Michael Rodrigues P9 – Final Report IS4300 - Fall 2013

[Problem]:

At the start of the semester, I was unaware of a great way to manage a list of assignments for all of my courses. In previous semesters, I tried using an agenda, but it lacked many nice features such as easily finding the due date of a previous assignment. This would become difficult if you did not know when it was assigned.

Another method that I tried was sending myself an email where the subject was the course and the message was the description and due date of the assignment. This attempt only lasted for a few days because of the fact that it was impossible to mark an assignment as completed without sending another message. After sending that new message, I would also have to mark the email as read so that I wouldn't think that it was not yet completed. This approach lacked many of the basic features for task management, but did allow for filtering if your email client allowed for searching. Unfortunately, the advanced features are used far less frequently than the basic features so I decided against this approach as well.

I then proceeded to try a third method. This approach was using a text file to denote which courses had assignments. I denoted my assignments in the following manner:

IS4300 – Read Chapter 14

This approach worked fairly well, but when I wanted to search for a previous assignment, it failed. The way that I was managing the text file involved deleting the lines that represented the assignments that I had completed. This was due to the fact that if I left them in the file, it became very hard to distinguish what was completed and what was not.

After completing my first Co-op, I decided to develop a basic website that would allow me to manage my tasks. I wanted this approach to be able to display previous tasks that had already been completed, allow for easy searching and filtering of tasks by given criteria, and be extremely easy to use. During the month that I had between when I stopped working and prior to starting classes, I spent a reasonable amount of time writing HTML, CSS, JavaScript, and PHP scripts that would interface with a text file to allow its user to perform the following tasks:

- View Tasks
  - Filter Tasks (based on given criteria)
- Add, Update, Deleting, and Completing Tasks
- Prioritize Tasks

Prior to the start of this course, my intentions were to only support a single user since this was a personal project. Once I started this term project, I decided to change the scope to support multiple users concurrently. This will be discussed later on in this report.

At the onset of this project, my primary stakeholders were students who would be using this system to manage their tasks. As the project progressed, the concepts that were being developed applied to other potential users such as employees and professors and therefore they could also be representative primary stakeholders in this project. The primary stakeholders are the users who will enter data into the system in the form of tasks (or assignments), and will receive the output from the system. The output includes the task list, usage statistics and graphs, etc. Since these users will be entering data and receiving the output, they are also the secondary stakeholders of the system. The tertiary stakeholders include multiple different potential groups. For example, if multiple people start using the system, then I will have to upgrade the hosting plan to maintain a consistent interaction speed as the number of users increases. The primary stakeholders could also be affected by the success of this project because they may experience more lag due to the increased number of users. The facilitating stakeholders include myself, who developed this website, and the representative users that were asked for opinions of various ideas.

[Design]:

The final design of my interface looks very different from the initial design that I had when I first proposed this project idea. First of all, the home page is now a login screen instead of bringing the user directly to their task list. This login screen contains a toolbar at the top of the page which all of the pages for the interface contain. This was a design choice that I decided to test during paper prototyping because it was initially going to be used to display the interface functionality. After testing it though, users were not a fan of a pop-up window per feature and therefore the interface was converted to use tabs as my initial design used prior to the course. Regardless of that, I decided to keep this toolbar so that on every page for this project, the "Task Tracker" name would appear in the top left area of the screen. As we learned in class, this is the most common spot for users to look to identify what site they are on if they become lost.

This toolbar becomes more useful in the Task view screen, because it allowed me to put the Help and Logout links in a clearly visible position. This was done after one the heuristic evaluators noted that the logout functionality was hidden and therefore the exit was not clearly marked. This task bar was also used to add an exit for the Stats screen since that was implemented as a pop-up window due to evaluators accidentally closing the entire interface because they did not realize that the link they clicked on loaded in the same page.

The default tab supported the "Add" functionality and was chosen so because it was the primary task that users were expected to perform during interactions. The "Filter" tab which allowed users to query their tasks for a specific category as well as viewing completed and/or uncompleted assignments was expected to be a less frequently used function since all user tasks are displayed. Additional tabs were added to allow for the usage statistics to be displayed and to support functionality specific to a user's account. I expected that most users would not view their usage statistics very frequently, so I placed that tab farther from the left. The user account settings were placed farthest right, because that is generally where most websites place this particular functionality and therefore this was done primarily to maintain external consistency. During the paper prototyping exercise, this was also placed farthest right in the toolbar for external consistency purposes. The reasoning behind why I chose tabs, was that it allowed for a single, specific functionality to be displayed at one time while hiding the other unnecessary features. This simplified the interface for novice users and was something that the subsequent evaluators of my interface appear to have liked.

In each tab, I always used a drop-down selection if possible to prevent users from having to recall data. I also used a Datepicker to prevent users having to recall what the calendar looks like and enter the date from memory. By providing this ability, users could just select the date from the calendar which ensured that the dates were always in a consistent format, which prevented malformed dates and potentially errors. This proved to be very useful during the paper prototyping, heuristic evaluations, and during the final usability testing. In all cases, at least one user mentioned how these made the interface simpler to use.

The task table was chosen over using a calendar, because a calendar would require a large amount of screen space and would not allot enough space by itself to display all of a user's tasks. I was afraid that may cause some users to forget about a task that they needed to complete which was one of the items that this interface was designed to avoid as the example scenarios demonstrated. The layout of the table changed from my initial implementation in two ways. First, the table headings were changed to be more in-line with the user's language. For example, "Date Due" became "Due Date", "Date Assigned" was changed to "Date Created", and the "Course" heading was changed to to "Category" since my primary stakeholders could include people other than students. Secondly, the order of the columns was changed. I moved the category column out to be the left-most column since users are likely more apt to scan their tasks for a certain category first. Then, the dates appear in the subsequent columns so in the case where a user does not use the ability to filter, they can easily scan the tasks themselves. I added an additional icon to the right-most column which was named "Commands" since it was to be used for updating and deletion of tasks. I then decided that in order to allow users to prioritize their tasks to make the most efficient use of their time, I added a new column on the left side of the table which simply had an icon which externally denotes the ability to drag-and-drop an item. This icon in this new column was used to denote the ability to drag-and-drop your tasks in a new order of your choosing. Although I did not explicitly have users perform a task to test this icon's visibility during this last evaluation, I did ask users during the follow-up if they had noticed any sorting functionality while they interacted with the interface, and two of the three users did notice the icon. Only one of the two users scrolled over the icon and realized based on the tooltip that the icon provided that feature.

I also chose the colors to be very consistent with other popular websites that follow the latest trends. I performed a checkmycolours.com test on these choices as well, and it passed with a rating of ~91% which ranks far better than Google (51% to 79%) and Facebook's (57.5%) results. The layout is also a fairly common approach where the primary information is displayed in a larger section primarily in the center of the screen while additional functions or navigation items are displayed at the top and left side of the page.

[Implementation]:

As I previously mentioned, the initial prototype that was developed prior to the start of this semester, only supported a single user. In order for this system to truly be useful though, it needed to support multiple users concurrently. I decided to purchase a hosting plan from DigitalOcean.com to host the server so that it would be available 24/7. Essentially, DigitalOcean provided me with a Virtual Machine running a Linux operating system of my choice. Since I use Ubuntu 12.04LTS on my laptop daily and have a decent amount of experience with it. I also decided to use a MongoDB database to manage user data. I chose a NoSQL database because I had experience with it during my previous Co-op where I needed to track data in a similar manner. I found MongoDB to be extremely capable and easy to use, especially with PHP. I am running Apache's httpd web server to host this project. Again, I chose this over other options because I had previous experience with it and it is fairly easy to setup and use.

At the implementation level, I have three different databases of user information. The first database contains user account information. The signup form on the home page submits the user entered information via HTTP's POST protocol to a PHP script which interfaces with the database. It stores the user's name, username, and password encrypted via PHP's crypt function. The login form also submits it data via POST to a PHP script which then queries the database for the specified username. If it finds a match, then it retrieves the encrypted password, encrypts the user entered password, and compares the two. I wanted to ensure that this implementation was secure, so I went online and researched common MongoDB attacks, such as injection and array attacks. I then proceeded to test the examples that I found to ensure that the system could not be easily broken. As far as I know, the system is not vulnerable to these attacks, because I could not break it even though I know how it works.

Once a user is logged in, I use sessions as provided by PHP to keep track of which user the system is performing tasks for. This prevents users from specifying another username and viewing tasks that they do not have permission to see. By doing this, I tried to eliminate any malicious use of the system to modify another user's task list. I decided to use sessions over cookies or specifying the username in the URL due to how they can be easily modified on the user's computer and allow for malicious attacks.

Another database that I have setup manages the user's categories. My system allows users to manage their categories, which supports adding and deleting. This ability is handled via a PHP script on the server that interfaces with this particular database in MongoDB. This database is also accessed for the drop-down selections in the Add and Filter forms.

The third and last database that I used was the one that I originally set up. This last database contains the tasks that users enter into the website. Again, I used a PHP script to interface with the site's Add form. The data was passed to the server and the script managed the interaction with the database. I decided to separate the different information into separate databases to avoid documents with a similar key appearing incorrectly in a query for different information. For example, if the categories and tasks were in the same database and I queried by username to retrieve a list of tasks, I would retrieve the categories as well which would cause errors with the interface and make it unusable.

For the website itself, I tried to use HTML5 compatible tags and CSS3 compatible style attributes. I also used various JavaScript libraries such as jQuery and Bootstrap to provide the features such as tabs and the toolbar respectively. I also used JavaScript libraries from amcharts.com which provided the pie chart functionality. This library required that the data displayed be stored in a JavaScript variable, which would have required a great deal of work to code since I would need PHP scripts to retrieve the data and then the JavaScript would need to parse it and format it properly. Instead, I decided to write a PHP script that generates the necessary JavaScript code and therefore the necessary variables for the chart. These libraries all provide functionality as well as the look-and-feel that most popular websites currently use. In order to stay externally consistent, I decided to use these instead of writing the features myself which would have taken far more time.

[Evaluation]:

The usability testing via paper prototyping, heuristic evaluations, and prototyping the actual interface allowed me to learn a number of things. First, I learned how important usability testing it. Looking at my interface, I thought it was a great design, but by testing it with users, it definitely made me realize the saying "You are not your user" is completely true. There were multiple usability issues that appeared in the heuristic evaluations that were not found in the paper prototyping exercise because of the new level of fidelity that was added. For example, color choices were not tested via paper prototyping because it was meant to be a gray-scale, low fidelity mockup which would test the functionality and not necessarily the entirety of the aesthetic appeal. It is also hard to test jQuery dialogs versus a JavaScript alert window due to the limited nature of the paper mockup.

Another thing that I learned is that finding usability bugs later on in the process does cost more. In particular, after the heuristic evaluations found a variety of different items I started working on fixing those right away. It took me a substantial amount of time to fix many of those items due to the fact that some required researching libraries and reading the documentation as to how to add new, necessary items. For example, one of my heuristic evaluators noticed that there was no constraint on what the user entered for a date. Since I was not sure how I would go about verifying that a date is valid due to the differing number of days per month, as well as the possibility of leap years, I looked online for a library that would provide that functionality. This particular usability issue took me almost as long as it took me to write the entire help documentation. In the process of fixing this issue, I needed to identify where in my JavaScript I should be checking for incorrect data which involved reading through all of the code. Once I wrote the function, scanned the code, and added the constraints where necessary, I also needed to go through the interface and test that the interface still provided the same functionality without any new errors appearing in my JavaScript or PHP code. This process did uncover a few issues that required a substantial amount of time to debug.

By investing as much time as I did in fixing the items that had been discovered by the evaluations, it luckily helped my interface to fare better through actually user testing during Project Assignment #8. During that particular usability testing phase, the only major item that was discovered was that users had difficulty with the small icons. The small icons caused users to take more time to precisely place the mouse. This was something that was mentioned during the follow-up as well as in the questionnaire that the users were asked to complete.

Unfortunately, there was one usability issue that I was unable to tackle during this semester. This was the suggestion to use jQuery dialogs instead of a JavaScript *alert* or *confirm* window. After researching this online, it appeared that jQuery dialogs did not natively support the *confirm* functionality which would require developing it myself. Since this would be fairly involved and a large amount of time, I felt that it was outside the scope of the project for this semester, but may subsequently be added in. All other usability issues were fixed briefly after they were found though.

[Reflection]:

This course has been extremely helpful. Over the course of the semester, I learned many new strategies and methods to test interfaces that I develop. I also found that the topics of this course were great points of discussion during Co-op interviews at Microsoft, MITRE Corporation, and Hewlett-Packard: Autonomy. It was interesting to hear the point of view of Software Developers at corporations that develop large software applications with interfaces that users end up working with.

The iterative design process confirmed to me that creating an application is not something that you do in one or two sittings and just deploy once it is "complete." During my last Co-op, that was seen as the norm which seemed strange to me. I'm not sure if that was due to the fact that the websites that were developed were only internally in scope and the primary users would be my coworkers and myself.

At each stage of the iterative design process, I felt that I learned something new. First, during requirements analysis, I realized that by documenting the scope of the project and the end users it helped me to structure my thoughts about representative tasks and based on those tasks, potential interface layouts. This is extremely helpful when starting a project since there seem to be infinitely many design paths that you could follow. By structuring the project and its scope, the more favorable design choices become more apparent which leads to a better starting prototype. As we learned in class, usability testing will only help improve upon the initial design layout that was chosen and therefore it is important that the initial design is a good one.

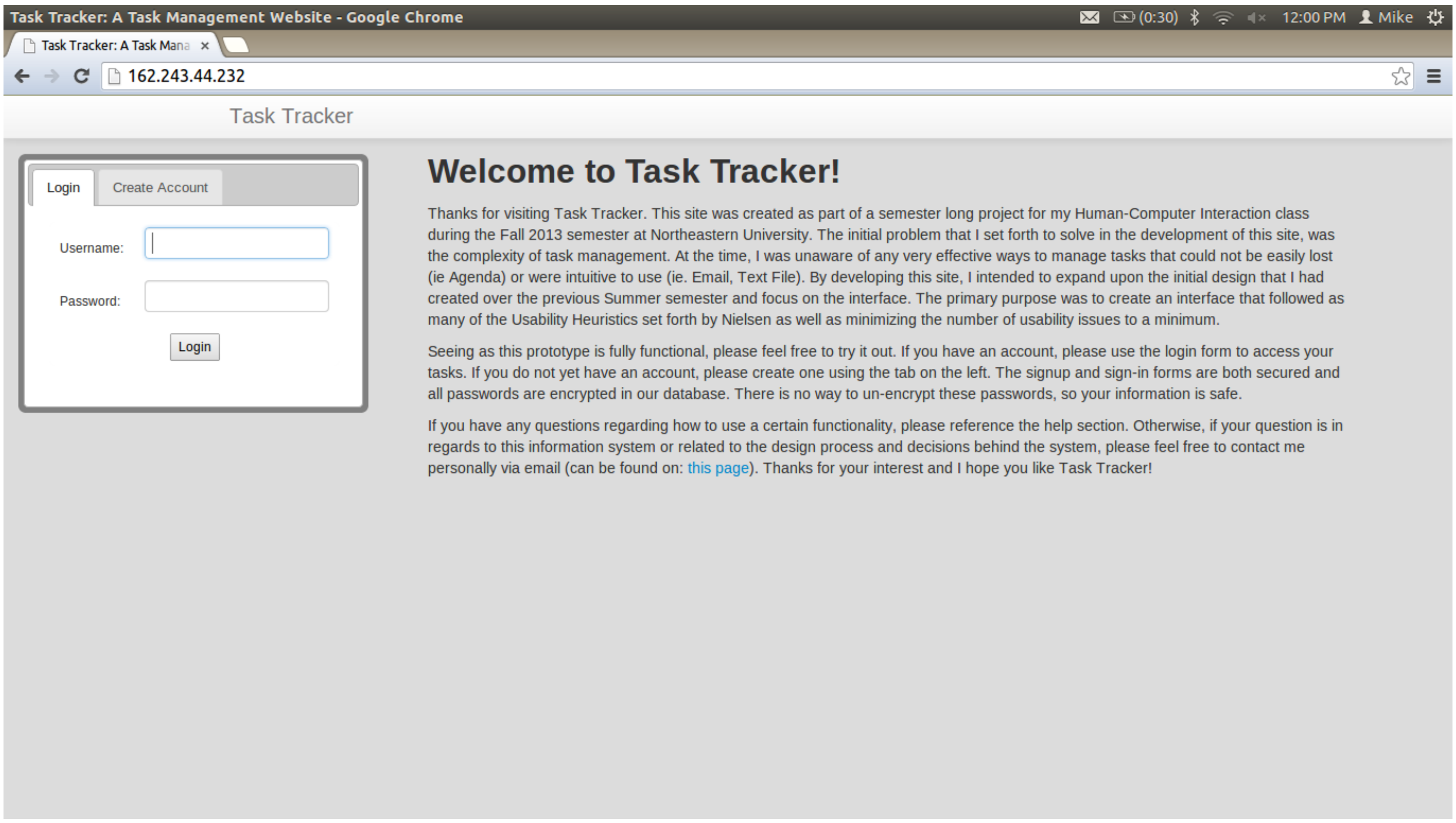
By breaking the supported interface tasks down via hierarchical task analysis, it helped me to determine exactly what process users currently used to perform the same actions on another interface. I then used this to try and simplify the process so that it would require less effort on the users part and therefore allowing for simple interactions. This process then helped me with the initial design sketches. By deciding how I could simplify a user's task, I needed to think about their interaction with the graphical interface. I was able to then conceptually sketch out the forms and create a story summarizing the interaction. Thinking through the users interaction directly like this, puts the designer in a representative user's mind and requires a lot of thinking to make the interface truly usable.

This mindset allowed me to build a decent model for my paper prototype to test with users. I thoroughly enjoyed the paper prototyping exercise because it was not like anything I had ever done before. I learned how important this, rather basic, form of testing is especially since it unveiled to me that I forgot to add a “Login” button to the homepage. Something as simple as that may have been overlooked if I had not tested the interface with users.

High fidelity prototyping was also very intriguing. I had never built an application before so that it would support just a few very limited functions in order to test the interaction rather than the specific features that the interface should provide. By doing so though, I learned that the functionality of the system is not as important as the ability of its users to actually be able to use the system. If the interface is unusable, then the features that were supported are of no use to anyone and the project was nothing but a waste.

If I had the opportunity to complete this semester's course again, I would have added more representative tasks. Although I tested the core features, I wish that the additional advanced functions would have been tested as well. For example, I was only able to add the ability to prioritize tasks rather late in the semester and therefore the only testing that occur with that functionality supported was the final usability test. Unfortunately, in order to evaluate my usability requirements, I decided to keep the representative tasks the same as I had previously. Therefore, the only testing of this advanced feature was that of the usability test users exploring the interface after the test session. Although I would not change the tasks that were tested, I would add another round of usability testing on this fully functional interface in order to explore the usability issues with the advanced features. As much as I feel the paper prototyping found useful issues, I believe that it is very limited in its overall scope and therefore the usability issues discovered represented that. Overall, I learned a lot of very useful information in this course and would like to thank you for a great semester.

[Screenshots]:



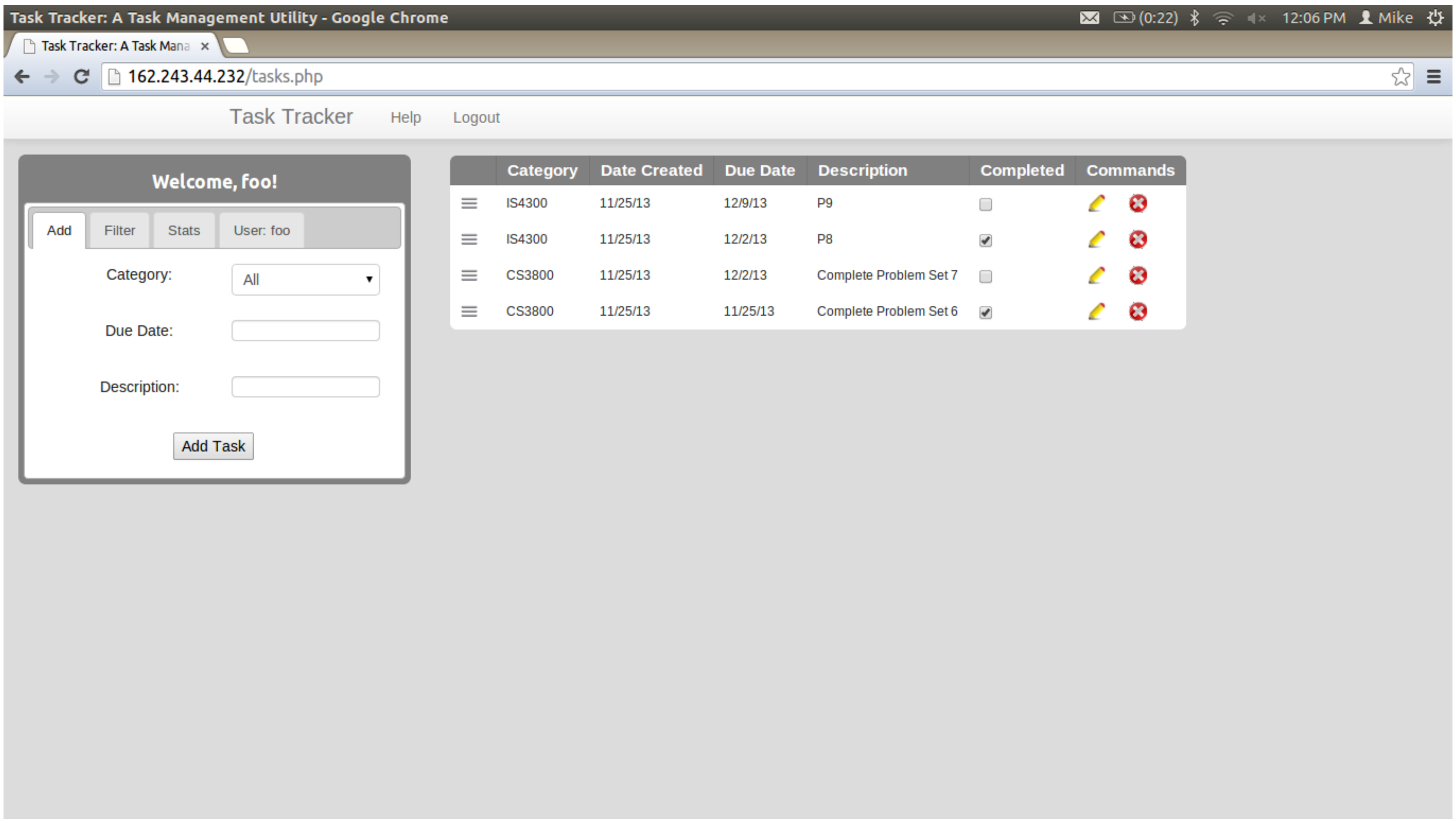
Welcome to Task Tracker!

Thanks for visiting Task Tracker. This site was created as part of a semester long project for my Human-Computer Interaction class during the Fall 2013 semester at Northeastern University. The initial problem that I set forth to solve in the development of this site, was the complexity of task management. At the time, I was unaware of any very effective ways to manage tasks that could not be easily lost (ie Agenda) or were intuitive to use (ie. Email, Text File). By developing this site, I intended to expand upon the initial design that I had created over the previous Summer semester and focus on the interface. The primary purpose was to create an interface that followed as many of the Usability Heuristics set forth by Nielsen as well as minimizing the number of usability issues to a minimum.

Seeing as this prototype is fully functional, please feel free to try it out. If you have an account, please use the login form to access your tasks. If you do not yet have an account, please create one using the tab on the left. The signup and sign-in forms are both secured and all passwords are encrypted in our database. There is no way to un-encrypt these passwords, so your information is safe.

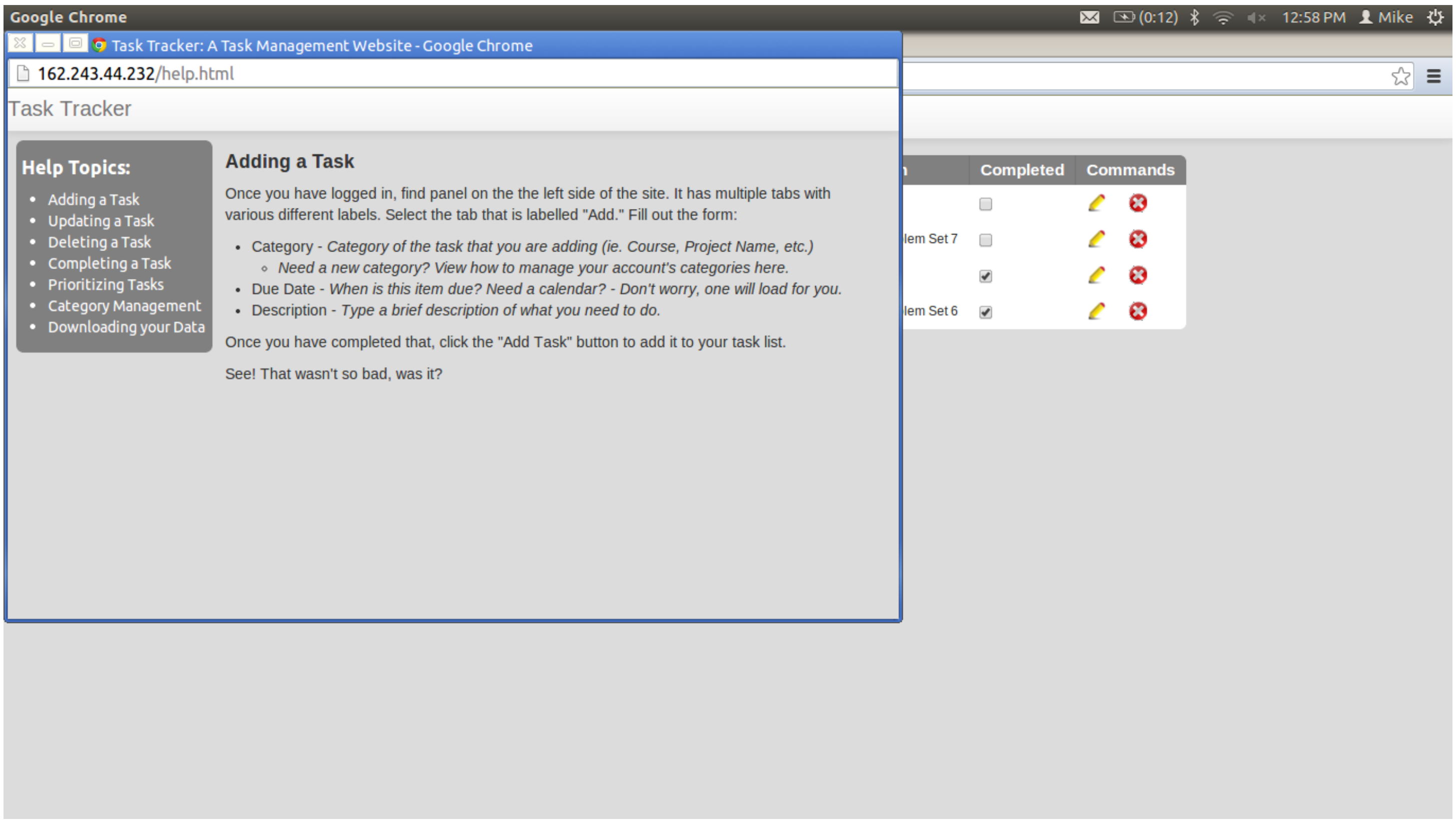
If you have any questions regarding how to use a certain functionality, please reference the help section. Otherwise, if your question is in regards to this information system or related to the design process and decisions behind the system, please feel free to contact me personally via email (can be found on: this page). Thanks for your interest and I hope you like Task Tracker!

Screenshot 1: Login Screen



Category	Date Created	Due Date	Description	Completed	Commands
IS4300	11/25/13	12/9/13	P9	☐	
IS4300	11/25/13	12/2/13	P8	✓	
CS3800	11/25/13	12/2/13	Complete Problem Set 7	☐	
CS3800	11/25/13	11/25/13	Complete Problem Set 6	✓	

Screenshot 2: Task Screen



Help Topics:

- Adding a Task
- Updating a Task
- Deleting a Task
- Completing a Task
- Prioritizing Tasks
- Category Management
- Downloading your Data

Adding a Task

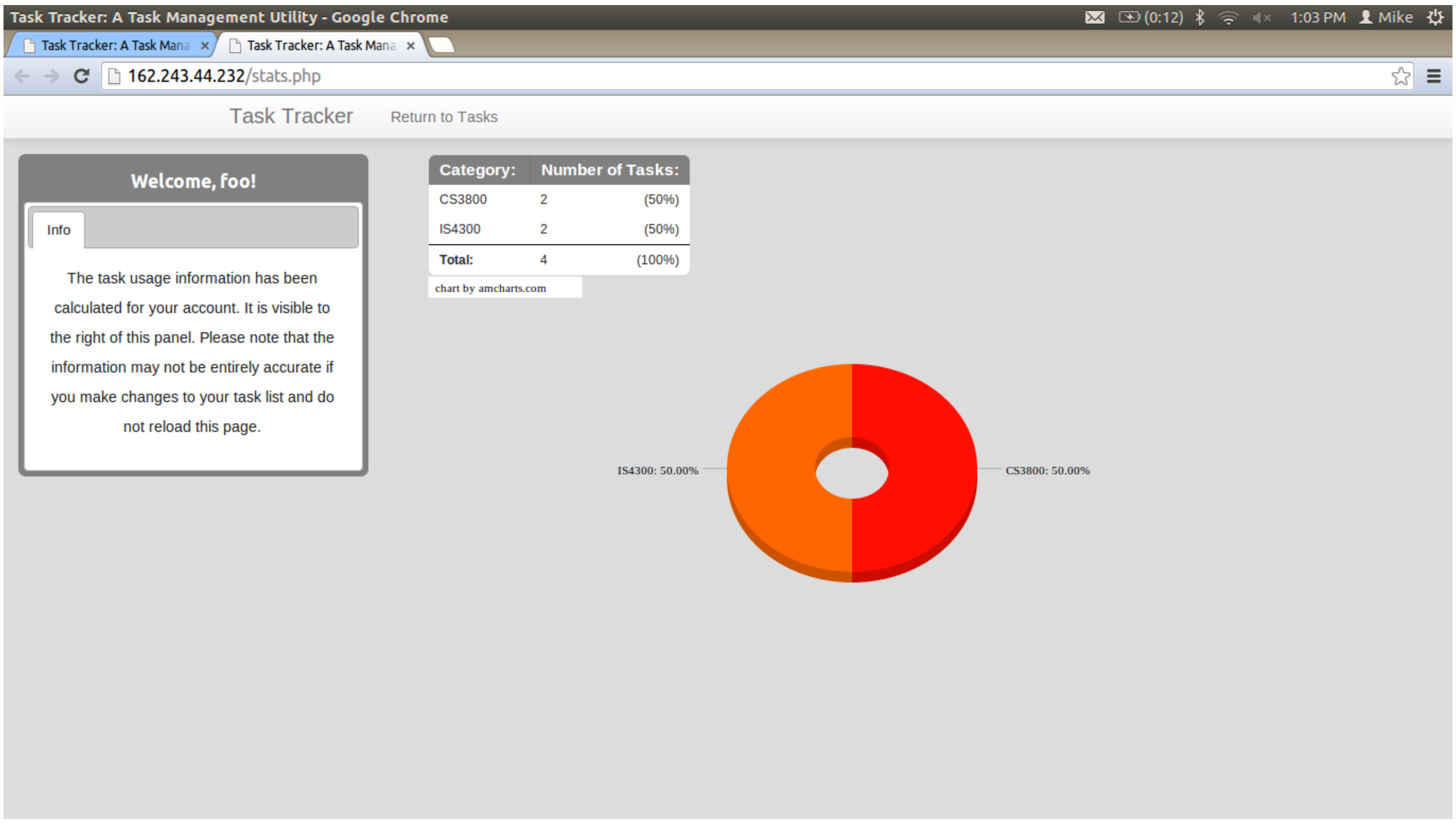
Once you have logged in, find panel on the the left side of the site. It has multiple tabs with various different labels. Select the tab that is labelled "Add." Fill out the form:

- Category - *Category of the task that you are adding (ie. Course, Project Name, etc.)*
  - *Need a new category? View how to manage your account's categories here.*
- Due Date - *When is this item due? Need a calendar? - Don't worry, one will load for you.*
- Description - *Type a brief description of what you need to do.*

Once you have completed that, click the "Add Task" button to add it to your task list.

See! That wasn't so bad, was it?

Screenshot 3: Help Pop-up Screen



Welcome, foo!

Info

The task usage information has been calculated for your account. It is visible to the right of this panel. Please note that the information may not be entirely accurate if you make changes to your task list and do not reload this page.

Category:	Number of Tasks:	
CS3800	2	(50%)
IS4300	2	(50%)
Total:	4	(100%)

Screenshot 4: Stats Screen