Authentication part

Look at front-end and see what we need to login

Code trace for front-end, see how it connects to back end

From this, see what fields are needed for backend db

Figure out what queries we need for back end

Need database tables and their names, the data they hold

Populate databases with mock data

**MongoDB:**

Users

* Added 5 users with the data fields:
  + Id
  + Name
  + Email
  + Address
  + Postcode
  + Balance
  + Password
* Chris will be the user that is far from the proximity of the other users.

Jobs

* Changed ‘location’ field to ‘address’ for better synergy with queries.
* Job Status: {job.jobStatus === 1 && "Listed Job"}
* {job.jobStatus === 2 && "Applied"}
* {job.jobStatus === 3 && "Active"}
* {job.jobStatus === 4 && "Completed"}
* Adding additional ‘jobs’
* We need to devise some kind of metric for how much a job should cost. Maybe the estimated time taken + effort?

Research what post requests are

Having problems with the fetch-post part

Data fields:

Users:

* Id
* Name
* Email
* Address
* Postcode
* Balance
* password

Queries and their use cases:

Check the proximity between users

Select postcode.a, postcode.b from user

15/08/2021 Action Items:

* Research ‘role’ field from front-end.
* Mongoose: how to get collection and how to display into the model
* How to put the collection into a model schema; how to put the collection on mongodb into a schema
* Try connection js file to db using mongoose + mongoose research
  + Try and retrieve collection

Other notes

* All the controllers and models are organised into different folders.
* For routes: routing controls whether there’s a call or fetch to a specific URL.
  + E.g. under index.js if we look at the code that fetches from /login (the POST method). The router for /login will see if there’s a method going to the URL (/login), fetches the information from the server, and routes to the login function in the controller. In the login example, the request in the controller contains the name and password in the body field.
* We were having trouble with login. When we click the submit button on the login page it’s a POST request, it’s sending a request to the server (localhost 3200).
* Displaying jobs on the home page wasn’t working because it wasn’t getting jobs, it was displaying

19/08/2021:

* Set postcode field to required when there was no value for it, hence the failures.
* Need to add more error handling

List of front end tasks:

* For registration, need to add in postcode and anything where there’s and API
  + Graphical user interface, text, application

    Description automatically generated
  + In swapstreetlogin.controoler:
    - Checks if user exists
    - Checks if password is for the user
    - Every user must have a jwt (json webt token)
  + Change jobs in homepage.js to read from schema
  + Every Graphical user interface, text, application, chat or text message

    Description automatically generated