The Address Organizer

Created by Crista Hill

Executive Summary

Project Description

The Address Organizer is an address management tool that sorts addresses based on information needed, automatically emails address requests, and generates envelopes using Microsoft Word Mail Merge.

Pain Point

Every time I have a sibling get married, they complain about how long they spent mailing wedding invitations. My sister manually typed each address on a word document in just the right place on the document so the envelopes would print correctly. My brother, after spending an entire day working on envelopes, frantically emailed me late into the evening the day he was printing envelopes, asking for help because he couldn’t get it to work. Though I explain the Mail Merge process to them, this process is not intuitive and usually requires significant reformatting of the work they’ve already completed.

Solution

Whenever I have a sibling announce their engagement, this spreadsheet will be my first wedding gift to them. It will help them collect and organize their addresses. Sending envelopes will require a mere click of a button.

Features

The organizer is designed to be very intuitive. There are no instructions along with the spreadsheet. All of the features are performed either automatically when the user clicks on a spreadsheet, or by clicking a clearly labeled button located in a visible area on the worksheet where the button would be needed. For example, the contact updates are completed with a form that automatically appears when a user clicks on the data. To add new contacts, the user clicks the “New Contacts” button that is located in a very visible area on the worksheet.

- Create new contacts using a Userform
- Update contacts using a Userform
- Identify contacts that “Need an Address”
- Identify contacts that “Need an Address and an Email Address”
- Identify contacts that have the needed information and are “Ready to Mail”
- Email an address request letter to contacts on the “Need an Address” list
- Create envelopes using Mail Merge

<u>Detailed Feature Description</u>

Create New Contacts Using a Userform

The Userform allows new contacts to be created quickly and easily. The Userform is designed to ensure that new contacts are listed in the next available row regardless of where the active cell is located when the user clicks the “Create New Contact” button located on the “Master List” worksheet.

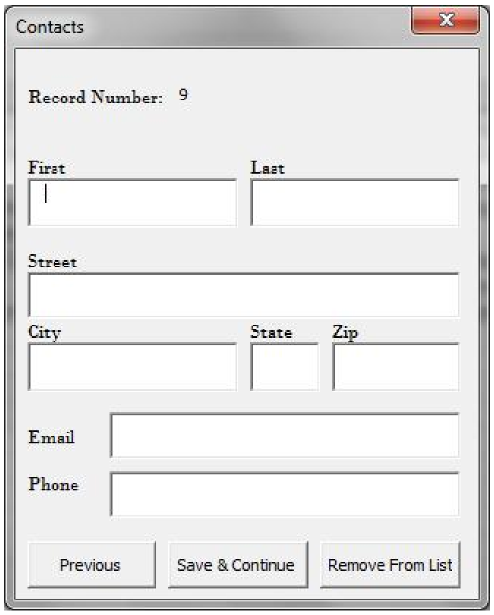


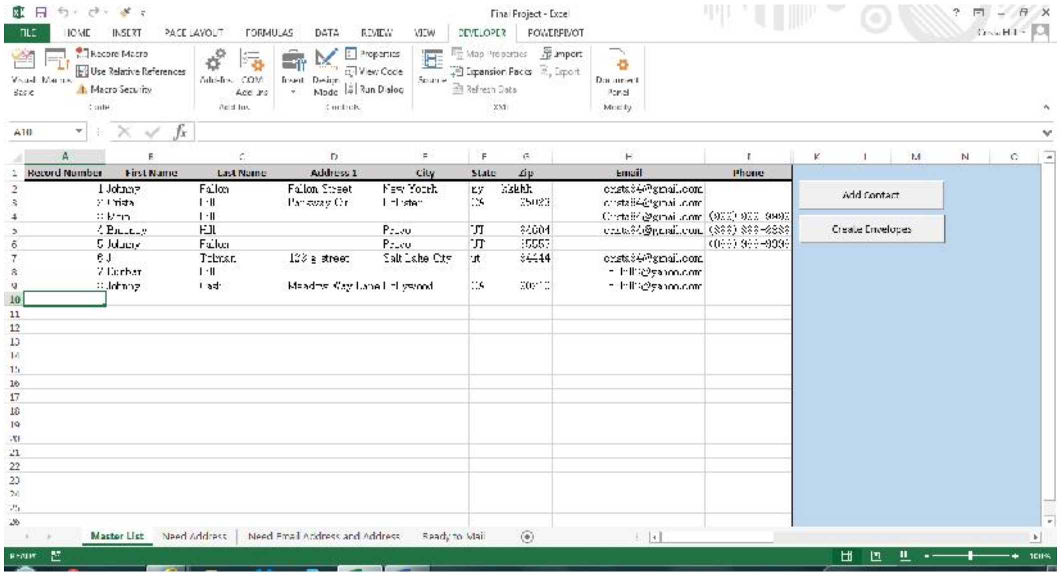
Figure 1: Contacts Userform

The Userform assigns each contact a record number based on the row where the contact will be located and includes fields for the contacts’ Name, Address, Email, and Phone. When the user clicks “Save & Continue”, the new contact will be saved on the next consecutive row under the existing data. The user can scroll to previous contacts using the “Previous” button, or discard the changes either by clicking “Remove From List” or simply closing the Userform without saving.

Update Contacts Using a Userform

Contacts can easily be updated with the Userform. When the user selects a cell that already contains data, the Userform shown above will automatically show and the user can change information and select “Save & Continue”. The user can also remove contacts from the list by selecting “Remove From List”. This will delete the row where the contact is located and move the other contacts up to remove the space. The Record Number cannot be changed and is directly related to the row number where the contact is located. If a contact is deleted, the Record Number for contacts below will change. All contact edits must be made on the “Master

List" worksheet. If a user tries to edit a contact sorted to another sheet, the user will see a message box that says, "Please make all contact edits on the Master List".

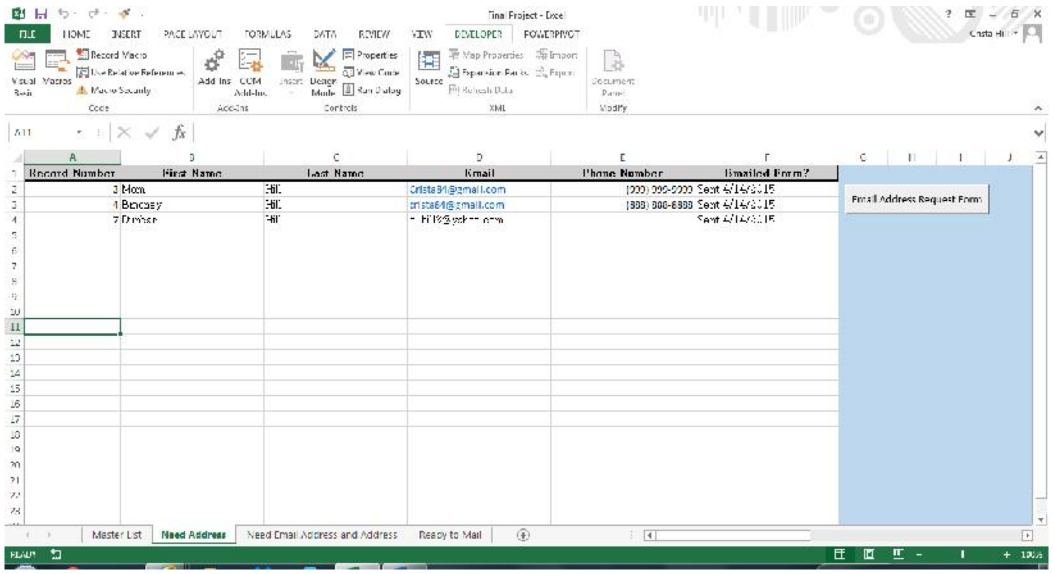


Record Number	First Name	Last Name	Address 1	City	State	Zip	Email	Phone
1	Johnny	Fallon	Fallon Street	New York	ky	[illegible]	[illegible]@gmail.com	
2	Crista	Hill	Parkway Cir	[illegible]	[illegible]	[illegible]	[illegible]@gmail.com	
3	Mom	Hill					[illegible]@gmail.com	[illegible]
4	Brittney	Hill		Provo	UT	[illegible]	[illegible]@gmail.com	[illegible]
5	Johnny	Fallon		Provo	UT	[illegible]		[illegible]
6	J	Tolman	123 s street	Salt Lake City	ut	[illegible]	[illegible]@gmail.com	
7	Dunbar	Hill					[illegible]@yahoo.com	
8	Johnny	Cash	Meadow Way Lane	Hollywood	[illegible]	[illegible]	[illegible]@yahoo.com	

Figure 2: Master List worksheet

Identify Contacts that Need an Address

In order for the user to know who they will need to collect addresses from, the contacts must be sorted. This is accomplished automatically when the user selects the worksheet titled "Need Address". This list is updated each time the user selects the sheet. Because the contacts can only be updated on the "Master List" worksheet (all other worksheets are password protected), this list should always be up-to-date when the user is viewing it.

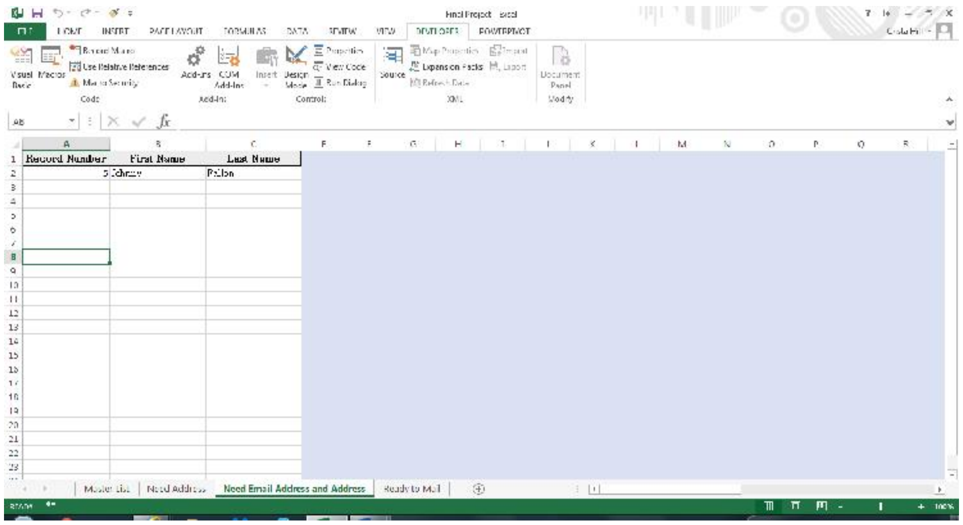


Record Number	First Name	Last Name	Email	Phone Number	Emailed Form?
3	Mom	Hill	crista84@gmail.com	(999) 999-9999	Sent 4/14/2015
4	Brittney	Hill	crista84@gmail.com	(888) 888-8888	Sent 4/14/2015
7	Dunbar	Hill	[illegible]@yahoo.com		Sent 4/14/2015

Figure 3: Need Address list

Identify Contacts that Need an Address and Email Address

The Address Organizer will automatically email contacts that need an address, however the user will need to know which contacts don't have an address and will not receive an email request. These contacts are sorted onto the "Need Address and Email" worksheet. The user will need to contact these individuals through another medium (phone, Facebook, in person) in order to get their address. As with the "Need Address" worksheet, these contacts are sorted automatically when the user clicks on the sheet. Because contacts can only be updated on the "Master List" worksheet, this sorted list should always be up-to-date.

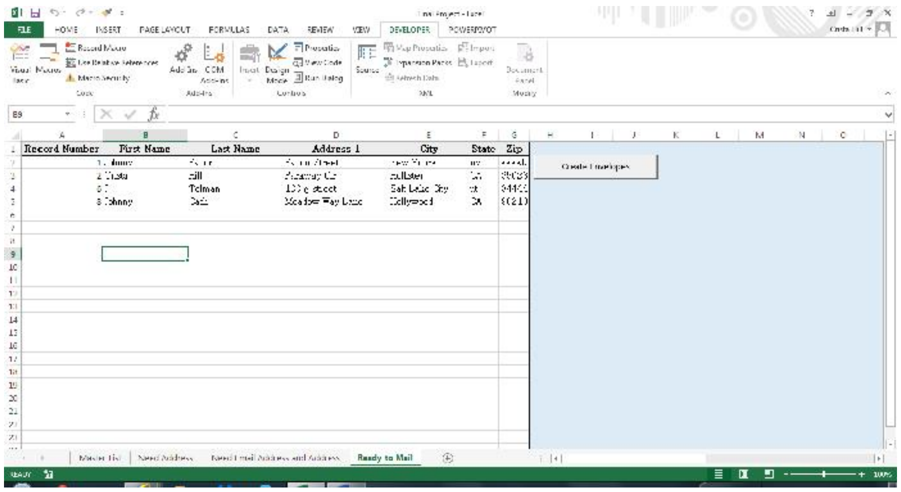


Record Number	First Name	Last Name
5	Johnny	Pallon

Figure 4: Need Address and Email Address List

Identify the Contacts that Have the Needed Information and are Ready to Mail

The Address Organizer will automatically create envelopes for contacts that have the required information. The list automatically populates when the user selects the "Ready to Mail" worksheet. As with the other lists, these contacts can only be updated on the "Master List" worksheet, so this list should always be up-to-date while the user is viewing the list.



Record Number	First Name	Last Name	Address 1	City	State	Zip
1	[illegible]	[illegible]	[illegible] Street	New [illegible]	ny	[illegible]
2	Crista	Hill	[illegible]way Cir	Hollister	CA	[illegible]
6	[illegible]	Tolman	100 [illegible] street	Salt Lake City	ut	844[illegible]
8	Johnny	Cash	Meadow Way Lane	Hollywood	CA	[illegible]

Figure 5: Ready to Mail list

Email an Address Request Letter to Contacts on the "Need an Address" List

The "Need Address" worksheet includes a button called "Email Address Request Form". This automatically sends the following email to the listed contacts:

> Johnny,
>
> I need your address. Please send it to me so I can get it added to my list.
>
> Thanks,
>
> Crista

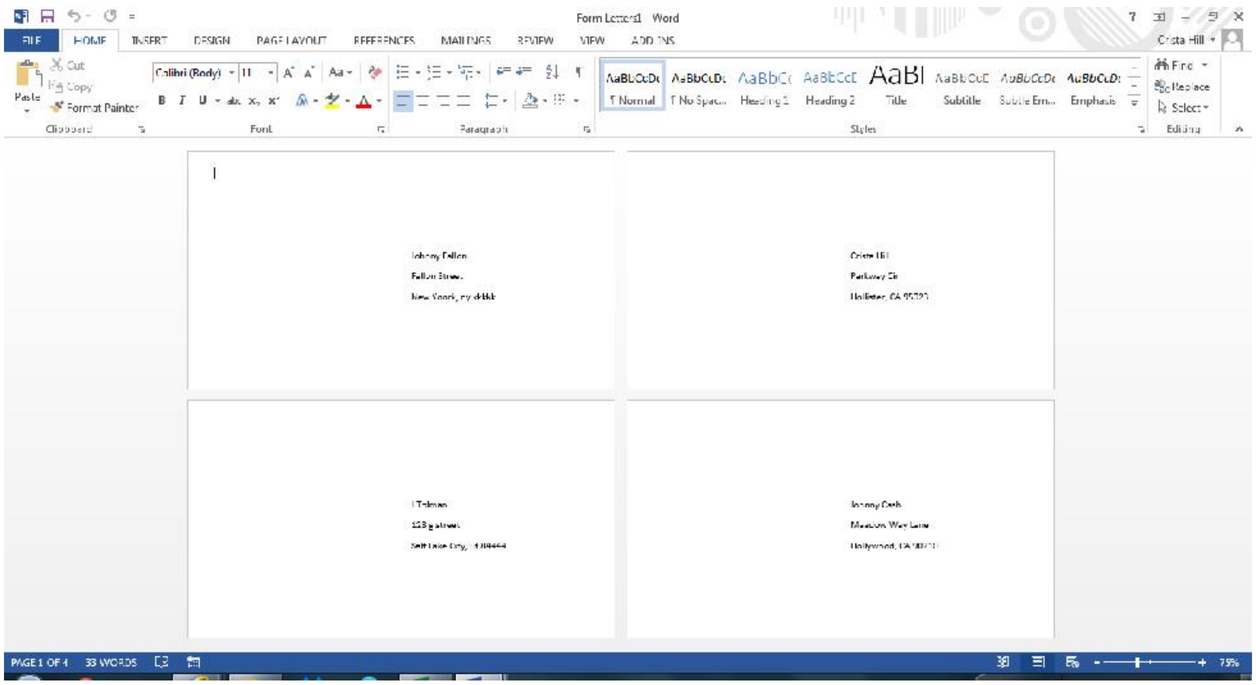
The User can customize the letter by modifying the text file in the same folder as the Excel file.

Once the email is sent successfully, the Organizer will population the field "Email Sent" with "Sent" & Date. This will also populate on the Master List so that the information will not be lost when the user selects a different worksheet.

Create Envelopes Using Mail Merge

The "Ready to Mail" worksheet has a button called "Create Envelopes". When the user clicks this button, the program will automatically open word and create envelopes for each contact in the "Ready to Mail" list. Because the button is located on this worksheet and contacts can only be updated on the "Master List" sheet, this list should always be up—to-date when the user clicks the button.

The Mail Merge file defaults to a 3 5/8 inch by 6 ½ inch envelope.



Johnny Fallon
Fallon Street
New Yoork, ny kkkkk

Criste Hill
Parkway Cir
Hollister, CA 95023

J Tolman
123 g street
Salt Lake City, [illegible]84444

Johnny Cash
Meadow Way Lane
Hollywood, CA [illegible]

Discussion of Learning

I discovered that learning to program is a lot like learning a language. There are infinite possibilities as to what you can do with code, but you have to know how to say it. The most difficult part of the project for me was knowing that something *could* be done, but not knowing the exact coding language that needed to be used to tell the computer what to do.

The Mail Merge part of my project was something that we had not done in class. I looked online to find examples of how to create this code, and found many examples, all of which were different and hard to understand for a beginning programmer. By studying the different code, I was able to piece together something that I thought *should* work, but I kept getting errors. Finally, after a lot of frustrated attempts, I took the code to the TA. He showed me that I had mixed up the Application with the Document and was trying to tell the Application to do something that only a Document could do. However, once we got this figured out, we still could not get the Mail Merge to run successfully.

I returned home and studied the examples closely, and running the code step-by-step decided it must be failing at the SQL line. After some creative Googling, I discovered that the SQL line needed to be written with brackets around the named range – which was not clear in ANY of the examples I had studied online. Once I added the brackets to the code: SQLStatement:="SELECT * FROM [Data]", my mail merge finally completed. I do not know of another endeavor where missing brackets can cause you hours of headaches.

I also learned the value of adding code at the worksheet level. I knew that I wanted my lists to sort automatically so that the worksheet would be very intuitive and the lists would always be up-to-date. I didn’t want the user to have to remember whether or not they had sorted the list since last updating the contact information. In order to do this, I learned how to add code to the worksheet level so that whenever the user clicks that worksheet, the list automatically updates. However, this required that I also really think through the layout of the program. If the list only sorts when you click on the worksheet, then it's important that the user can't push a button expecting the list is up-to-date unless the button is located on the worksheet with the list the button uses. Thinking through all the possible actions of the user and what the consequences could be is the most important part of programming, and in order to do that properly would require hours of testing by various users to ensure that all possible scenarios had been tested.

Another example of needing to think through the actions of the end user is Contact Updates. As I was writing the code, I realized that my list sorted the Master List by making a copy of the data on new worksheets. However, if the end user tried to update the data on the new worksheets, all of that data would be lost. I had to add protection features to these worksheets to ensure that all edits were made on the Master List. Had I not thought through this potential error, the end user would likely make updates throughout the workbook and would end up very disappointed in the results of all their hard work.

I also learned that it's important to consider whether a feature will be useful, or if it will just cause additional work for the end user. I had originally planned to include a feature where the user could import contacts from Gmail. However, when I ran a report showing all the contacts from my gmail account, I found that there were over 700 contacts, most with duplicate entries. Most of the contacts were not people I would send any kind of invitation to – many were not even people – and I decided that the user would likely spend more time cleaning up this list than they would creating a list on their own. So, I decided that including this feature in the project was not worth the time or the effort.

However, I also learned that programming is never really done. There are many different "tweaks" the end user could request and many unforeseen problems that could occur after more testing. Maybe I will find that users really would like an importing feature and that I should add that. What I have created is the beginnings of a code that will grow and evolve over time to create additional features, make it more user friendly, and ensure against failure.