Report 2

Viktor Enghed

February 26, 2021

This iteration focused mostly on implementing the things we had skipped over during the first iteration. This included authorization, transactions and data migration. We divided this work between us and discussed the different solutions available to us after experimenting with them for a few days.
For authorization we chose to use JWTs as this is a powerful way of keeping track of a client between requests without the need of a state in the backend, which we want to avoid building a REST api. At first, we planned on using access tokens and refresh tokens with expirations to keep track of each user, but one could skip using the refresh tokens and instead keep track of how long the user's login session is active by keeping it in the user's row in the database. A column was added to the user tables which contains a timestamp. This timestamp is set to one hour in the future when the user logs in using the login endpoint or accesses the backend using his or her access token. The decoded token contains the user's username and the user's role, so as to avoid an applicant accessing the employer functionality on the site and vice versa. We also implemented, in the front end, routing so that when a applicant logs in, he or she will be routed to the make-application page and employers will be routed to the search-for-application page.
For transactions, we were not clear at first what it meant, but we chose to think of it as a mutex when working with multithreaded programs accessing the same resources. We need to make sure that each transaction is atomic and that if anything fails, we do a rollback as if nothing happened. This could be done quite easily as we are using sequelize, which have this functionality built in. However, as we are using promises and not async/await the code became a bit hard to read; we ended up in "recall hell". We are thinking about changing this, but are not sure yet.
As for data migration, it is still a little bit of a work in progress. As we are using postgres, we can easily copy the sql code from the project description and use it but there is some data missing here and there. We could enable using email as a login or send an email out to all users with missing info, there are options but nothing has been set in stone yet.