CS465P-002 Spring 2023
Full Stack Web Dev
Christopher Snow

Term Project Journal

This journal is organized by each week of the term! I tried to get started on my project early and worked pretty consistently throughout the term.

Week 1:

Project Description

My full stack web application is going to be a website which allows people to create public profiles for their Final Fantasy 14 (FFXIV) characters. Each user will have a single private personal page where they can add and manage their characters. And each individual character will have it's own public profile page with a shareable link. The website will allow users to create an account and log in, and to customize their characters. The characters will be customized by sending API requests for the official character data from the FFXIV servers, as well as by allowing the users to add their own images and text description of their characters.

Github Repository

I started by creating a public Github repo as requested. It is located at:

https://github.com/pearstopher/465-Project

All of my progress on my term project will be recorded here as I work on it in the virtual machine, alongside the Doggr project, throughout the term.

Auth0 Authentication

I have the all of the Doggr stuff working in the virtual machine on my PC already, so I am all set to start working ahead! Luckily there has been a little bit of advice for working ahead:

> *"at some point you're going to have to build an authentication connection to some third-party service (authy, auth0, fusionAuth, etc) and make use of it for your site"*
>
> - Casey (Discord)

After looking around a little bit and researching the different options, I decided to go with auth0 as my third party authentication service. I started working through the tutorials provided on their site, and learned a little bit about how everything works. After downloading the tutorial onto the virtual machine

and following the instructions, everything worked pretty well and I was able to use the demo to log in with the email and password of the test user that I had set up.

For my final project, I am going to configure auth0 to allow login via two social services, Discord, and Twitter. I could add more, but it appears that my free account will only allow a maximum of two social connections when the free trial wears off, which is (sadly) before the project is due. Thankfully, Discord and Twitter usage is very high among the target demographic for my web application. I anticipate that users of my site would be most likely to want to use Discord and Twitter anyways. In fact I am so confident about my users' preferences that I am going to disable the ability to log in with a username/password entirely to streamline the process even more. Nobody wants another username and password to keep track of in 2023.

While Twitter didn't initially require any additional configuration, the Discord login did require me to set up a new Discord application. Thankfully the process is pretty simple and I have made one before for fun. I got Discord set up and was able to log in and authenticate from the virtual machine successfully.

Although I was initially able to log in using auth0 and Twitter, I did notice later that my Twitter login was using a test developer application that was not recommended to use in a live site. After realizing this, I also signed up for a Twitter developer account and created and configured a Twitter app of my own so that users of my website could log in using that app instead. I also got this set up successfully and was able to authenticate myself from the virtual machine. The appearance of using my own app for this was definitely more professional and less confusing (the name of the test app was not very recognizable).

Week 2:

This week I got started on the actual project. I began by copying over the files from Doggr in their current state and making sure that everything worked. Then I immediately tore everything apart and started replacing it with my own code! My plan is to try to keep my site updated alongside Doggr for the rest of the term without pulling any more actual Doggr code. I think it will be much better practice to use this site to help figure things out in a little bit more of an unsupervised setting, where I don't have such a nice set of Markdown pages that I can pull code from if I get stuck.

<u>Database</u>

For my database, I emptied out the original database and deleted all the old schemas and seeders and migrations. I haven't changed the database name yet because I don't want to mess up my database connection right away, but that's something I would like to change later. After the old database code was cleared out, I created a new basic database schema for Characters, I created the initial migration, and once I made sure that everything was working, I also made the seeder file to automatically add some users to my new database.

Figuring out the migrations was really time consuming and awful. I had the same problem with this site as with Doggr, that at some point the migration "got off the rails" and the database wasn't in the same state that Mikro-orm thought it should be. As a result, every attempt to migrate up and down resulted in errors. Either it was trying to create things that already existed, or it was trying to delete

things that weren't there. I'm not a big SQL expert, but I did take PSU's database class, so I know a few of the basics thankfully. Eventually what I had to do to get the migrations working was to edit the database manually to get it back into the correct state. Once everything was back "on track" so that the database was in the state that Mikro-orm expected, then everything started working great again. I could make new migrations, and migrate up and down. Even though this was miserable I definitely used this painful opportunity to become a lot more familiar with Mikro-orm so that part was good at least. And it was so nice to have everything working again.

The database itself is pretty simple for now (thankfully). As of right now, I only anticipate needing two database tables – a table for characters and a table for users. And I didn't even need to add the users table yet because I wouldn't have any users or user data until later when I finished setting up the actual authentication and people can actually log in.s

Routes

I also added the first basic routes this week. The CRUD for Character creation / reading / updating / deleting. And the Custom search route for searching for characters, just like the one from Doggr. We just started to set the search routes up in class but they didn't work yet, so there are still some bugs to fix here.

Week 3:

This week I started working on testing! I made a test suite to test the user CRUD and search functions. I enjoyed making the tests and I think that I made a good set. I did notice some errors too and used the tests to figure out what was wrong and fix them so that my routes worked correctly! So that was great.

Using my tests and the help from class this week, I also got my Search route working correctly after continuing to learn about how plugins work and getting the same route working in Doggr finally. Custom HTTP requests are pretty cool. There should be more of them!

After I had my tests working, I also did some work with the documentation since we had just talked about that. Iadded and configured the ability to generate Typedoc documentation. I had to fix some errors in the code which were breaking the documentation generation. It turned out that even though my code was working great, there were still a few bugs with typing that made the documentation unhappy. So it was good to realize how important that was, and to fix those errors too. In the end I got everything working great and I got the first set of documentation generated! I still have to go through and add better comments to everything sometime though so that I get better documentation. That's not a priority yet because I'm not supposed to include docs in the final assignment and because I am not even sure if you'll try to generate them when grading. But it's still on my to do list because documentation is important.

Week 4:

Week four is my first Frontend week! After having my basic backend routes working and tested, it's time to start on the front. This is all a little overwhelming right now, I almost wish that this class was split into a frontend and a backend class. Well, I say that, but I remember that there is a Frontend class too! I only took Intro to Web Dev before this though, so this is my first Frontend class.

Starting on the Front end, I created some initial routes to test the waters. I got a home page, a profile page, an example character page, and a search page up and running. If only it was as fast to make all those pages as it was to tell you about it! I think I started to get the hang of it though.

The most time consuming part of the frontend this week wasn't getting the first components working though, it was just getting the whole shell of the site up and running around them. I had to get some basic HTML going, and some CSS. I added some initial CSS that I wrote by hand just to get everything to a place where you could visually see what was going on around you. I wouldn't say it's pretty, but it's functional at least. There's a header, a menu, and a footer. Everything is responsive using CSS grid and flexbox, and it's good enough for now, for sure.

Like I mentioned in my proposal, Tailwind.CSS sounds really neat and that's what I want to use for the final version of my site. But I'm going to wait for now until we go over that more in class, so I can see how it works and maybe get a feel for it in Doggr before I try it out on my own. I don't mind writing a little CSS on my own for now anyways, at least I'm a little bit familiar with that already.

This was a fun week for me all things considered. It was busy but it was really special seeing my site go from just some ugly backend routes that I was testing in my test suite and with Postman, to an actual visible site that you could click around on and use.

Week 5:

With a very basic Frontend up and running, this week my goal was to integrate my Frontend and Backend routes. I didn't have all the routes I would need for the final site yet, but the ones that I had, I wanted to get them fully working together so that the Frontend made the correct calls to the Backend and I could get all the information I needed.

After getting Axios set up, the initial frontend calls that I made in my components were pretty easy. I love when things are easy! All of my requests got me the data I wanted and I was able to add it to the pages without too much trouble. The character display page was working great and getting characters like it was supposed to.

The search page was a beast though. Every time I tried to make the SEARCH request, it would fail because of CORS. I ended up spending almost the whole week just on figuring out how to make the SEARCH request go through. More than a few times I just thought about ditching it and returning my data from a different type of request, but I didn't want to give up I wanted to solve the problem! In the end I had to figure out a whole mess of CORS stuff and spent way too many hours working on changes to the frontend and backend. But I did get it working!!! Now my search page gets the search results successfully using Axios and a custom SEARCH request. No more getting shut down with error messages. I definitely was relieved that I got that over with. I wish we'd done some of that stuff on

Doggr lol I didn't know when we set up the SEARCH method in the backend that it was going to be such a nighmare when we got to the front.

Now all of my routes work. I can get, create, and search for characters using the frontend interface. Things are starting to really come together. The only big problem left is that my frontend interface can only create or update a single static character in a pre-determined way. So I will need to put some serious work into those components next week.

Week 6:

I spent a lot of time on the character creation and updating forms this week. I had to use typescript to handle the forms. I also spent a lot of time trying to make the forms work intuitively, where it would fill out each field with a description or with "sample text" which would disappear when you clicked on the form. There were a lot of cases and this was a lot harder than I thought it would be. But I was excited to get everything working in the end.

If you try out the forms I think you'll see that they work pretty intuitively. Every time you click on a form element, the default text disappears and is replaced with your cursor. If you delete everything that you type in the field, the default text returns. And even though each form has a "submit" button, you can also just press Enter from any point in the form to submit the form.

(It was only after getting all of this working that I suddenly wondered if I should have found some sort of package or plugin to help do forms! Oh well it was a great learning experience.

I also noticed this week that I had ended up with a lot of saved state variables by this point. Some of them were redundant and unnecessary, and others were just unused. I was also able to use some of my extra time this week to refactor an simplify some of the Frontend logic in order to reduce the amount of redundant saved state variables.

Week 7:

Made a bunch of big updates this week!

I added a new Frontend and Backend route for Featured Characters. This is a special route where some characters are labeled as "Featured" and their profiles show up on the home page. This is a great way to show off profiles or characters that are special in some way, or to give people recognition. It as also the perfect component to add to the home page which was looking a little empty and which I thought needed something special.

I had to add the backend route and the new database field and migrations for this. It was pretty painless though after all the work I did on the database and migrations in previous weeks. Was excited to not run into any unforseen issues with migrations this time. And had to create the frontend component to get the featured character data and to display it the list of featured characters.

After I had the Featured Characters working I was able to finish the design of several of the pages of my site. First I finished designing and implementing the homepage. In the current ersion, there is a welcome message which gives a description of the site, followed by a search module, and then

followed up by new component which displays the brand new list of featured characters. This is pretty much just how I want it to work.

I finished search results page as well this week, so that each character in the search results correctly displayed the character's picture and name, and a link which would take you to the profile page for that character! I'm glad to have the search results page working too, because searching is one of the most important parts of the site, and the search module is featured not just on the Search page but on the Home page as well. So that's two of the most important main menu pages done!

Week 8:

I was sick this week. :(

Week 9:

This was the AUTH week for me! Wow, looking back at this journal, I thought AUTH was easy in week 1 but I just had no idea how hard it would be! Back then I thought I just had to set up the links and get the token and save it in the frontend. How naive.

Feeling better after being out for a week, I finished getting caught up on all the videos and learning how the authentication is supposed to work. Now I could see how authentication worked from the frontend and the backend, and I udnerstood that I didn't just have to take care of the frontend authentication. I also had to take care of the authorization in the backend routes, to add token verification and decoding and user verification from the decoded tokens. This took allllllllll week In the end I used fastify-jwt-verify to lock down the backend. And I got the frontend to send the bearer token with its requests so that it could get permission to make the requests!

This was the most horrible and longest week but it was also the most fun and I learned a lot. The biggest hassle here was just the callback page. For the life of me, I could not figure out how to get the token directly from the callback. There must be a way, there just must. But I think I might have just used all of my brain cells up already, because I couldn't figure out how to get the token. So I had to use a separate function to get the token. Luckily there was a function "GetAccessTokenSilently" which promised to get me the token whenever I needed it. So I just included that function in the callback page, and once I got to the callback page I just called that function to get the token in a format where I was smart enough to figure out how to access it.

Unluckily, when I tried using this function, it didn't work. I just got an error on the frontend. After some more research, I discovered that the documentation said that that function doesn't work when your host is localhost. And of course that's where I'm working. SO I had do do an annoying thing and use an alternate function which got the token but which triggered a popup window. This function was called "GetAccessTokenWithPopup" instead of "GetAccessTokenSilently." This will be unnecessary in production, but if you are testing on localhost, you will need to expect a popup to confirm that you are logged in, on the callback page, after you are logged in. On localhost, I just made my site print out the access token on the callback page so that you could visually make sure that you got the token successfully. It does work great though!!! Once you confirm the extra little popup, my app gets the token and saves it in a nice little cookie, and everything proceeds after that point according to plan.

Week 10:

This week I finished all of the remaining pages! Now that authentication was working, I finished adding all of the menu logic. I successfully updated the Log In and Log Out and My Profile page links in the menu, setting them up so that the correct links show up depending on whether the user is authenticated or not.

Then I finished the profile page which allows you to Create and to Update your characters. I realized that I needed the page to show the Character Creation component when the user didn't have a character created yet. And I needed the same page to show the Update Character component when the user already had a character. So I quickly created a new backend route to solve this problem. Now the component sends a request with the token for the authenticated user. The response returns false if the user does not have a character, and the response returns true and returns the user's character ID if the user does have a character. This allows me to load the correct component for the user's current situation. This was a pretty quick fix thankfully, I realized that I was becoming a lot more familiar with the process of creating pairs of backend routes and frontend components.

This week is also the week that I did the most major refactoring of the code. Since my code split from Doggr fairly early (Week 2) I had a lot of catching up to do. While I had implemented most of the things we learned as we went, there still were a few things I had missed. Doggr had undergone a major refactoring where components and routes were split into separate files and everything had been cleaned up and made a lot more intuitive and manageable.

So that was the biggest thing that I had to do too. I got started putting my frontend components and backend routes into their own separate files. I had known for a while that my Components.tsx and my routes.ts were just becoming a giant mess. Like way over a thousand lines, one of them. So I was already having trouble finding stuff that I was looking for at times, and I was really looking forward to these changes. After organizing and cleaning everything up and putting everything in well-named files and modules, everything still worked great and it was 1000% less overwhelming trying to find the right code that I needed. I wish I'd done this part sooner for sure. It's just like cleaning in real life, I guess I always procrastinate it 'til the end.

Note: I also did something else really horrible this week: I really wanted a custom rich text editor to allow people to format their character descriptions without having to, like, type their own HTML into their character description box. I worked on this for hours and hours, using draft-js. I finally got a few things working, the editor would appear on the page and you could type stuff in it. But it was just a mess. I mentioned this in my "Future Goals" section of my presentation video. There's also a branch in my Github called "text-editor-mess" if you're curious. I have the Draft JS Rich text editor working and you can type in it. And I know how to load and display the editor state in the public character page to display the content. I just need to finish figuring out how to integrate it into my forms so that I can actually save the data to the database. One small piece missing but it's just been a nightmare! When I read about Draft JS It really sounded like it would work out of-of-the-box. But Holy Guacamole it sure didn't.

Week 11:

Woooooo finals week! This week was just tidying up finishing all the last little bits and pieces.

I have been working on my own Linux install all term because the virtual machine was just so slow on my computer. So my main goal was just to make sure my code would work when it was graded! So I booted up the VM again and downloaded my repo, and made sure everything worked. I'm glad I checked because there were some errors at first, but once I updated the version number of a few packages and remembered to copy my .env files, everything did work great on the VM! So I am confident that if you docker compose up on the VM now, it'll work for you just like it worked for me.

Once I had the VM working, I recorded my presentation and then just came back here to finish up the last part of my journal here. I'm feeling pretty good. I'm going to write a few more little instructions in a README file in my project just to be safe, but besides that, now there's only one thing left on my to-do list.

Multiple Characters Per User

This is the last part of my project that I need to do. I explained how I was still working on it when I recorded my presentation video, and that I would have it finished before I turned in the final version of my program. In order to support allowing each user to have multiple characters, I updated my frontend and backend routes so that multiple characters could be both created, updated, and displayed on the site. Thankfully I already had the database set up so that this would be an easy change. I just needed to update the user search route to send back a list of all the characters instead of one. And to update the Profile page so that it showed a list of all of a user's characters. Now that's finished! I tested all the new features and they work great no matter how many characters a user has. So, I guess I'm done!!!

My presentation video is at:

https://youtu.be/4Bd1iFxa6Dk

My project repo is at:

https://github.com/pearstopher/465-Project

(And the very basic instructions to run my program are in the README file which you'll see as soon as you go there.)

I will be uploading this journal to canvas alongside a copy of my project files, another link to my repository, and another link to my presentation video as well.

This has been such a crazy class. I learned so much, and I definitely cried a few times haha. Thanks for reading and for watching my video, and have fun testing things out!

-Christopher Snow