

FINAL PROJECT



Jason Howell
MBA 614
Spreadsheet Automation / Modeling
Sec 001
April 11, 2012

Executive Summary

Business Description

For my final project, I chose to work with a gourmet cupcake bakery where my wife is currently employed, Cupcake Chic (pronunciation: \kŭp-kāk' shĕk\). Cupcake Chic is located in Orem, UT and their cupcakes are made from scratch with the finest ingredients to create a fresh and delicious product.

At the end of every quarter (3 months), every restaurant in the state of Utah needs to file two tax forms (TC-62S and TC-62F) to properly account for the combined sales and use tax, grocery food tax, and prepared food (restaurant) tax that they owe to the state. Since Cupcake Chic is a for profit business in the restaurant industry, they need to track their daily sales so that they can properly file these quarterly tax returns.

Overview of System Built

The VBA system I built for Cupcake Chic has been designed to streamline their process for tracking sales and filing both quarterly tax returns. The program I wrote consisted of three main processes: 1. tracking daily sales, 2. generating monthly reports, and 3. compiling quarterly tax returns.

1. Tracking Daily Sales

At the end of each day, the cash register at Cupcake Chic produces a receipt that has the total sales for the day, broken into several categories. In order to easily and effectively transfer this information into an excel spreadsheet, I created a user form that allows the user to simply input the total daily sales for each category of products at the end of each day. (Please note that all sales figures contained within this report and within the excel file are completely fictional and do not correspond in any way to actual sales figures for Cupcake Chic.)

2. Generating Monthly Reports

Along with tracking daily sales, the owners at Cupcake Chic also wanted to be able to track their total sales at the end of each month. To generate this report, I created a sub procedure that gathers all the sales information for each day during the month and creates a report that shows the total sales for each month, broken down by product category.

3. Compiling Quarterly Tax Returns

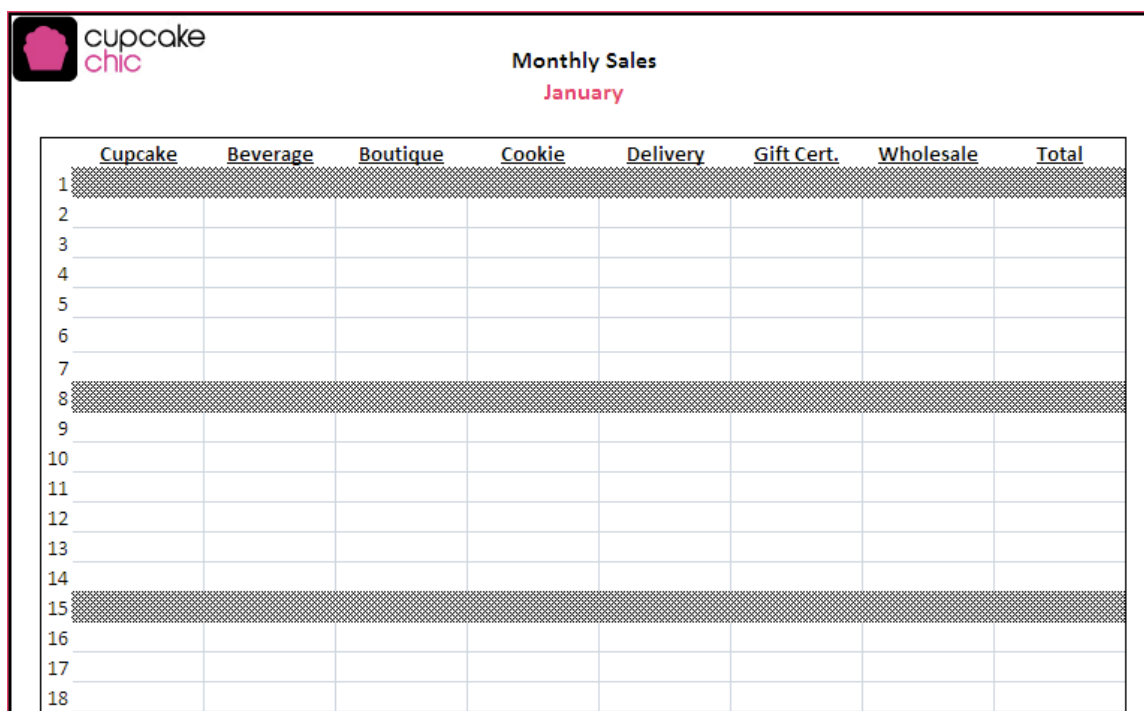
Prior to completing this project, the owners at Cupcake Chic had to fill out the two quarterly tax returns by hand. I created a sub procedure that will completely and

correctly fill out both quarterly tax returns, based on the sales of the prior three months.

Implementation

Tracking Daily Sales

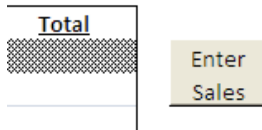
At the end of every business day, a daily sales receipt is printed from the cash register at Cupcake Chic. This report lists the total sales for the day, categorized among seven groups: cupcake, beverage, boutique, cookie, delivery, gift cert., and wholesale. Since I am not aware of any device that will get information on a printed receipt to copy to excel, I figured these numbers would have to be inputted manually by the user. I decided the easiest way to accomplish this (from the users standpoint) would be to create a user form that the user can use to input sales figures. The first thing I did is create twelve tabs in an excel file that correspond to each month. I then listed the seven categories across the top, with the days running down the side.



	Cupcake	Beverage	Boutique	Cookie	Delivery	Gift Cert.	Wholesale	Total
1								
2								
3								
4								
5								
6								
7								
8								
9								
10								
11								
12								
13								
14								
15								
16								
17								
18								

This would serve as my template, and these cells would be populated as the user interacts with the user form. I also decided to cross out all the days in each month that were Sundays, since the store is closed on the Sabbath. There are no Excel functions used in populating this form.

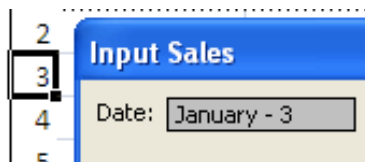
The next thing I did was create a button ("Enter Sales") that the user can push and it will bring up the user form. This button is placed near the top right corner of the monthly sales template. I considered putting the button in the ribbon, but when given the choice, the client specifically said that they would prefer to have the button on the page.



The next step was to create the user form that would be used to input the daily sales numbers. The following is what the finished user form looks like:



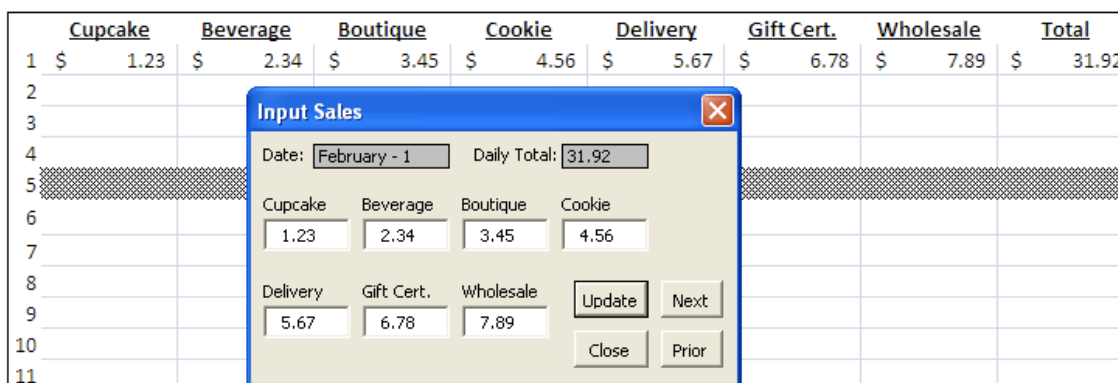
There are several features of this user form that make it especially useful for the user. When the user pushes the "Input Sales" button, this form will pop up and the active cell will become the date that corresponds with the row the active cell was on when the button was pushed. Also, the date on the form will correspond with that new active cell.



If the active cell was on a row higher than the row that contained the first day of the month, then when the user pushes the button the active cell will become the first day of the month (i.e. January – 1). Also, if the active cell were on a row below the last day of the month, the new active cell would become the cell that contains the last day of the

month, and the date on the form would correspond with that date. I accomplished this by writing an “if” statement where if the active cell were above the first row of the month (or below the last row of the month) then the active cell would become the first row of the month (or last row of the month).

The form has seven text boxes that correspond with the seven categories that Cupcake Chic uses. If the user is on a row where sales numbers have been previously input, these values will show in the seven text boxes, and the daily total label will show the total for the day.



	Cupcake	Beverage	Boutique	Cookie	Delivery	Gift Cert.	Wholesale	Total
1	\$ 1.23	\$ 2.34	\$ 3.45	\$ 4.56	\$ 5.67	\$ 6.78	\$ 7.89	\$ 31.92
2								
3								
4								
5								
6								
7								
8								
9								
10								
11								

Input Sales

Date: February - 1 Daily Total: 31.92

Cupcake: 1.23 Beverage: 2.34 Boutique: 3.45 Cookie: 4.56


Delivery: 5.67 Gift Cert.: 6.78 Wholesale: 7.89

Update Next Close Prior

The user can quickly input the daily sales for each category and tab across the text boxes. When all the sales have been entered for the day, the user can hit the “Update” button on the form to save the values to the spreadsheet. The “Update” button will also add all of the seven categories together to get the total amount of sales for the day, and then it will display this number in the daily total label and on the spreadsheet. Hitting the close button will unload and close the form without saving anything to the spreadsheet.

Once the user has entered all the sales data for a particular day and has hit the “Update” button, the user can then hit the “Next” or “Prior” button to move to the next or previous day. When the user moves to the next or prior day, the active cell will update to the corresponding date cell on the appropriate row. If the user is on the first day of the month and then hits the “Prior” button, the active cell will stay on the first day of the month. The same is true if the user hits “Next” and they are already on the last day of the month, they will simply stay there. This was accomplished through a simple “if” statement.

Another neat feature of this form is that if the user is on a date that falls on a Sunday, all of the text boxes will show "-". If the user changes the values in the text boxes to sales numbers and hits "Update", nothing will happen. This will not write the sales numbers to the form. This is useful because the store is never open on Sundays. I was able to accomplish this by writing an "if" statement where if the background of the active cell row was "crisscross" then the update button would not save to the spreadsheet.



The user can also use this form to edit previously input sales data for any day. As stated above, if the user is on a date that already has had data put in, that data will show in the form. The user can change those sales figures and hit "Update" to write the new sales figures and daily total to the form. When the user is finished inputting new daily sales or updating old figures, hitting the "Close" button will unload and close the form.

Generating Monthly Reports

Along with tracking daily sales, I also created a sub procedure that would populate a monthly sales form. To accomplish this, I first created a button (named "Monthly Report") on each of the 12 monthly sheets in the excel file. I also chose to put this button on the sheet instead of the ribbon because of client preference. When the user pushes this button, the monthly sales form will be populated with the total monthly figures.

Monthly Report	
April	
Total	
Enter Sales	
Monthly Report	
Cupcake	\$ -
Beverage	\$ -
Boutique	\$ -
Cookie	\$ -
Delivery	\$ -
Gift Cert.	\$ -
Wholesale	\$ -
Total	\$ -

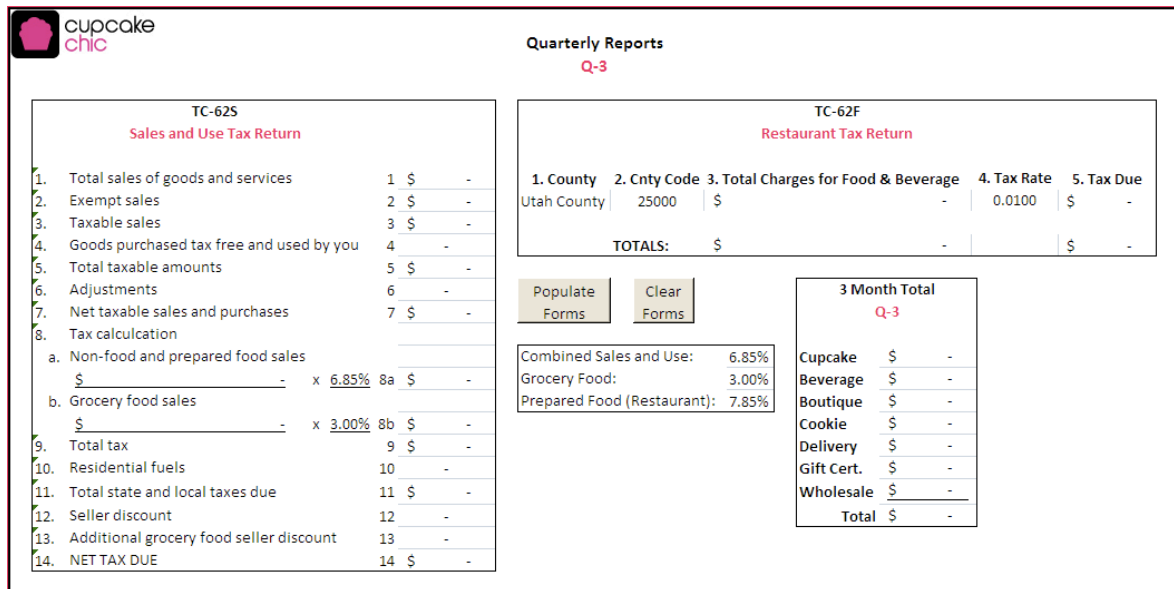
The monthly report is located on the same sheet as the daily sales figures. In order for this report to populate, I had to create a sub procedure that would take all of the daily sales figures for each category for the month, add them up, and return the values to the form on the spreadsheet. I was able to accomplish this by utilizing the "next" statement in VBA to add up all the rows for each category. I was also able to set up a variable called "lastRow" that would find the last row of the month. This way, I could use the same coding to add up all the values for each month, even though each month has a varying number of days.

Monthly Report	
May	
Enter Sales	
Monthly Report	
Cupcake	\$ 3.57
Beverage	\$ 7.58
Boutique	\$ 7.68
Cookie	\$ 11.57
Delivery	\$ 12.30
Gift Cert.	\$ 7.90
Wholesale	\$ 9.30
Total	\$ 59.90

Compiling Quarterly Tax Returns

For the final portion of my solution, I created a sub procedure that would populate the two tax forms that are due every quarter. Before I started programming, I created four more sheets in my workbook (Q-1, Q-2, Q-3, Q-4) and placed them at the end of every

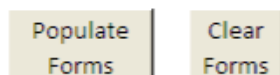
three months. On these quarterly sheets, I made a section for a 3 month total, the two tax returns, and a tax rate area.



The only values that the user would directly input into this spreadsheet are the tax rates, located in the middle of the sheet. Since the program used to calculate the figures for the tax forms draws its tax rates from the spreadsheet (they are not hard coded in the program), the user would simply need to update these cells if the tax rates change (they will most likely remain constant for years).

Combined Sales and Use:	6.85%
Grocery Food:	3.00%
Prepared Food (Restaurant):	7.85%

There are two buttons on this sheet, "Populate Forms" and "Clear Forms". The populate button will call a sub procedure that will populate all three forms (two tax forms and 3 month total). The clear button will clear all the values in all the forms.



I created one sub procedure that will populate all three forms. In order to populate the tax forms, I first had to gather data for the preceding three months. The program I created will grab data from the three monthly reports on the previous three sheets.

After adding all the totals per category, the results are placed in the “3 month total” form of the quarterly worksheet. I did this by creating variables that were used to store and add values from all three months. It is also important to note that the same sub procedure is used when completing any of the four quarterly reports. I accomplished this by having “if” statements in my solution, where each “if” statement condition was whether the sheet was named “Q-1”, “Q-2”, “Q-3”, or “Q-4”. Depending on which statement was true, the program would only grab data from the three months that correspond with the correct quarter.

3 Month Total		
Q-4		
Cupcake	\$	4.88
Beverage	\$	11.78
Boutique	\$	8.77
Cookie	\$	5.91
Delivery	\$	7.96
Gift Cert.	\$	8.55
Wholesale	\$	15.10
Total	\$	62.95

Once the totals for each of the seven categories have been added for the past three months, the tax forms are now ready to be filled out. To complete this, I created variables and used some rudimentary math equations to populate the tax forms. Some lines of the tax forms include all the sales numbers, and some lines only include certain categories of the sales figures. My program correctly identifies which sales figures belong on which lines on the tax forms, and it will compute how much tax is due on each form. It is important to note that no excel functions have been used in this program, and no excel functions exist on these quarterly sheets.

TC-62F				
Restaurant Tax Return				
1. County	2. Cnty Code	3. Total Charges for Food & Beverage	4. Tax Rate	5. Tax Due
Utah County	25000	\$ 10.79	0.0100	\$ 0.11
TOTALS:		\$ 10.79		\$ 0.11

TC-625				
Sales and Use Tax Return				
1.	Total sales of goods and services	1	\$	62.95
2.	Exempt sales	2	\$	31.61
3.	Taxable sales	3	\$	31.34
4.	Goods purchased tax free and used by you	4		-
5.	Total taxable amounts	5	\$	31.34
6.	Adjustments	6		-
7.	Net taxable sales and purchases	7	\$	31.34
8.	Tax calculation			
	a. Non-food and prepared food sales			
	\$ 19.56 x 6.85%	8a	\$	1.34
	b. Grocery food sales			
	\$ 11.78 x 3.00%	8b	\$	0.35
9.	Total tax	9	\$	1.69
10.	Residential fuels	10		-
11.	Total state and local taxes due	11	\$	1.69
12.	Seller discount	12		-
13.	Additional grocery food seller discount	13		-
14.	NET TAX DUE	14	\$	1.69

Once the tax forms have been populated, all the user has to do is transfer the numbers from the spreadsheet to the actual tax forms for filing. There is no need for the user to make any calculations of their own when filing their quarterly tax returns.

The system that I have built will greatly reduce the amount of time and energy that is required to track daily sales, generate monthly reports, and compiling quarterly tax returns. Prior to the completion of this system, the owners were doing almost all of this by hand, which was a very tedious endeavor. This system will not only reduce the amount of time greatly to complete the desired tasks, but it is also error proof (mathematically speaking).

Learning and Conceptual Difficulties

Learning

There were several real world lessons that I learned as I worked through and completed this project. The first is that real world problems/projects that I actually have a vested interest in are far more enjoyable to complete than homework or projects from class. I really enjoyed having the freedom to create a solution that I came up with on my own, not one where I had to follow certain guidelines. Even though it took more work, I enjoyed figuring out my solution from scratch by mapping out what needed to be done

and making adjustments to my plan as needed. Since I really cared about my final product, I took the time to make sure that my program worked perfectly and that my end spreadsheets/workbook looked professional.

I also learned that when creating a VBA program, you need to make sure that the user interface is not very complicated, especially if you have low tech users (users who do not have a lot of computer experience). VBA does a great job of allowing a programmer to make a complicated program that will run with the click of a button, so usually this isn't an issue. But I did discover that most users want the interface to be as simple as possible. For example, I originally had put my buttons on the ribbon, but after meeting with my client and giving them the option, they chose to have the buttons on each spreadsheet. This might not be the ideal placement, but for the client this made more sense for them when they worked with the workbook.

One last lesson that I learned was that in the real world, usually a client will prefer a final product that looks flawless but is not as powerful, as opposed to a final product that is very powerful but looks less professional. Even though I did not write a program that is super powerful or doing crazy stuff, my client was ecstatic with the finished product because it simply did what they wanted and the interface looked very professional. The important lesson here is to remember to make sure that the finished product looks great, no matter what the actual program does.

Difficulties

In working through a solution for Cupcake Chic, there were two major difficulties that I encountered that I wasn't able to completely solve. I was able to work my way around both of these issues, but I would have really liked to have figured out a better solution to both of them.

The first issue was that I wanted to find some other way for the user to transfer the daily sales data from the receipt to the excel workbook. My solution (user form) was a good solution, but I would have liked to have done some more research to see if there is some sort of device that could read the information on a daily sales receipt and directly transfer that to the VBA program. I assume that I would need to use some sort of alternate input device (such as the barcode scanners we used in class) that would scan the receipt and enter the data into excel. If there is such a device, I would have liked to have utilized it in my solution. That aside, I do think my user form was a good way to have the user input the sales data.

The second difficult issue that I encountered had to do with the two quarterly tax forms. My actual solution filled out this information in an excel sheet, and then the user would copy that information to the actual tax forms by hand. I wish I would have been able to have my program take the numbers that belong on the tax forms and actually put them in the .pdf tax forms, so that all the user had to do was run the program and print off the completed forms. This would eliminate the step of having the user hand fill out the tax forms. I'm pretty sure there is some way to accomplish this, but I was not able to figure it out. The solution that I settled with still saves the client a lot of time in completing this process.

Assistance

I did not receive substantial assistance from another person or persons on this project.