TRIVIA GAME SITE

SENIOR PROJECT FINAL PAPER

JEFFREY KLAMERUS

APRIL 26, 2019

Introduction:

When I first started this senior project, I admit I wasn't sure what to do with it. In fact, it took until the proposal deadline to come up with an idea. Many fields of Computer Science interest me, from robotics to software design, but one area I've always found interesting is the networking side. Considering I had recently finished a course on client-side web programming, I decided to start from a chat program and build my way up.

Inspired slightly by some sites, such as skribbl.io, I decided to integrate my chat program with a game as well. However, aside from the one course I took on web design, I was admittedly rather new to it. One of my goals going into this was to get a better understanding of several tools available to web designers, including Socket.io and JQuery (which were covered in class but which I had not had much hands-on experience with beyond basics), as well as some other resources such as MySQL and BCrypt. Unfortunately in the case of the latter two, I got to that aspect of the project very late in development, and didn't leave myself enough time to find any alternatives, for all attempts to install MySQL were met with failure for reasons I haven't been able to discern. As a consequence, one of my proposed features, a login system, ended up not making it into the final result, at least at the time of this writing (I hope to touch it up some before my presentation, as it's a little messy right now).

I think I would have to say the greatest difficulty I had in this project was self-inflicted. I waited too long to get started, and ran out of time to implement a fair few features. But I'll discuss those as I get into the various moving parts of this project.

Rooms:

This was honestly intimidating to start with, despite the simple concept. Were this a standalone chat program (one of my initial ideas for the project), servers would have been hosted by clients. In the case of a website, however, rooms have to be entirely server-side. This meant that the first challenge I had to tackle was setting up socket.io to send messages only to certain users in certain rooms. In the end, I settled for giving each user a username (since logins don't work, these usernames are simply "Guest#" followed by a number), and creating an object called "client", which held the guest's username, as well as their room number and their socket. I stored these client objects in a Set. I chose to use a Set instead of an array because clients would be connecting and disconnecting all the time, and it would be cumbersome to resize the array and shift clients around to fill empty spaces.

Because I gave each client their own roomNumber variable (with -1 being the lobby), I could simply broadcast messages only to clients with certain room numbers. Then, by storing the rooms in an array, I could use the room's array index as its unique ID, and make sure that the client's room number matched the ID of the room that they were in.

One challenge I faced that I was not expecting was the difficulty in making comparisons between strings. I've only ever used Javascript before in a few places, and this caused me serious problems with password authentication for joining password protected servers. I ended

up finding a method I'd never even heard of prior, called localeCompare(), which eventually did the trick. I'm sure there was a better way to compare strings, but time was running short for me and I was forced to settle for what worked.

Chatroom:

The chatroom was probably one of the simpler parts of this project. I'd already built one of my own two times since I've come to Northern—once as an extremely threadbare website and once as a standalone application—so I already knew the basics. Most of my issues here came from my lack of understanding of javascript and JQuery. It took me a long time to figure out how to automate the scroll bar to keep it scrolling down and keep up with the chatroom. While I am happy with the ability to quickly and easily switch between rooms, I was only able to work in one chat command, "/whisper" or "/w", which allows a user to send a message to another user within the same chatroom that nobody else can see. Due to the failure of setting up the login system, I will likely get the "/nickname" or "/nick" command working before my presentation.

One aspect of the chatroom that was surprisingly tricky was dealing with when to close it, and when to clear it. I knew that leaving the chatroom would result in it being closed, but I had not considered how the room creation dialogue and the room join dialogue might get in the way.

The Game:

This is the aspect I deliberated on for the longest time. I knew that a simple chat website would be too feature-bare for a senior project, but I struggled with finding a game that really fit

the idea. Fast-paced games would defeat the purpose of the chat room. Complex games, like board games (chess, checkers, connect four), I wasn't even sure where to begin with. I felt sure games like that, depending on the level of complexity, could potentially be worthy of being senior projects on their own. More to the point, however, I felt that board games were too limiting—many only support up to four players. I wanted to find a game that had no upper limit of players.

I settled on Trivia for that reason. I was inspired by a trip to Buffalo Wild Wings with my family, where we always watch the trivia screen. My frustration with the idiosyncrasies of Javascript led me to create a poor implementation of the questions and answers, but with just a little work it could be made to be modular, with no upper limit of questions. If one implemented file reading, they could probably even have question "packs" they could switch in and out. None of this is relevant here, at least for now. My questions are built directly into the code. At the time of this writing, there are only three (for testing purposes), but that could easily be increased, and I likely will find several more before presenting my work here.

At this point in programming, I was cutting it right down to the wire. As a result, while I did implement a timer (30 seconds between questions, 15 seconds to answer each), I was unable to add a countdown timer, nor could I get to finishing the leaderboard or time-based score. The latter would be fairly simple with a message back from the server telling the client how much time was left. The countdown timer actually gave me a lot of trouble—I really tried to use setInterval() to count down second by second, but when used client-side it resulted in desyncing between clients who joined the room at different times, and when used server-side it caused many strange crashes. I do not know why these crashes occurred, it could be that

setInterval() interrupts other processes and causes problems, or it could be that I was using it wrong at the time, but either way, I was forced to settle for a double-loop using setTimeout() instead. I would have needed to get setInterval working to have a countdown.

Organization/Presentation:

My website files are organized in a fairly simple manner. Since I am running Node (for Socket.io), I have a separate server.js and client.js files, the former handling what runs server-side and the latter drawing and changing what the client sees. The HTML page is threadbare—it contains only a message that if you see the text, the page failed to load. The server.js is in the parent folder, while everything else is in a "pub" folder.

Organization within the code is unfortunately another matter. Due to the amount of work I had to rush through, I didn't give myself much time to plan it out. I will be going back over it to clean it up somewhat with comments, and to get rid of unnecessary code as I spot it, but I fear I'll only be able to do so much without rewriting a lot of code.

What brings the code together is socket.io. There are no post/get methods, no page changes, everything takes place on one html page. Instead, I work through socket messages and JQuery edits to make the page change on the fly. Clicking on a room will bring up a room, or a room join dialogue if the room is password protected. I make liberal use of .hide() and .show() in my code—all of the panels are actually there the moment the page is loaded; they are simply hidden from site. I made an effort to make it intuitive what can be interacted with and what can't be. Buttons are obvious, but anything that can be clicked on will also change in some way when the mouse is over it. There are a few problems with it—I never put a cap on name the

length of a room name, so one could feasibly spam it and lead to an ugly, thick room. I was unable to finish the sorting or searching systems, so one must scroll through the list of servers by hand. I did at least work in the password filter and refresh button, so that unless the user is looking for a room created by a friend, they can see only servers they can join.

Experience:

I learned several things in this project, though most of them were unrelated to the tools I used. I suppose the biggest thing I learned is that I don't think web design is for me. I wasted a lot of time that I could have spent making progress just tweaking the JQuery objects and the css file, trying to make shapes appear where and how I wanted them to. It was endlessly frustrating—many of the css properties and html element names aren't intuitive at all, and oftentimes their behavior borders on the bizarre. Setting the background color of a , for instance, doesn't change the background inside the , but the background *behind* the . In fact, no color command I could find would change the color inside a . I ultimately had to resort to putting a <p> inside the and changing *its* background color. As such, nearly every on this webpage has an invisible <p> inside it. And that's without getting into moving all of the parts into places that make sense.

And the worst part of html/css is that when something goes wrong, there's almost no capacity for debugging. The web developer tools in Opera helped me somewhat, but they would only tell me when a shape had simply been *hidden* and not, say, shrunk to an invisible size, moved behind another element, or moved offscreen. Even then, they wouldn't tell me *why*.

I suppose the bigger lesson I learned on this project is not to procrastinate. It sounds obvious, I know, but I underestimated just how long it would take to make headway on this project due to javascript and html being needlessly finicky in strange places. You just need to see my aforementioned issues with comparing strings for an example of that. As a result, what I thought I could do in three weeks (with a little hard work) ended up being something I could only half do, and it was only through non-stop work with no free time for the final week that I got it to a presentable state.

In terms of what I actually did learn about the technologies, I learned a fair bit about JQuery, for one. JQuery is a lifesaver, making it incredibly easy to edit objects on the fly. In particular, its .append(), .hide(), and .show() methods can practically build entire sites on their own, and the ease with which you can edit text fields made it far easier to manage the chat part of the website. I can't imagine what a mess the code would have been for the room listings, either.

I guess I learned more about javascript and css as well. Though in their case, it's mostly that I learned annoying design choices I had to work around. In the case of javascript, I came to hate how often I had to trace through the code to find the *actual* source of any given error, and I learned to *dread* the word "undefined", because it meant I had to scour my code to find out where I made a mistake that resulted in that. The line provided by the error message was no help in those situations.

In the case of css, I learned how to make things work, after a fashion. I'm sure the solutions I came to were not optimal, and that there are likely well-known techniques among web designers to move elements to exactly where they need to be, and to make them behave

properly, but it honestly seemed like even when I looked online for help (as I often do), the tutorials and tips that others game just *wouldn't* work for me. And like I said before, there is basically no troubleshooting css. If I try to give a margin to a button to get it away from the edge of a panel, and it doesn't move, I've simply got no recourse at all.

Conclusion:

In conclusion, I could have done far better. In retrospect, and with the experience of actually having seen how long it takes to set these things up, I suppose I'm surprised I even accomplished as much as I did. I'm not exactly proud of what I made, but I'm not completely disappointed, either. I wish I could have managed to finish more features, and if I had begun work sooner I probably would have.

I'm proud of how the chat room turned out, if a little disappointed by its lack of additional features. I was able to work in the ability to submit with the enter key, and for it to automatically scroll down with text as it appears. The room list could have seen some improvement, especially in terms of features. Sorting algorithms and a search bar go unused. But the rooms themselves, and the ease of their creation, work well for me. The trivia game is perhaps my biggest disappointment. I'm happy I managed to get it automatically going for any room, randomly selecting questions (per room, too), *and* shuffling the answer spaces around so they aren't always in the same space, but unfortunately it's a mindless diversion at present without score or leaderboards. To reiterate: not proud of the result of this project, but not disappointed either. I will try to clean it up and maybe add a small feature or two before presentation.