					Produce	
12	Chuck roast, USDA Choice, boneless, per lb. (453.6 gm)	Meats, poultry, fish and eggs	Deli Meats & Cheese					Snacks/Chips/etc.	
13	Round roast, USDA Choice, boneless, per lb. (453.6 gm)	Meats, poultry, fish and eggs	Deli Meats & Cheese						
14	All Uncooked Beef Roasts, per lb. (453.6 gm)	Meats, poultry, fish and eggs	Deli Meats & Cheese						
15	Steak, round, USDA Choice, boneless, per lb. (453.6 gm)	Meats, poultry, fish and eggs	Deli Meats & Cheese						
16	Steak, round, graded and ungraded, excluding USDA Prime and Choice, per lb. (453.6 gm)	Meats, poultry, fish and eggs	Deli Meats & Cheese						
17	Steak, sirloin, USDA Choice, boneless, per lb. (453.6 gm)	Meats, poultry, fish and eggs	Deli Meats & Cheese						
18	Beef for stew, boneless, per lb. (453.6 gm)	Meats, poultry, fish and eggs	Deli Meats & Cheese						
19	All Uncooked Beef Steaks, per lb. (453.6 gm)	Meats, poultry, fish and eggs	Deli Meats & Cheese						
20	All Uncooked Other Beef (Excluding Veal), per lb. (453.6 gm)	Meats, poultry, fish and eggs	Deli Meats & Cheese						
21	Bacon, sliced, per lb. (453.6 gm)	Meats, poultry, fish and eggs	Deli Meats & Cheese						
22	Chops, center cut, bone-in, per lb. (453.6 gm)	Meats, poultry, fish and eggs	Deli Meats & Cheese						
23	Chops, boneless, per lb. (453.6 gm)	Meats, poultry, fish and eggs	Deli Meats & Cheese						
24	All Pork Chops, per lb. (453.6 gm)	Meats, poultry, fish and eggs	Deli Meats & Cheese						
25	Ham, rump or shank half, bone-in, smoked, per lb. (453.6 gm)	Meats, poultry, fish and eggs	Deli Meats & Cheese						
26	Ham, boneless, excluding canned, per lb. (453.6 gm)	Meats, poultry, fish and eggs	Deli Meats & Cheese						
27	All Ham (Excluding Canned Ham and Luncheon Slices), per lb. (453.6 gm)	Meats, poultry, fish and eggs	Deli Meats & Cheese						
28	All Other Pork (Excluding Canned Ham and Luncheon Slices), per lb. (453.6 gm)	Meats, poultry, fish and eggs	Deli Meats & Cheese						

This is the final product of the Data Sheet, after completing the Do Loop to extract the information and reassigning the departments.

- Once the Data Sheet tab was completed, it was time to create the userform and write the code to make it useable. The first object I worked on was the drop-down list from which the user could select an item. I needed a procedure to populate this list with the items from the BLS website as well as items the user would want to add later. The main code I used for this procedure was, again, a Do Loop, which cycles through and adds each BLS grocery item, skips a buffer line, then cycles through and adds each user-added item to the drop-down list.

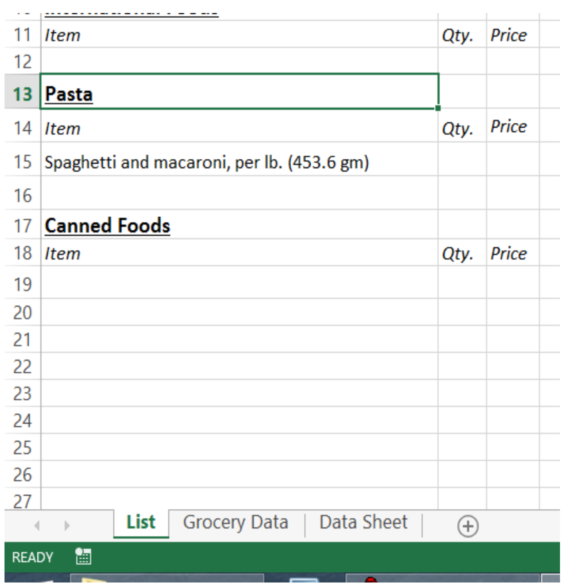


Please select a grocery item from the menu

```
Flour, white, all purpose, per lb. (453.6 gm)
Rice, white, long grain, uncooked, per lb. (453.6 gm)
Spaghetti and macaroni, per lb. (453.6 gm)
Bread, white, pan, per lb. (453.6 gm)
Bread, whole wheat, pan, per lb. (453.6 gm)
Cookies, chocolate chip, per lb. (453.6 gm)
Ground chuck, 100% beef, per lb. (453.6 gm)
```

The user can either select an item by clicking on it, or else by typing the first few letters and the item will appear.

- One of the greater challenges of this project was figuring out how to go from the selected item, to finding the correct department and adding the item under its proper department in the “List” tab I created. I finally decided to use the ‘cells.find’ feature to find the correct department under the Data Sheet tab, save the department as a variable, then find the variable under the List tab and activate that cell.

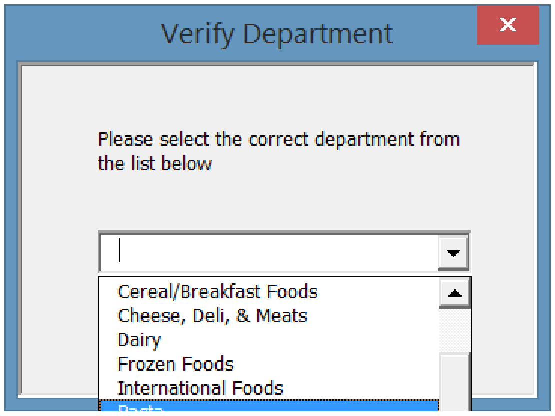


		Qty.	Price
11	*Item*	*Qty.*	*Price*
12			
13	**Pasta**		
14	*Item*	*Qty.*	*Price*
15	Spaghetti and macaroni, per lb. (453.6 gm)		
16			
17	**Canned Foods**		
18	*Item*	*Qty.*	*Price*
19			
20			
21			
22			
23			
24			
25			
26			
27			

- Once the cell with the correct department was activated, I could then insert a row of cells below the “Item,” “Qty,” and “Price” rows using the range.offset().insert method.

- Once I figured out how to insert rows in the proper location, it was quite easy to print the item in the newly inserted row. Some items from the BLS website have very long names, so to make sure that the list fit onto one page, I chose to cut off the item name at 47 characters, assuming that 47 was enough to choose the correct item. I saved the item as a variable, then printed it at the end of the procedure, with quantity and price together.

- I also added a check feature to ensure that the item was saved under the correct department in the Data Sheet tab. If the user selected the check box on the userform, a Message Box appeared, asking the user if the current department (saved as a variable earlier in the code) was correct. If the user chose yes (using the vbYesNo feature), the procedure would continue to insert the rows and print the data. If the user chose no (or if a newly added item had no department assigned to it yet), a new form would open up.



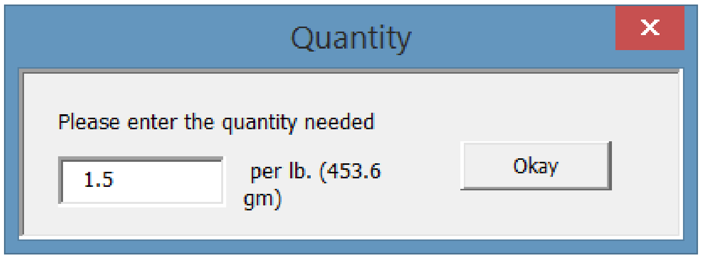
Verify Department

Please select the correct department from the list below

Cereal/Breakfast Foods
Cheese, Deli, & Meats
Dairy
Frozen Foods
International Foods
Pasta

- Using the same type of coding as I did to populate the Grocery Item list, I used a Do Loop to populate the Department list, from the list under the Data Sheet tab (see first screenshot). The user can select a new department, and upon clicking “Okay,” that department will be updated in the Data Sheet, and the department variable will be updated to represent the department selected by the user.

- The next item I needed to consider was how to allow the user to enter the correct quantity and how to print that properly. I created a userform that would initialize after the correct department was chosen. I thought it was important that the user be reminded of the unit used to measure each grocery item, so I tried to figure out a way to print that unit of measure (e.g. “per lb.”) onto the userform (more details in the challenges section), and then have a text box in which the user could enter the desired quantity of each item. I included an If statement to ensure that only a number could be entered into the text box. I then saved the entered quantity as a variable, so that it could later be printed, multiplied by the appropriate price, and added to the total price at the end of the procedure.

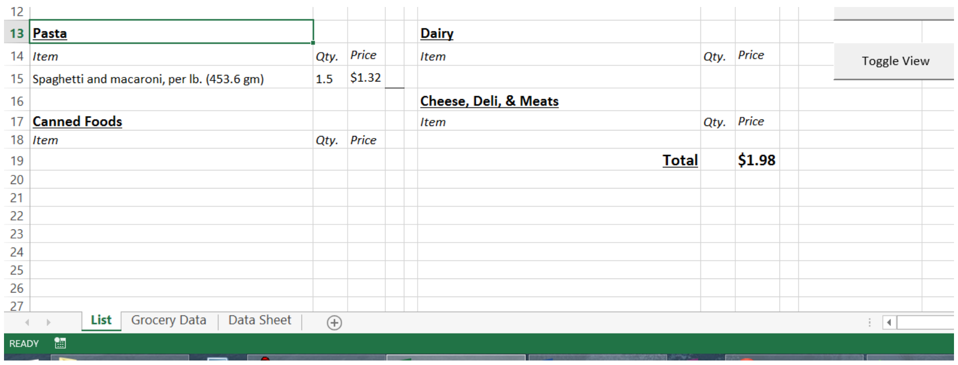

Quantity

Please enter the quantity needed

1.5 per lb. (453.6 gm)

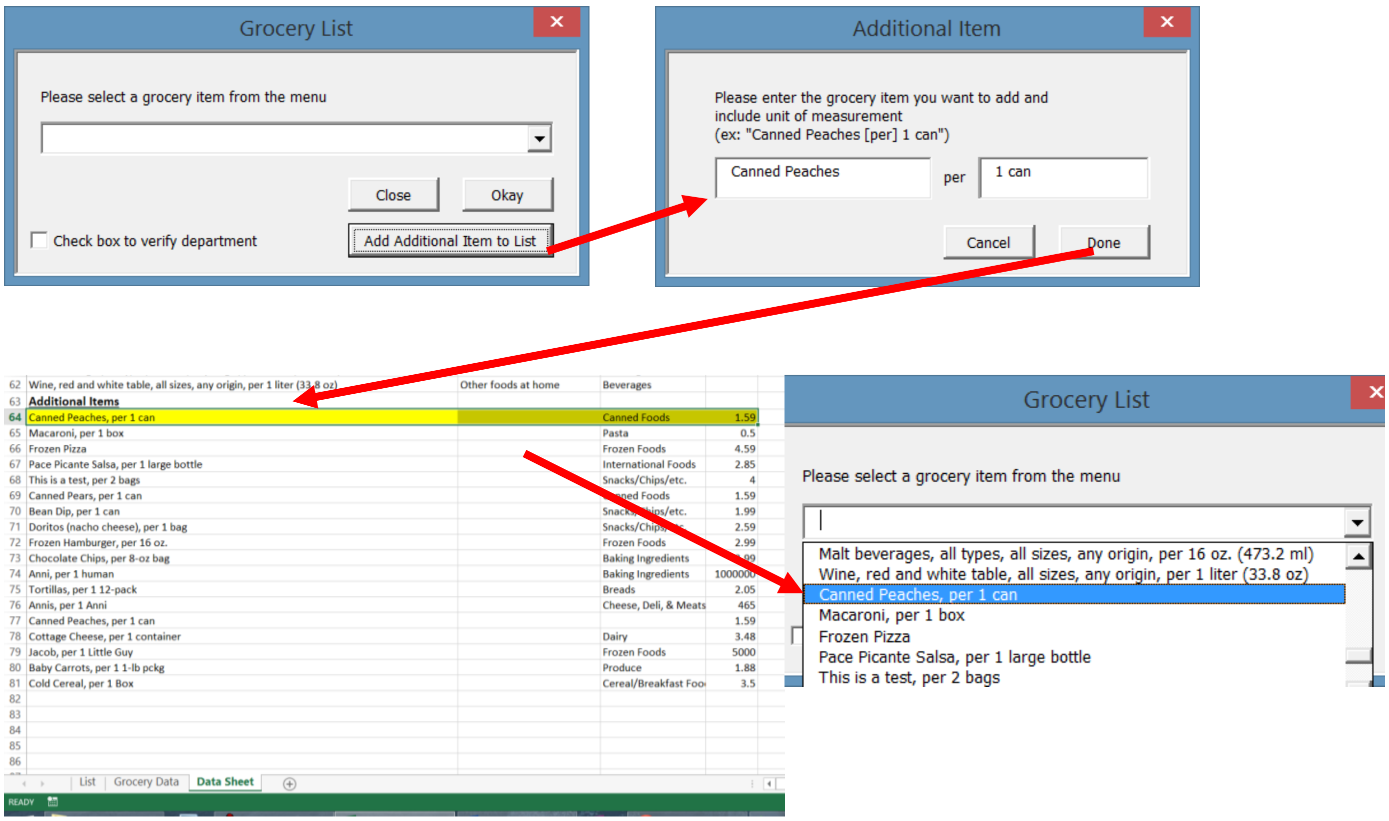
- Naturally, the next component of the project was determining the price of the items. For the items from the BLS website, this was simply a matter of using the cells.find feature again, finding the item on the Grocery Data tab, and then using range.offset to find the most recently updated price. For items added to the list by the user, I created a userform similar to the quantity form, asking the user to enter a best guess for the price. I saved this price as a variable, which was added to the Data Sheet tab next to the item (so that in the future, an If statement would save the user time by not prompting for a price each time the item was selected). The variable could also be printed on the List tab.

- Now that Item, Quantity, and Price were all saved as variables, and the correct department was still the activated cell in the List tab, it was a simple process of using the range.offset feature to print the three variables in the proper places below the active cell. I also used simple mathematics in the VBA code to multiply the price variable times the quantity variable, and then add that to the value already existent next to the “<u>Total</u>” cell.



12						
13	**<u>Pasta</u>**			**<u>Dairy</u>**		
14	*Item*	*Qty.*	*Price*	*Item*	*Qty.*	*Price*
15	Spaghetti and macaroni, per lb. (453.6 gm)	1.5	$1.32			
16				**<u>Cheese, Deli, & Meats</u>**		
17	**<u>Canned Foods</u>**			*Item*	*Qty.*	*Price*
18	*Item*	*Qty.*	*Price*			
19					**<u>Total</u>**	**$1.98**
20						
21						
22						
23						
24						
25						
26						
27						

- Now that the user could choose an item and successfully print it out on the List tab, with a running total being calculated, the next thing I needed to fine tune was the user's ability to add an additional item to the list. This involved adding a button to the first "Add Item" userform that allowed the user to click and enter in a new item. It also requested that the user include the unit by which this item was measured. The form would then concatenate the new item with a comma (to allow the Split function to work with newly added items) and with the unit of measure, and add it to the Data Sheet tab under "Additional Items." Once the item was added to the Data Sheet tab, the initial "Add Item" userform would re-initialize, thus repopulating the item list to include the newly added item.


Grocery List

Please select a grocery item from the menu

Close | Okay

Check box to verify department

Add Additional Item to List

Additional Item

Please enter the grocery item you want to add and include unit of measurement (ex: "Canned Peaches [per] 1 can")

Canned Peaches per 1 can

Cancel | Done

62	Wine, red and white table, all sizes, any origin, per 1 liter (33.8 oz)	Other foods at home	Beverages	
63	**Additional Items**			
64	Canned Peaches, per 1 can		Canned Foods	1.59
65	Macaroni, per 1 box		Pasta	0.5
66	Frozen Pizza		Frozen Foods	4.59
67	Pace Picante Salsa, per 1 large bottle		International Foods	2.85
68	This is a test, per 2 bags		Snacks/Chips/etc.	4
69	Canned Pears, per 1 can		Canned Foods	1.59
70	Bean Dip, per 1 can		Snacks/Chips/etc.	1.99
71	Doritos (nacho cheese), per 1 bag		Snacks/Chips/etc.	2.59
72	Frozen Hamburger, per 16 oz.		Frozen Foods	2.99
73	Chocolate Chips, per 8-oz bag		Baking Ingredients	[illegible].99
74	Anni, per 1 human		Baking Ingredients	1000000
75	Tortillas, per 1 12-pack		Breads	2.05
76	Annis, per 1 Anni		Cheese, Deli, & Meats	465
77	Canned Peaches, per 1 can			1.59
78	Cottage Cheese, per 1 container		Dairy	3.48
79	Jacob, per 1 Little Guy		Frozen Foods	5000
80	Baby Carrots, per 1 1-lb pckg		Produce	1.88
81	Cold Cereal, per 1 Box		Cereal/Breakfast Foo	3.5
82				
83				
84				
85				
86				

List | Grocery Data | Data Sheet

READY

Grocery List

Please select a grocery item from the menu

Malt beverages, all types, all sizes, any origin, per 16 oz. (473.2 ml)
Wine, red and white table, all sizes, any origin, per 1 liter (33.8 oz)
Canned Peaches, per 1 can
Macaroni, per 1 box
Frozen Pizza
Pace Picante Salsa, per 1 large bottle
This is a test, per 2 bags

This concludes the project for which I submitted my original proposal. Beyond the scope of this project I continually added features that were not required, but that I thought would make the process even easier and more helpful. If I had more time, I would delve into those in more detail, but given the time crunch, I will let the following suffice:

- *An option to clear all data (including the running price total) and restore the List tab to its original format.*
- *An option to change the view from "Normal" to "Page Break Preview" at the click of a button.*
- *A button allowing the user to quickly access the "Print Preview" feature*

- *Another feature I added to improve helpfulness is including next to the "Price" cell an additional cell, whose only purpose is to have a bottom border. Thus, when at the store, the user can easily place a check mark next to each item that is already selected and see what still needs to be purchased.*

All of these additional features are done in the hope that a Mac user who is unfamiliar with Microsoft Excel (such as my wife) can use the program without much hassle or confusion. I will be continually adding more features to make it even easier and more useable, such as a recipe section that automatically populates the list with ingredients needed for recipes.

Discussion of Learning and Conceptual Difficulties Encountered

I met and had to overcome many challenges throughout this project. The concepts ran anywhere from minor confusion to extreme, frustrating puzzles that I felt had no answers. I will limit my discussion to the more extreme cases.

Unit of Measure – The problem I ran into with units of measure involved the "Quantity" userform that asked the user to input a desired quantity for each item. I wanted to include the unit by which the BLS website measured the grocery items, but I had no idea how to accomplish this without hard-coding the unit of measure for each item. This would take a very long time with 60 items, and in my mind it defeated the purpose of writing the procedure in the first place.

Finally, I realized that every item had at least one comma within the string, and many items had several commas. I went through each item and confirmed that the final comma always preceded the unit of measure for that item. I remembered there was a function from our array section that would separate a string using a given criteria as the separator. I quickly opened up the array homework project in which we did this, relearned how to do it, and wrote a similar Split function in order to separate each line item. I then set the label in the "Quantity" userform equal to the Upper bound (the string after the final comma), such that it displayed the unit of measure for the selected item, and nothing else.

This gave me an idea on how to format the "Additional Item" userform, where the user entered a new item. I realized after some thought and experimentation that if I only provided the user with two text boxes to fill, I could add a comma myself (in the procedure) and not have to worry about the user remembering to include the comma or not. This way I could use the same split feature for additional items, and not just the original grocery items.

Repeat Items – I quickly realized that as my wife was looking at recipes and meal plans, she would want to add multiple quantities for the same item. For instance, if one recipe required a lemon and she added it to the list, then she came across another meal that required a lemon, the way my code was initially set up would add a second line for the same item (the lemon). So, though it wasn't required by the scope that I specified in my proposal, I went about trying to fix that issue.

Again, I spent a lot of time trying to figure out how to do this. I ended up realizing that after about 15 characters, every item in the list became unique to itself, even the very long ones. So all I did was create an If statement that compared the selected item to all of the items in the "List" tab, under the selected item's department. If there were no items in that department, the If statement simply ends, and the procedure continues to run as normal. However, if there are items under that department, I ran a Do Loop that compared the selected item to each item under the department in turn. If there was no match, the procedure would continue to run and insert rows where needed. If there was a match, the procedure would not insert any rows, but it would prompt the user for the desired quantity, save that quantity as a variable, then add that variable to the value already existing in the quantity cell.

This way the procedure could still multiply the additional quantity by the pre-existing price and add that to the total. The first quantity of that item would already have been calculated into the total, so the procedure only needed to calculate the new quantity added to the old. This also took me some time to figure out. One of the greatest challenges of this feature was solving the issue of where to include the code to insert rows and add a new item, versus simply adding to the existing quantity. I moved the code around and copied it several times throughout the procedure until it worked. I am not sure I did it in the most efficient way possible, but I got it to work, so I was satisfied with it.

<u>The Ribbon</u> – The final challenge I will discuss before submitting this is the Ribbon X feature. Again, this concept was a source of great frustration. The day we discussed this topic in class I was excited to learn about it. I thought it would be useful to know. I downloaded the required program from Learning Suite, installed it on my computer without any problem, and began to follow along in class. As soon as I tried to open the program, however, an error occurred and I could not go any further. This was especially discouraging to me because it wasn't the first time that I ran into technical difficulty, fell behind, spent the rest of class trying to catch up, and got to the end feeling like I learned nothing.

Exasperated, I made the decision that trying helplessly to follow along would simply not be worth the time, and it would be better spent working on my VBA project. So I did that instead, only to find out at the end of class that it was an "expectation" that we included this Ribbon feature in our project. I was quite disheveled and even more discouraged now. I had absolutely no idea what a Ribbon X was and how it worked, and now I was expected to include it in a project that constituted a significant portion of my grade. I made up my mind that I would talk to Professor Allen about it, make sure I even needed to include it in my project (he mentioned that some projects that called specific sheets might not need it), and get whatever help I needed if I still was required to include the Ribbon X feature.

So I went to his office on Friday afternoon when I finally had some spare time, and I saw that he was gone in meetings that originally were to end by the time I got there, but then were extended to the end of the day. Annoyed, I decided I would just download the document and give it my best shot. Sad mistake. So I downloaded the explanation, understood very little about it, and gave my best attempt at following the instructions and including the Ribbon X feature. This is the once concept/challenge that I faced but had no success at overcoming. At this point I became annoyed again, decided that I had more important features to improve/perfect in my project, and finished those, knowing that I gave the best attempt I could on understanding and implementing the Ribbon X feature. Since the requirement was sprung on us in one of the last days of class, and I had no understanding of the Ribbon X, I made my decision with the hope that the expectation to include it was a somewhat lenient one.

Assistance

Besides Google searches and previous class assignments, I received no outside help with this project. That is my best explanation for the relatively simple-minded, basic code found in this project. While I worked on the project, it occurred to me that I am no Excel VBA master. I can get the application to do what I want it to do, but I implement much less complicated code and tend to stick to the basics that I know and remember how to use.