* [Link to assignment](https://fullstack.instructure.com/courses/1059/assignments/56586?module_item_id=339802)
* Needs to be handed in by 5pm EST on Wednesday.
  + “I would recommend working on this career simulation and submitting what you have at the end of the day. You can always work on it more tomorrow and push up the code then.”
* Breaks
  + Take a 5 minute break for every 25 minutes of work. Take a 20 minute break for every two hours of work.
  + Started
    - 10:12 AM
  + Breaks
    - 10:59
      * 5 mins
    - 12:58
      * 20 mins, bank 5 mins

1. ~~Make a planned order of what I’ll do~~
2. [~~Make a mock-up plan with drawSQL~~](https://drawsql.app/teams/fullstack-20/diagrams/restaurant-reviews)
   1. ~~Three tables~~
      * ~~Restaurants~~
        + ~~Relationships~~
          - ~~One restaurant can have many reviews~~
        + ~~Columns~~
          - ~~Id primary serial~~
          - ~~Foreign key to reviews~~
          - ~~Etc think about more in drawsql~~
      * ~~Reviews~~
        + ~~Relationships~~
          - ~~One review can only be about one restaurant~~
          - ~~One review can only be by one user~~
        + ~~Columns~~
          - ~~Id primary serial~~
          - ~~Score~~
          - ~~Review text~~
          - ~~Created at~~
          - ~~Edited at~~
          - How do I note that there is a unique combination constraint on the user\_id and restaurant\_id foreign key columns in drawSQL?
      * ~~Users~~
        + ~~Relationships~~
          - ~~One user can have many reviews~~

~~A user can only have one review per restaurant~~

* + - * ~~Columns~~
        + ~~Id primary serial~~
        + ~~Foreign key to reviews, must be unique~~
        + ~~Username~~
        + ~~Password~~
        + Token?

I don’t think that’s in the db

* + - * + ~~Etc think about more in drawsql~~

1. ~~Set up basic backend files~~
   1. [Create a fullstack application](https://docs.google.com/document/d/1fGxVK61IntjflKWK5bpYxZnfvjFPQYp_0bYM4XVMTAQ/edit)
   2. ~~Don’t forget to add a .gitignore file manually and add in your node\_modules and .env file~~
      * DATABASE\_URL in .env didn’t work in client. Watch video about how to do .env and and come back to this
2. ~~Make a github repo and do initial commit~~
3. ~~Make branch for new feature~~
   1. ~~Create tables~~
4. Think when setting up the following: will your user data be secure so that no one can maliciously or unintentionally manipulate the information?
   1. Review stuff from yesterday
5. ~~Make branch for new feature~~
   1. ~~Seed restaurants~~
6. ~~Make branch for new feature~~
   1. ~~Seed users~~
7. ~~Make branch for new feature~~
   1. ~~Seed reviews~~
8. ~~Make server~~
   1. [Create a fullstack application](https://docs.google.com/document/d/1fGxVK61IntjflKWK5bpYxZnfvjFPQYp_0bYM4XVMTAQ/edit)
9. ~~Make branch for new feature~~
   1. API- anyone can get all restaurants with these details:
      * ~~Name~~
      * ~~Cuisine type~~
      * Average score
        + Come back to how to correctly apply average score to each restaurant in array. I’m guessing I’ll map through the array, band then the big question is how to match up the restaurant id in the review with the correct restaurant in the array.
10. ~~Make branch for new feature~~
    1. API- anyone can get one restaurant’s full details
       * ~~Name~~
       * ~~Cuisine type~~
       * ~~Average score~~
         + ~~In reviews.cjs, make a function that selects all scores for a restaurant. That should return an array of objects like this. { score: X }. Reduce through the array and divide the total by the length of the array.~~
       * ~~All reviews~~
         + ~~Add to restaurant object~~
           - ~~objectName.newKeyName = value~~
         + Come back to this to attach username to review?
11. Make branch for new feature
    1. API- register, i.e., create user. What was route and API request for bookbuddy?
12. ~~Make branch for new feature~~
    1. ~~Encrypt/decrypt passwords~~
       * Steps to encrypt when creating a user
         + npm i bcrypt
         + Import into users.cjs with back-end syntax
         + *const encryptedPassword = await bcrypt.hash(password, 10);*
         + Insert *encryptedPassword* into database record
       * Steps to decrypt when authenticating log in credentials
         + Make async, await, try-catch getUser function in users.cjs
         + Check 1: SELECT user FROM table WHERE username matches, deconstructing the result.
         + Check 2: Can’t check log-in password in table because it’s encrypted, so check that with:
           - *const isPasswordMatch = await bcrypt.compare(password, user.password);*

We have *user.password* from the user object we deconstructed from the client.query in the previous step

* + - * + If we get a true result from both, i.e., *if(user && isPasswordMatch)*, we authenticate the log in attempt

1. Make branch for new feature
   1. API- log in. What was route and API request for bookbuddy?
      * Steps to authenticate user in backend from API request
        + Need *app.use(express.json());* near top of file in server.cjs to be able to make json objects sent into server readable for req.body
        + Post request with body in postman
        + req.body in app.post to get request
        + Destructure body object and use values as arguments for function you imported from users.cjs and called to authenticate the user
        + Return if user matches
      * Steps to give a JWT upon log in that travels with user around site as proof of authentication after they've logged in
        + *npm i jsonwebtoken*
        + In users.cjs, *const jwt = require('jsonwebtoken');*
        + Inside function definition that authenticates user, the true condition of *if(user && isPasswordMatch)*, is to create a token variable and return it
          - *const assignedToken = await jwt.sign({* payload, typically userID, must be stuff that is safe for anyone to see *},* secret*)*

*return assignedToken;*

* + - * Then, in server.cjs, in your app.post to log in, create an *assignedToken* variable here too to take the result of that function you imported from users.cjs and called to authenticate the user
      * *res.send(assignedToken);* to send token to frontend from server upon proper API call

1. Make branch for new feature
   1. .env for JWT secret PICK UP HERE
2. Make branch for new feature
   1. API- logged-in user can get all reviews they created
   2. Require token 12:59? How to handle if no user exists and inform user around 11:45
   3. .env?
3. Make branch for new feature
   1. API- logged-in user can create review
   2. Require token 12:59? How to handle if no user exists and inform user around 11:45
   3. .env?
4. Make branch for new feature
   1. API- logged-in user can delete specific review they created
   2. Require token 12:59? How to handle if no user exists and inform user around 11:45
   3. .env?
5. Make branch for new feature
   1. API- logged-in user can edit specific review they created
      * Edit score and review text
   2. Require token 12:59? How to handle if no user exists and inform user around 11:45
   3. .env?
6. Double check: do you have secured user data so that no one can maliciously or unintentionally manipulate the information?
7. Compare to requirements
8. Clean ups
   1. Console logs
9. When you're finished, submit a GitHub link that contains the following
   1. This plan
   2. An image or a link to your complete database schema
   3. All the relevant code and commits for the project
10. Make branch for new feature
    1. Stretch Goals
       * Access the website via the internet to browse and read reviews.
       * Write comments on reviews that others have written.
       * Observe a list of all the comments this user has written.
       * Edit and delete comments this user has written.

Things to improve at