**FEATURES AND CUTS**

We have successfully completed the major functionality outlined in our original SRS document. A few additional features were initially planned but did not come to fruition, most notably “qtracks,” a plotting tool intended to give users detailed information about queue activity as a function of time. One reason this was dropped was simply time—as the final release approached, we decided as a team that our time could be better spent improving features more integral to the core functionality of the website. Evan Whitfield had written a successful plotter, but it was done using Python and we realized that we’d have to do some restructuring to use that code. This is because our server returns exclusively JSON objects and, therefore, either files would have to be created and sent to the front end, or someone would have to learn enough Javascript to rewrite the plotting code and run on the application side. The latter would have made a lot more sense, but by the time we figured it out, time was short. Realistically, we could probably have thrown together and released an implementation of qtracks in a matter of a few days to a week, but it wouldn’t have been up to any of our standards. To include that feature—fully fledged, thoroughly tested, and aesthetically pleasing—would require at least another couple weeks of work.

The other feature that we were not able to include was the ability for users that create queues to upload a photo that would be linked to their queue’s profile. The idea behind this feature was to allow queues to include a small photograph that was representative of the organization or person creating the queue. A straightforward example of this would be a company including their logo so that users could easily identify the queue. The reason this did not make it into the final product is that we simply did not see it as a priority and the other features and bugs took up all of our time. In order to do this, we would have to come up with a schema for storing the images on our server and associating them with each queue. Additionally, we would have to learn how to transfer files from to the server using javascript. This could be done easily within another week.

**TASK ASSIGNMENTS**

**Bryan**

Of all the team members, Bryan’s role was the most fluid and thus remained true to what was projected in the original SRS. Bryan was the project manager throughout the quarter as originally planned and spent most of his time implementing administrator controller functionality, though he also put in effort to UI design (chief among this the progress bar for the queue info page) and server side implementation (minor tweaks to views.py). As project manager he also put a fair amount of time into producing documentation and coordinating team work.

**Evan L**

Evan ended up being the person to create and manage the website. The majority of his time was spent continually reworking the site to get the appearance we wanted. However, he also spent a fair amount of time improving/fixing glitches that arose as we implemented features on the site. A good amount of his time was also spent on helping to create front-end test cases and deliverables, like the user information page and our many documents. From pretty early on Evan was assigned to be the web programmer for the team. Managing the website seemed like a task for about one person.

**Evan W**Initially EvanW was assigned to the UI/frontend team, but it was later decided that since he was interested in queue analytics, that it might make sense for him to be more directly involved in the in-memory Queue classes from the beginning. He also had some previous experience with Python which was useful for this task. After starting work on the Queue classes, he became increasingly interested in maintaining these classes, and continued to be the main person to update them as the project progressed.   
EvanW also spent some time contributing to various required documents, writing Unit Tests for the Queue classes, and working on making a plotter, which probably will not be in the 1.0 release, and will instead be pushed to a later release.

**Nicholas**

In our initial requirements document Nicholas was given the role for back end design and implementation. That was not the role he took on once work on the project started. He worked closely with Thomas to define many of the interfaces between the server and client side for requesting JSON data but the focus of Nicholas’ work was in the client side JavaScript. Nicholas wrote the prototype queue page, which was client side only, so he continued in that work once coding started.

Nicholas spent more time than we initially expected working on support and administration tasks in relation to the documents needed for each assignment and the production server configuration and management. This took a lot of time but his major role was the client side JavaScript. This included the Angular controllers and services used throughout the webpage.

**Nora**

Nora’s initial plan was to be heavily involved in the object-oriented design of the queues and the queue server, since she has basically no experience in anything web-related or SQL-related. However, Evan W ended up implementing the queue classes while Nora learned about Flask and regained familiarity with Python. She even dabbled a little in the database functionality, tackling bugs as they emerged on the server side (she had no involvement whatsoever with anything on the client side).

**Simone**

Simone was the main point-of-contact for the GitHub bug tracker, frontend test plans, frontend testing, and UI design. Originally, she expected to spend the first third of the quarter on UI design and the last two thirds on testing and keeping the bug tracker updated, but she found that UI design occupied way more of her time than originally planned, extending into the very last week of the quarter.

**Stephen**

Stephen described his current role as “junior angular developer.” The majority of his time was spent creating and editing documents that we had to turn in. Toward the latter half of the quarter, once he’d had enough time to learn how to use AngularJS, he did started implementing features with it, but never implemented anything that was very complicated. His original expectations were that he would become very familiar with the framework. To explain that further, for example, he would have expected that by the end of the project he would be comfortable putting AngularJS on the skills section of his resume. Contrarily, he learned enough to speak generally about it with someone, but he won’t be including it on his resume.

**Thomas**

Overall, Thomas’ current role ended up being a server side utility person. He was able to work in any part of the python code to quickly address client side needs. The majority of his time was spent coordinating with client side on the format of the JSON interface, and the proper handling of errors. However, a good portion of his time was spent initially on implementing database utilities.

His original expectation was to implement database utilities. However, since our server side design was not stable enough to allow as much parallel development as would be desirable, he ended up implementing a lot in route handlers as well, with a few contributions to the queue server. He also spent a vast amount of time simply coordinating with client side about what data was going back and forth, and how to handle errors. He did not expect to be spending time on errors and data

**SDS**

* 1 days working with Stephen, Evan W, and Thomas to create architecture diagrams and class/object design.
* 0.5 developer days researching test frameworks for JavaScript code.
* 0.5 developer days editing SDS documents.
* 2 days building a prototype example page.

**Zero Feature Release**

* 0.5 days adding boilerplate client side code, organizing the github repository
* 0.5 days updating the use cases from the SRS.
* 1 day researching automated testing and working on ways to email the group every night.
* 1 day working with Stephen and Thomas to get the production server to host our system and developing a procedure for deployment.
* 0.5 days editing documents.

**Beta Release**

* 0.5 days building a controller and service for creating a new queue.
* 0.5 days building a controller and service for requesting and displaying popular queues
* 0.5 days building a controller and service for creating a new account and logging in.
* 0.25 days building a controller and service for joining a queue.
* 1.5 days building a page a diagnostic test page for all these controllers to demo their use.
* 1 day connecting the controllers to the HTML UI.
* 0.5 days revising production and development deployment procedures.
* 0.5 days revising testing procedures.
* 0.5 days editing documents.

**Feature Complete Release**

* 0.25 days building a controller and service for Search/results
* 0.5 days building a controller and service for requesting the queues you manage and own.
* 0.25 days updating controller for the creating a queue.
* 0.25 days building a controller and service for editing an existing queue.
* 0.25 days building a controller and service for requesting queue view details (queued and not queued).
* 0.25 days Setup Coveralls for code coverage
* 0.5 days creating a header to be included on multiple pages.

**Release Candidate**

* 0.5 days redesigning error handling and responses with Thomas.
* 1 day implementing the new error handling in create queue, edit queue, join queue, queue management, and create user account pages.
* 1 day fixing bugs on all pages of the site.
* 0.25 responding to the code review and writing additional documentation for Angular code.

**Time Needed for Extra Features**

* 0.5 days to connect and debug QTracks analytics to be displayed in the queue view and admin pages