Ryan Jones

VBA Final Project

MBA 614

Mel's Auto Referral Manager

Executive Summary

Mel's Auto of Springville is a local auto mechanic shop. They opened their doors just over a year ago and they are hoping to increase their customer base. Thus, Mel's wants to implement a customer referral rewards system to incentivize their customers to refer a friend to this auto mechanic business. Any individual who refers a friend will receive a credit toward their next purchase equal to 10% of the referral's order. This credit can accumulate. The original idea was to track referrals and credits using a notepad next to the register. A notepad is very user-friendly, yet not so efficient. The most important objectives for this project include:

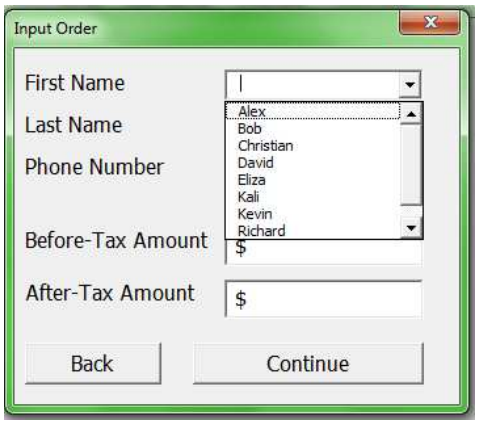
1. The system must be very user-friendly to allow staff mechanics to easily operate.
2. Accurately track referrals as well as the credits generated and used by the customers.
3. Facilitate the input of special offers.
4. Allow management to manipulate, modify, and fix customer information, credit amounts, referrals, and special offers.
5. Report data on number of referred customers, total credit outstanding, etc.

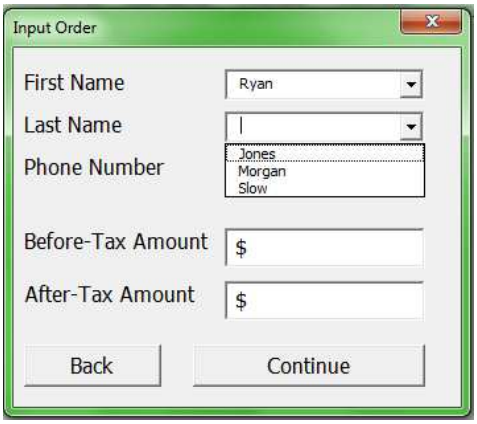
The program accomplishes these objectives through a simple database and multiple user forms, with extra care and focus on ease of use. The creation of combo boxes to allow the quick and easy selection of existing customers helps facilitate the user friendly nature of the forms. The many forms follow a step by step process of customer order input, referral claimed or unclaimed, and credit usage. Beside the input of new orders, another password protected user form allows the management to update the database and apply special offers to individual customers. All program interaction is done through forms to remove any accidental formula deletion or other errors.

Implementation documentation

This VBA project consists of 8 user forms and 3 modules. When the worksheet is opened, the MainMenu form is loaded. The form merely contains 2 buttons that lead to the two main functions of the program: Input Order and Edit Data. First, the Input Order function uses 3 forms:

1. frmInputOrder
2. frmInputReferral
3. frmUseCredit





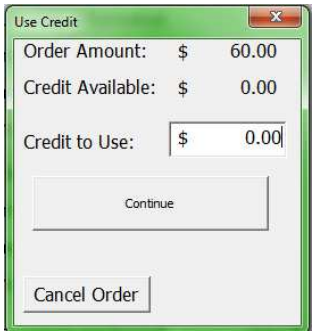
1. The InputOrder form is used as a starting point for new orders. The combo boxes are populated with options from the table. The boxes allow the user to quickly find existing customers through the use of auto complete, as well as utilization of the updateComboBoxes module I wrote to narrow down the options in the other combo boxes to only show options that match previously input data. If only one option is available, the box also uses that value automatically. Hence, if only one customer (Zeek) has a first name that starts with "Z", if a Z is entered into the first name combo box, the combo box is auto filled with "Zeek", the last name combo box is filled with Zeek's last name and the phone number combo box is filled with Zeek's phone number. The auto complete feature of a combo box was used for the active box being updated, but many subroutines were programmed to facilitate the update and auto completion of other boxes. The combo boxes' lists are refreshed after any one of the combo boxes is updated. At that point, the program goes through each category and copies the values to a new sheet, narrows down the values by going through each and testing it with the current information. Duplicates are removed and finally the remaining items are added to the combo box. If only one item remains, instead of adding a .list value to the box, a single item is added. This is one of many "what if" scenarios that had to be programmed for.

To more fully explain, the list in each combo box is updated for entries that match currently input data. If two Zeeks are in the system, the last name box will still not show any value, but the dropdown options will be narrowed down to only the two last names that match with a first name of Zeek. If a phone number is then selected, the Last name field is again narrowed down and the one remaining last name would be selected automatically. The same narrowing process can be done with any of the three boxes in any order.

The user then completes the order amount box and clicks the continue button. At this point, many tests are run to ensure all the fields are filled, the customer is in the system, whether that customer has been referred or has referred others, etc. If the customer is not in the system, it is inserted into the table after receiving confirmation. If the customer has already claimed a referral or has claimed to have no referral, the program moves on to the UseCredit form. Otherwise, it first stops at the InputReferral user form.

2. The referral input form utilizes the same three combo boxes and uses much of the code of the order input form. Rather than copy and subsequently modify the prior code, I created the code in such a way as to allow each form to call the update procedures with a unique value to change the way the code is executed. It was very helpful to use objects in this way so if I had to go back and change some coding in the future I only had to change one copy, instead of one copy per form using the procedure.

This form also has to check a few things before moving on. It is possible for a customer to be claimed as a referrer before making a purchase themselves, so this form can also potentially add a new customer to the table and then set the referrer reference to this new customer. If the current customer doesn't know the phone number of the referrer, then the program automatically sets the number as 999-999-9999. When that customer comes in to use their credit, the system will recognize this phone number as a place holder and will ask the user to input the actual phone number. After the referral has been established, the UseCredit for is used.



3. The UseCredit form pulls the order amount and the customer's current credit amount from the underlying table. The value in the credit to use field is initialized as the lesser of these two amounts yet allows this to be changed in case the customer wants to save some credit. If this amount is above the available credit, a message will instruct the user to input a lower amount. When the user clicks continue, the user is notified of the completed order; it will also show the amount of credit given to the referrer, if any. The form then retreats to the input order form to be ready for the next order. At this point the workbook is automatically saved to ensure the order isn't lost. Although it takes a moment to save, the mechanics garage isn't pumping customers out

like Walmart so the slight save delay is fine. The workbook saves the file in the current directory, and is tagged with the date to allow for data backup and history. The date is acquired using the =today() formula in a cell, then using the vba format(expression, format) method to take out slashes and add in dashes for delimiters, as slashes would not be allowed in the file name.

The second main program feature is the data edit function. This function utilizes 4 forms:

1. frmEditInfo
2. frmNewCustomer
3. frmSearchCustomer
4. frmData

The EditInfo form is the main form for this section. From this form, the user can navigate to each customer entry using the next and previous buttons, as well as the search customer button. This search button brings up the SearchCustomer form which allows the user to input information and search for an entry. If the entry is found the form unloads and we return to the EditInfo form with that customer selected.

Back at the EditInfo form, we see all of the fields that pertain to a customer and these fields are all available for update. Any changes are saved when the Save button is clicked. When this is clicked, first the procedure checks the referral information to make sure the customer exists. Having verified the customer, the procedure then proceeds to update the correct fields in the table according to the corresponding fields. In addition, the specials list feature was requested to allow management to apply specials to specific customers and record what has been given.

Finally, the NewCustomer form appears when the button is clicked. It is a simple form to allow the entry of a new customer. Nothing fancy there. The Data form brings up a few points of information that might be useful for Mel's Auto. This form is basically just an info box; although, the data includes a chart of the top 5 credit earning customers extracted from the customer table. This chart was created by pulling the data from the table, pulling it to another sheet, formatting the data, removing excess information, creating a chart, formatting the chart, saving the chart as a .gif, then setting the image on the Data form's picture as that .gif.

A few other buttons exist on this form but I believe they are self explanatory and they were rather simple to program for; so, I won't discuss them here.

Learning

This was a fun project. Never have I had so much fun spending dozens of hours on homework. Yet the fun came mostly from overcoming obstacles that were extremely frustrating. The bulk of my time was spent in learning user forms. I learned that they aren't always easy to work with, especially when you want to accomplish something outside the normal function. Making the combo box lists automatically update when another field was updated took quite

some time. I had to run many loops to do this. I used object types I haven't used before to avoid duplicating lines, instead adding a small function to allow variation based on which form is calling the procedure. For the longest time I thought the forms name would have to be hard coded into the vba code. But then I found using the object model could simplify so many things. Just - set form = frmInputOrder – then using form.show – etc instead of duplicating efforts. But I found this out hours and hours into my coding. For some procedures it made sense to go back and modify it so I could use the modified code for future forms.

Another thing I learned was the importance of keeping numbers and names in line. A couple of times the program was putting in the wrong cell references and I couldn't understand it, but then I would notice one part using the wrong variable; which came from a difference of two characters in the variable name and led to an hour of hair pulling. In the future I will have to be more careful in making unique names more unique.

I found the most difficult part of the project was making it user-friendly. It is really easy to tie one customer to another and to give a credit to one or the other. User error, user needs, and variations in needs and circumstances add an exponential amount of complexity to the work. I had to ask myself questions about what might be unique about a customer or what the managers might want the program to handle. The program has so many checks and if statements in an attempt to account for all the variety in situation.

I got the dollar values to format just fine. It took a little time to do so. Especially since I used a long variable at first… for some reason I thought a long could hold a decimal. Boy was I wrong. I learned my lesson though. I still couldn't get the phone number to go in and out right. Maybe I could if I gave it some more time, but every now and again the phone number won't format with delimiters, but it will be a boring string of 10 numbers. But that isn't the only thing I need to learn about VBA.

I originally thought I might be able to work with a customer database to pull some data, but I was unsuccessful in this regard. Not because I couldn't figure out the programming, but I couldn't get access to the database to even try.

This project took me a long time. I enjoyed it and I learned a lot. It sounds simple now that I have put my blood and tears into learning it, but now I could probably program a similar project in far fewer hours. I suppose that is the point of learning. Progressing in the knowledge and skills needed to better perform.