Week 4

I shared SSH credentials for the Hetzner server with my teammates.

I installed Node.js on the server and documented the steps in the README.

I got an Express.js "hello world" app running on node and exposed to the public internet on port 8080.

I learned that port 80 is what we need for HTTP traffic and found that Linux restricts access to those ports for non-root processes. I informed my teammates of the issue.

I learned that logging out of SSH will cause processes created in the shell to be killed, but we need the OnlyBruins server to keep running after we log out. I wrote a shell incantation using `nohup` that would keep the Node process alive (it uses `sudo` to work around the port 80 issue).

I will find a solution to the Linux privileged ports issue.

I will work on serving the frontend HTML and JS from the server.

Week 5

I worked on database and API design with my team.

I aimed to create RESTful endpoints, where resources are described hierarchically and accessed with appropriate HTTP verbs. For example, GET /users/:username/posts should return a list of the user :username's posts, and POST to that endpoint would create a post for that user.

I also advocated for HATEOAS in server responses. For example GET /users/:username/posts/:postid should return information about a post, including an endpoint to be used for tipping that post.

I will continue the database and API design next week. One problem is designing a good resource URI for the tips on a post.

Week 6

I continued work on the database schema. I aimed for extensibility by using normalized tables. I also included a timestamp column in many tables, besides the necessary one in the posts table, because I think it would be cool to collect data on OnlyBruins and see trends over time.

My solution for the tipping problem is the endpoint /users/:poster_username/posts/:post_id/tips/:tipper_username. HTTP PUT to this resource will tip the post, and HTTP GET will return the value tipped, if any.

I will work on more of the implementation of these ideas next week, since we're in a good place with planning.

Week 7

I fixed a bug in a query that was susceptible to the N+1 problem, where the server would make one query to get a list of user IDs and then one query to find the username of that ID, for each of those IDs. The fix was to JOIN two tables and return the list of usernames in only one query.

I migrated existing queries to use a safer way to query the database that wouldn't leak DB connections if an exception occurred.

I implemented an API endpoint to list the users who are following a given user. In line with the RESTful API design, this endpoint is /users/:username/followers.

Next week I will work on image uploading, which is core functionality for our app. Michael did some preliminary research on the API we need to use (files are often sent using an HTTP form), but we don't have working code yet.

Week 8

I worked on the server-side requisites for our images. We needed to send images through Express from the server's file system to the client. It's possible to use dummy images for testing this, whereas testing image uploading is annoying to test without image viewing (you'd need to scp the images to view them since the server is headless).

We decided to follow Git's example in storing files by their identifier, where we distribute the files by the first two characters of their GUID. I learned an interesting trick to create an "empty" directory in Git, which tracks only files: by putting a .gitignore in the directory, it will be tracked without any other content inside.

I got image viewing working and next is to implement image uploading, so users can actually make posts to OnlyBruins.

Week 9

I worked on image uploads to the server. Michael had already made a nice React component to select an image, I needed to craft the HTML form and receive it on the server. Multer is a useful library for this task because it can write images to the filesystem for you - all I had to do was give it a file path based on an image's GUID.

Next week I will work on other features that we need on the frontend, including tipping posts and notifications. These will be less technically complex than the images but of course it still needs doing.

One problem I am thinking about is how to implement notifications on the backend, since we don't want to give the user the same notifications multiple times (this can range from annoying to outright incorrect, if we claim they got bruinbux twice when it was only once).

Week 10

This week was a whirlwind of features. My part was the tipping and notifications, which both needed changes to the database schema and several new API endpoints.

I am proud of my notifications design, whereby the server tracks the last time a user checked their notifications and only returns the notifications that have been made since then. The whole notifications system came together nicely on the backend since I'd build up a good amount of experience writing SQL queries and endpoints by then.

We got everything together and prepared our presentation for demo day.

Post-demo day report: it went great! Our server didn't have any showstopping issues that brought the website down, and we got our first dozen users 🙂