David Avery
MBA 614
Final Project

Automated Bookkeeping in Excel

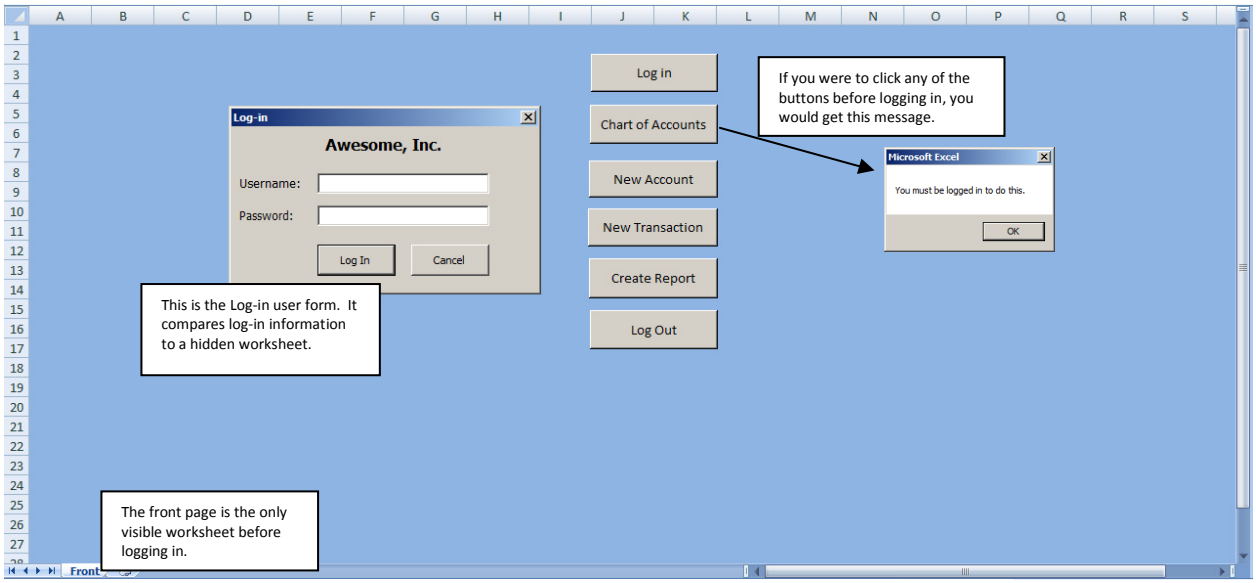
Executive Summary

Many accounting students try to practice their accounting skills by working as contract bookkeepers throughout school. Many times, clients that hire these student-bookkeepers do have not purchased bookkeeping software because they don't feel like they can justify the expense. In these cases, the student is forced to cover the costs of a bookkeeping program, such as QuickBooks. My completed project is a simple Excel-based bookkeeping program that performs many of the same bookkeeping functions that QuickBooks does.

This report will take you through a step by step guide of what this program does and how you operate it.

Front Page

Upon opening the workbook, you are immediately asked to log in. The log in screen itself is a user form, but it verifies log-in information off of a worksheet that is hidden within the workbook. I've taken extra steps to protect the log-in information by writing an event-based subprocedure where any time someone attempts to access that worksheet it immediately hides itself, thereby making it completely inaccessible. Also, in order to protect company information I've written code which hides everything but the front worksheet until acceptable log in information is submitted, and I've also blocked any other function from working without valid log-ins. The following screenshot illustrates the front page:



For all workbooks, the default username and password are both "admin".

It is also important to note that the workbook is protected from any sheets being added, as are all worksheets protected from any changes being made. I actually wrote subprocedures for protecting and unprotecting the workbook and worksheets which I use frequently throughout the program whenever any changes need to be written. There are several examples of when this would happen, which I will address throughout this paper.

After logging in, all of the functions of the workbook are available, and all worksheets are viewable, with the exception of the users worksheet. This is shown below:

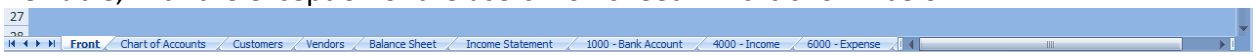
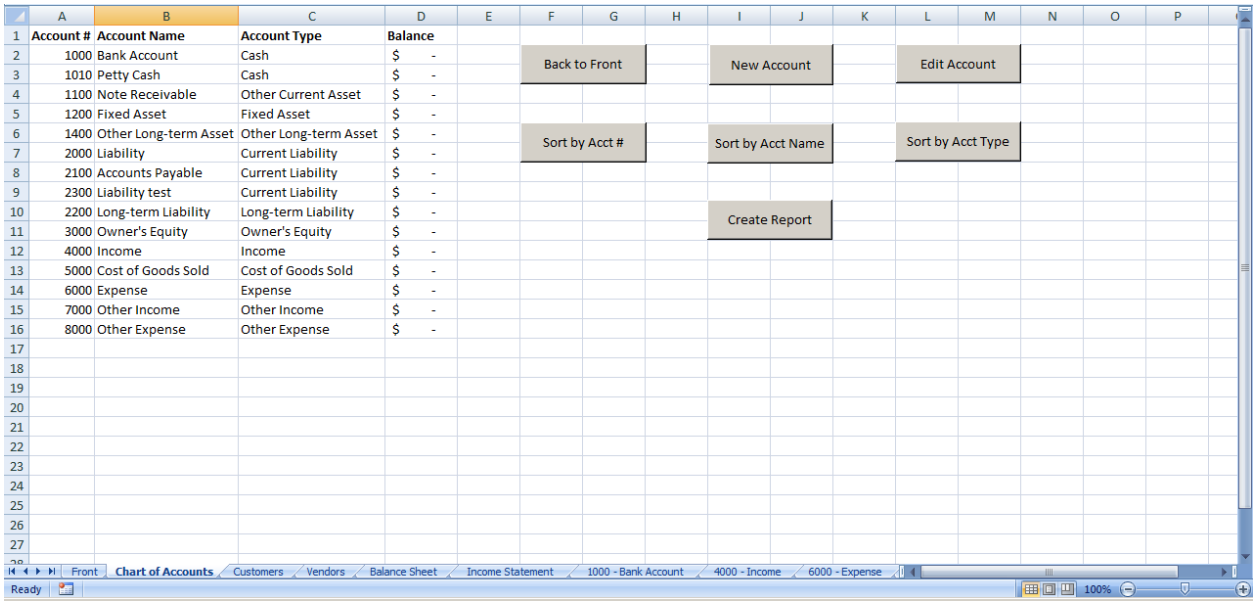
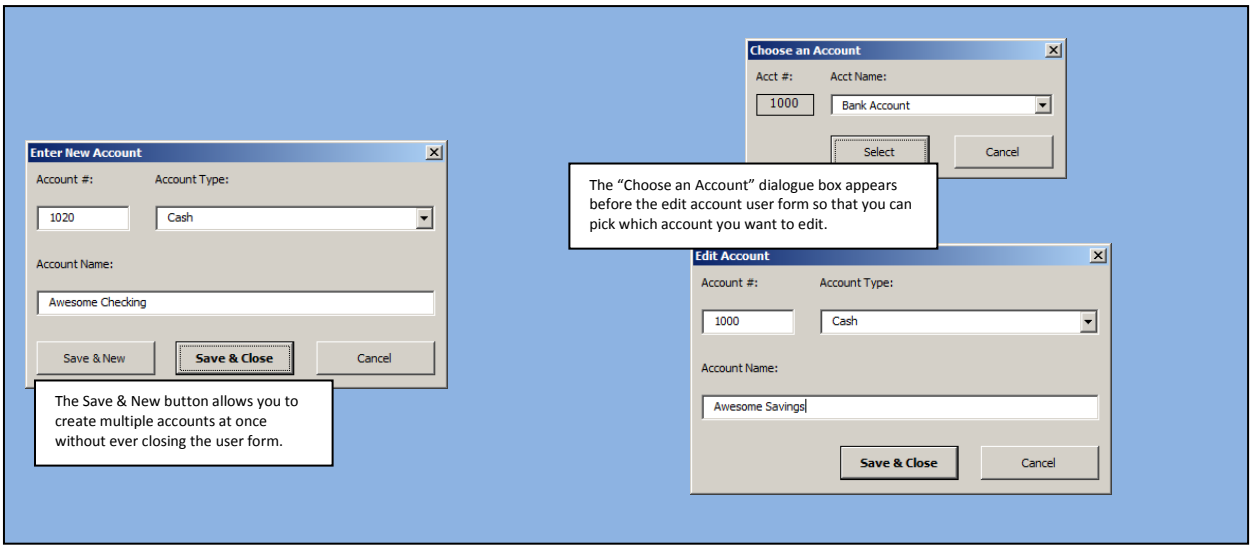


Chart of Accounts

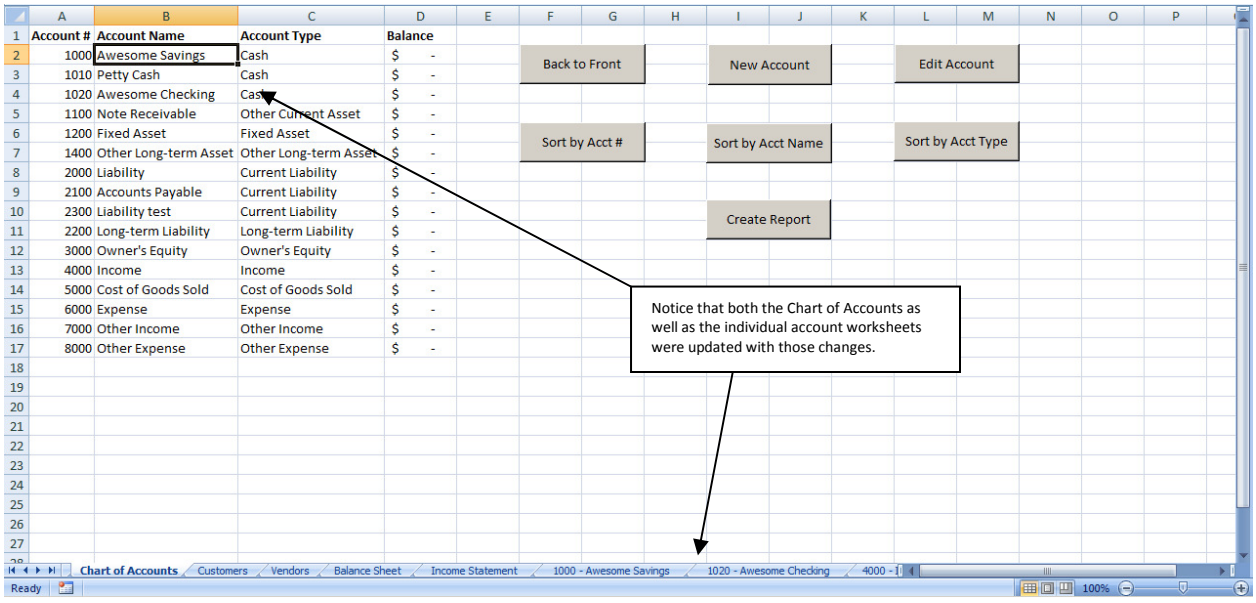
A necessary function of the program is having the chart of accounts. This is the central place in the workbook where any bookkeeping function can be accessed. The screenshot below shows the chart of accounts:



The chart of accounts automatically looks up each accounts balance and refreshes it every time the chart is activated. This is a helpful snapshot of company financial information. Also, to help organize information, I've included buttons that will sort the accounts by name, number, or by the order in which they appear on the balance sheet/income statement. Also, if you wanted to access an account, you would just have to double click on that account's name, and excel will automatically activate that worksheet. To do this I had to unlock the cells in column B. I was hesitant to unlock those cells because it would allow a user to change the account name without using the existing automated process. To mitigate this, I wrote an event based procedure that prevents changes from being made to any unlocked cell on the chart of accounts. The chart of accounts also allows users to add new accounts to their company's chart, as well as edit existing ones. The user forms for these functions are shown below:



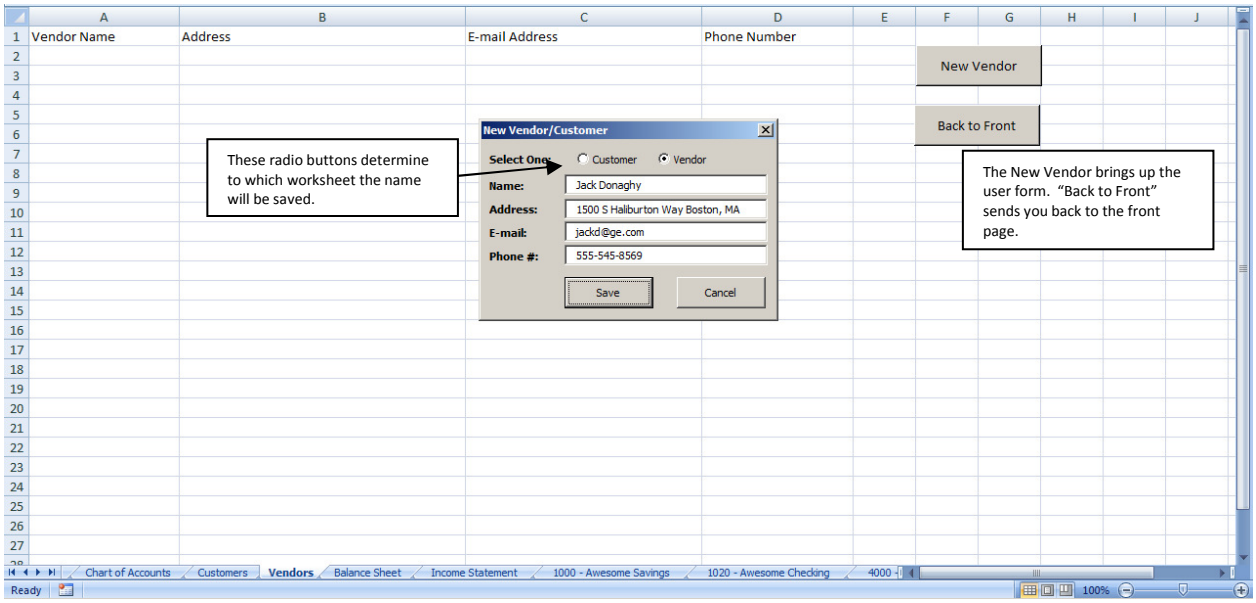
The account type combo boxes on these forms are pre-loaded with the different types of accounts you would regularly find on a balance sheet or income statement. You can only select from those specific types of accounts, and cannot create your own account type. Upon saving the new or edited account, the chart of accounts will be updated, and a worksheet with the account's number and name will be created (or updated). Note that in the above example, the name of "1000 – Bank Account" account was changed to "1000 – Awesome Savings". The following screenshot shows the updated chart of accounts:



Customers and Vendors

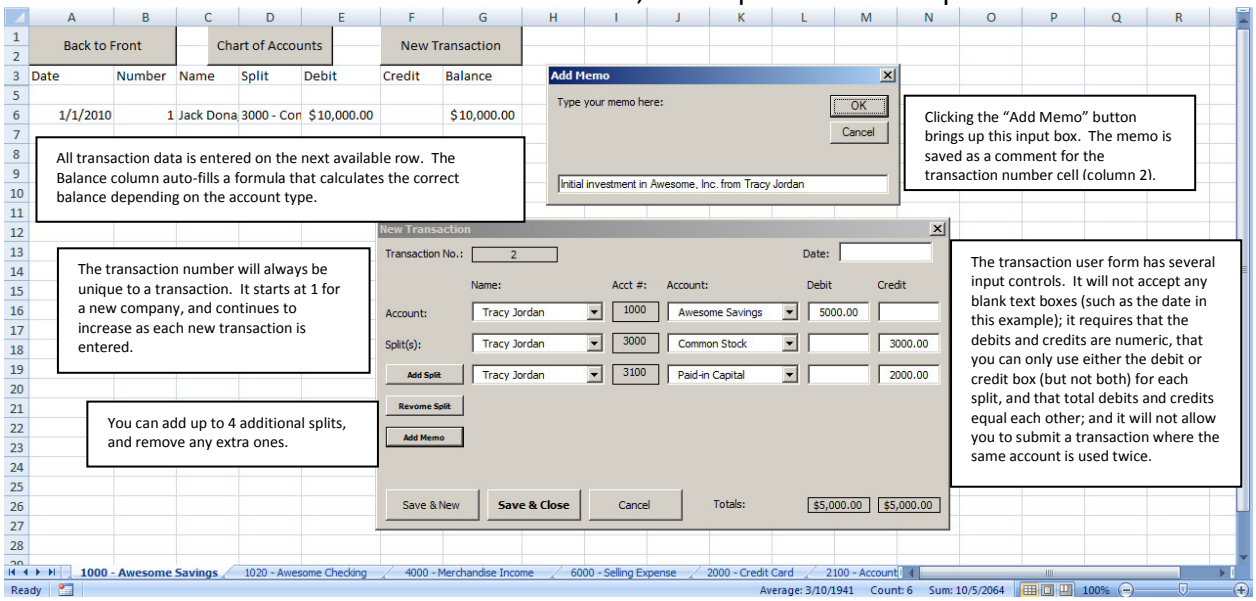
There are two worksheets, one each, for tracking both customers and vendors. The vendor sheet is important because it is the source for all names on the "Enter Bills" user form, which is part of the Accounts Payable worksheet. New vendors and customers are entered through a

user form, which automatically saves the information on the appropriate worksheet. The "Vendors" worksheet is displayed below as well as the user form which is used for entering in information for a new vendor.



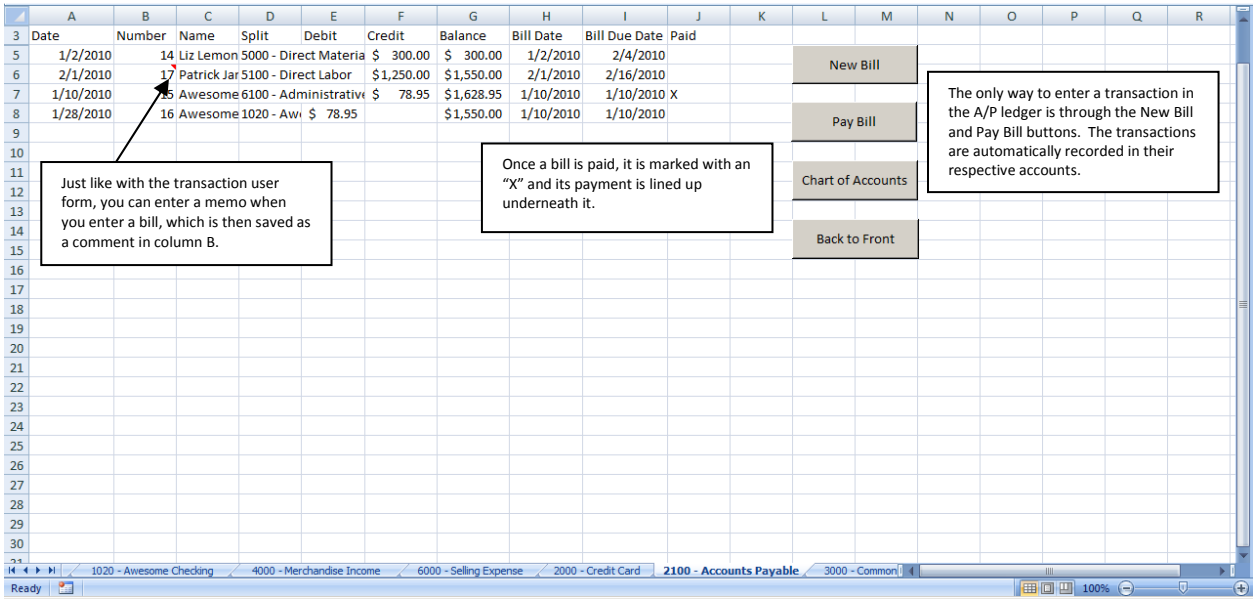
Account Ledgers

Any time an account is created, a new worksheet is created to serve as that account's ledger. It tracks all transactions that were entered for each account, and provides all unique information for each transaction. Each ledger has three buttons, one to send you back to the front page, one to send you back to the chart of accounts, and one which brings up the user form to enter a new transaction. The screenshot below shows the ledger for the "1000 – Awesome Savings" account as well as the new transaction user form, and explains different parts of each:

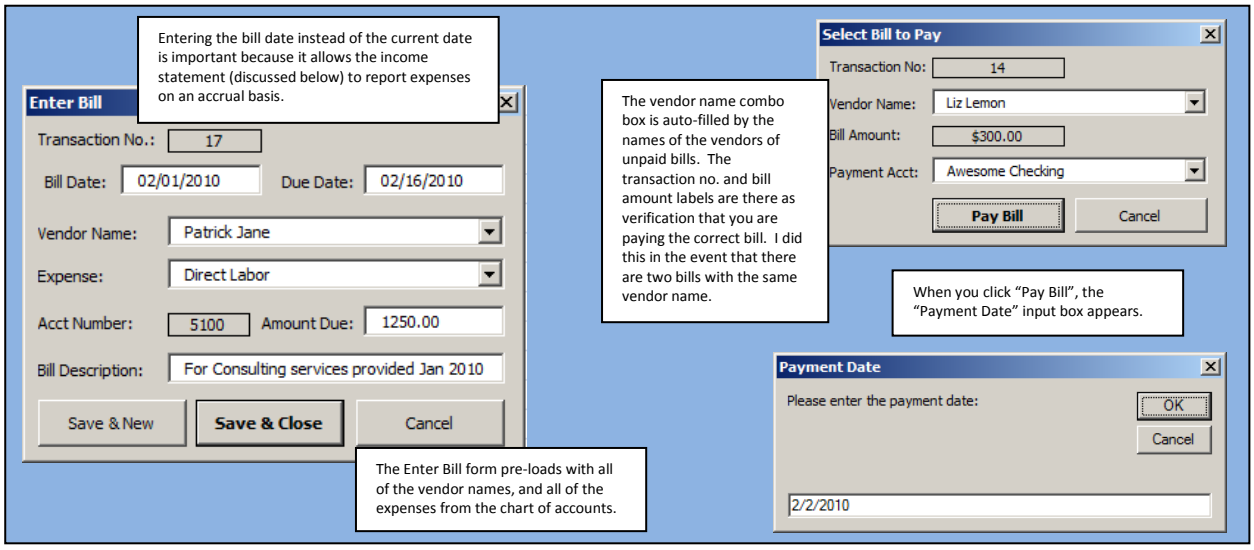


Accounts Payable Ledger

Accounts payable is a necessary account for any business. It allows the business to track what bills are unpaid, to whom they are owed, how much they are, and when they are due. When the accounts payable worksheet is activated, the worksheet automatically sorts itself so that unpaid bills are moved to the top, and are sorted by due date. From the accounts payable worksheet you can both pay bills and enter new ones. The screenshot below shows the worksheet:

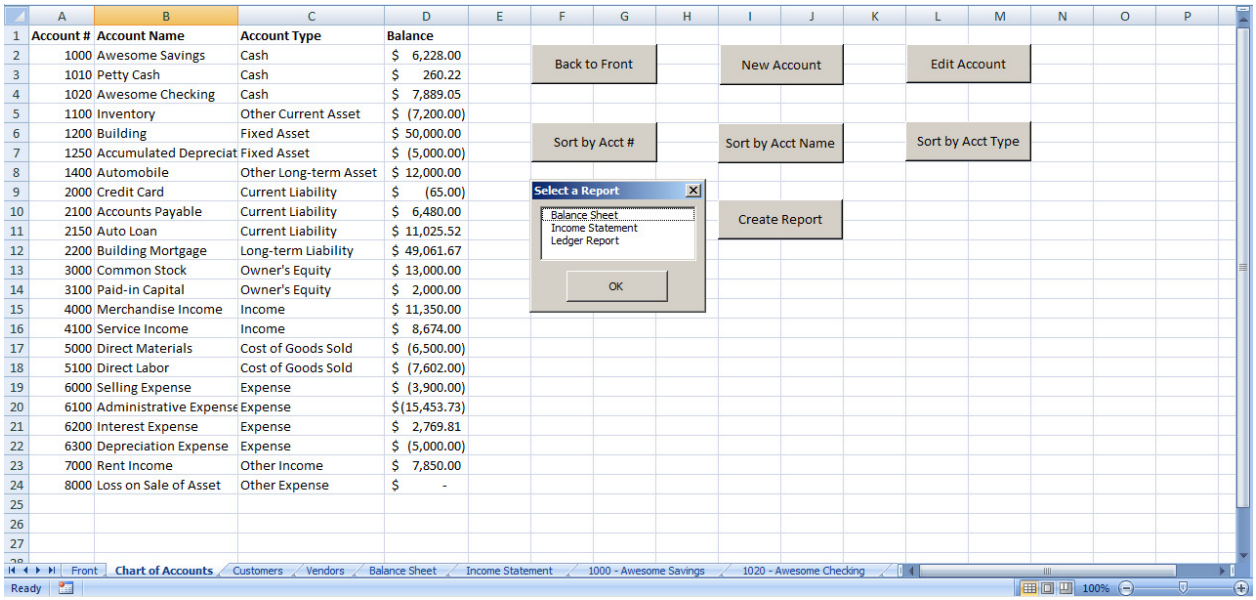


Below are the Enter Bill and Pay Bill user forms as well as explanations of their features:



Reports

Using the functions described above, I was able to put together one year's worth of data for Awesome, Inc. The last function that this program performs is that it can dynamically produce three different types of reports: the balance sheet, the income statement, and ledger reports. Below is an updated chart of accounts with one year's worth of data entered, along with the create reports dialogue box:



Account #	Account Name	Account Type	Balance
1000	Awesome Savings	Cash	$ 6,228.00
1010	Petty Cash	Cash	$ 260.22
1020	Awesome Checking	Cash	$ 7,889.05
1100	Inventory	Other Current Asset	$ (7,200.00)
1200	Building	Fixed Asset	$ 50,000.00
1250	Accumulated Depreciat	Fixed Asset	$ (5,000.00)
1400	Automobile	Other Long-term Asset	$ 12,000.00
2000	Credit Card	Current Liability	$ (65.00)
2100	Accounts Payable	Current Liability	$ 6,480.00
2150	Auto Loan	Current Liability	$ 11,025.52
2200	Building Mortgage	Long-term Liability	$ 49,061.67
3000	Common Stock	Owner's Equity	$ 13,000.00
3100	Paid-in Capital	Owner's Equity	$ 2,000.00
4000	Merchandise Income	Income	$ 11,350.00
4100	Service Income	Income	$ 8,674.00
5000	Direct Materials	Cost of Goods Sold	$ (6,500.00)
5100	Direct Labor	Cost of Goods Sold	$ (7,602.00)
6000	Selling Expense	Expense	$ (3,900.00)
6100	Administrative Expense	Expense	$ (15,453.73)
6200	Interest Expense	Expense	$ 2,769.81
6300	Depreciation Expense	Expense	$ (5,000.00)
7000	Rent Income	Other Income	$ 7,850.00
8000	Loss on Sale of Asset	Other Expense	$ -

When you select a report to create, one or two input boxes will appear depending on the report you choose. The balance sheet will ask you for a balance sheet date, while both the income statement and ledger report will ask you for a "From" date and a "To" date (i.e. from 1/1/2010 to 12/31/2010). The reports will then gather information from the chart of accounts, and from each individual account to produce what you want. The best thing about these reports is that they get rebuilt each time you create them, so if you create a new account for example, the balance sheet will pick it up and include it the next time you refresh the balance sheet. I've attached the reports on the next few pages, first with the balance sheet as of 12/31/2010, followed by the income statement from 1/1/2010 to 12/31/2010, followed by a ledger report for the checking account from 1/15/2010 to 2/14/2010.

Balance Sheet

Awesome, Inc.
As of 12/31/2010

ASSETS	Balance
Current Assets	
Cash	
Awesome Savings	6,228
Petty Cash	260
Awesome Checking	(11,825)
Total Cash	**$ (5,337)**
Other Current Assets	
Inventory	10,800
Total Other Current Assets	**$ 10,800**
Total Current Assets	**$ 5,463**
Long-term Assets	
Fixed Assets	
Building	50,000
Accumulated Depreciation	(5,000)
Total Fixed Assets	**$ 45,000**
Other Long-term Assets	
Automobile	12,000
Total Fixed Assets	**$ 12,000**
Total Long-term Assets	**$ 57,000**
Total Assets	**$ 62,463**
LIABILITIES	
Current Liabilities	
Accounts Payable	6,480
Other Current Liabilities	
Credit Card	-
Auto Loan	11,026
Total Other Current Liabilities	**$ 11,026**
Total Current Liabilities	**$ 17,506**

Long-term Liabilities	
Building Mortgage	48,985
Total Long-term Liabilities	**$ 48,985**
Total Liabilities	**$ 66,490**
EQUITY	
Owners Equity	
Common Stock	13,000
Paid-in Capital	2,000
Retained Earnings	(19,027)
Total Owners Equity	**$ (4,027)**
Total Equity	**$ (4,027)**
Total Liabilities + Equity	**$ 62,463**

Income Statement

Awesome, Inc.
1/1/2010 - 12/31/2010

	Amount
Revenues	
Merchandise Income	11,350
Service Income	8,674
Total Revenues	**$ 20,024**
Costs of Goods Sold	
Direct Materials	(6,500)
Direct Labor	(7,602)
Total Costs of Goods Sold	**$ (14,102)**
Gross Margin	**$ 5,922**
Operating Expenses	
Selling Expense	(3,965)
Administrative Expenses	(15,454)
Interest Expense	(8,380)
Depreciation Expense	(5,000)
Total Operating Expenses	**$ (32,799)**
Net Operating Profit/(Loss)	**$ (26,877)**
Other Income	
Rent Income	7,850
Total Other Income	**$ 7,850**
Other Expense	
Loss on Sale of Asset	-
Total Other Expenses	**$ -**
NET INCOME/(LOSS)	**$ (19,027)**

Ledger Report: 1020 - Awesome Checking

1/15/2010 - 2/14/2010

Date	Num	Name	Split	Debit	Credit	Balance
Previous Balance						**0**
01/15/2010	3	Awesome, Inc.	1000 - Awesome Savings	$ 10,000.00		$ 10,000.00
01/17/2010	11	Awesome Bank	Multiple Splits		$ 150.00	$ 9,850.00
01/20/2010	12	Awesome Bank	Multiple Splits		$ 857.00	$ 8,993.00
01/21/2010	13	Awesome Bank	2000 - Credit Card		$ 25.00	$ 8,968.00
01/28/2010	16	Awesome Bank	2100 - Accounts Payable		$ 78.95	$ 8,889.05
02/02/2010	18	Liz Lemon	2100 - Accounts Payable		$ 300.00	$ 8,589.05
02/05/2010	19	Things, Inc.	1100 - Inventory		$ 9,000.00	$ (410.95)
02/08/2010	20	Kenneth Parcell	Multiple Splits	$ 350.00		$ (60.95)
02/12/2010	21	Jenna Maroney	Multiple Splits	$ 1,348.00		$ 1,287.05
TOTAL						**$ 1,287.05**

Learning and Conceptual Difficulties

There were several difficulties in getting this project to work. The first problem I really ran in to was how I would be able to authenticate users without making that information readily available. I wanted there to be an option to have multiple users, so hard-coding a username and password into the VBA code itself was not an option. Instead I found out how to create an event-based subprocedure which would prevent you from ever opening the users worksheet. The code is as follows:

```
Private Sub Worksheet_Activate()
  Users.Visible = False
End Sub
```

It turned out to be pretty simple, and has proven to be a good option for authentication.

Another problem I kept running in to was how to format the worksheets for new accounts. I had the Chart of Accounts linked to each new account so that it could read each account's balance. The problem, though, was that when there were no transactions in an account, the chart of accounts wouldn't report a zero balance. In the end, I decided to hide the 4^{th} row of every account, enter 0 into column G to signify a zero balance, and then in the 5^{th} row, column G I put the formula to calculate the balance. This solved my problem.

Several times throughout writing the code, I found myself repeating large blocks of code. Eventually, I'd find problems within that block of code and then I'd have to go through and repair each individual block of code with that same problem. I found this to be time consuming, and even wasteful. Eventually I remembered that I could create subprocedures with pass-through variables. This made the programming much easier. One example is in the new transaction user form. The form allows for up to six lines of information to be saved on six different account ledgers. I started by writing the code for each individual line. I kept running into errors, and I had to go back and fix each line, one by one. Eventually I was able to write a subprocedure which required you to pass on each piece of information (account, debit, credit, etc.), but then it took each piece and posted them to their respective account. This was extremely helpful because it cut down on the code I had to write and because I was able to just go to one subprocedure to fix any errors, instead of poking around in six different places.

The biggest obstacle I ran into was in the creation of the reports. I really wanted them to be dynamic so that they could be created based on the dates that the user chose. This meant that sometimes there would be new accounts created between the generation of these reports. I needed to write a code that would look at each type of balance sheet/income statement account, and then go find every account that fit under that classification. It took me a while, but I was able to figure it out. The code is its own subprocedure that requires a string which represents the account type to be passed through. It then looks at each item in the chart of accounts, and picks out only those accounts which match the given account type, and then goes to the ledgers of those accounts to get the appropriate balance. I was pretty proud of that

code because it allowed me to create these dynamic reports without missing any newly created accounts.

I hope to be able to continue to build on this. Some functions that I want to add include: editing and adding new users; editing existing transactions; and creating a cash flow statement. I didn't realize how much I could do with excel before starting this project, and am amazed that the whole thing was able to work out.