B.J. Saunders
Brigham Young University
MBA 614 – Winter 2010
Final Project

Executive Summary

This project involved the creation of an Excel workbook intended to be used as a database and management system for running a home salon. The final product began essentially as an 'upgrade' to an existing workbook used by my sister who works as a hairdresser out of her home. The old workbook was outdated and 'clunky' with various glitches occurring. The old workbook used only worksheet formulas to keep track of basic income and expenses on a monthly basis with summary information on a master sheet. The new workbook created for this project provides all the functionality of the old workbook with improved ease of use as well as additional useful functions and features. The new workbook uses multiple user forms to make it easier to enter income and expenses to better manage the finances of a home salon business and the worksheet have a much improved appearance. Additionally, the new workbook adds functionality to store data such as contact info, hair color formulas, notes, etc. on all clients entered in the client data worksheet. Information on each client is conveniently displayed and can be edited on a user form, where new clients can also be added and managed. The user can also choose to send e-mail and text alerts to clients reminding them of their next appointment.

Home Salon Management System

Introduction

A significant amount of time was put into this project to create a user friendly and stable database system that can be used to track income and expenses and manage client information for a single hairdresser running a salon out of her home. The basic user needs for tracking income and expenses came from a workbook already used by my sister for that purpose. The new workbook is really a major overhaul of the old work book and provides a much more useful system for managing finances, and now, clients.

Workbook Features

A brief description of the original worksheet should help to better portray the changes in the new workbook. The old workbook essentially had a Master Sheet which served as a dashboard for the important summary financial information from the data entered on the worksheets created for each month of the year. Screenshots of the original workbook showing the Master Sheet and a sample month worksheet are provided below in Figure 1.

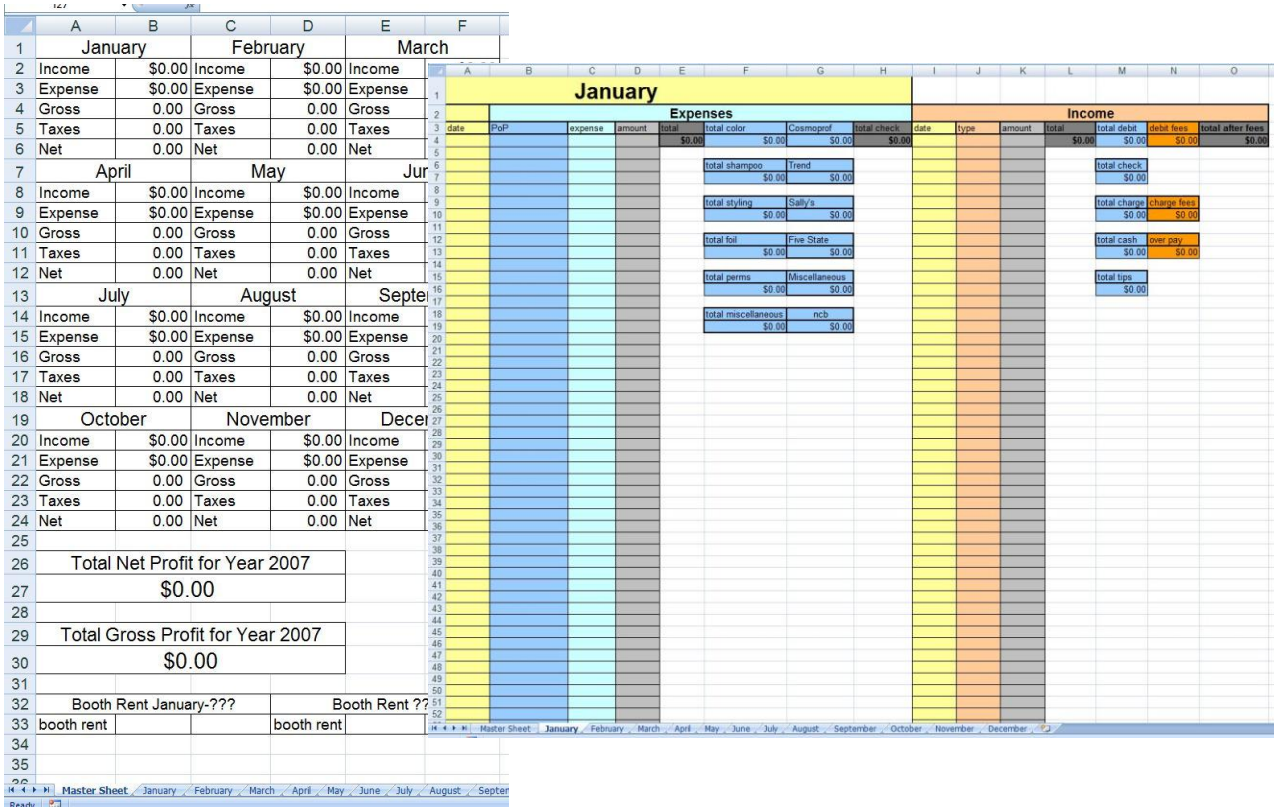

Figure 1. Screenshots from the original workbook

In the old workbook, the user had to enter the data directly onto the worksheet which created potential for error, especially in the subtotals using "SUMIF" statements on each month's worksheet. The same basic concept of a Master Sheet and a worksheet for each month was used in the new workbook, although the layouts have changed and the appearance has been enhanced. Screenshots of the new Master Sheet and a sample month are shown in Figure 2 below.

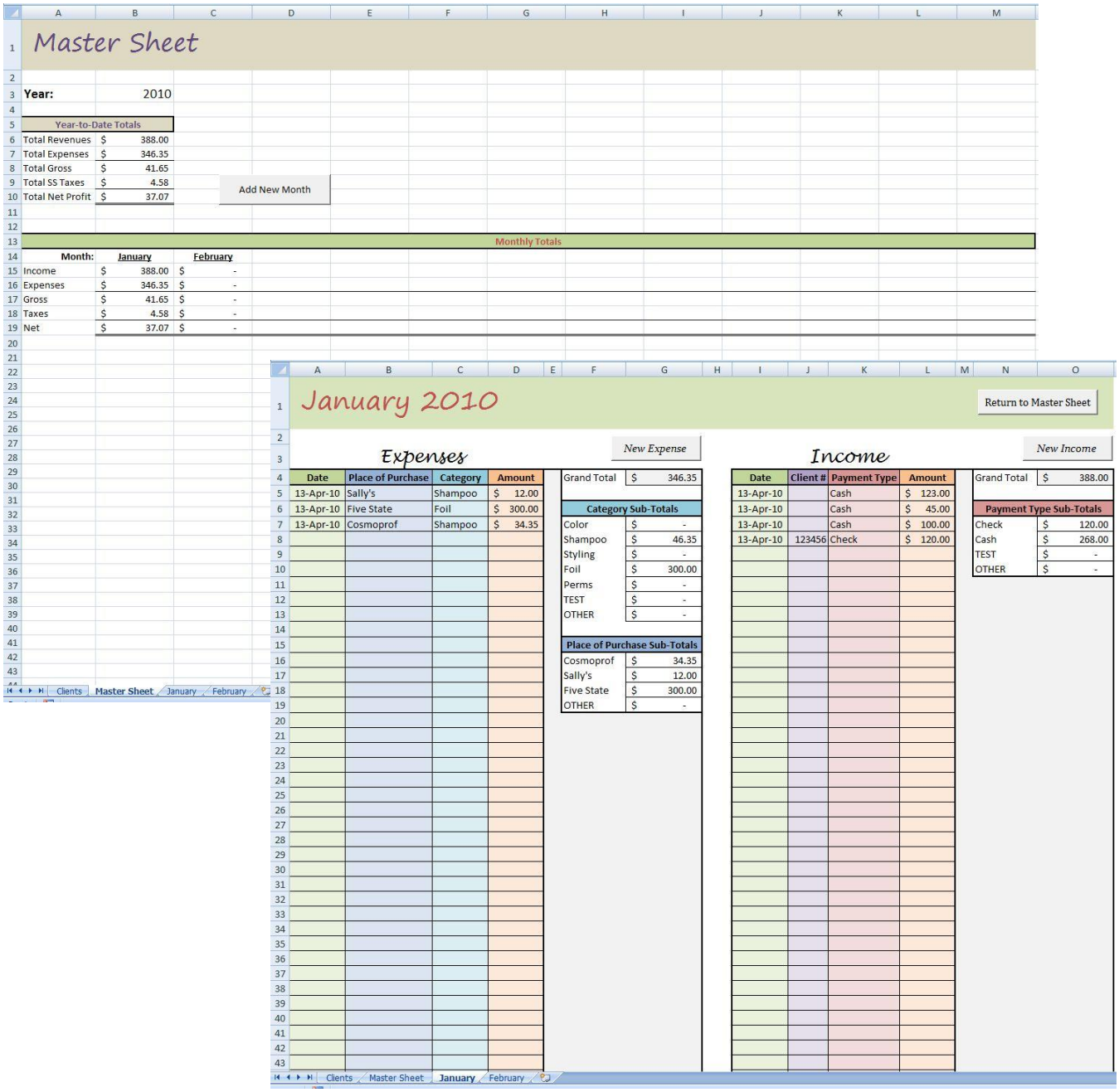

Figure 2. Screenshots from the new workbook

The basic functionalities of the new workbook created for the project are listed here:

- The workbook starts with only the Client and Master Sheet worksheets (and a hidden Data sheet that should not be altered except through the options presented in user forms.
- An 'Add New Month' button placed on the Master Sheet lets you add a new worksheet for the next month.
- Each month worksheet is built from "scratch" and buttons to open user forms for entering new expenses and new income are created. Each month worksheet also has a button to return to the Master Sheet.
- Within the new expense and new income user forms, the user can add new sub-total categories to be used for data entry. The active worksheet and user forms are updated with the new information and all subsequent worksheets will include the addition.
- A new column for summary data from each new month created is automatically created on the Master Sheet and formulas are entered into the appropriate cells and cell formatting is applied.
- The significant new feature in the new workbook is the functionality to manage client data. A 'Clients' worksheet is included with a button to access a user form which allows the user to store contact and other personal information on each client, essentially serving as a database for managing clients.
- A new year can be created simply by performing a 'Save As' and saving the workbook with a different name. The month sheets and month data on the Master Sheet can then be deleted and the year value on the master sheet updated. This approach allows you to easily start a new workbook for a new year while keeping the client data intact for a new year.

Creating New Month Worksheets

Each new month worksheet is built "from scratch" when you click the 'Add New Month' button. This includes all the formatting, formulas, and buttons. This approach keeps the workbook more orderly and ensures that new month worksheets have consistent formatting. The code for this operation was not simply created using the macro recorder but was personally built from the ground up so the code is "clean". To improve functionality, efficiency, and more robustness in the code, the code is built in a modular fashion in several logical sub routines and functions. This applies to all the code generated for this project.

Some of the features created with respect to new month worksheets and associated user forms are described here:

- When the 'Add New Month' button is pressed on the Master Sheet, the program checks to see if you already have all the months for the year (i.e., through December). If all the months have already been added it will not allow you to add another month.
- It also checks that you have entered a year on the Master Sheet before it will add a new month worksheet since this information is needed to add the label at the top of the new worksheet.

Screenshots of the new user forms created to enter new expenses and incomes are provided below in Figure 3 and Figure 4.

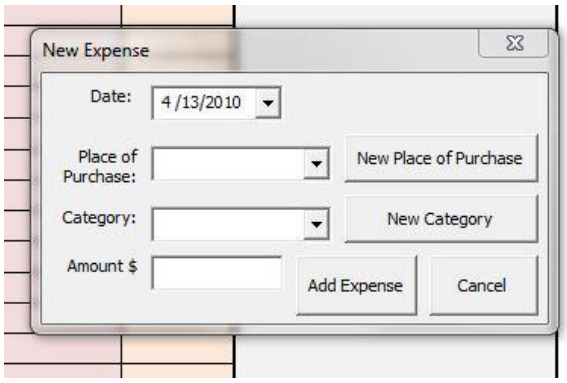

Figure 3. New Expense User Form screenshot

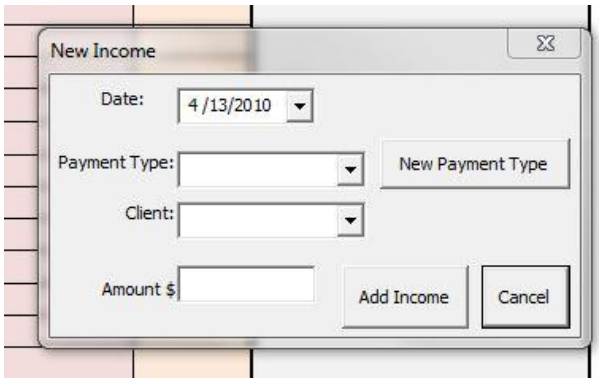

Figure 4. New Income User Form screenshot

Through these user forms you can add new categories used for tracking sub-totals. When you click the associated buttons in each user form a new form appears that allows you to add a new sub-total category. Those forms check to make sure that a duplicate is not being added. Once a new item is added, the combo boxes are updated and the sub-total area on the active worksheet is also updated to include the new entry. A modular piece of code is used that updates just the sub-total output area. Generating the code to automatically create the new worksheets, with buttons and allowing the user to add new sub-total categories proved to be one of the most time consuming portions of the project.

The code from a module that creates the “backbone” for adding new sub-total categories is provided here:

```
Option Explicit
Const dataTypeLbl = "Data type:"
Const dataTypeLblColumn = 1
Dim dataTypeRow As Integer
Dim typeMatchColumn As Integer
Dim dataEntryRow As Integer
Dim currentNumEntries As Integer

Public Function entryAttempt(dataType As String, userEntry As String) As Boolean
   findData dataType

   If findMatch(userEntry) = True Then
      MsgBox "This entry is already in the data set. Please enter a new " & dataType & ".", vbExclamation,
"Duplicate Entry"
      entryAttempt = False
   Else
```

```
      addData dataType, userEntry
      entryAttempt = True
   End If

End Function

Sub findData(dataType As String)
   dataTypeRow = 1 'start search at top of sheet

   'Finds the row where data type names should be
   Do Until dataSheet.Cells(dataTypeRow, dataTypeLblColumn).Value = dataTypeLbl
      dataTypeRow = dataTypeRow + 1
   Loop

   'Finds the matching data type column
   typeMatchColumn = dataTypeLblColumn
   Do Until dataSheet.Cells(dataTypeRow, typeMatchColumn).Value = dataType
      typeMatchColumn = typeMatchColumn + 1
   Loop

   dataEntryRow = dataTypeRow + 1
   currentNumEntries = 0
   Do Until dataSheet.Cells(dataEntryRow, typeMatchColumn) = ""
      currentNumEntries = currentNumEntries + 1
      dataEntryRow = dataEntryRow + 1
   Loop
End Sub

Sub addData(dataType As String, newEntry As String)
   If dataType = PeroxideTitle Then
      dataSheet.Cells(dataEntryRow, typeMatchColumn).Value = newEntry
   Else
      dataSheet.Cells(dataEntryRow - 1, typeMatchColumn).Value = newEntry 'Enters the new entry
      dataSheet.Cells(dataEntryRow, typeMatchColumn).Value = "OTHER"
   End If
End Sub

Function findMatch(userEntry As String) As Boolean
   Dim i As Integer    'for the For loop
   For i = 1 To currentNumEntries
      If UCase(dataSheet.Cells(dataTypeRow + i, typeMatchColumn).Value) = UCase(userEntry) Then
         findMatch = True
```

```
        Exit For
      Else
        findMatch = False
      End If
    Next
End Function

Public Function getNumEntries(dataType As String) As Integer
    findData dataType
    getNumEntries = currentNumEntries
End Function

Public Sub setDataArray(dataType As String)
    Dim X As Integer

    findData dataType

    Select Case dataType
      Case Is = PoPTitle
        ReDim expensePoP(currentNumEntries - 1)
        For X = 0 To UBound(expensePoP)
          expensePoP(X) = dataSheet.Cells(dataTypeRow + 1 + X, typeMatchColumn).Value
        Next
      Case Is = PayTypeTitle
        ReDim incomePaymentType(currentNumEntries - 1)
        For X = 0 To UBound(incomePaymentType)
          incomePaymentType(X) = dataSheet.Cells(dataTypeRow + 1 + X, typeMatchColumn).Value
        Next
      Case Is = CategoryTitle
        ReDim expenseCategory(currentNumEntries - 1)
        For X = 0 To UBound(expenseCategory)
          expenseCategory(X) = dataSheet.Cells(dataTypeRow + 1 + X, typeMatchColumn).Value
        Next
      Case Is = PeroxideTitle
        ReDim clientPeroxideType(currentNumEntries - 1)
        For X = 0 To UBound(clientPeroxideType)
          clientPeroxideType(X) = dataSheet.Cells(dataTypeRow + 1 + X, typeMatchColumn).Value
        Next
      Case Else
        MsgBox "error"
      End Select
End Sub
```

Clients

The major new edition to the workbook overhauled for this project is the client database portion. Generating the code for the desired functionality took some innovative thinking. All interaction with the data stored on the Clients worksheet is designed to be done through the 'Clients' user form that is accessed by pressing the 'Edit/Add Clients' button on the worksheet. A screenshot of a portion of the Clients worksheet is provided in Figure 5. A screenshot of the Clients user form is also provided in Figure 6.

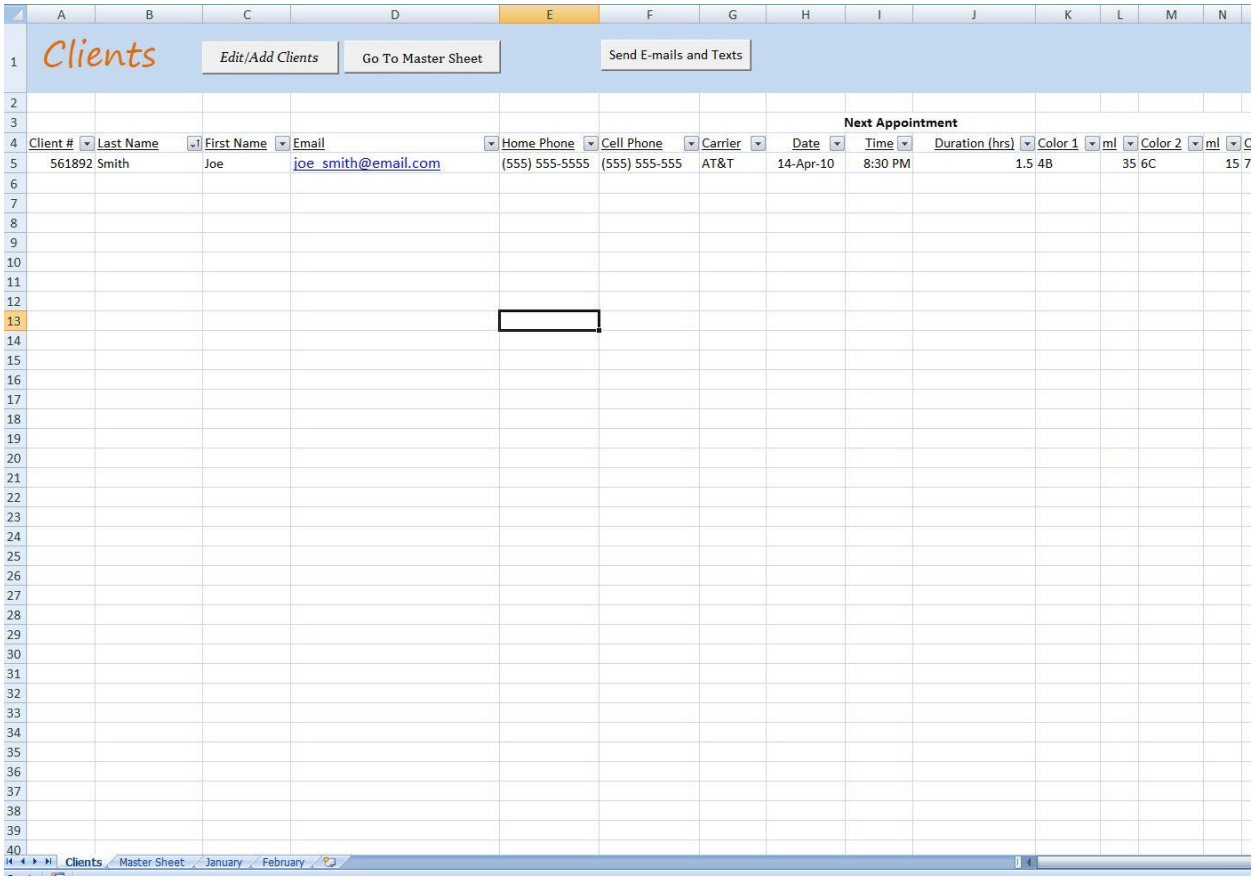

Figure 5. Screenshot of the new Clients worksheet

An important and somewhat challenging feature to create was the ability to use the same user form to edit existing clients as well as adding a new client. When you add a new client the caption for the form changes, the navigation buttons and combo box are grayed out and made unavailable, and the behavior of the save and cancel buttons change. The ability to navigate with either next and previous buttons or a combo box that is updated with all name in the database proved somewhat challenging. An example of some modular code that was used can been seen in the "setNextPrevButtons" sub procedure code provided on the next page that can simply be called to make sure that the navigation buttons do not allow you to navigate past the rows where data is currently stored.

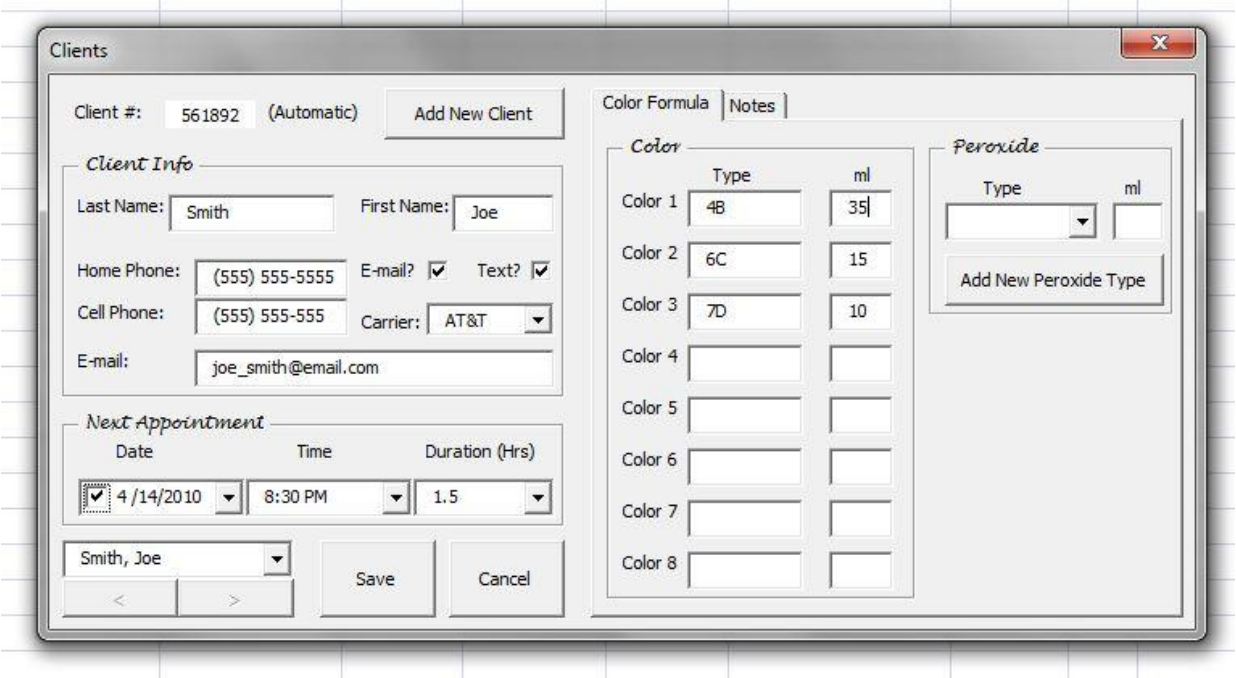

Figure 6. Screenshot of the Clients user form. Interaction with client data is intended to primarily take place through this form.

```
Sub setNextPrevButtons()

    findLastEntryRow

  If currentRow = lastEntryRow Then

    cmdNext.Enabled = False

  Else

    cmdNext.Enabled = True

  End If

  If currentRow = lblRow + 1 Then

    cmdPrev.Enabled = False

  Else

    cmdPrev.Enabled = True

  End If

End Sub
```

One of the unfortunate drawbacks of the program in its current state is that new client data is simply entered at the end of the list, not in a sorted, alphabetical fashion. However, the code is designed such that the user can close the client user form, sort the data on the worksheet and then reopen the user form and the data will appear in the sorted order.

E-mail and Text

One feature not yet fully developed in the program is the ability to send email and text to clients to remind them of their next appointment. The basic code and "infrastructure" has been put into the program but are not yet fully operational. This lack of functionality is largely due to scope creep that occurred with the rest of the project. In its current state the program will send an automated email message to every client in the database with information on their appointment date and time currently listed in the database using the user's gmail account by simply pressing the 'Send E-mails and Texts' button on the Clients worksheet. The automated message which is sent is currently of the form shown here:

Dear B.J.,

I just wanted to remind you about your appointment on 4/14/2010, at 8:30 PM.

Other challenges included generating the time and duration lists in the combo boxes on the worksheet.

Conclusion

In all, despite some of the few shortcoming and lack of functionality that still exists, specifically with sending email and texts, the workbook provides a user friendly and useful resource for someone running a small hair cutting business out of their home to manage basic financials and client information.