

Final Report List.it

Michael Lopez

Advisor: Dr. Laura Grabowski
Semester: Fall 2020

December 9, 2020

1 Introduction

The project that I proposed and completed for my Senior capstone was List.it, a website that aimed to simplify the selling experience for people by allowing them to list items that they want to sell or trade for other users to offer or bid on. They can also organize events such as group yard sales or bake sales to invite other users to, communicate and send payments through the site for completed listings. My motivation in completing this project was due to my enjoyment in web development and wanting to provide a simpler solution to a problem, but the idea for the site was from seeing my mom and late aunt using Facebook Marketplace to sell and trade their items. Both had trouble using Marketplace to sell their items, as they had to join several different groups in order to reach a larger audience and track their bids and offers themselves. I felt that the process could be simplified so that users can view all bids or offers for a listing at once, which gave me the idea for List.it.

I also had inspiration from other sites such as Amazon, eBay, Carousell and LetGo, which are several different sites and apps people use to sell and trade items. My goal was to try to combine a few of the sites' features into one site that can provide most of the functionality they offer. The site helps a user by allowing them to manage all their items in one place and provides them the choice of listing their own items for trade or cash. Users who are not interested in listing items can search for and offer on other user's listings. It also allows communication on site without having to give personal contact information to others and provides users a way to make payments onsite using PayPal. A history of their transactions and payments are kept on site after for reference should that need it.

2 Background

2.1 Preparation

My preparation for this project included taking Software Engineering and Computer Security at SUNY Potsdam while practicing with a few independent projects in my spare time outside of classes. I feel that Software Engineering helped to prepare me by teaching me how to use Django for back-end web development, when previously I had only used Ruby-on-Rails when practicing outside of school. The course also taught me about how to develop a site using test driven development, which was extremely helpful during my project as it helped me to find errors within my code easily. Lastly, Software Engineering taught me about Scrum which was very useful as it helped with organization of tasks for the project and to review what I accomplished and hoped to accomplish with my advisor. I also feel that it helped me to stay focused as I was able to create a goal and work towards it without getting distracted. Computer Security otherwise taught me the importance of keeping information safe and secure, which was incredibly important to consider in my project as users would be exchanging money through PayPal and their information needed to be kept secure. If the wrong person was to obtain that information, it would create a bad image for the site and for me as a developer, so security was one of my biggest concerns throughout the project.

Outside of classes I had practiced developing sites using Ruby-on-Rails as mentioned earlier. I had some previous experience with test driven development using Ruby-on-Rails, but not as much as I had after taking Software Engineering. However, it did help me to practice using HTML and CSS when developing the design and layout of my sites and introduced me to using JavaScript and some JQuery, which I was able to reference back to for List.it. I was also introduced to Bootstrap, which is extremely useful in designing layouts as it provided me with pre-written CSS components and was heavily utilized in the project. Lastly, developing sites using Ruby-on-Rails helped me to get familiar with a few tools such as Stripe, which I used in one project for processing payments by users so that they could subscribe to other users. The previous knowledge with Stripe did help me with implementing PayPal's software development kit (SDK)[2] into List.it.

2.2 Practice

Developing List.it proved to be very rewarding when it came to developing new skills and building upon previous skills. I feel that I am more confident in working with Django over Ruby-on-Rails when it comes to back-end web development. Although Ruby-on-Rails does provide benefits that Django does not, I started to feel that Django was easier to use during development and provided better organization. Other skills that I feel more confident with include writing tests and commenting my code. Previously, I would write tests after writing the code, which was not as useful as writing tests beforehand. However, during the development of the project, I got more into the groove of writing tests before my code, which helped me to pay focus to parts of code I may have ignored and to include tests for it. My code commenting did improve as well, as I included comments for classes explaining them, and then included small comments throughout to explain what was being done. This helped me to make sure that I remembered all parts of my code, which meant less time was spent going through to add forgotten code. Lastly, a few other skills I was able to improve upon was documenting features as I implemented them so that users would know exactly how that feature is meant to work. As mentioned earlier, I was able to detect and correct errors more easily thanks to testing, although there were a few times that errors took longer to solve but they were few and far between. Those errors were typically results from implementations of code that were new to me, so after being able to correct the errors I learned how I could avoid them in the future.

New skills that I feel that I have gained after finishing List.it include managing my time, solving problems on my own and working with a few new tools. My time management skills were not that great before beginning my project, as I would tend to put off tasks until the last minute for some assignments and projects. During development though, I was able to stick to a schedule more easily to work on List.it and only had a few time management issues due to events outside of the project that were unexpected and out of my control. I learned how I can solve certain problems on my own without needing to discuss it amongst a team or another person and now feel I can reference those problems later on to provide help to others. After completion of List.it, I now know how to use a few new tools such as PayPal's SDK and GeoIP2, and how to use AJAX calls to return data to a view or to change the data a view displays without a user needing to refresh their page.

3 Methods

To develop List.it, I worked with a number of tools, some that I were familiar with before starting on List.it and several that were new to me. The programming languages that I worked with in development were Python/Django to develop the back-end of the site, and HTML/CSS for the front-end. I did use Javascript and JQuery as well which were useful for both the back-end and front-end of the project. I learned a lot more about JQuery during development such as with working with AJAX calls, and it has made me want to start working with it more for other projects. It sort of became a replacement for my plans in using React.js as it provided me with the functionality I needed that I would have used React for. I still did want to work with and implement React into several parts of the sites, but did not have the time to learn it and implement it unfortunately.

To build and code List.it, I used Atom as my text and code editor while using GitHub for version control. Atom allowed me to view the directory of my project in one place and work with and view several files at once. It has built in integration with Git and GitHub so I was simply able to make and push commits from Atom to GitHub or fetch previous commits if I needed to. Atom was very helpful as it allowed me to see what changes I had made before making a commit and what features I had previously implemented as well.

The coding standards that I followed while developing the site through Atom is that class names would be in camel case while method names would be in lower case, with words separated by underscores. For variables, the names were made to be meaningful, model class fields would be in mixed case while variables for other classes and methods would be lower case separating words by underscores. Commenting standards included having a header comment above classes and methods to describe what they were meant for, and smaller additional comments would be included within to explain separate instances of code. For model

classes, I made sure to implement the class variables first, then I would define any properties needed and then methods such as name representations when referencing an object. Lastly, I made sure to indent new blocks of codes with tabs throughout the project, separated different methods and classes with a new line between them, and made sure that all imports needed were kept at the top of each file.

The last few tools I had worked with include GeoIP2 for getting a user's location, PayPal's SDK for making payments and MySQL for the project's database. I choose to use MySQL for the development database as I had worked with it before in past projects, but my knowledge about it was getting rusty as it was a while since I had last worked with it. I also wanted to learn how I could implement MySQL into my own machine and felt that it would be a better database than SQLite should List.it were to ever be used regularly or expanded upon. GeoIP2 and PayPal's SDK were completely new to me in this project, and I enjoyed learning about how to use them and have them be implemented into my site working with the rest of the site's features.

As for management tools, I utilized Trello, Discord, Zoom and Excel. Trello was extremely useful in the organization of the project, as I could create a backlog of tasks that were needed to be met and organize weekly sprints to work towards. It was also useful for referencing old sprints later on so I could review what I had implemented before. After organizing tasks for a sprint, I would then use Excel to design sprint burndown and velocity charts so that I could track how much I had accomplished each week. If I noticed that progress had slowed down, I would try my best to make up for it in the next sprint. Lastly, Discord and Zoom were used to communicate with my advisor to review weekly sprints and for discussing upcoming sprints or projects or for either party to ask questions.

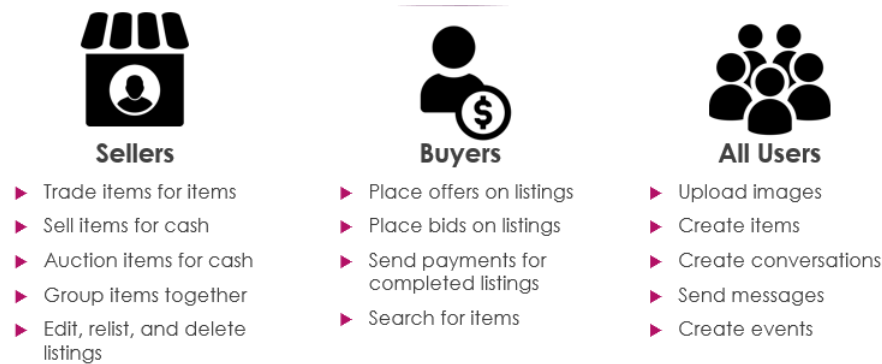
My design and implementation process for List.it was kept simple. At the start of each week, I would organize a number of tasks into a list that would then become my sprint goal for the week. I would try to prioritize the most important tasks first if possible, but previous unfinished tasks would come before them should there be any. As I went through implementing each task needed, I would make sure that it would work correctly using test driven development, and should any code be affected by it, I would go back to the code and change it so that it would work with the new implementation. After implementation, I would add documentation on site for it so that users would know how its intended to work and how to use it. Lastly, I would do a final run through of the newly written code to make sure that it's as simple as I could make it and that it also remains simple to use for users. There were a few times due to time constraints that I could not go back to simplify things, but if I was to continue working on List.it after the semester I would make sure to go back to those instances to make them easier to understand.

The development methodology I used for developing List.it started with me designing the model class for the feature to be implemented. This included any subclasses if they were needed, such as with listings to differentiate the kinds of listings and notifications for different models. After designing the model class, if users are intended to create the objects, I would implement the creation form and view for the model. If it isn't needed, I would just simply move onto designing the list and detail views for the model. Helper methods would be written as well for the views if needed in order to change data, such as setting a message to read, or to update the list of objects based on given parameters such as with the item search feature. Once the list and detail views were finished, update and deletion views would be implemented so that users can modify or delete the objects that they own. Lastly, the documentation covering the feature would be written on site so that users can reference it and learn how each feature works.

4 Results

For the results of List.it, I was able to complete approximately 94 percent of the estimated workload for the project. The only tasks that I was not able to complete were improving the user interface of the site using React.js, but otherwise, the site is functional and is working as intended.

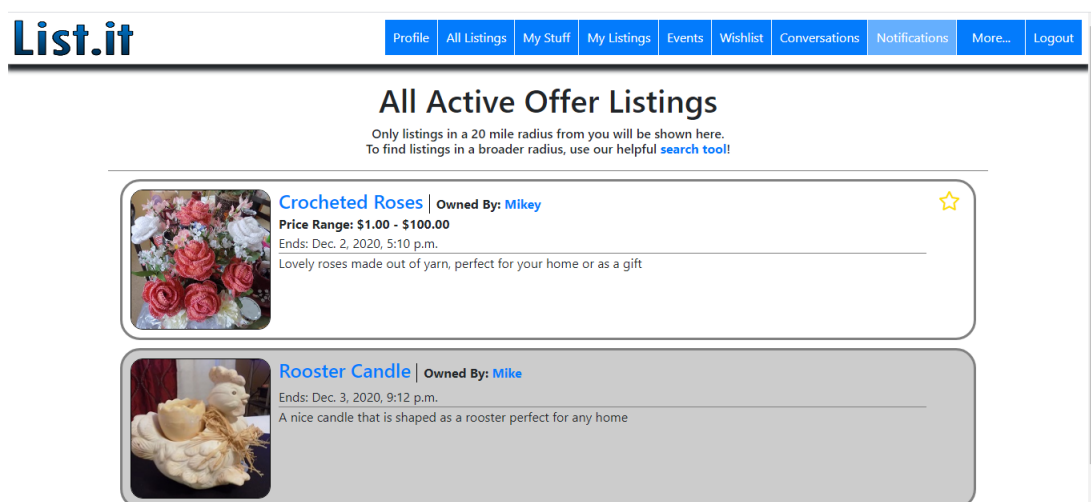
The graphic below gives a brief overview of what users are able to do on the site:



In addition, sellers and buyers are able to leave ratings for one another after they have completed a listing between each other. All users can also search for items using parameters such as names, tags, etc. and report inappropriate listings, users, etc. that they come across on the site. They can also favorite any listings that other users own.

When a user first access the site, they are able to sign up and create an account. Once they have made an account, they are able to start uploading images to the site. They then use the images they have uploaded to create items in which they use to create either an offer, auction or wishlist listing. Offer listings allow users to choose an offer they like most for their items, whether it be other items and/or cash while auction listings allow users to bid on a listing until it ends or a user places an autobuy bid which wins the listing. A wishlist listing is simply for users to advertise items that they are searching for that other users may own so that they can contact the user about the item. Users must first create a wishlist though in order to create a wishlist listing. A wishlist allows a user to create a list of items that they are interested in trading for or buying from other users, which helps them to separate the items from ones that they own.

The below graphic shows a view of offer listings that users are able to view:



Users are able to see the name of the listing, the price range the owner of the listing wants if they're open to money offers, when the listing ends and a short description of the listing. Auction listings display similarly except they show the starting bid for the items, the minimal increment for bidding and the autobuy price if the user set one. Users can favorite a listing for reference later by clicking on the star icon in the top right corner of the listing's container. To view more details about the listing, a user can click on the name of the listing to go to the detail view, where they can also place offers or bids and for the owner to view the current offers and bids. The listing owner can accept offers through the detail view as well, while users that have placed offers and bids can easily track them through separate list views.

Search Listings


Name:

Listing Type:

Search Radius:

Tags:

Home
Kitchen
Living Room
Bedroom




Crocheted Roses | Owned By: [Mikey](#) ★

Price Range: \$1.00 - \$100.00

Ends: Dec. 2, 2020, 5:10 p.m.

Lovely roses made out of yarn, perfect for your home or as a gift



Rooster Candle | Owned By: [Mike](#)

Ends: Dec. 3, 2020, 9:12 p.m.

A nice candle that is shaped as a rooster perfect for any home

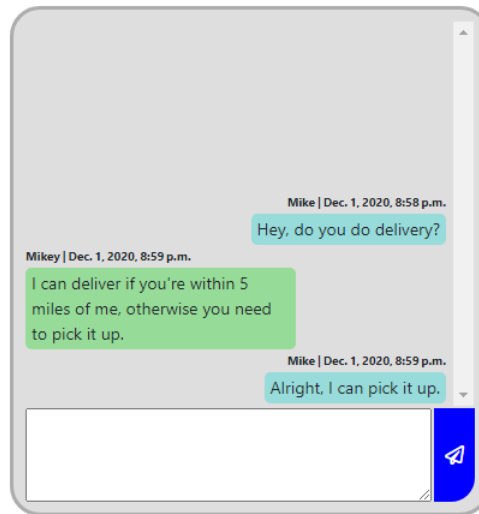
Users can find a greater range of listings or find more specific listings by using the search function on the site. I was able to implement this function thanks to a tutorial on [Openfolder.sh\[1\]](#) which explains how to use Javascript, JQuery, and AJAX calls with Django to populate a list based on parameters passed in by the user. Users can search for listing using a name of a listing, and can search with additional names by separating parameters with commas. They can filter listings by what type of listing they are and find listings in a certain range from where they live. Lastly, they can select tags for the types of items they want so that they only find listings that include items for a home or bathroom or whatever tag they may select.

Once a listing is completed between users, a receipt is made for each user giving them details on what is being exchanged between users. Through receipts, users that have offered money to another user can send a payment through PayPal to the other user. As long as the PayPal email address the receiving user has given is valid, users can then enter their PayPal credentials and review the details on the transaction before sending the payment. If they are satisfied, they can complete the payment to finalize the exchange. Each user is also able to leave a rating for another user on their profile after a listing is completed between them. Ratings are not able to be made until the completion of a listing to help reduce false ratings. They are able to rate a user from a scale of one to five and leave feedback for the other user.

Crocheted Roses

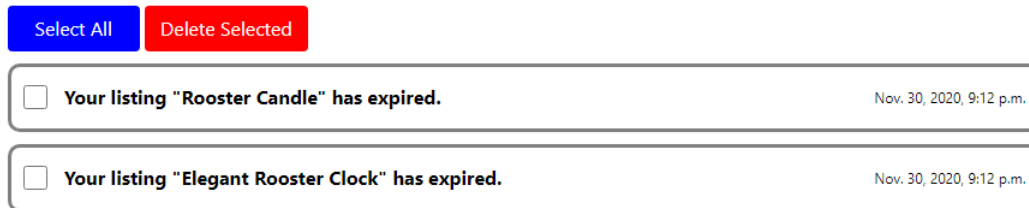
Conversation with Mikey

REMOVE



If needed, users can contact another user by starting a conversation with them. They can start a conversation several ways by messaging a user through a listing they own, through a users profile or through a receipt. Conversations allow users to negotiate where they want to pick up or deliver items, ask questions about items, see if a user may be interested in an item or listing or to discuss events. Events are a way of users to organize occasions such as yard sales, bake sales, etc. and invite nearby users to participate in them. As all conversation is through the site, a user does not need to exchange their contact information such as e-mail or phone number with another user.

Notifications



Page 1 of 1.

Users receive notifications on the site for a variety of reasons for their convenience. They can receive notifications telling them that their listing has ended without accepting an offer or having a bid, an auction ending with a winning bid, a listing has received an offer or a user has edited/retracted their offer. Other users receive notifications if their offer has been accepted or rejected or if their bid has won an auction. Notifications are also made if any objects are removed if they have been reported by other users for being malicious or false or for any other reasons. There are also other notifications made when being invited to events or leaving ratings just to name a few.

Lastly, there is a documentation page provided on site for users to reference. They can find details on the FAQ page such as how to go about creating a listing, how to place an offer or bid on a listing, how to create items/wishlists/images/etc. and many more references for the features on the site. There are also some FAQs provided as well in case a user may have any questions regarding a particular feature.

5 Conclusion

Overall, developing List.it has taught me a lot and has introduced me to a lot of new tools. I now feel more comfortable using Javascript and JQuery and am now familiar with PayPal's SDK. I feel that I have improved in commenting and code standards and feel comfortable with tackling any sort of feature that may need to be implemented in future projects. I now understand how to work with a schedule and deadlines and how to keep myself organized. I have accomplished a lot during the development and am confident that List.it may be used by people in the future. I would love to continue working on List.it if that was to become a reality. My advice to future students that are planning on starting their senior project is to choose something they're passionate about in Computer Science. If they like building AI, or developing sites like me, or developing software, go for it. Make sure to ask people around you for opportunities as well as they may know of some you won't learn about otherwise. Also, don't be afraid to look for help when you get stuck on something or simply can't push yourself anymore, we've all been there and there's always someone who will be happy to help.

References

- [1] MANSOUR, S. K. Django tutorial: as-you-type search with ajax.
- [2] PAYPAL. Paypal checkout, 2020.