

Goodbay

“A website specializing in finding new homes for unwanted items.”

Customer: Jonathan Horton

Date: 2017-06-09

Project B, Group #7

Isaac Neibaur

Johannes Pikel

Berry Semexan

Akshay Subramanian

Ivan Xa

Table of contents:

<u>Introduction</u>	3
<u>First XP Cycle</u>	
<u>User stories</u>	3
<u>Story task and effort estimates</u>	5
<u>Priority list</u>	5
<u>Summary of pair work</u>	6
<u>Summary of unit tests</u>	7
<u>Summary of acceptance tests</u>	9
<u>Second XP Cycle</u>	
<u>Summary of changes</u> (user stories, estimates, and priority list)	11
<u>Summary of pair work</u>	11
<u>Reflection</u>	12
<u>Appendix A - ER diagrams</u>	14
<u>Appendix B - Pros and cons of waterfall vs extreme agile</u>	17
<u>Appendix C - Example of acceptance/regression testing</u>	19
<u>Appendix C - Step by step instructions on how to run the project</u>	19

Introduction

Goodbay introduces a new marketplace where unwanted, light-weight, and easily shippable items can be donated. Users interested in available products can select a particular item and receive it free of charge. While the donor will pay the shipping cost, they will in turn receive a tax write-off. Goodbay revolves on the idea of recycling. Items that do not have enough value to re-sell can now find a home with a user looking for such an item on this marketplace.

First XP Cycle

User stories

In order to collect our initial list of user stories we “chunked” the vision statement our customer Jonathan created. Once we had worked our way through his vision statement, we then invited our customer via e-mail to review the user stories and to add user stories that we may have missed when looking over his vision statement. We made sure to emphasize that we did not intentionally order the vision statements in our document that we sent to him.

In order to enforce a 3x5 note card size for each vision statement we used a single-celled table per user story in our document, whose dimensions were set to approximately 3” x 5”. Our customer, Jonathan, did have another user story to add to our list. The full list of user stories that we initially collected are as follows:

User Story Title	Detail
Upload an item for donation to the site	<ul style="list-style-type: none">• Imagine a database where users can list items in their possession, each item with a headline, optional text description, optional picture, and three keywords.• Key Features:<ul style="list-style-type: none">○ Creating listing must be fast, needing only:<ul style="list-style-type: none">■ Headline■ Optional description■ Optional picture■ Three key words
A user can browse and select an item from the database	<ul style="list-style-type: none">• Users could browse this database, find something they want, and simply click a button that tells the system that they want that item• The system then notifies the person with the item and gives the shipping information.• Key Features:

	<ul style="list-style-type: none"> ○ Shows a summary of latest listings when logging in to browse ○ Search functionality for keywords
The system should have limits in place	<ul style="list-style-type: none"> ● Someone could list and give away as many items as they wish. A user would only be allowed to request a certain number of items per month in total and not too many from the same person. ● Purchase of the postage would need to be verified. The donation receipt would only be sent after the recipient acknowledges that they got the item. ● If too many donations are reported incomplete, then the user might be blocked or reviewed by staff.
The system should provide a feedback link	<ul style="list-style-type: none"> ● There would have to be a feedback link available for reporting any suspected misuse of the site or other issues ● Communication between users would be necessary for questions and/or messages of thanks.
Provide a program to recall all successful transactions for tax documentation	<ul style="list-style-type: none"> ● We would need a program with the ability to recall all the successful deliveries at the end of the calendar year to summarize the tax write-offs for all users
System interoperability, scalability, financial requirement	<ul style="list-style-type: none"> ● Technological requirements shouldn't be steep. Many of these services are already bundled together for server rental for an "auction" website, which may serve as a starting point. ● This may be ideal since the website should be able to scale easily with popularity without wasted investment in equipment. ● The site must be cheap to operate since the funding might be unsteady.

User Information should be retained	<ul style="list-style-type: none"> • May be deleted on request from user • Should include shipping information • Should include number of items donated
Investigate Shipping Connections (this one the customer generated after reviewing the other user stories we pulled from his vision statement)	<ul style="list-style-type: none"> • Send shipping information to auto-fill the correct form. • Integration with USPS, UPS, or FedEx.

Story task and effort estimates

In terms of work unit estimates, we decided one work unit would be the equivalent of a pair working on the project for 5 hours. So for the first sprint, we let the customer know we expected we would be able to complete between 3 and 6 work units. These were rough estimates of a unit of time since we did not know what each group member's skills were in programming this system. Further, we did not know if what ideas we had in mind were complicated or simple to achieve. We presented the customer with the below table.

List of User Stories from Vision Statement		
#	User story title	Estimated work units (1 unit = 5 hours of pair work)
1	Upload an item for donation to the site	2.5
2	A user can browse and select an item from the database	3
3	The system should have limits in place	4
4	The system should provide a feedback link	3.5
5	Provide a program to recall all successful transactions for tax documentation	3.5
6	System interoperability, scalability, financial requirement	?Not Clear, is this a user story or Fit criteria?
7	User information should be retained	3
8	Investigate Shipping Connections	4

Priority List

On the 16th of May, we emailed our user stories with estimated time to completion, and within 24 hours, our customer, Jonathan, responded back with his input on which user stories to

implement first. The priorities are listed in the table below and were selected entirely by the customer with no influence from the group.

List of User Stories from Vision Statement		Client Preference, please prioritize with 1 being the highest priority and 8 being the lowest
#	User story title	
1	Upload an item for donation to the site	2
2	A user can browse and select an item from the database	1
3	The system should have limits in place	6
4	The system should provide a feedback link	5
5	Provide a program to recall all successful transactions for tax documentation	4
6	System interoperability, scalability, financial requirement	* (Fit Criteria)
7	User information should be retained	3
8	Investigate Shipping Connections	7

From the table given back by the customer above, the utmost priority was to get a page where the user of GoodBay can browse and select an item from a database. In our case, the database would be presented in the form of a table. users can browse and select from the entire selection of donated items. After an item is selected the donor receives a notification of this person's address to ship to.

The second most important task was making sure that users can upload items to the database to donate. This was followed by the third task, to make sure that user information is retained. To do so there needs to be a login system where people have usernames and passwords to keep their information tied to an account. The rest of the tasks in order of priority are as follows: Provide a program to recall all successful transactions for tax documentation, system provides a feedback link, system should have limits in place, investigate shipping connections. It was determined by the customer that the user story "system should have interoperability, scalability, financial requirement" was in fact a fit criteria and not a user story.

Summary of pair work

We used a google doc with a table of availability that all group members filled in so that we had a better idea when we had overlapping availability. We had to base our common time as UTC because we had members on PST, EST, and one group member in Saudi Arabia. We used Slack to communicate on best times for meeting up. Finally we used joinme to share our

screens and teleconference so that one person could code while the other person could be actively engaged, seeing the coder type code while they were doing it and both coder and co-pilot could talk over their respective headsets. We used Cloud9 as the development environment and the simultaneous editing allowed for coding pairs to easily swap duties from coding to co-piloting.

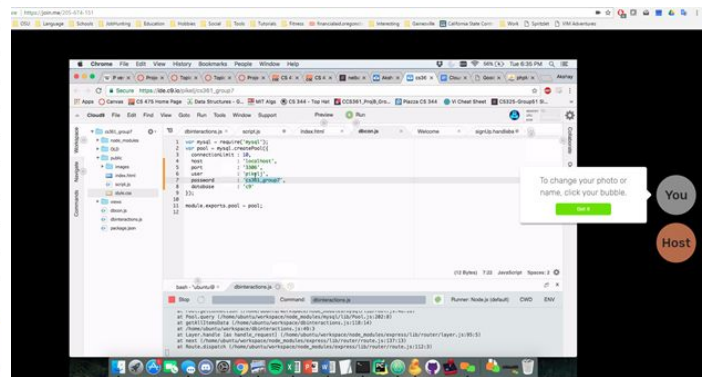
The issues we had with this type of communication were minor. On occasion a group member had some last minute time conflict, or technical difficulty like installing windows updates at the wrong time or having one's headphones run out of batteries. However, for the most part members were able to pair off organically and a lot of progress was made without needing to put too much effort on coordinating everyone's availability. While all members participated in group sessions, the distribution could have been better balanced and if this was a professional environment instead of a school project, we would also want to keep track of how many hours each member contributed to ensure all programmers were given a fair share of the load.

A snippet of our shared calendar below as well as a screenshot of a joinme session:

IN = Isaac
Jp = Johannes = Ideal times. If needed I can be available during 2100 - 0300 UTC on any given day before work.
Sb = Berry
AS = Akshay subramanian (available weekends as well but left blank for now)
Ix = Ivan (Im available all weekends but prefer mornings. I'm pacific time, so UTC is +7 hours)
JH = Jonathan Horton = client

The first UTC to other time calculator I found is [here](#), if you have a better one please update

Time UTC	M	T	W	R	F	Sa	Su
0000	AS	AS	AS	AS			
0100	AS	AS	AS	AS			
0200	AS, IN	AS, IN	AS, IN	AS, IN	IN	IN	IN
0300	AS, IN	AS, IN	AS, IN	AS, IN	IN	IN	IN
0400	AS	AS	AS	AS		AS	AS
0500	AS	AS	AS	AS	Jp		
0600	AS	AS	AS	AS	Jp	Jp	
0700					Jp	Jp	
0800					Jp	Jp	
0900					Jp	Jp	
1000					Jp	Jp	
1100					Jp	Jp	
1200					Jp	Jp	Ix
1300	JH	JH	JH	JH	Jp, JH	Jp, Ix	Ix
1400	JH	JH	JH	JH	Jp, JH	Jp, Ix	Ix
1500	sb, JH	sb, JH	sb, JH	sb, JH	Jp, sb, JH	Jp, Ix	Ix
1600	Jp, sb, AS, JH	Jp, sb, JH	Jp, sb, JH	Jp, sb, JH	Jp, sb, JH	Jp, Ix	Jp, Ix



Summary of unit tests

Unit tests were written before any coding was done in a collaborative google doc. Unit tests were written down for small functions or for key features that we were trying to code. Some of the unit tests we had are as follows:

For user story #2 (priority #1)

1) Unit test for isWanted component

Input	Verify
a checkmark in "wanted" button	item "object_wanted" variable set to "True"
click "wanted" button	system checks if item is still available and sets variable to True if available otherwise notifies the user item is no longer available
A User has selected a number of items as "wanted" and ends their browser session without checking out.	item(s) "object_wanted" variable set to False on exit of session

2) Unit test for placing an order'

Input	Verify
an order with "wanted" items is submitted	donor(s) are notified of customer's shipping address and asked to upload a shipping receipt to the order for a particular item customer is notified to mark receipt of item

3) Browsing and search component

Input	Verify
input known keywords for items in the db	items with known existing keywords are displayed on screen
input keywords that do not exist in db	db returns that no item exist with those keywords
loading the webpage for browsing items	the latest donated items should be shown on the screen in summary fashion

For user story #1 (priority #2)

1) Unit test for create listing

Input	Verify
create new listing with headline and three keywords only	new listing entered in system with headline and keywords only
create a new listing with optional description	new listing added to system with headline, keywords and description

create new listing with optional picture	new listing added to system with headline, keywords and picture
create new listing with both description and picture	new listing added to system with headline, keywords, picture and description
create a new listing	new listing should be associated with a particular donor

For user story #7 (priority #3)

Input	Verify
user selects to delete profile	requires username and password user is only disabled and items are removed from availability User information is retained for X years for tax purposes
User registers on website	address information is required and stored with the user profile
user views their uploaded items	items that user has uploaded are displayed
user views their donated items	items that have been received by recipient are shown and a total count displayed

Besides the above listed unit tests, after doing some pair coding, it was discovered that some additional testing should be performed at times due to unanticipated complexity and at times due to the project moving in a direction that wasn't envisioned. Some examples being, it would be good to remove the implementation from the dbinteractions.js app.post() of the actual functions that are sending back data to our page, so we can build them incrementally and a second example being the current sql statement that returns all rows from the table donated_items, should check the object_wanted flag, so if it's 0 then it is available if 1 it is wanted / part of another order for instance and does not get returned to the browsing page.

Summary of Acceptance Tests

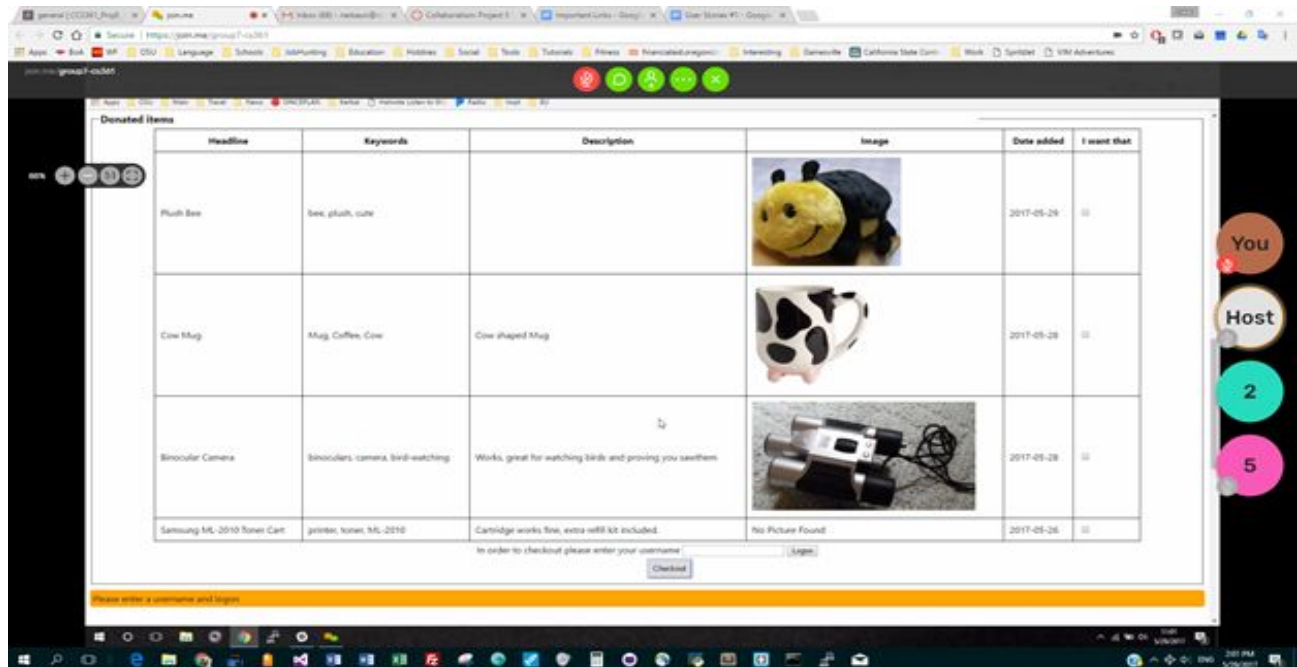
Among the first of the features to undergo acceptance testing was the main page table. This table comprised of all active items on the application that had been donated and were still available for user request. There were two main features of the main table that had to be tested. First, all of the required information to identify the item needed to be seen within the table. Second, all items were to be organized by posting the date to allow users to easily access new donations. The acceptance testing done ensured that all requirements for an item to be added to the application would be checked before adding the item to the page. For example, our

customer wanted each item to have a headline and three key words. We ensured that the add page item would refuse any additions to the database that lacked any of these essential items. We then tested whether the optional items would affect the table layout. During this testing we discovered that a large image file would make the table unreadable. To finish testing the main table, we removed an item from the database and then added it back to ensure that this re-inserted item would be at the top of the table in the main client application.

Further acceptance was done to ensure that the goals of the latter of our two highest priority user stories were met. This included functionality to allow other users to be able to request the items that were in the database. We first tested the button that allowed users to request an item. We ensured that checking the box would correctly remove the item from the main table without disrupting the remaining items on the table. Through our acceptance we realized that multiple users working on different cached versions of the site would be able to request items resulting in a single item being promised to multiple customers. To fix this, a checkout order button was implemented. This would allow the client to double check with the database if the item was still available to ensure than item was only requested by one user. To test this new functionality we had two developers operating on different computers try to request the same item as simultaneously as possible.

Our final acceptance testing for the first XP cycle involved getting in touch with our customer. On May 29th two team members and our customer met for some acceptance testing. At first the group introduced the customer to what we had been working on and provided a broad overview as well as some basic instructions on how the site should work. Afterwards the customer was allowed to walk through the site and provide feedback as he was trying to implement features or find bugs to the system. Overall the customer was pleased with the work done. However, through their acceptance testing he discovered that customers could donate items to themselves. While the customer had some thoughts on how to move forward, this was the only major bug discovered.

Below is a screenshot showing acceptance testing from our customer and some of our group members participating as observers



Second XP Cycle

Summary of changes (user stories, estimates, and priority list)

At the end of the first XP cycle none of our user stories were completely finished. As such, the customer did not have any changes suggested to the priority list. As the second XP cycle went on and some of the integral user stories were implemented, we realized that the work estimates for successive user stories did not always stay the same. User stories that were more related to the user functionalities already in place had reduced work estimates as opposed to stories that required implementing entirely new features. Although this was taken into account when the original work estimates were drafted, the degree to which this took place was unpredictable for each user story. For example, one of the lower priority user stories included keeping track of all of the transactions a user was involved in. After finishing the user story to allow for requesting items, we realized that one of the work tasks for keeping track of the transaction had already been finished. Conversely, the work units for a different (lower priority) task, providing feedback, had not changed at all. As the second XP cycle moved on we were able to identify areas where we could more easily implement additional functionality through changing work unit costs.

The customer's acceptance tests revealed additional refinements that we took into account when moving forward. One such suggestion was that a user should not be able to donate and receive their own item. Also, he thought it best to restrict requestors from requesting for more than 3 items at a time. He further suggested that requestors provide proof of items received after passing the 3-item limit before they could request more items. This would help prevent customers from abusing the Goddbay marketplace services. The customer also suggested that donations be put behind a username and password for safety reasons. He pointed out that if we didn't use username and passwords, one might find that a malicious person was donating illicit or illegal items under other people's usernames. Finally we also

discussed how long records would be maintained, although we weren't sure about the specific tax code, it was made clear by the customer that we should account for some set time period, after which records could be removed from the database. We also continued to do our own internal acceptance/regression tests, an example of this can be found at Appendix C.

Summary of pair work

Overall, pair work was smooth on this second cycle. There was not as much activity this second week as the bulk of the progress was done on the first sprint. At this point some of the more productive coders continued to make the highest impact on getting the website refinements put in place. We divided into pairs the same way we did for the first sprint. Users updated available time in a shared google doc. When there was overlapping time we reached out to each other in Slack to confirm appointments, and we met using joinme so that we could share screens and communicate live as one person coded and the other watched and actively communicated with the active coder.

The second cycle was easier generally. During the second cycle, there were also more places for the pairs to work and focus on. During the first cycle, a large part of the project was setting up the database, creating the node server and generally just getting all the parts of the system to work together in Cloud9. Once these core mechanics were done, it was easier for everyone to take on tasks associated with the user story that we were implementing. In addition, during the second cycle there were also many more opportunities for pairs to further develop user stories. In addition, by having completed some of the foundational user stories, we were able to access and branch out to develop the stories that added new features to the application.

Another difficulty that was diminished in the second cycle was peer reviewing code. In the first cycle, for the other members of the team, reviewing the code that a pair and implemented in the application took considerable time. This can be attributed to the fact that major changes were happening to the application: fundamental database tables were being updated, new server code functions were being created etc. As the second XP cycle began, however, peer reviewing the code became a quicker task. We understood what the server functions were doing. New user stories were generally building on existing code in a way that included additional functionality instead of a completely new features.

Reflection

Throughout this project using the XP Agile development process we certainly found that we could quickly get a something up and running for the customer to view. Being, that this process relies more heavily on coding and building incrementally, instead of spending time on documentation, we can see where there can be cost savings. Programming in pairs allowed us to bounce ideas off of each other and although it did take a bit longer in general to write the code, having two individuals looking at a section of code when there is some fault or other error helped to increase the speed at which we were able to debug a section of code. In addition, pairing a member who was more familiar with implementing a technical aspect of the project with a less experienced member allowed for the less experienced member to catch up to speed quickly.

We spent extra effort to stay on task as much as possible, staying within the unit tests that we wrote and the user stories that we were focusing on. We have a number of areas where we could definitely spend time to refactor the code base in order to reuse code in other areas of the site. Due to the fact that we were specifically focused on those particular user stories and unit tests, sometimes when moving on to another user story we found areas of the code that we could adapt to use as part of the next user story. Because we tested extensively as we wrote the code, we were more confident that the code would work appropriately from there on out.

Initially we did spend a bit of time putting together a basic ER diagram of the database tables we planned to create. By the end of our two cycles the tables had changed quite a bit, we even dropped an entire table from the schema and added a number of attributes to the other tables (Appendix A). All part of refactoring, but this could definitely be seen as a shortcoming of the XP, in that if the group does not communicate well, then as changes happened through the refactoring process other pair teams might be left unintentionally out of the loop and thus more time winds up being spent on reviewing code changes.

Overall, the rapid deployment of some kind of functionality is certainly enticing, such as being able to show the customer some part of their software system even after just the first cycle. However, in the long term because code is written to pass the unit tests and there is not a lot of time given to documenting defects, this could result in large coding projects becoming difficult to manage. In the future going into a project of this nature with a team, an adherence to a coding standard in regard to commenting, variable names, and levels of abstraction expected would help in the long term maintainability of a project.

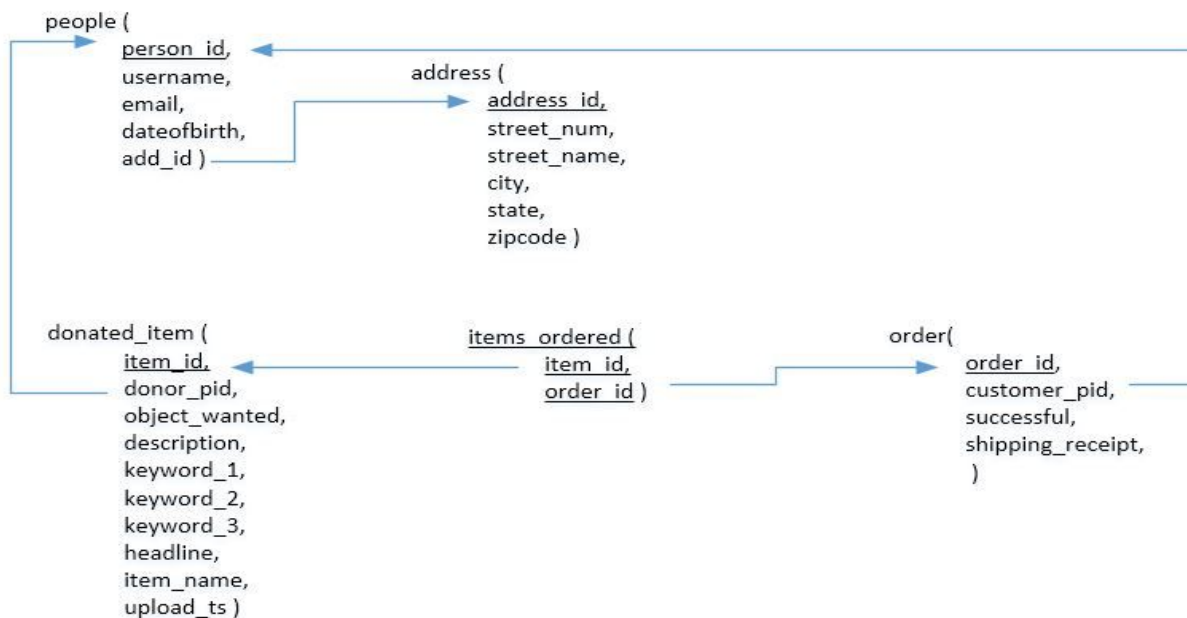
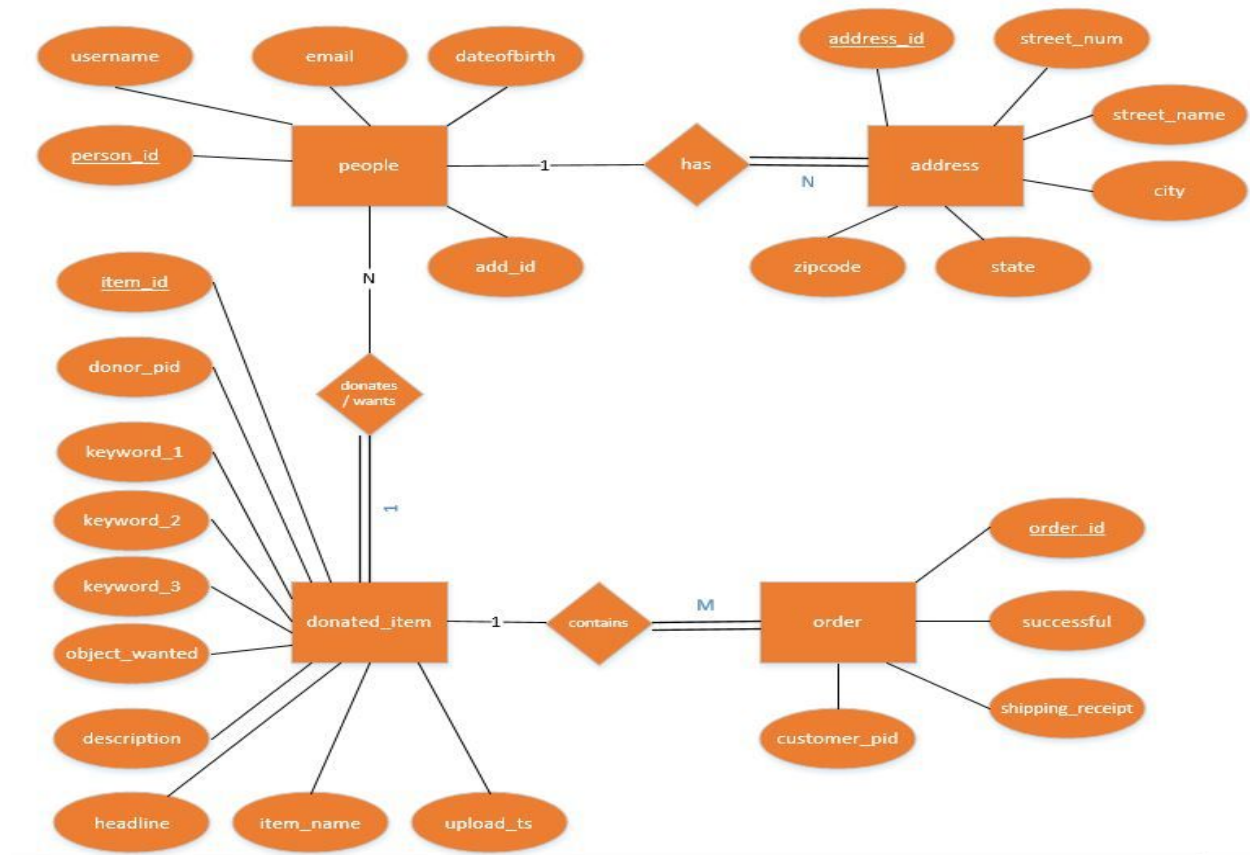
In comparison to Waterfall, XP feels a bit rushed. Using XP on a huge complex project may be extremely challenging, although it certainly has its benefits. In our case developing a web site XP seems to shine through. The speed of development produces something to look at immediately. Also as different unit tests are produced or new constraints are implemented, that part of the system can be refactored easily to accommodate them. Whereas, with Waterfall because of the static nature of the development process, if it is not part of the original design then it will be difficult to integrate a design change into the final product.

Given the restricted nature of Waterfall, it may not be the best fit for something like the project we had. Starting with an initial vision statement, functionality and expectations can all change fluidly through the process as the customer performs acceptance testing, perhaps there are new features that should be added that were not part of the initial vision. Allowing a customer the flexibility to change the final product on the fly is certainly a benefit to the XP process, because the project can be tailored to the customer's requirements.

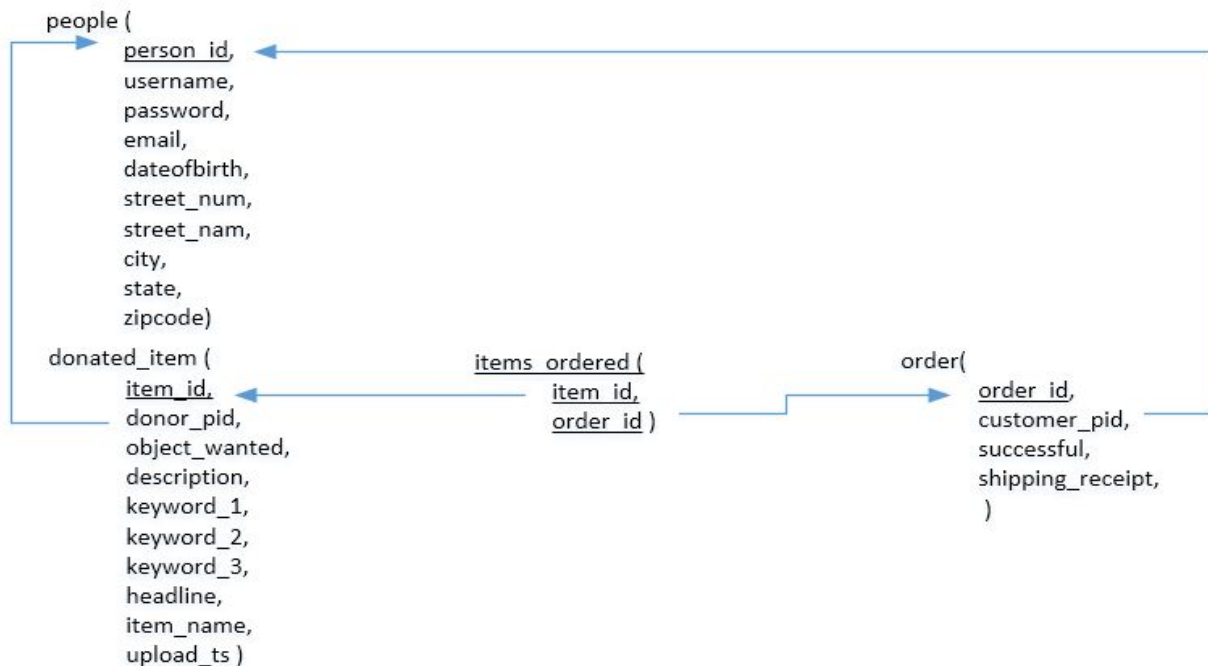
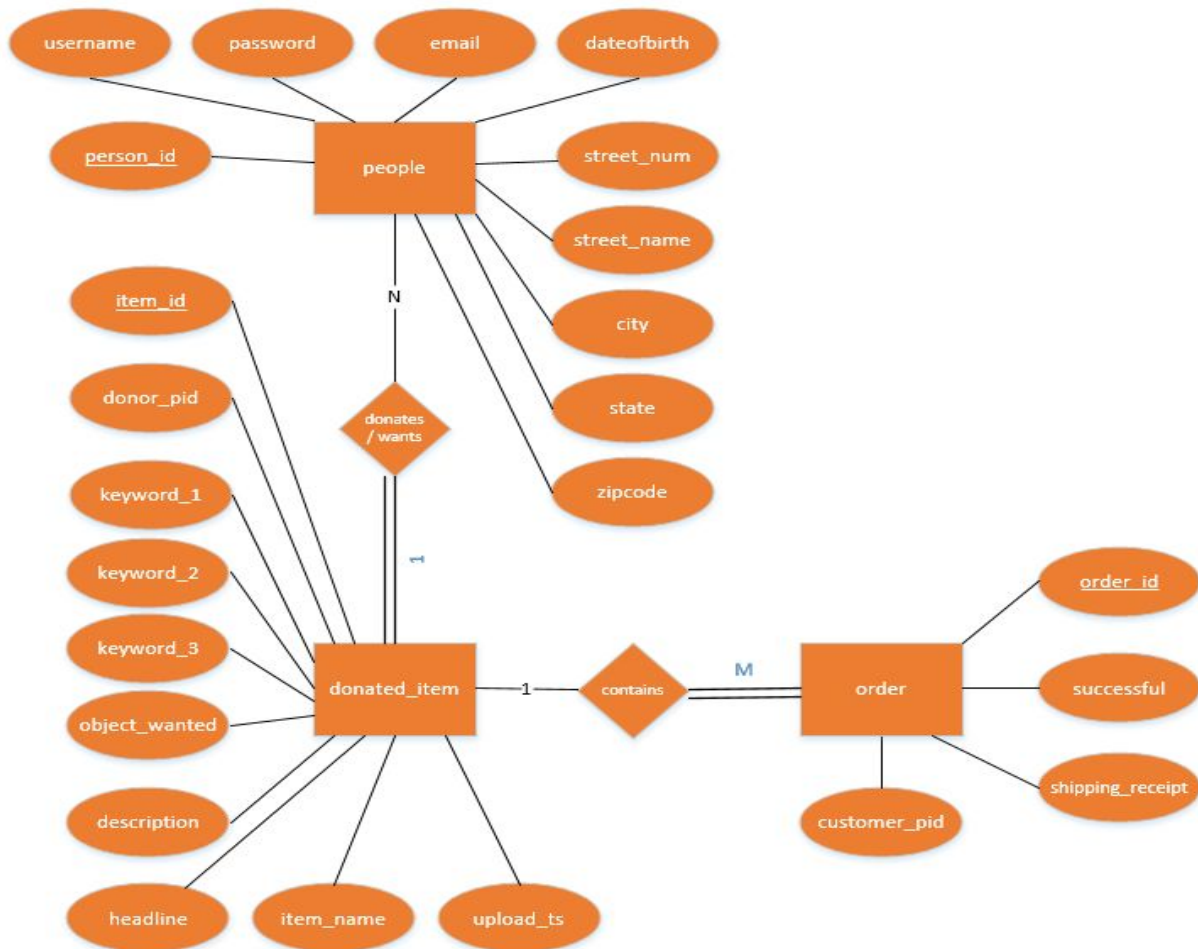
There are pros and cons to both types of development (Appendix B). It is important then to not impose one particular software development process on all projects. Instead the development process should be chosen based on the type of project that is being worked on. As stated frequently throughout the course, a navigation guidance system for a rocket might work using an Agile process, but can it be trusted to never fail? As opposed to a website, that goes through many different permutations from start to finish would not fit well with Waterfall because it restricts the creative process throughout the development cycle.

Appendix A - ER diagrams

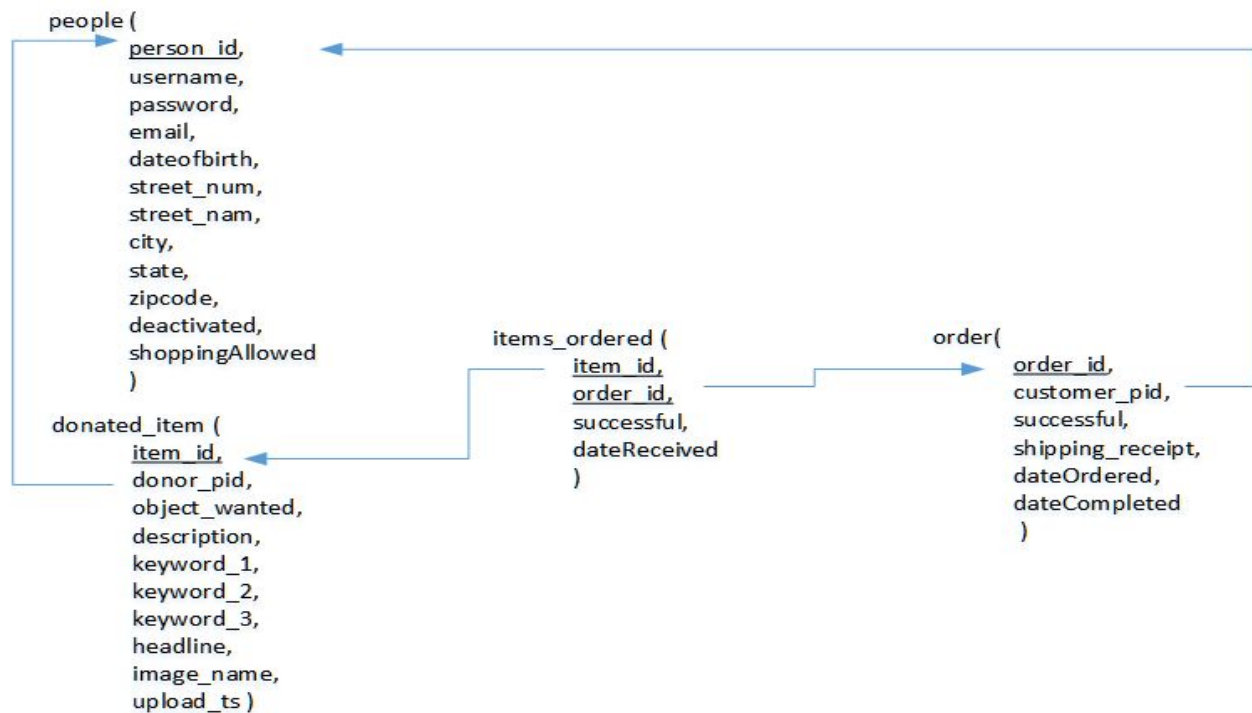
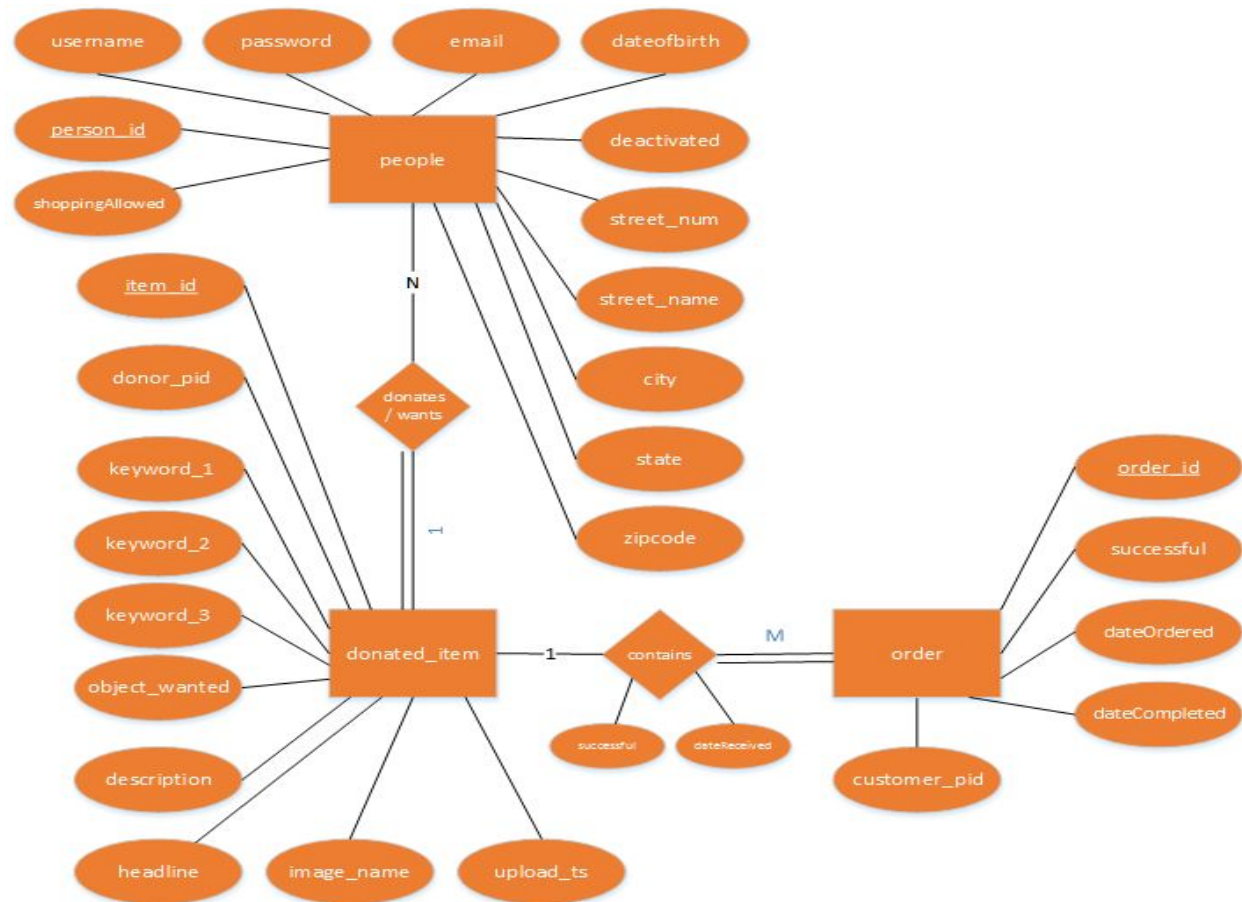
First iteration of ER diagrams used in the first cycle



Second iteration of ER diagrams used in the second cycle



Final iteration of ER diagrams at the end of the two cycles



Appendix B - Pros and cons of waterfall vs extreme agile

Pros of waterfall

1. **Clear deadlines:** Waterfall's static nature and predictable workflow make it easy to estimate costs, create timelines, and stick to deadlines.
2. **Disciplined by design:** Since each phase has a clear start point and a requirement review gate at the end of it, the team is forced to complete all tasks before the project as a whole can proceed.
3. **Well-documented:** Waterfall requires documentation and a clear paper trail for each phase of development. This makes it easier to follow the logic of past projects and lay the groundwork for future projects.
4. **Clear communication:** Predictable timelines and well-documented projects make it easy to give status updates to upper management, stakeholders, or clients with strict requirements.
5. **Easy learning curve:** As the traditional approach to project management across industries, teams usually don't require any prior knowledge or training in order to start working on a project with the Waterfall method.

Cons of waterfall

1. **Change can be costly:** The major downside to Waterfall's rigidity is the hampered ability to handle change. Testing occurs late in the project life cycle, and if you find out that your end users don't like the product you're building, it can be too late to pivot.
2. **Slow delivery times:** As many as four phases of development need to be completed before any coding begins—which means stakeholders and customers won't see a working product until late in the life cycle.
3. **Gathering requirements too early is risky:** Customers and stakeholders often don't know what they really want until they've had a chance to play with a working prototype. Since Waterfall handles all the requirement gathering upfront, there's a real risk of missing the mark and causing further headaches down the project line.
4. **Tendency to neglect testing:** Saving all the testing for the end of a project can be risky, because of the temptation to rush through it when there's a looming deadline. A poorly tested product can lead to a disastrous launch. You also lose out on valuable data you could have gained earlier in the project.

Pros of XP

1. **Cost Reduction:** Extreme Programming is code centric and trims some activities to reduce costs and developers focus on coding instead of needless paperwork and meetings. Feature based approach also ease the cost estimation as well.
2. **Better Risk Management** XP is reduces the risks related to programming. Normally programming depends a lot on individuals key team members. In XP, Using module

structure, and pair programming XP spreads the risk and mitigate the dependence on individuals

3. **Robustness:** In software simplicity always brings quality and contributes the robustness. By bringing simplicity, XP manages to create faster software with less defects. Test driven development at the coding stage and the customer UAT validation leads to successful development completion.
4. **Resilience:** Continuously changing requirements makes the software development harder but XP with its accommodation using user stories at the start and through feedback during the course of iterations.

Cons of XP

1. **Not Enough Against the Complexity:** Contemporary software is complex since the high requirements of the new world and XP's incremental approach doesn't respond well to this complexity just by itself.
2. **Detailed Planning:** XP requires a detailed planning from the start due to changing costs & scope.
3. **Weak Measurement:** XP doesn't measure/plan Quality Assurance of coding.
4. **Duplication Issues:** XP is practiced with pair programming which might usually lead to too much duplication of codes and data.
5. **Hardness:** To convince developers to accept this tough practice is not always easy. It requires more discipline in the team and devotion of your customers. Project management might experience difficulties related with the practice that changes during the life cycle.
6. **Code Centric:** XP is code centric rather than design centric development and this can be tiring in larger projects. Also XP requires too much refactoring and time consuming.

Also, It is very difficult to test the codes due to not structured and also defects are not well documented.

Appendix C - Example of acceptance/regression testing

#general

☆ | 5 | 0 | Company-wide announcements and work-based matters

June 3rd

pikelj 3:10 PM ☆

If everyone has the time and you all think this is necessary, it would be awesome if we could do some acceptance/regression testing. I am hoping to do some of this over the next day or so and I'll document any bugs found. some of these have already been tested, but regression testing was mentioned in the lectures so here's some of the things I've done and if you have a few minutes would be cool if you could run through this or any other you think of to break the site.

- register as a new user (passwords must be 8 or more characters, letters and number only and must match)
- try to register as a new user when logged in
- try to upload an item when not logged in
- log in, then upload an item
- try to review order with your own item
- try to review order more than 3 items
- try to review order with 3 items, 1 or 2 of which are your own items
- try to review order with items, then after checkout button is activated click another item (checkout should not be allowed)
- complete a successful order (if you used a gmail account or something other than osu email you should receive an email with order details)
- view my donated items/pending orders page
- select several items then close the tab, going back to the page the items should still be there
- try to deactivate an account without logging in first.
- save for last: upload a new item(s), browse to see your items, deactivate your account (only possible when logged in), those items should be gone and you should be logged out.
- after deactivating an account try to logon again.
- keep in mind if you click the "ADMIN" that over rides everything and items will appear again, we should probably remove that before submitting. (edited)

semexanb 3:30 PM

tried registering as login user: Please log out to register a new user

tried adding an item when not logged in: Please go back to the main page and logon first to upload an item.

logged in, uploaded an iphone for donation

try to review order with your own item:

Removed items that you donated you may now checkout.

try to review order more than 3 items: Too many items, no more than 3 per order

try to review order with 3 items, 1 or 2 of which are your own items: Too many items, no more than 3 per order

try to review order with items, then after checkout button is activated click another item :it doesn't let me checkout

completed a successful order n received an email

select several items then close the tab, going back to the page the items should still be there: they are there

try to deactivate an account without logging in first:Please logon first to deactivate the account

upload new item, checked my items, deactivated my account and the items are all gone

tried to re register with same info and say its already exists, but no way to activate account

June 4th

pikelj 12:03 AM

Thanks @semexanb you bring up an interesting point on your last item there. I suppose as a security measure we would not want to allow that without a direct request to an admin type, right? What if someone got a hold of a deactivated account and used to post illicit things.

Appendix D - Step by step instructions on how to run the project

Navigate to the webpage at : <http://104.36.18.164:8081/>

The webpage above is a permanent hosting solution. The contents of the mySQL databases are different when you view at the webpage above versus the development site.

We developed the project using a Cloud9 Workspace, we have invited Professor Rooker and TAs and the editing link is https://ide.c9.io/pikelj/cs361_group7 Once you request access we will approve the request.

To get the Cloud9 development site up and running you will need to run a couple of commands.

From the bash terminal at the bottom of the workspace run the following:

(if it does not appear click the green plus symbol and open a new terminal)

```
$ sudo service apache2 restart
```

```
$ mysql-ctl start
```

Then in the workspace tab (on the left of the screen) right click on dbinteractions.js and select “run”

At this point you should be good to navigate in a new browser tab to

<http://cs361-group7-pikelj.c9users.io:8081/> you may get a notification from Cloud9 about temporary use. Accept that and you will be passed on to the development site

If you would like to view the phpmyadmin to access the mySQL database used with Cloud9 navigate to <https://cs361-group7-pikelj.c9users.io:8080/phpmyadmin/>

User: pikelj

Pass: cs361_group7

The schema we are using is c9. Here you can view the tables

Website User SOP

To navigate through the GoodBay website (hosted [here](#)) you will need an active username and password. First click on Register, then fill in your information (use a gmail e-mail address). After registering you can now browse items and checkout up to three items at a time. After an item has been received you can mark as such and continue to checkout more items. At any time you can also post items to donate. If a user no longer wishes to be a member of GoodBay they can click on Deactivate User, enter in their log-in credentials and then any items that user had posted will no longer be available on GoodBay.

Starting on the next page are more detailed screenshots describing the navigation options mentioned here.

Below is the homepage

[GoodBay](#) [Register](#) [Deactivate User](#) [Add an item](#) [My Items](#)

Username:

Password:

[Logon](#) [Logout](#)

Welcome to GoodBay!

Notifications:

Most recently Donated Items

Optimus Prime
Samsung ML-2010 Toner Cart
Mickey Mouse
rubber ducky
Cow Mug
Binocular Camera
Neighbor's Cat
Firetruck


Search by keyword

Keyword:

[Search](#) [Reset](#)

Search Message:

Donated items

Headline	Keywords	Description	Image	Date added	I want that
rubber ducky	toy, bathtub, kids	bath toy		2017-06-04	<input type="checkbox"/>

[Review Order](#) [Checkout Order](#)

Message:

Click Register

GoodBay	Register	Deactivate User	Add an item	My Items	Username: <input type="text"/>
					Password: <input type="password"/>
					<input type="button" value="Logon"/> <input type="button" value="Logout"/>

Type in your registration information

[Back](#)

User Sign Up Page

User Details

Username*	<input type="text"/>
Password*	Minimum 7 characters, only letters and numbers.
Confirm Password*	Must match Password
Email*	<input type="text"/>
Date of Birth	mm/dd/yyyy

Address Details

Street Number*	<input type="text"/>
Street Name*	<input type="text"/>
City*	<input type="text"/>
State*	<input type="text"/>
Zip Code*	<input type="text"/>

Please note: unfortunately oregonstate.edu email addresses are not working with the current SMTP server.
A gmail account works well.

After successfully creating an account you will see the below screen

[Back](#)

Welcome to Goodbay, neibaur!

You will need to logon before you can checkout.

Username:

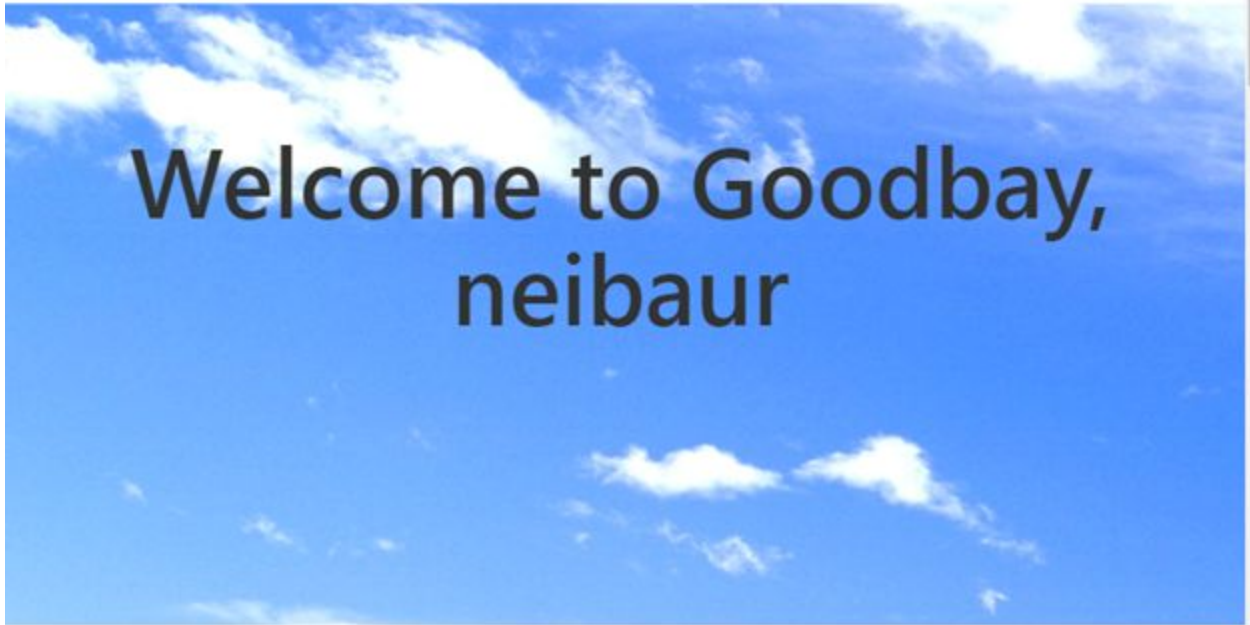
Password:

Click Back and type in your new login information
Below is after successfully logging in

[GoodBay](#) [Register](#) [Deactivate User](#) [Add an item](#) [My Items](#)

Username:

Password:



Welcome to Goodbay, neibaur

Notifications: *Logon successful. Welcome, neibaur your id is 9*

Most recently Donated Items

Note, now you can check out items, however if you select more than three and click review order you will see a message saying “Too many items, no more than 3 per order”

Festivus Sweater	Clothes, Sweater, Warm	Red sweater		2017-06-03	<input checked="" type="checkbox"/>
iphone	mobile, ios, iphone	iphone se		2017-06-03	<input checked="" type="checkbox"/>
fidget spinner	fidget, toy, fun			2017-06-03	<input checked="" type="checkbox"/>
Plush Bee	bee, plush, cute			2017-03-07	<input checked="" type="checkbox"/>

Review Order

Checkout Order

Message: Too many items, no more than 3 per order

Note below the message is different as this time only two items are selected

Review Order

Checkout Order

Message: Item reserved. Please review your order to checkout.

With three items or less selected after clicking Review Order the Checkout Order button will be active, click it to check out your selected items.

Review Order

Checkout Order

Message: You may now checkout.

At the top of the screen click on My Items, you will go to your User Profile page

Back

Welcome to your User Profile, neibaur

Donated Items Pending Ordered Items Items Received Items to be Shipped

Donated items

Here clicking on Pending Ordered Items will show you the items you have checked out

Back

Welcome to your User Profile, neibaur

Donated Items Pending Ordered Items Items Received Items to be Shipped

Pending Orders (Items not received)

Order: 21	Order Date: Fri Jun 09 2017 15:08:51 GMT+0000 (GMT)				
Headline	Keywords	Description	Image	Date added	Item received
Plush Bee	bee, plush, cute			March 07, 2017	<div>Received</div>
Order: 21	Order Date: Fri Jun 09 2017 15:08:51 GMT+0000 (GMT)				
Headline	Keywords	Description	Image	Date added	Item received
fidget spinner	fidget, toy, fun			June 03, 2017	<div>Received</div>

You can click the Received button to indicate the item has shipped successfully, after which you can click on the Items Received to review what you have successfully ordered.

[Back](#)

Welcome to your User Profile, neibaur

Item marked received and moved to Items Received

[Donated Items](#)

[Pending Ordered Items](#)

[Items Received](#)

[Items to be Shipped](#)

Completed Orders

Order: 21	Order Date: Fri Jun 09 2017 15:08:51 GMT+0000 (GMT)			
Headline	Keywords	Description	Image	Date Received
Plush Bee	bee, plush, cute			June 09, 2017

If you click Back to get to the homepage you can click “Add an item”, here you will be asked for more details to fill, in this example I’m donating the Plush Bee that we just received

[Back](#)

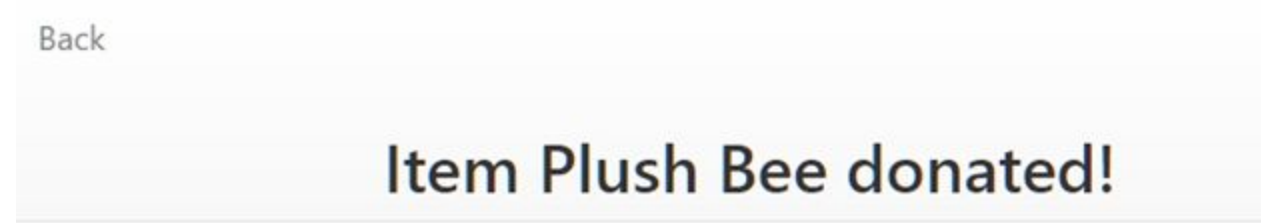
Welcome neibaur, please upload your item.

Item Headline*	Plush Bee
Description	
Keyword 1*	bee
Keyword 2*	plush
Keyword 3*	cute
Picture of your item	Choose File BeeStuffy.jpg

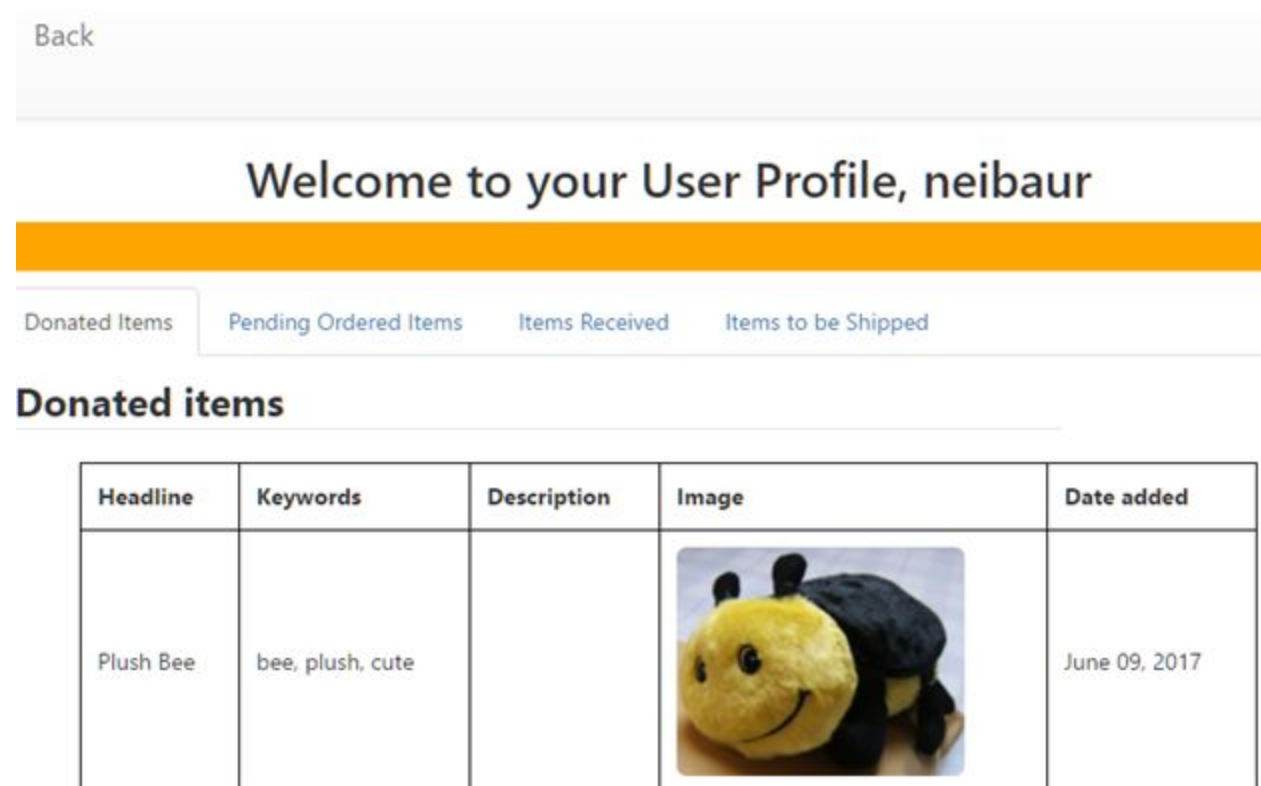
JPEG or PNG, 1MB Max Size

[Upload!](#)

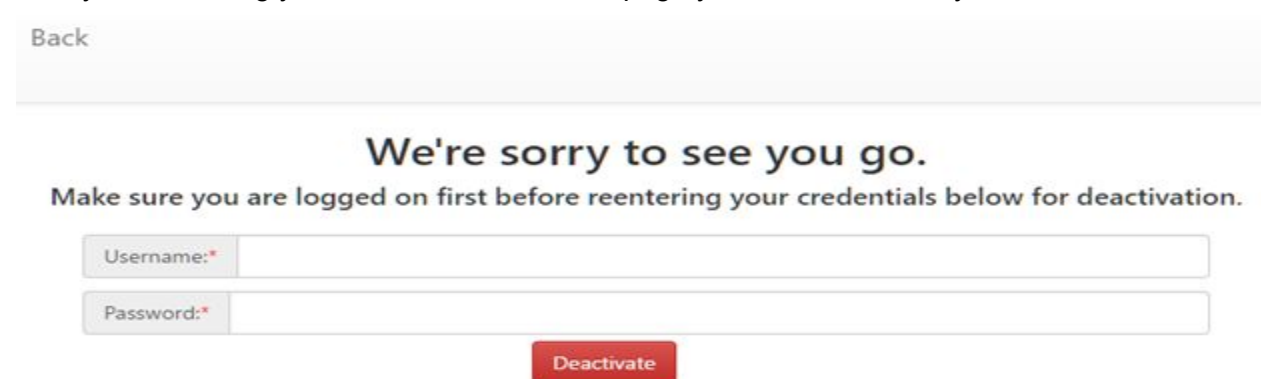
If successful you will see the following screen



Back at your User Profile you will also see the item under Donated Items



Finally the last thing you can do is on the homepage you can Deactivate your account



If successful you will see the below screen and any items you were actively donating will no longer be available for other users to select from.

[Back](#)

User neibaur was deactivated, and all items removed from available.