VBA FINAL PROJECT: RESTAURANT FINDER

David Douma

April 15, 2013

Executive Summary

Background and Overview

If one has ever moved to a new and unfamiliar city, he/she has run into at least one major problem: not knowing what good restaurants there are to eat out at. This is especially a problem in Provo, Utah where it seems, at first glance, that there is nothing but your main staple chains such as McDonalds, Red Robin, Subway, Taco Bell, etc., that are located almost everywhere in the world. Little does the newcomer know that there are many small, local, mom and pop restaurants, as well as local joint favorites that survive only by word-of-mouth, scattered across the city. And, if they do know, they may not know a quick way to find out about them other than by years of experience. Therefore, unless one has spent considerable time looking, or has a local friend with a familiar knowledge of the area, they may never stumble across these small gems.

My task was to ease the pain of foodies, restaurant lovers, and people in search of a good restaurant for date night. The program I wrote takes what would be hours of searching, comparing, and asking around for the best places to eat, and simply loads a list of literally hundreds of restaurants in a city ranked in order from best to least best in a matter of minutes. Best of all, once the user has told the program the city desired and given all the required information, the program works on its own and does not need any further input or assistance from the user.

The program is simple. The user clicks on the button "Find Restaurants", and then inputs the required information into the user-forms that appear. The program then fetches the list of the desired number of restaurants given. The list includes the restaurant's name, address, neighborhood, price range, category, and rank position in the city according to Urbanspoon.com. Once this data has been stored, the program starts the heavy work: gathering the restaurant ratings from Urbanspoon, Yelp, and Google. The program uses mathematical weighted averages to sort the list and place it in a neatly formatted table that can be easily filtered and manipulated by the user for his/her convenience in finding the "perfect" local restaurant to eat at. Plus, there is no need to stand and wait around for the results. The user can go make a ham sandwich and relax while the program works; it will send them a text message when it is complete.

Implementation

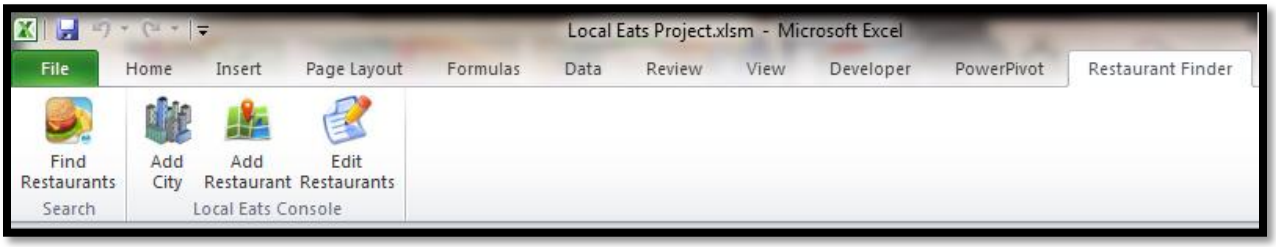
The implementation used in the solution of my program was to the benefit of making the program easier, user-intuitive, and efficient. The main parts used in my solution include:

- Ribbon Buttons
- UserForms
- Sub Procedures
- Web Agent
- ModGmail

Each of these parts is important and required to make the program work properly. The program also includes extra Ribbon Buttons, UserForms, and Sub Procedures to make the program even more user-friendly and helpful to the specific needs and wants of the user.

Ribbon Buttons

The first step to the program is for the user to locate the “Restaurant Finder” tab on the Excel ribbon. This tab contains the “Search” label and the “Local Eats Console” label that contains all the buttons needed to initiate and run the program.

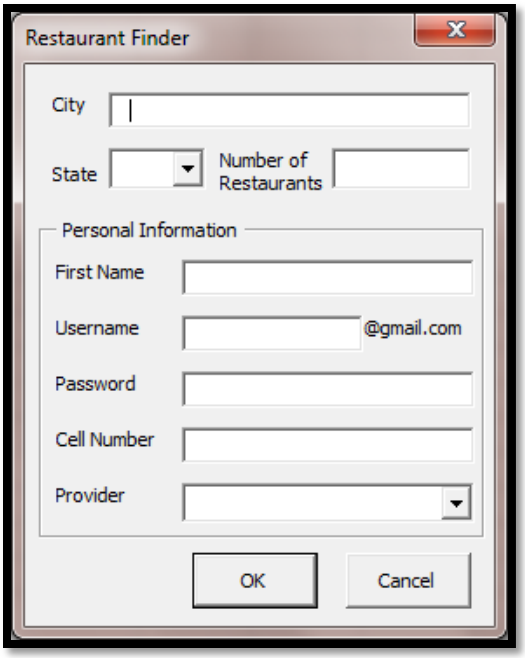


The “Search” label contains the main button needed to build the list of restaurants. This button is all that is needed to generate a list from best to least best restaurants in a specific city. The “Local Eats Console” is solely there to allow the user to manipulate the data set to their personal likings. For example, a user may feel that Provo, Utah is too small of a city and may wish to add surrounding cities in the Utah County area to build a more extensive list of restaurants within a reasonable driving distance. The user may also wish to add a restaurant that may not have appeared in the list originally just to see how it compares to those that are listed. Lastly, maybe a restaurant was categorized a little differently than the user feels is accurate and would like to edit that particular instance to reflect that in the list order. All of this adds to an easy, customizable way for a user to interact with the data set in order to find that best restaurant.

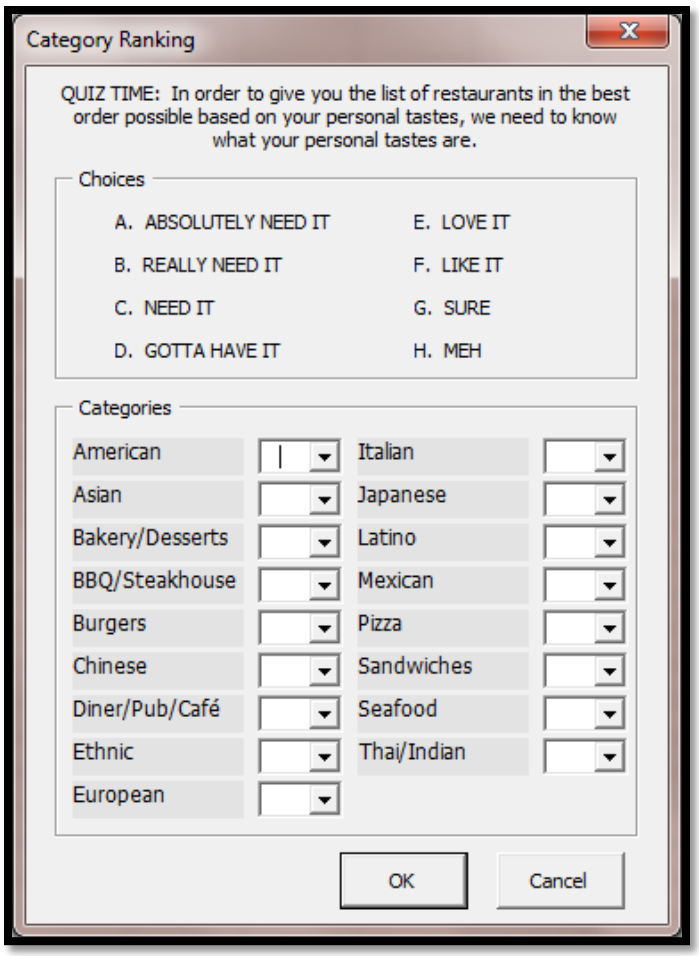
UserForms

Once the user has selected the “Find Restaurants” button, a UserForm appears for the user to interact with. The UserForm is designed to look professional and intuitive. It asks all of the main information that is required in order for the program to run correctly.

The user is able to type in the city and state where they are searching for somewhere to eat. The state box is a dropdown menu with a list of all 50 states in the United States of America along with the 14 provinces and territories of Canada. Data and lists used to populate the dropdown boxes are stored on a hidden tab called, “System Info”. After selecting the city and state, the user inputs the number of restaurants he/she would like to search and compare, and the list is generated. The recommended number is 100, but the more the better! Also keep in mind that the longer the list, the longer it takes to load. In the user-form the user will enter his/her first name for personalization. The username and password of a Gmail account is also required for means to accessing Google restaurant reviews and to send a text message to a phone once the program is completed. The “Cell number” and “Provider” dropdown box of all the major wireless carriers is also required in order to receive a completion text message.

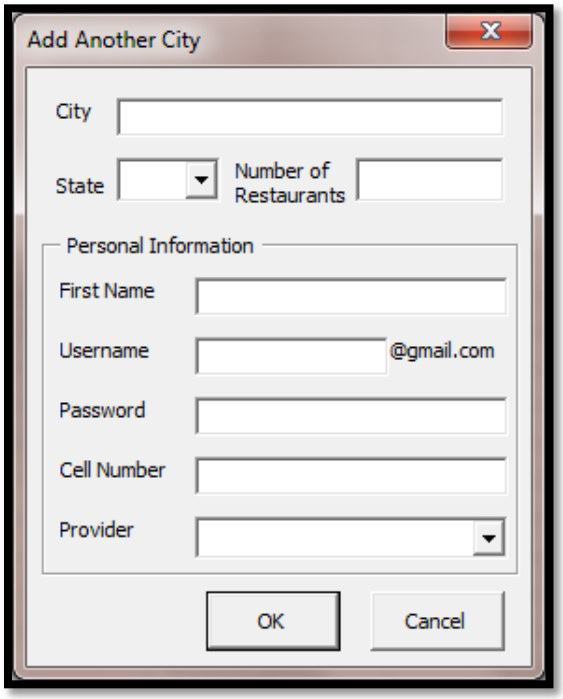


Once the "OK" button has been clicked a second UserForm appears with the intent to make sure the list of restaurants is as personalized and customized as possible. This second UserForm is used to allow the user to rank his/her preferred foods from favorite to least favorite.



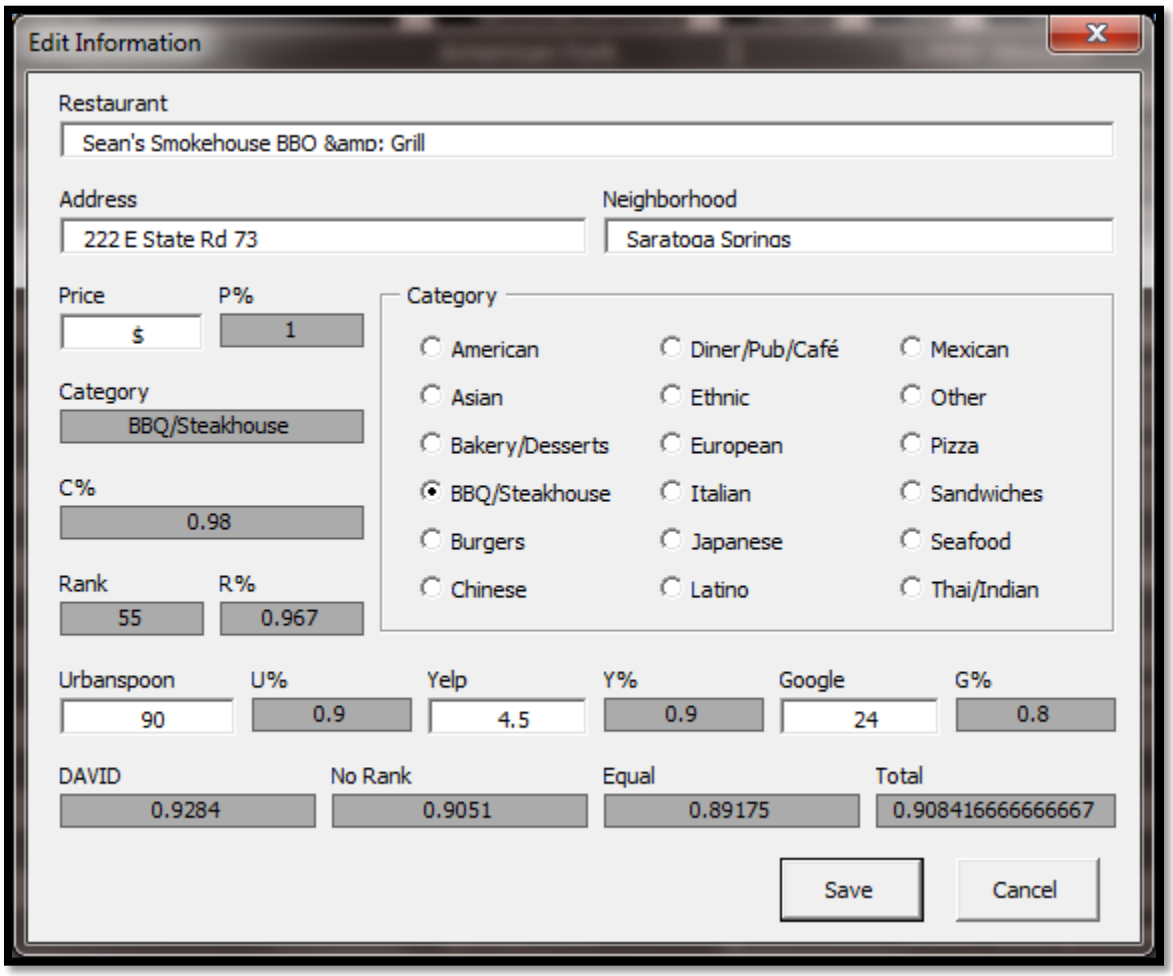
Not all categories of food are created equal. For example, some might like BBQ and Burgers and dislike Japanese and Seafood. Therefore, if the user truly wants to know what restaurants are best for him, it all depends on what he/she likes. Now most people don't hate food, so the multiple choice answers range from "ABSOLUTELY NEED IT" to "GOTTA HAVE IT" or "LOVE IT" all the way down to "SURE" or "MEH". Each choice is given a rating from 80% as the low and increments up evenly to 100%. This way the user can generate a list that categorizes the restaurants based on what they like pushing the less enticing categories lower on the list. It won't, however, exclude any of them from the list. It will keep all of the restaurants in the city the same, but only in a different order for the most part for any given person from another. These choices are saved onto the "System Info" for later use if necessary by other procedures the user initiates for their needs.

As for the other "Local Eats Console" buttons. The "Add City" button brings up the same UserForm as at the beginning. The only difference is the caption at the top reads, "Add Another City" rather than the original "Restaurant Finder". But all the same information is still required to re-run the program and add it to the previously generated list. It does not, however, force you to re-rank your category order. Once the first list is generated, the category rankings selected by the user, for customization, are automatically recorded into the hidden "System Info" tab, as stated earlier. This allows the user easy and quick access to editing the data set and formatting it to more fully fit the needs and wants they are seeking from the program. Once again, this makes for an easier, more user-friendly program to find that perfect restaurant.



The “Add Restaurant” button brings up a very simple UserForm that allows the user to type the name of a restaurant so it can add it to the list in similar fashion to the restaurants that have already been generated. This is meant only for the user who hears of or sees a restaurant and wants to know how it rates and compares to the other restaurants in the city.

The last button, “Edit Restaurants”, brings up a much bigger UserForm that displays all the data in that row for the selected restaurant. It allows the user to quickly edit any part of the name, address, neighborhood, price range, category, and the rankings if they need to be updated for any reason. When the “Save” button is clicked it submits all the new information and re-runs the formatting to identify whether or not the edits have changed its score and requires it to be moved up or down on this list.



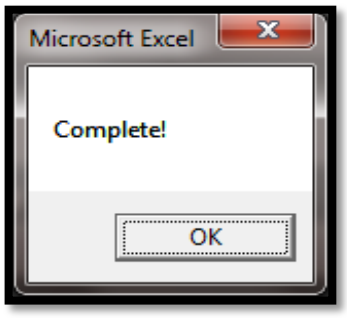
Sub-procedures

There are multiple sub-procedures found in this project that are required to make the program work correctly. The main sub-procedure, however, is "getRestaurants". This sub-procedure first, sets up a new worksheet called "Restaurants". It then calls upon a sub-procedure, "urbanspoon", which fetches the list of restaurants from Urbanspoon.com based on the information submitted by the user. It grabs all of the name, address, neighborhood, price range, category, and Urbanspoon ratings out of 100 and puts it on the worksheet that was created. The next sub procedure called on is, "yelp", which searches on Yelp.com for the rating out of 5 in the neighborhood/city for each restaurant that was previously listed from the "urbanspoon" procedure. The program then calls on the "google" sub procedure which searches on plus.google.com/local for the Google rating out of 30 in the neighborhood/city for each restaurant that was previously listed from the "urbanspoon" procedure.

An important note is that not all restaurants have a score on every rating site. If there is no relevant score available, an "NR" is placed in its spot to signify there is no rating. Now that all the information required has been gathered, the "getRestaurants" procedure calls on one last sub procedure called, "formatting". This sub-procedure formats the data set and inputs the ratings and other information into three mathematical weighted averages. The percentages are each given based on the information that was pulled. For the amount of "$" the percent goes down the more "$" there are, because it signals a less favorable place as it costs more. The Rank is given a percent from 100% to 85% in even increments depending on the amount of restaurants it is compared to. As mentioned earlier, the category ranking is in even increments from 100% down to 80% depending on what the user selected as their category rankings. Urbanspoon is rated out 100, Yelp is rated out of 5 stars, and Google is rated out of 30.

Because the ratings are out of different numbers, some deserve more weight than others. For example, Urbanspoon is out of 100% where a high 80 or low 90 is a good score, so it does not make a huge difference in the overall score. However, Yelp is only out of 5 stars and a good rating on Yelp is around 4 out of 5 stars which only equates to 80% and really skews the score. Thus a weighted average approach is necessary to put more weight on Urbanspoon than Yelp. The first weighted average is done using every relevant data set: price range, category, rank, Urbanspoon, Yelp, and Google. This makes the score be in favor of what the user likes, because it includes their category ranking so that the information is more personalized. The second weighted average only uses objective data such as rank, Urbanspoon, Yelp, and Google. This gives you a more well-rounded average of what restaurants are better than others without the personal tastes of the user, while still using a weighted approach. The third and last weighted average is meant to be an equalizer weight. It puts an even 25% weight on each of the four objective data sets: rank, Urbanspoon, Yelp, and Google. The final score is the average of those three weighted average scores; the highest score is best, the lowest score is least best.

The last thing the "formatting" sub-procedure does is to put all of the data into a neat table. The table is nice, because it allows a user to filter and see only specific information for what they are looking for at a certain time. The sub-procedure also color coordinates the restaurants into thirds (green, yellow, and red) so that when the data is filtered you can still see how a particular restaurant compared against the whole rather than just the filtered data set. Once completed, the "getRestaurants" sub-procedure sends off a text message, beeps, and displays a message box to inform the user that the program is "Complete".



Web Agent

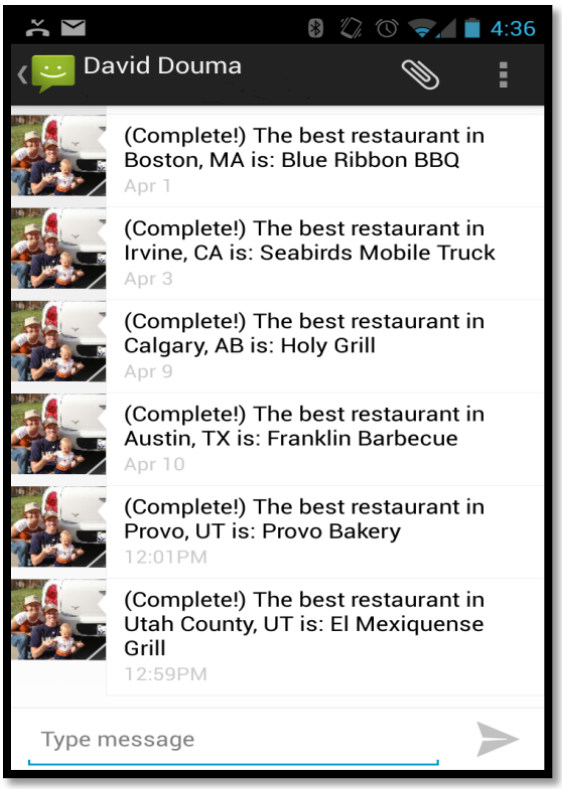
This project relies heavily on using the web agent class module to extract information from the web. The web agent allows for use of navigating a website for specific information. The agent contains methods for the user to make it easier to navigate through a web page source and html file in order to find the data. Methods I used most include:

- openPage – to open the web browser to a specific page. I often used loops to change parts of the URL to include restaurant name, city, state, etc. That way I could use the html file and not actually have to interact to much with the built in search engines of a page.
- position – to set the position or cursor back to 1 when I move through the html source and grab only a specific portion of the source.
- moveTo - to move the cursor through the html source to different spots in order to fetch the useful information.
- moveBackTo – to move backwards through the html source in order to get exactly where I needed to be in the source to fetch the information.
- getText – to actually fetch and store the information and assign it to a variable within VBA.

modGmail

I used the modGmail module to send a text message via a Gmail account to a cell phone in order to inform the user that their list is complete and ready for their viewing. I made use of the sendGMail method that passes through variables that are necessary to actually send the text message:

- sendGMail (sendTo, from, pw, subject, body)
  - sendTo – I made use of the getAddress method contained in the module and included the variables for a cell phone number and provider such that it would grab the number that the user entered in and pull the provider information from the hidden "System Info" tab.
  - from – was the email account the user entered as a string.
  - pw – was the Gmail account password the user entered as a string.
  - subject – since this text only sent when completed, the subject line read, "Complete!" as a string.
  - body – the body of the text read, "The best restaurant in (City, State listed by user) is: (the name of the first restaurant in the sorted table)."



The final result with the information and formatting looks like this:

RESTAURANT	ADDRESS	NEIGHBORHOOD	PRIC	P%	CATEGORY	C%	RANK	R%	URBANSPOON	U%	YELP	Y%	GOOGLE	G%	DAVID	NO RANK	EQUAL	TOTAL
El Mexiquense Grill	74 N W STATE RD	American Fork	$	1.0000	Mexican	0.9500	22	0.9868	94	0.9400	4.5	0.9000	25	0.8333	0.9422	0.9360	0.9150	0.9311
Provo Bakery	190 E 100 N	Provo	$	1.0000	Bakery/Desserts	0.8300	84	0.9496	93	0.9300	4.5	0.9000	29	0.9667	0.9146	0.9399	0.9366	0.9303
Sundance Foundry Grill	8841 Alpine Scenic Highway	Provo	$$$	0.3000	BBQ/Steakhouse	0.9800	97	0.9418	94	0.9400	4.5	0.9000	28	0.9333	0.9157	0.9375	0.9288	0.9273
Dlf Cheesesteaks	933 W 500 N	American Fork	$	1.0000	Sandwiches	0.9300	40	0.9760	91	0.9100	4	0.8000	29	0.9667	0.9329	0.9328	0.9132	0.9263
Lehi Bakery	172 W Main St	Lehi	$	1.0000	Bakery/Desserts	0.8300	76	0.9544	96	0.9600	4.5	0.9000	26	0.8667	0.9160	0.9413	0.9203	0.9259
Daylight Donuts	125 E State Rd	Pleasant Grove	$	1.0000	Bakery/Desserts	0.8300	92	0.9448	98	0.9800	5	1.0000	NR	NR	0.9079	0.9212	0.9456	0.9249
Mountain West Burrito	1796 N 950 W	Provo	$	1.0000	Mexican	0.9500	29	0.9826	92	0.9200	4.5	0.9000	25	0.8333	0.9344	0.9248	0.9090	0.9227
Eliane French Bakery	1750 S State St	Orem	$	1.0000	Bakery/Desserts	0.8300	61	0.9634	94	0.9400	5	1.0000	24	0.8000	0.9092	0.9290	0.9259	0.9214
Yogurtland	534 E. University Parkway	Orem	$	1.0000	Bakery/Desserts	0.8300	121	0.9274	95	0.9500	4.5	0.9000	27	0.9000	0.9105	0.9332	0.9194	0.9210
BYU Creamery	1209 N 900 E	Provo	$	1.0000	Burgers	0.9800	36	0.9784	92	0.9200	4.5	0.9000	24	0.8000	0.9377	0.9185	0.8996	0.9186
Chubby's Cafe	670 West State Road	Pleasant Grove	$	1.0000	BBQ/Steakhouse	0.9800	18	0.9892	94	0.9400	4.5	0.9000	NR	NR	0.9330	0.9062	0.9147	0.9180
J Dawgs	858 N 700 E	Provo	$	1.0000	American	0.9300	1	0.9994	91	0.9100	4.5	0.9000	24	0.8000	0.9259	0.9198	0.9024	0.9160
Zub's Pizza & Sub's	520 N Main St	Springville	$	1.0000	Pizza	0.9500	101	0.9394	94	0.9400	4.5	0.9000	24	0.8000	0.9294	0.9168	0.8949	0.9137
The Smoking Apple	70 N. State St.	Lindon	$	1.0000	BBQ/Steakhouse	0.9800	5	0.9970	86	0.8600	4.5	0.9000	26	0.8667	0.9271	0.9041	0.9059	0.9124
Bombay House	463 N University Ave	Provo	$$	0.9000	Thai/Indian	0.9000	4	0.9976	91	0.9100	4.5	0.9000	24	0.8000	0.9130	0.9193	0.9019	0.9114
Joe's Café	1126 S. State Street	Orem	$	1.0000	Other	0.8300	34	0.9796	93	0.9300	4	0.8000	26	0.8667	0.9056	0.9289	0.8941	0.9095
Sean's Smokehouse BBQ & Grill	222 E State Rd 73	Saratoga Springs	$	1.0000	BBQ/Steakhouse	0.9800	55	0.9670	90	0.9000	4.5	0.9000	24	0.8000	0.9284	0.9051	0.8918	0.9084
Joe Coffee Shop & Espresso	145 E Utah Ave Ste 2	Payson	$	1.0000	Other	0.8300	173	0.8962	97	0.9700	5	1.0000	NR	NR	0.8938	0.9006	0.9266	0.9070
India Palace	98 W Center St	Provo	$$	0.9000	Thai/Indian	0.9000	11	0.9934	90	0.9000	4	0.8000	26	0.8667	0.9103	0.9180	0.8900	0.9061
Thai Mango	2250 N University PKWY #4	Provo	$	1.0000	Thai/Indian	0.9000	23	0.9862	94	0.9400	4	0.8000	23	0.7667	0.9179	0.9209	0.8732	0.9040
Nicolitalia Pizzeria	2295 N University Pkwy	Provo	$	1.0000	Italian	0.9000	8	0.9952	87	0.8700	4.5	0.9000	25	0.8333	0.9069	0.9036	0.8996	0.9034
Village Pizza	3545 Ranches Pkwy	Saratoga Springs	$$	0.9000	Pizza	0.9500	69	0.9586	93	0.9300	5	1.0000	20	0.6667	0.9164	0.9026	0.8888	0.9026
Hickory Kist Gift & Deli	1533 N Main St	Spanish Fork	$	1.0000	Bakery/Desserts	0.8300	75	0.9550	96	0.9600	NR	NR	27	0.9000	0.8943	0.9027	0.9101	0.9024
Asahi Japanese Restaurant & Sushi	1470 North State Street	Orem	$$	0.9000	Japanese	0.8000	43	0.9742	95	0.9500	4.5	0.9000	NR	NR	0.8843	0.9078	0.9131	0.9017
Ernie's Sports Deli	182 W Center St	Orem	$	1.0000	Sandwiches	0.9300	90	0.9460	86	0.8600	4.5	0.9000	27	0.9000	0.9077	0.8938	0.9015	0.9010
Pantrucas Chilean Restaurant	3161 North Canyon Rd	Provo	$	1.0000	Ethnic	0.8500	83	0.9502	92	0.9200	4.5	0.9000	24	0.8000	0.8995	0.9101	0.8926	0.9007
Pho Plus Noodle House	908 South State Street	Orem	$	1.0000	Asian	0.8000	16	0.9904	92	0.9200	4	0.8000	25	0.8333	0.8934	0.9221	0.8859	0.9005
Pizzeria 712	320 S State St #185	Orem	$$	0.9000	Pizza	0.9500	9	0.9946	84	0.8400	4.5	0.9000	26	0.8667	0.9071	0.8934	0.9003	0.9003
Two Jack's Pizza	30 N Main St	Spanish Fork	$$	0.9000	Pizza	0.9500	107	0.9358	94	0.9400	4.5	0.9000	22	0.7333	0.9170	0.9057	0.8773	0.9000
Boudreaux's Bistro	47 S. Main ST., Payson, UT	Payson	$	1.0000	Ethnic	0.8500	124	0.9256	95	0.9500	4	0.8000	25	0.8333	0.9035	0.9177	0.8772	0.8995
Cubby's Chicago Beef	1258 N State St	Provo	$	1.0000	BBQ/Steakhouse	0.9800	51	0.9694	91	0.9100	4.5	0.9000	NR	NR	0.9162	0.8819	0.8986	0.8989
Taco Amigo	239 E State Rd	Pleasant Grove	$	1.0000	Mexican	0.9500	66	0.9604	90	0.9000	4.5	0.9000	23	0.7667	0.9162	0.8981	0.8818	0.8987
Sensuous Sandwich	378 E 1300 S	Orem	$	1.0000	Sandwiches	0.9300	59	0.9646	95	0.9500	3.5	0.7000	24	0.8000	0.9229	0.9194	0.8537	0.8987
Kohinoor	75 S. State Street	Orem	$$	0.9000	Thai/Indian	0.9000	45	0.9730	92	0.9200	4	0.8000	24	0.8000	0.9066	0.9119	0.8733	0.8973
Aiyara Thai Cuisine	224 E Main St	American Fork	$	1.0000	Thai/Indian	0.9000	114	0.9316	96	0.9600	3	0.6000	27	0.9000	0.9173	0.9245	0.8479	0.8966
Molly's	753 W Columbia Ln	Provo	$	1.0000	American	0.9300	10	0.9940	95	0.9500	4.5	0.9000	18	0.6000	0.9188	0.9082	0.8610	0.8960
Sundance Tree Room	8841 N Alpine Loop Rd	Orem	$$$$	0.2000	American	0.9300	106	0.9364	90	0.9000	4.5	0.9000	26	0.8667	0.8764	0.9059	0.9008	0.8944

Learning and Conceptual Difficulties

Probably the biggest thing I have learned doing this project is about thought process. I learned that it is best to think through the plan and work step by step. There were times when the search engines would not work the way I expected them to, and I had to change and manipulate the URL. Each time I encountered a problem like this, I was amazed at how I could come up with another route to the same destination as the one that I originally had thought of. There are many ways to solve the same problem.

I also realized how often I overlooked small parts to a program. Sometimes I did not realize how something as small as a hyphen, or the way a string was written, would affect the program as a whole. It forced me to develop patience and to experiment many different methods, including debugging techniques, to figure out what was wrong and what would fix it.

This project was exciting to me and I found myself just wanting to work on it all the time to make my program better and better. Besides a few lower level I SYS classes, this was a fairly new thing for me and I had to develop a whole new mindset. It seemed like a puzzle and whenever I figured something out and got my program to work as planned, I was extremely pleased and felt satisfied with the work I was putting forth.

Assistance

I received no assistance on the direct code of my program and project. I used the Web Agent file to help search and obtain data from the web. I also used the modGmail file to help send a text message to the user when the program was complete. Both the Web Agent and modGmail files were made by Professor Gove Allen. Other than those two files, I referenced past assignments I had done in class, and occasionally did a Google search. I did not, however, receive any help from any individuals and no one helped me on any of the direct code used in my own Sub Procedures and UserForms.