MBA 614 – Spreadsheet Automation & Modeling

Final Project – Expense Recorder & Analyzer

Orlando Lopez
12/9/2010

Executive Summary

The idea for the project I created came as a result of my family's need to carefully track our expenses. I wanted a means to track where most of our money is spent so as to ensure that our budget is balanced. With that in mind, the project that I created enables users to input data into the workbook that is subsequently exported via text file and stored in an "Expense Data" folder. The program stores each text file by date, making it possible for the user to store a new expense data file per day, but also giving the user the ability to decide how often he/she decides to do so.

Along with the capability to store expense data, I needed the ability to analyze that stored data. Thus, the program that I created has the ability to import data from the "Expense Data" folder for any user-requested date range. The program then sums up all the imported expense data and creates two charts – a bar graph that details the total per expense category and a pie chart that breaks down the total expenses into percentages per category. The program is able to create charts for many different date ranges at a time to enable the user the ability to analyze how one date's expenses affected the expenses for the overall range of dates. Again, the benefit here is that the program has given me the ability to analyze one day's worth of data or years of it, simply by entering a date range.

Implementation Documentation

For the "Expense Recorder and Analyzer" project, I created two forms, an input form and an analysis form, each initiated by a one of the two buttons on the "Home" worksheet in the workbook. The input data form informs the user to input the date in "mm/dd/yyyy" form and then allows the user to input expense amounts for different expense categories. Once the expenses are recorded via the submit button, the form copies the inputs into a worksheet called "Data". Immediately after the inputs are recorded into the "Data" sheet, the contents of the sheet are then saved into a text file which is named after the date that the user entered as the expense report date. The file is consequently exported into and saved in a folder on the C drive called "Expense Data" (path: C\Expense Data).

It is important to note two things here: (1) the input form checks to see whether the expense data folder exists and creates it if it doesn't. I wanted to do that to ensure that the program would work for everyone (i.e., there wouldn't be an error about the folder not existing). (2) I deliberately made the path simple so that anyone who downloads my program can use it without need for much customization. Below are snapshots of the home page, the input form, and the expense data folder's contents after inserting an expense report for the date of 12/4/2010.

Exhibit 1 – The Program's Home Page

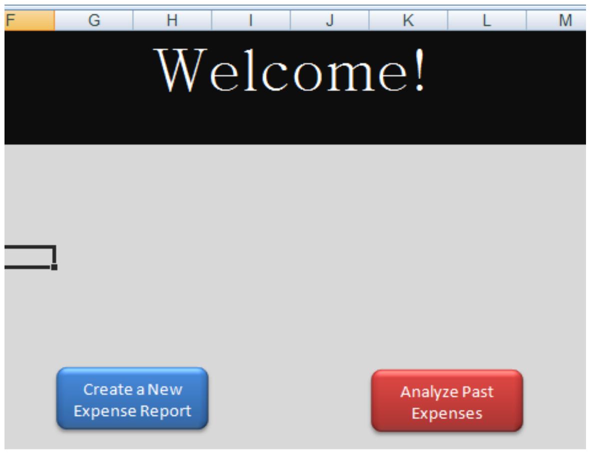
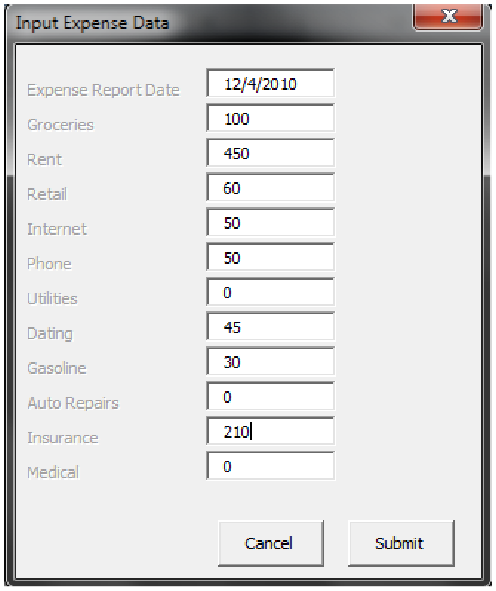
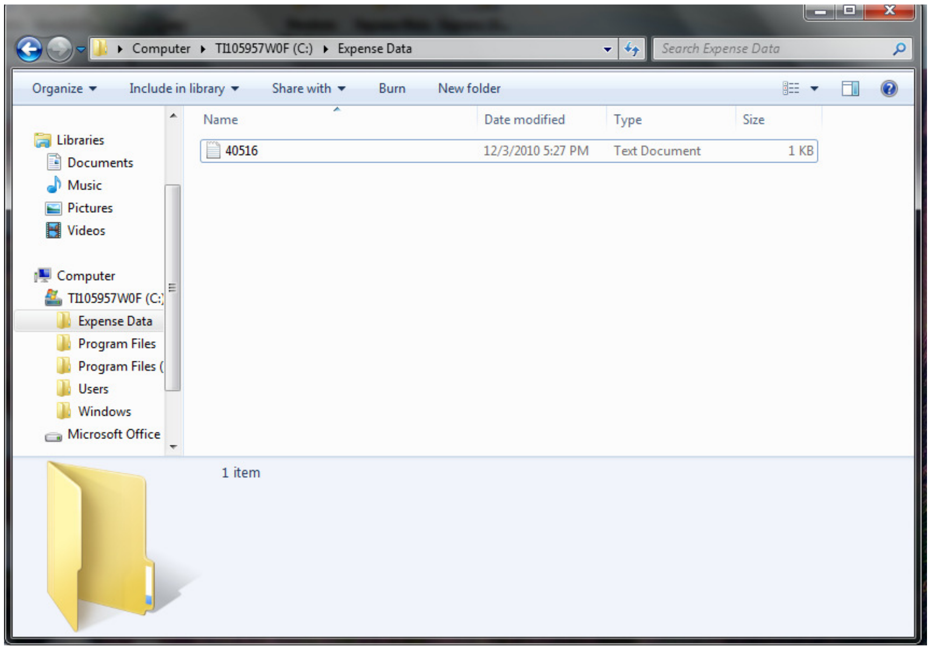


Exhibit 2 – The Filled Input Form


Input Expense Data

Expense Report Date	12/4/2010
Groceries	100
Rent	450
Retail	60
Internet	50
Phone	50
Utilities	0
Dating	45
Gasoline	30
Auto Repairs	0
Insurance	210
Medical	0

Exhibit 3 – The Expense Folder After the Inputs Are Processed

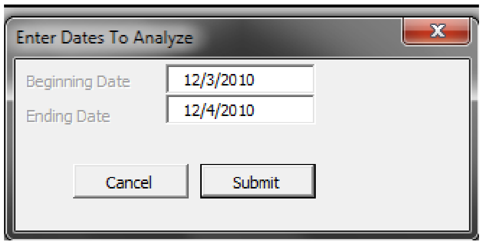

Computer ▸ TI105957W0F (C:) ▸ Expense Data

Name	Date modified	Type	Size
40516	12/3/2010 5:27 PM	Text Document	1 KB

1 item

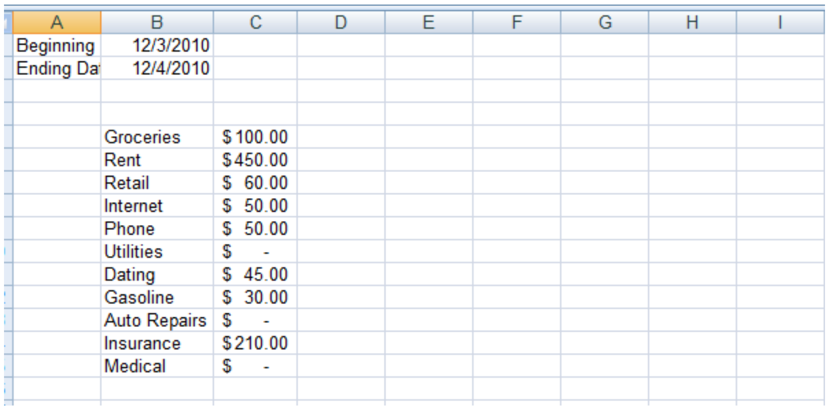
The analysis form does a few things in order to process the user's request. First, the form asks the user for a date range (i.e., a beginning date parameter and an ending date parameter) which will subsequently be used to import the data that fits within that range. To import the date, the program converts both user-input dates into numbers and then searches (via a loop) through all the files in the expense data folder in order to find and import the records that are needed and that fit the range that the user suggested. Once the files are imported, the program deletes all unnecessary data (i.e., the column names that are repeated) that was in the imported text files and then uses an algorithm to sum up the totals for each expense category. It then takes the category names column and the amount totals column and creates a bar graph that shows the totals per category, as well as a pie chart that shows expense percentages for each category. Each graph is then assigned to its own sheet (as opposed to being an object in the analysis sheet). Both graphs contain the user-inputted range on their title. The following are snapshots of the analysis form, the analysis sheet after the import but before the formatting, the analysis sheet after the formatting and the two charts that were created in the process.

Exhibit 4 – The Filled Analysis Form



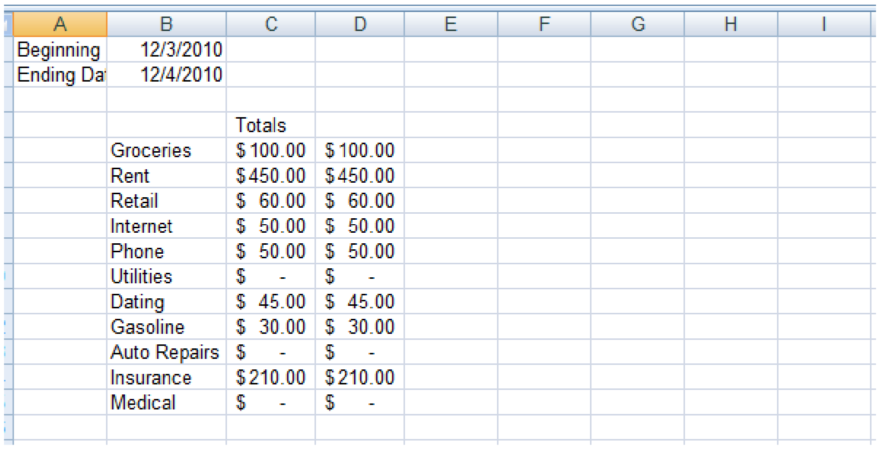
Beginning Date	12/3/2010
Ending Date	12/4/2010

Exhibit 5 - The Imported Information (Unformatted)



A	B	C	D	E	F	G	H	I
Beginning	12/3/2010							
Ending Da	12/4/2010							
	Groceries	$ 100.00						
	Rent	$ 450.00						
	Retail	$ 60.00						
	Internet	$ 50.00						
	Phone	$ 50.00						
	Utilities	$ -						
	Dating	$ 45.00						
	Gasoline	$ 30.00						
	Auto Repairs	$ -						
	Insurance	$ 210.00						
	Medical	$ -						

Exhibit 6 – The Imported Information (Formatted)



A	B	C	D	E	F	G	H	I
Beginning	12/3/2010							
Ending Da	12/4/2010							
		Totals						
	Groceries	$ 100.00	$ 100.00					
	Rent	$ 450.00	$ 450.00					
	Retail	$ 60.00	$ 60.00					
	Internet	$ 50.00	$ 50.00					
	Phone	$ 50.00	$ 50.00					
	Utilities	$ -	$ -					
	Dating	$ 45.00	$ 45.00					
	Gasoline	$ 30.00	$ 30.00					
	Auto Repairs	$ -	$ -					
	Insurance	$ 210.00	$ 210.00					
	Medical	$ -	$ -					

Exhibit 7 – Sample Bar Graph

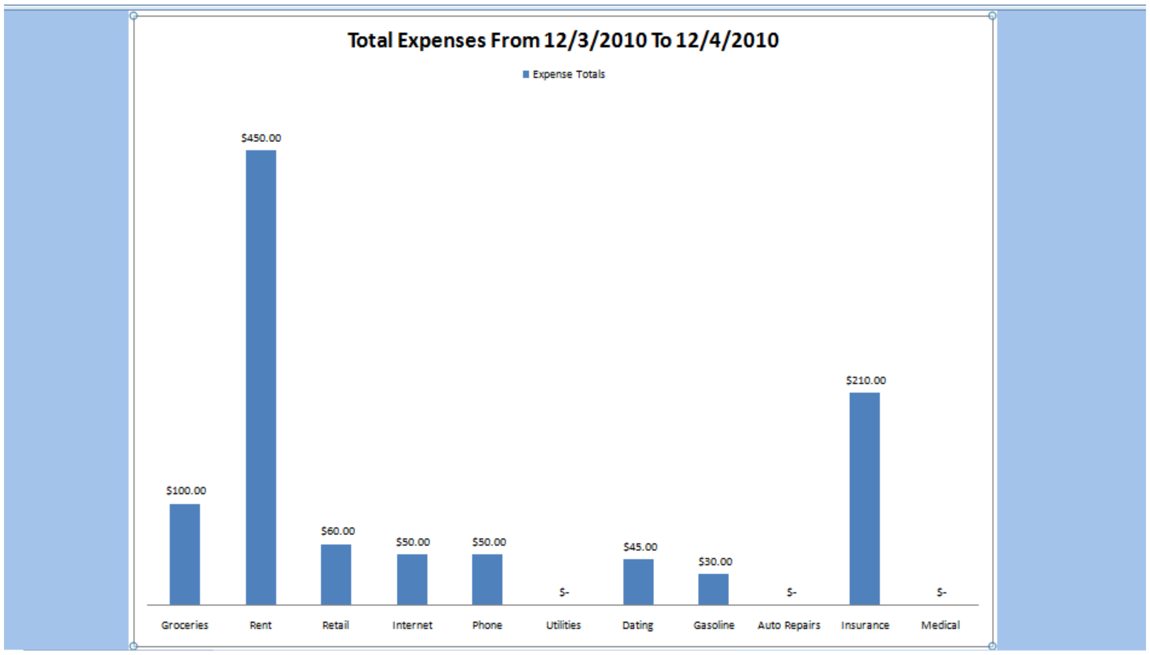
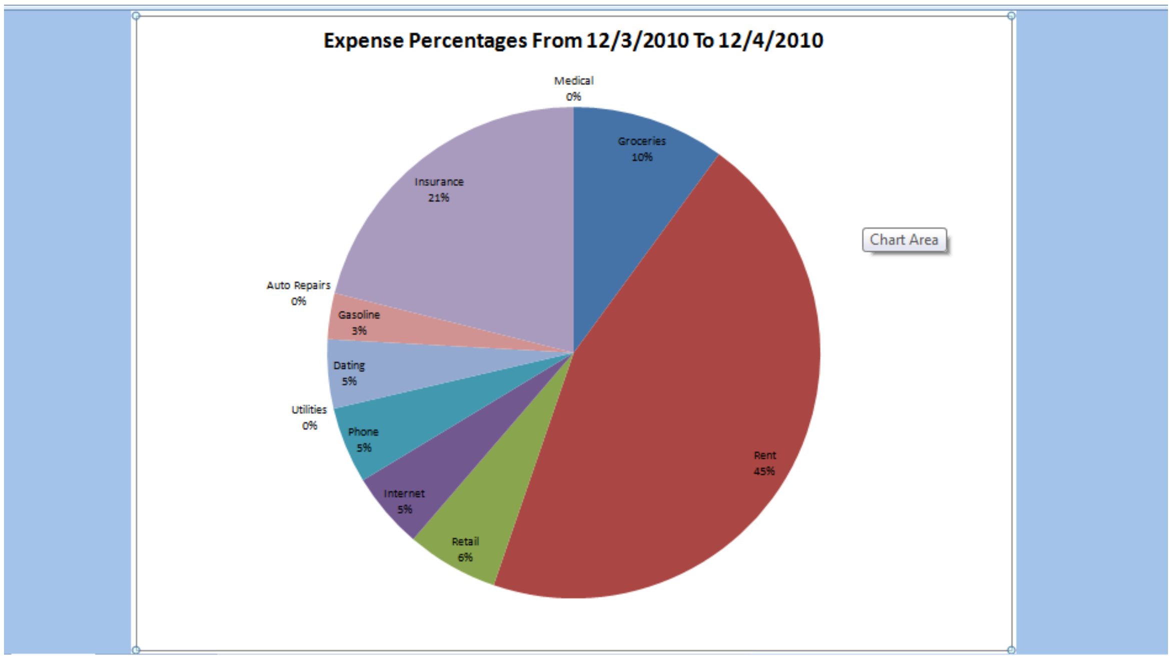


Exhibit 8 – Sample Pie Chart



Learning and Conceptual Difficulties

In retrospect, writing this program was a lot more complicated that I expected. I experienced the greatest conceptual difficulties figuring out how to save and export the text files, trying to properly format the expense data once it was imported, and working with the imported data to create the charts.

Saving and exporting the files was difficult because I was trying to do so in a way that would improve (or at least not hinder) my ability to import those same files in the future. Initially, I considered simply using the actual, user-input expense report date as a ways to save it but then I realized two things: (1) files cannot be saved if the name contains "/" and (2) attempting to create a loop that enabled me to upload files based on whether the would-be-imported text file name is larger than one date and smaller than another (both dates which could vary each time the user ran the program) seemed a lot more complex than it needed to be. That is why I decided to use Microsoft's date algorithm to my advantage. So, rather than using an "actual" date, I decided to let the user input the date and then convert it to "general" format, which simply turned each date into a sequence of numbers that could easily be imported in the future.

Properly formatting the expense data once it was imported was slightly related to the exportation process. When I was trying to figure out how to export the data, I decided that I didn't simply want a column of numbers. That is, I wanted to export the data along with the expense category titles so that I could open any expense report text file from the data folder and still know what each expense value was. Of course, that decision presented the problem of having repetitive information when the data was imported (i.e., the expense categories appeared once for each file imported); the main issue being that I never knew how many expense files would be imported. After a lot of pondering, I solved this problem by creating a for loop that would count the number of columns being used in the import and would then subsequently delete every other column (the "repeats") until all that was left was the numeric values for each import, except the first instance of the imports; which I left intact because I needed the category names to create the graphs.

Creating the graphs presented the final challenge. In creating the graphs I encountered two challenges. The first was that I needed the graph to somehow sum the values for each category in order to come up with a total for that category. I solved this problem by creating a loop that added a new value into the existing value of a variable until it reached the end of the row and then did the same for the next rows in the column until it got to the end. The second problem was that I wanted to enable users to create more than one graph, but creating an excel macros made it impossible to do so because excel assigns names to each graph that cannot be repeated when a new graph is created. I solved this problem with a mix of internet searching (e.g., how to create a new graph in VBA) and macros code (e.g., recording a macro to see how to format the chart into a format I like). I used both sets of knowledge to find out which attributes were necessary and which weren't; which lead to know that "chart name" was an optional attribute and, thus, not necessary to create a chart. Thus, the charts that are created keep their default, Excel-appointed name, which enables the user to create and delete as many graphs as he/she chooses.

After finals are over, I intend to improve the program by adding the ability for the user to also input pay data and try to create revenues vs. expenses graphs to create breakeven analyses.

Conclusion

In conclusion, although moments of difficulty arose during the process of creating this project, I feel that the amount of things that I learned far outweigh the number of moments of frustration I experienced. I hope that you have been able to see the effort that I put into this assignment and also the growth that has come as a result of it.