Visualizing Autograding and Office Hours Queues in Submitty

Alexa Orosz, Ryan Waddel

April 30, 2020

1 Introduction

For any student who has taken a Computer Science (CS) course at Rensselaer Polytechnic Institute (RPI), the Submitty Autograding Homework Submission Server, which will be referred to as Submitty from here on out, is part of everyday life. Almost every course uses it to share resources, take homework submissions and grade some of them automatically or allow Teacher's Assistants (TAs) or instructors to grade, provide quizzes, and show students their grades. Very recently, a feature for creating a queue for office hours was implemented, to help with organizing more difficult classes. Instructors can choose what features they want to use and how they want to set up their page, possibly allowing for its spread to non-Computer Science classes in the future. The main goal of Submitty is to create a better, more streamlined, and informational experience for everyone involved. Being longtime users of Submitty ourselves, we created a possible addition to the existing Submitty site, visualizations in the form of bar graphs that show wait times for the Autograding and Office Hours queues.

The visualizations we are creating show the weekly trends for the Autograding and Office Hours sections of each course's Submitty page. Our hope is that for Office Hours information will be useful to students, TAs, and teachers so they can make decisions about when to attend office hours, how much time people are waiting for help or how long they are getting help for. In this way, students can identify days with less waiting and TAs and instructors can see if they should add more help on busier days. Hopefully, this would help balance out the volume of students across more days, so students could get maximum help with least wait.

As for the Autograding page, different days and different homeworks have different homework loads. Some classes have tests that grade almost instantly, others have tests that can take a few minutes to do. Thursday nights are also a big night for homework to be turned in, with an 11:59:59 PM EST deadline. While students should not use the autograding feature for testing, what usually happens is that the output will not match the sample output correctly or something executed differently on the Submitty servers than the students' environments. This can lead to making tweaks as needed and submitting again. If a student is up against a deadline, is being helped in office hours, or is just wondering how long this suite of tests will take, the visualization will be able to provide them with a good estimate with a weekly and hourly average. As such students and those assisting them would be able to see that perhaps Thursday is not the best day to submit completed work for the first time or that from Monday's wait times that the autograding takes longer on this assignment, and decide to test earlier.

Our hypothesis, building on trends we and other peers have noticed from using Submitty and taking CS courses, is that when students notice the longer wait times for grading and help coincide with days that are closer to the deadline, proactive students will change their study habits or schedule so they can avoid wasted time and stress later on. At the very least, they can learn about their own, and their course's, study habits and gain a visual explanation as to why their submission is taking so long or the TA can only help for a few minutes. Hopefully we will have the best possible version of our ideas added to the current version of Submitty.

2 Background

2.1 Initial Peer Feedback

The idea we ended up choosing did not get much peer feedback, but we had a decent amount of instructor advice, which made up for it. One peer did recommend making a scatter plot correlation matrix with submission time, number of submissions, lines of code, number of functions, comments, and submission runtime to see the correlation between those things and grades. While we thought it could be a wonderful extension to this at some point, students currently have no idea what the submission server is processing or how long it has been taking, with much the same situation occurring for office hours. This would also be a great deal to parse and interpret. We knew we wanted it to be as accessible and easily readable as understanding the meaning of these visualization and making a change could really help some students with planning.

2.2 Related Works

"VISUALIZATION METHODS FOR TIME-DEPENDENT DATA - AN OVERVIEW" discussed a variety of techniques for presenting and visualizing time-dependent data. The authors, Muller and Schumann, define types of time and data and then supply visualizations that fit best in the intersection of time and data. For our project, we were using interval time to show the time dependence of static visual elements [1]. We are using intervals, such as hours, days, and weeks as our data is defined between two points in time [1]. The paper suggests the bar graph for cumulative data as a conventional method [1]. We gravitated to its simplicity, as we wanted to present the information in an easily understandable manner. Using a new design or less easily readable format could cause confusion in the students. Eventually, this would be affirmed by other papers we read, and we both used bar charts for our visualizations.

Donaldson, Ashley, et al. 2015 wrote in "DizViz: Visualizing Disneyland Wait Time Data With a Focus on User Experience" about their design for a new app to show current wait times for rides in Disneyland. Their background was human based design and engineering, so they focused on the best design based on human behavior [2]. Large elements of their final designs involved color-coded bar graphs, citing these as easily readable for people on the go and

that length is an easily readable metric for comparisons [2]. They reference color for distinguishing different rides and also helping to encode quantitative data, which we also had.

As a side note, when we were looking to include visualizing the traffic on the servers as well, we looked into "Visualization of Urban Mobility Data from Intelligent Transportation Systems" by Sobral, Thiago et al. which attempted to visualize the movements of people over time in a visual environment. Our reasoning was that something that could accurately visualize the complex movements of people over time would also be good inspiration for how to visualize traffic on the servers. However, we never got past the initial idea and the concept of using heatmaps and clustering during this project before decided to shift it into a stretch goal [3].

Submitty already used Plotly, which is built over D3.js and WebGL to create a charting library to more easily create various charts and graphs from existing data [4]. This was convenient for us to use, as all the necessary features were being called and loaded in. The various features of the types of graphs and charts they offer, with bar graphs being one of them, are easily combinable and customizable. They have a version for JavaScript which loads quickly on the Submitty pages. We were able to quickly plug the data into the graphs of choice and get a good result.

3 Visualization Design Evolution

3.1 Beginning Stages

Initially, we had envisioned a much simpler design where only the user's place in the queues (autograding or office hours) was shown. It was a horizontal bar graph and would change dynamically based on real time data. However, we did not think this would solve the problem of helping students do better on their assignments over all. We wanted to create something that would allow students to make more proactive decisions about when they got help or submitted work as well as showing them the data that they could not see so they could understand the process better. Since the structure of the CS department here at RPI, especially in early classes, is very homework based, where the homeworks are small projects, and the homework is due every week on Thursday, we decided the visualizations would be centered on that week structure. The data had to be anonymized, with personalized data only appearing to the relevant user or only averages or totals with no specifics shown.

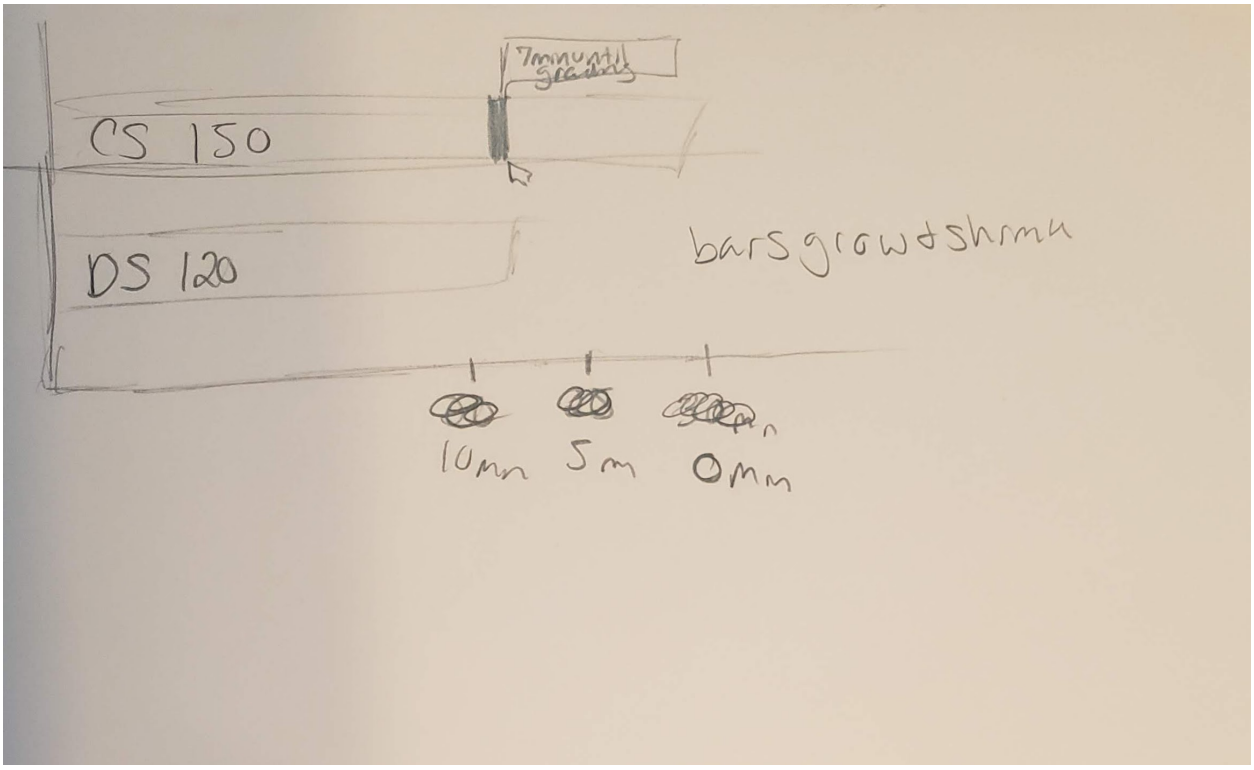

Fig. 1 Our initial design for how the user would be visualized in the wait for grading. This was revised in favor of a more readable graph to help change behavior.

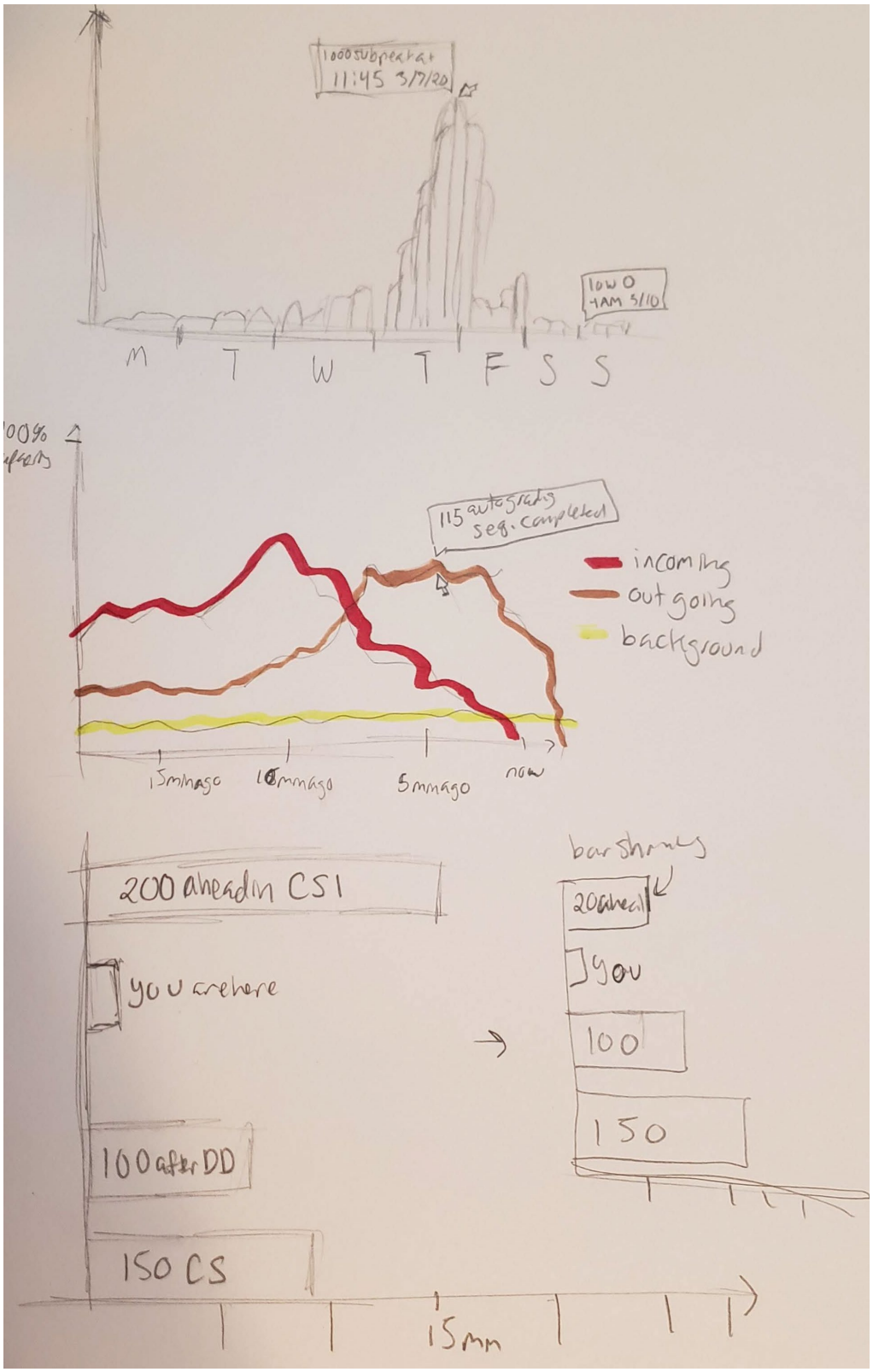
1000 subpeak at
11:45 3/7/20
low 0
4AM 5/10
M T W T F S S
100%
115 autoscaling
seg. completed
incoming
outgoing
background
15mn ago
10mn ago
5mn ago
now
200 ahead in CS1
you are here
100 after DD
150 CS
15mn
bar shrinks
20 ahead
you
100
150

Fig. 2 There are more concepts here, with the top being an early version of the bar chart format we ended up doing, the center being the scrapped design for server traffic, and the final was another iteration of the stacked bar concept

3.2 Work In Progress

As explained in section 2, we decided on bar graphs for our visualization, and after deciding that showing the individual student moving through each queue was not practical to represent for our purposes, we eventually switched to a regular vertical bar chart. Since Submitty has an established color scheme that signals “Good”, “Bad”, and “Medium” using green, red, and yellow respectively. We would encode our bars to show the severity of wait time using a combination of color and scale. Our base visuals were just colored graphs with high times in red and low in green. During the beginning of our project, we only focused on weekly averages, but had the idea to show students their custom averages based on their information, which we believed was a natural extension.

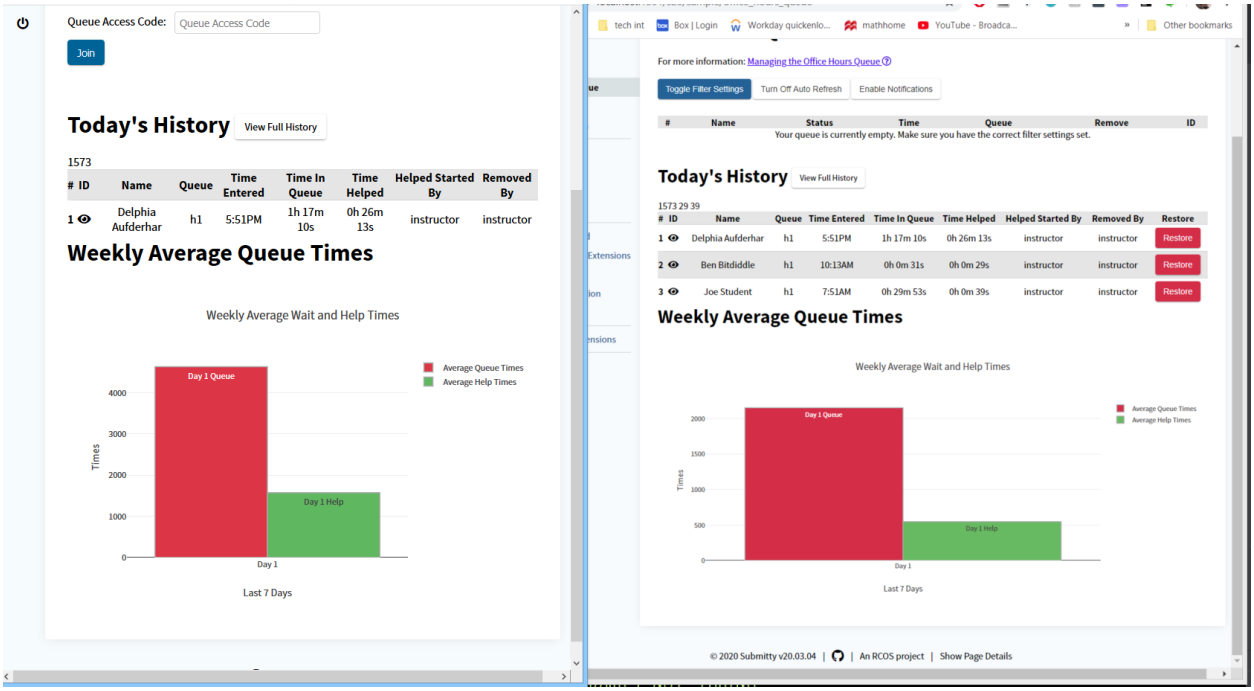

Fig. 3 The student (left) and instructor (right) views of the weekly average queue times in their first iteration.

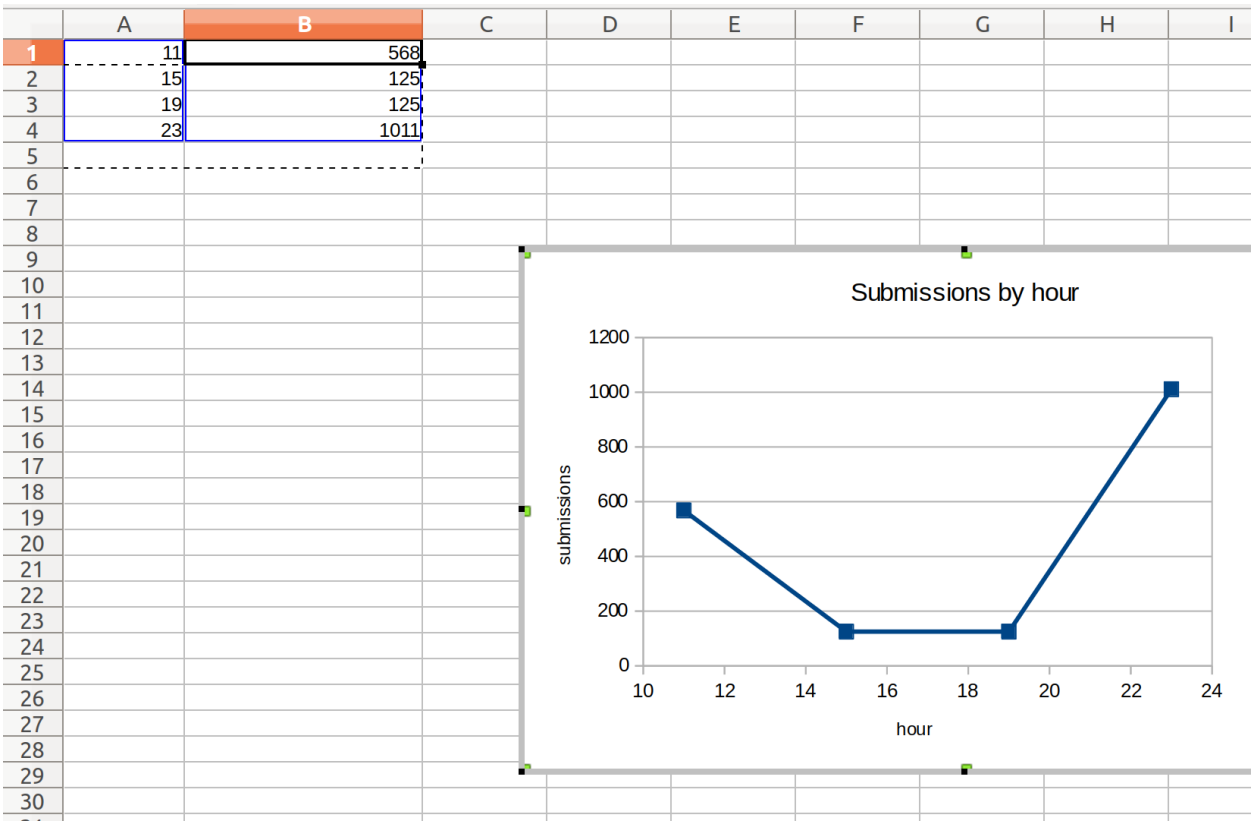

Fig. 4 Early log file parsing results for autograding before we totally switched to bar charts

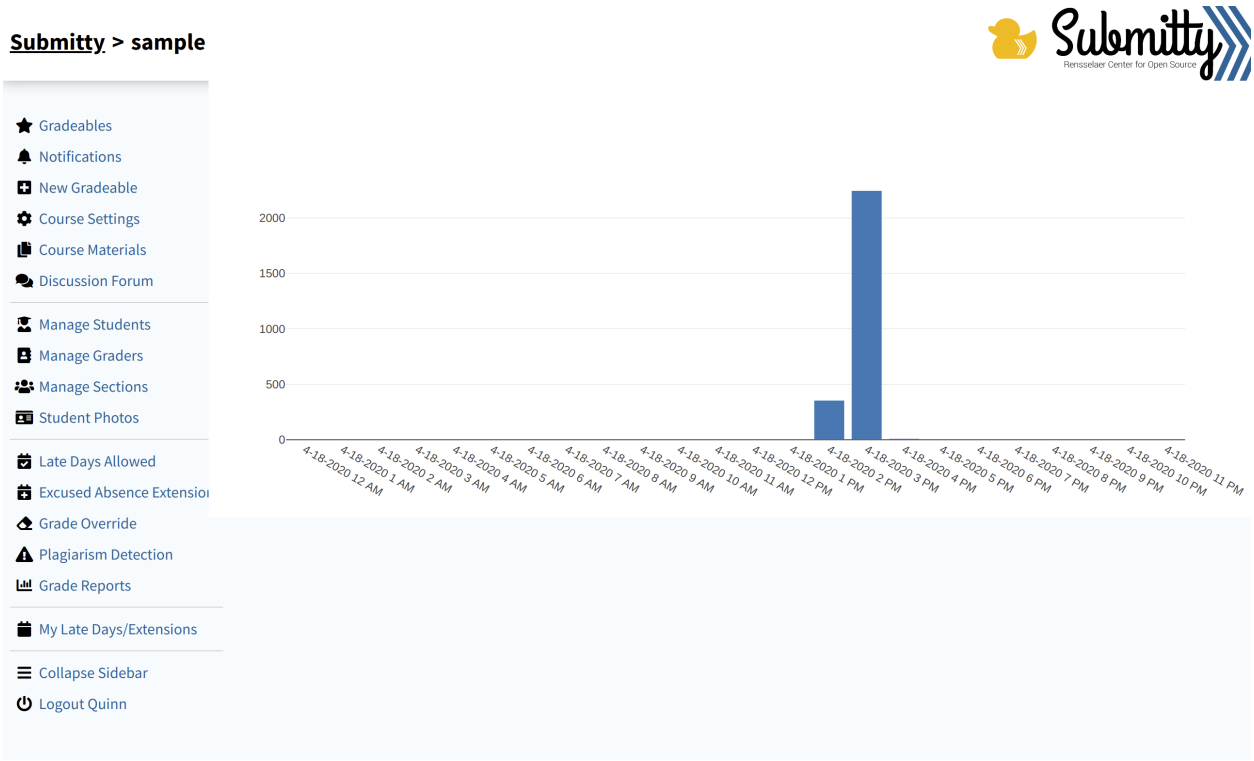

Fig. 5 This was the initial autograding graph before color was added

We were unfortunately not able to set up a user study or have a day for peer feedback within the course as they had in past years. We had to rely on ourselves and some minor familial input, which amounted to it looking nice, so we had no real chance to show anyone with the knowledge that our target audience would have until our presentation. At this point, we were able to self-refine the graphs. Our main changes were to the way we used color. For autograding, the longer the wait for grading to start or for how long the grading took, the longer the bar, with a gradient from green to red on the bar with the more red corresponding to the longest times and more green corresponding to shortest times. For the queue, longer bars also signified longer wait or help averages, but red was set to the waiting for help average and green was set to the being helped average. This way the two could be distinguished at a glance. They also had the average wait time for each bar written at the top to avoid confusion and to provide precise and accurate information. At this point it was discovered that it was fairly easy to make another database request to only get personalized information for the student in the Office Hours Queue page, even though the parsing took a bit of effort, so we implemented a weekly personal graph as well that was the same design as the general average graph.

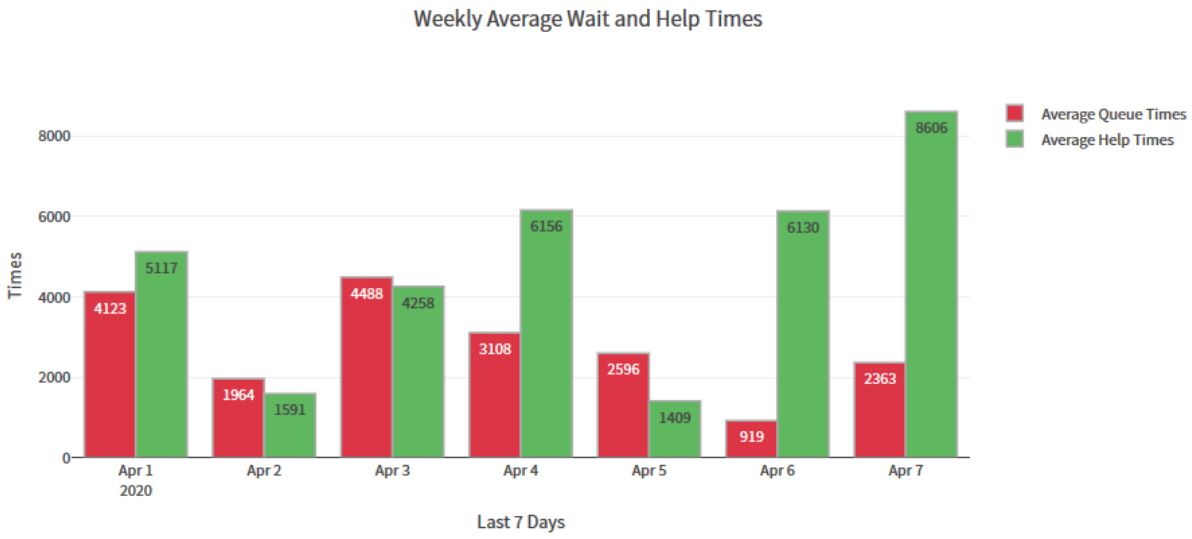

Fig. 6 Here is the final Queue page wait and help averages with sample data, note that it still is in seconds but every bar has the labeled value with a legible key beside it. The stark contrast allows for students to discern at a glance what bar means what.

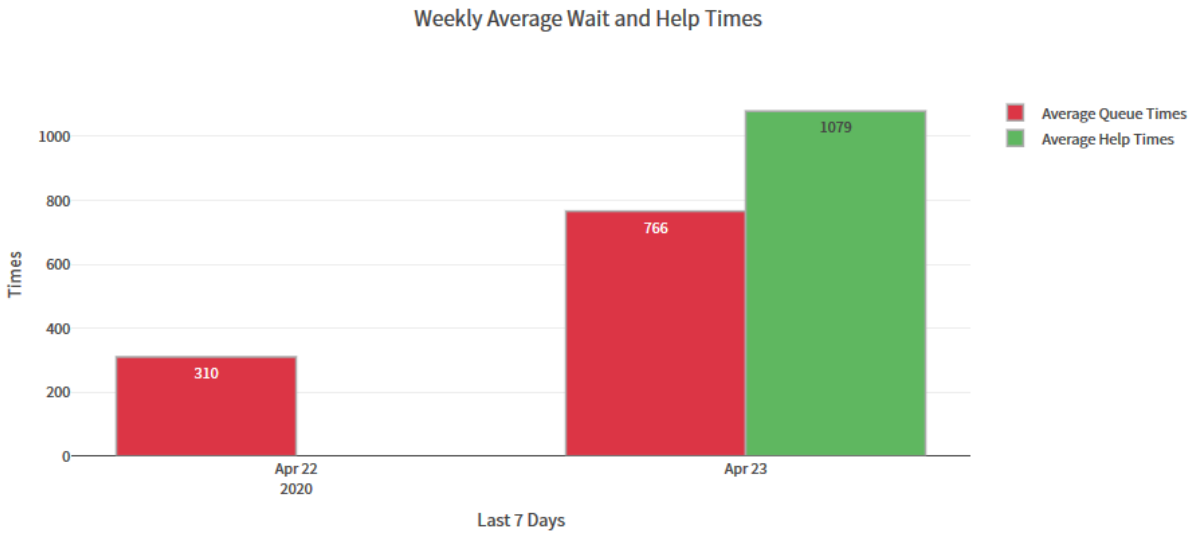

Fig. 7 is exactly the same as the one above in design but is showing the student's personal graph

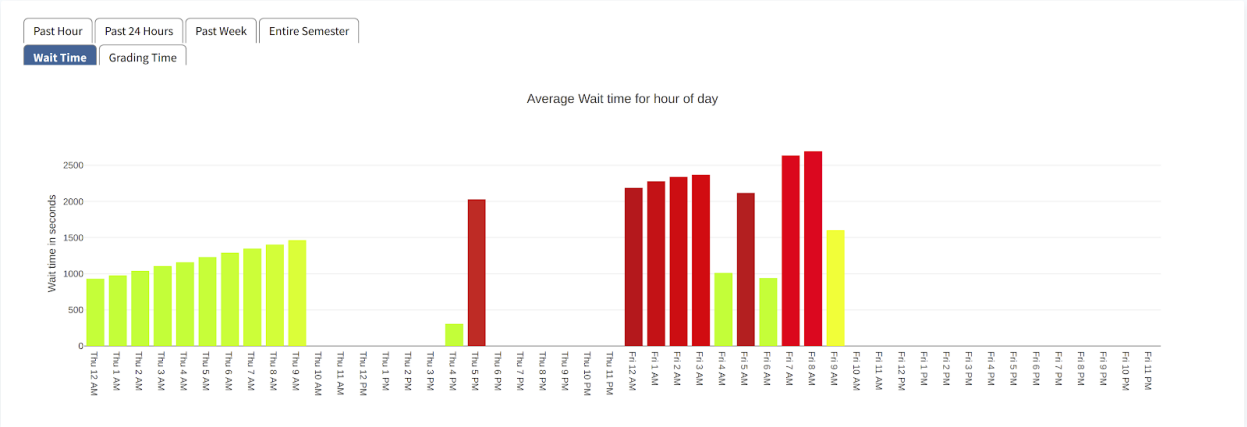

Fig. 8 shows the Wait Time graph for the Autograding page, showing an hour by hour breakdown of the highest and lowest volume times and using a gradient to show the longest and shortest times at a glance

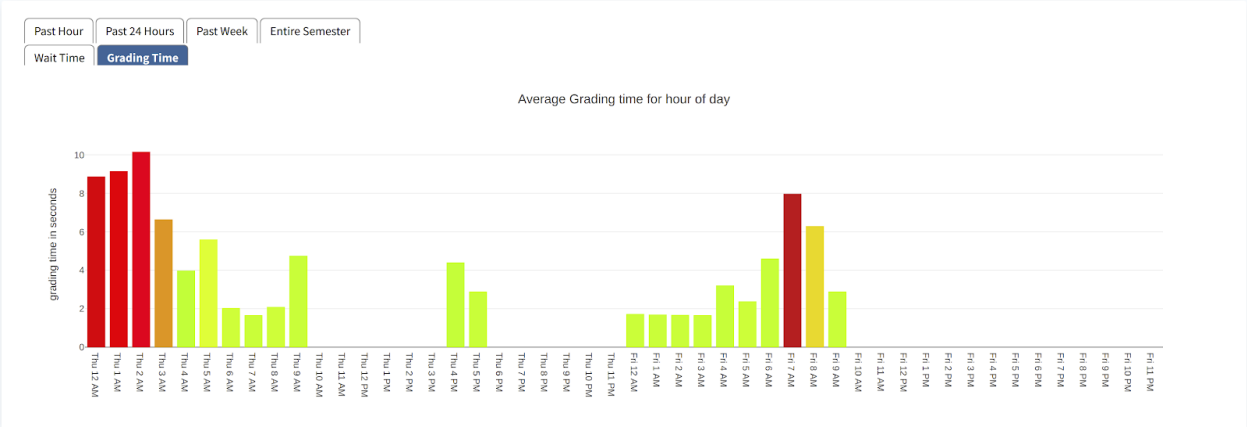

Fig. 9 shows the time it took to grade each submission on average. Please note the same gradient color scheme as above

4 Feedback

4.1 Peer and Self Review

As aforementioned, due to current social events, we were unable to meet other students and get their opinions in person. We also did not have contact information for other students in the class. Therefore, the reveal of our designs took place during our final presentations where feedback was mostly focused on the presentation itself or the idea as a whole. Most responses saw this as a useful addition to the site, if a bit limited at the moment. They also thought it may help students who procrastinate work see why this would not be the best idea, but that the visualization in itself may not be enough to inspire change. Luckily for us, those in our peer group accurately represent the type of user we would want to use our visualization. None of the complaints were about the design itself, more on the volume of visualizations we could show from the data.

As far as self-reflection goes, we were pleased that our visualization accomplished our main goal of being easily understood for students. For the graphs, we believe that time in minutes will be the most easily understandable format to use for the measure of minutes waited or helped. Currently it was in seconds as that was how it was stored in the database or the most easy time increment to average, but that is extremely unhelpful. The office hours typically held do not exceed two hours in total so it is not unreasonable that the average wait time should not exceed that, and probably not exceed an hour. Currently, the office hours wait/help bar graph does not have the day of the week, only the date on it, which would also help students more easily associate peak or low average times with a certain day of the week, as from experience, the homework cycle seems to be quite cyclical. We have considered also doing a gradient for the office hours bar graphs, but think because of the large contrast in color between the two (wait and help) it is more useful to understand which is which as it stands.

4.2 Future Work

This visualization will hopefully become part of Submitty and move onto the production build so more students can use it and give feedback on it. In order to do so, we will have to finish squashing some bugs and the Office Hours Queue page will have to be fixed up. Currently it does not use the tab system to represent multiple graphs and does not show as much data as it could. In the future, we want to show the averages by office hour as well, so less popular times of day and more popular times of day can also be discerned. We could also show the averages accumulating over the whole semester a well, both for the course as a whole and for each individual student.

In addition to the existing and extended visualizations we could show either text statistics for each particular student as the semester progresses or use some kind of stacked chart to show how much time was spend doing what in the Autograding and Office Hours Queue pages. In the autogading it could be the average and/or total amount of time waiting for homework submissions to grade against the average and/or total amount of time spend actually grading the homework. Similarly there could be an average and/or total time spent waiting to be helped in office hours and the average and/or total amount of time spent being helped. It could also show total amount of time spent in office hours for them. There might even be an anonymized scatter plot showing the correlation, if any, of hours spent in office hours vs grade in the course.

Of course, this all can be implemented after we put in a pull request for both of these initial changes to be approved and migrated into the source code. We are both hopeful that this will be able to help students understand about how to structure their time so they can get the help they need.

5 Technical Details

This project created visualizations in two areas of the Submitty site. Both are a variation on the theme of time spent waiting for a process to happen and the time spent executing the process. For the Autograding page, there is a graph showing the average wait time to start grading for that particular assignment and then another graph showing how long the actual grading takes. Depending on the day/time, class, and assignment, this can vary greatly and even affect a student's ability to perform a final debug of the system and make the final submission on time. With this showing what the average times for those are, students should be able to see that certain days and times are poor choices to try and turn homework in quickly. In a similar vein, office hours tend to get crowded on certain days and times. Showing exactly what days have historically the lowest and highest wait and/or help times, students should be able to discover when the best times for them to go are.

For the autograding, there are log files that have been generated with the wait times and grading times for each submission. This can be parsed and using JavaScript and Plotly, was shown as a bar graph. Because of the formatting anf information density of the file, there was some difficulty in getting the correct information out and organized.

As aforementioned in 2.2 and in the previous paragraph, Plotly was the visualization package of choice based on its ease of use and compatibility with the web-based environment we worked in. The Office Hours portion of Submitty was not created by us, and many of the database requests created were based off of existing ones. Some of this data was parsed with existing functions as well. However, some new data had to be retrieved with new requests, which was a bit of a trial and error process. The transfer of the data from the Twig portion of the file to JavaScript was also a bit of a trial as sometimes things would fail silently or have vague errors. There was a bug that ended up being because a PHP date/time variable would not convert. Some of the parsing done was a bit convoluted to try and keep the order of dates without having to make any

complicated sorting happen. The data was pulled with a query and brought into the file and then the pertinent data was called out using existing functions. The Twig arrays were brought into JavaScript so we could use Plotly. Then three corresponding arrays hold the day, the help average, and wait average. There was a bug where null entries were not being counted as zeros, and caused some confusion.

The most frustrating challenge was that for one of us, Alexa, the reinstall to apply changes took approximately 7-11 minutes per run, whereas Ryan's took approximately 30 seconds, which is about the average time the install takes for most people. While it did not inhibit progress, it certainly did make debugging and testing a bit more difficult.

6 Conclusion

Driven by our own experiences with Submitty, we tried to develop visualizations that would help struggling students. The first few CS classes can be a struggle, and having to wait unknown amounts of time or not knowing the best times to receive help can inhibit a student's ability to succeed. Knowing this, we created a suite of visualizations attacking the two most time-consuming areas of debugging and problem solving relating to Submitty. Our visualizations can show past course and personal trends to allow students to take that data and make more proactive decisions for themselves if they are struggling getting help or doing their final debugs with Submitty in a timely manner. While there is still much expansion and improvement, we feel that the more information students are armed with the better they are able to make their own decisions and those who apply this knowledge will have a quality of grade and life improvement.

7 Who Did What

Ryan worked on the Autograding section. He parsed the log files containing all submission and grading data. He then also organized this data and transferred it into a format that could be put into Plotly and generated a multitude of graphs from that. Alexa created some queries to pull data from the database and used a combination of existing functions and new processes to organize and average out the times. She then also put the data in a format that worked with Plotly and created the visualizations for the Office Hours Queue page.

References

[1] Muller, Wolfgang and Schumann, Heidrun, editors Chick, S., et al. "Visualization methods for Time-Dependent Data-An Overview." Proceedings of the 2003 Winter Simulation Conference. 2003.

[2] Donaldson, Ashley, et al. 2015, "DizViz: Visualizing Disneyland Wait Time Data With a Focus on User Experience" https://userexperiencedesign2015.files.wordpress.com/2015/09/411final-2.pdf.

[3] Sobral, Thiago et al. "Visualization of Urban Mobility Data from Intelligent Transportation

Systems.” Sensors (Basel, Switzerland) vol. 19,2 332. 15 Jan. 2019, doi:10.3390/s19020332
[4] Plotly.js (n.d.). Retrieved April 29, 2020, from https://plotly.com