Outline of Project Work

Expectations:

1. React Pose to React Framer Motion migration (frontend)
2. Update depreciated TypeORM function findOne() → either find() or findOneBy()

Roadblocks + Solutions

(roughly in chronological order)

1. Documentation for starting development was incorrect/outdated
    a. Solution Steps:
        i. Checking package.json file
        ii. Unable to find previously used scripts mentioned in documentation
        iii. Found default scripts
        iv. Used default scripts instead
            1. *they did not replace complete functionality, but for the purposed of development it was enough for us :)
2. Unable to start development on windows machine
    a. *this wasn't thoroughly investigated as we only had one windows machine available to us, however all development progress was only possible via machines running macOS
3. Project unable to compile (known/expected issue)
    a. Our client informed us this was a known issue, and was expected to resolve via migration from Pose to Motion
4. Lack of documentation for migrating from Pose → Motion
    a. Solution Steps:
        i. cmd+F to find all uses of Pose
        ii. replaced with motion
        iii. Fixed resulting syntax errors
5. Project Compiles but does not function as expected
    a. Solution Steps:
        i. Investigate any possible error outputs via inspect elements
        ii. Discover that project is unable to make calls to server
6. Lack of documentation of project's serverside
    a. Solution Steps:
        i. Investigating files to gain understanding of overall structure
        ii. After describing issue to coach, discover that a change in .env variable is able to produce a different error
        iii. Determine that the serverside app starts with the same command as the clientside app but the code itself produces some error so that there is not expected functionality
7. Discovered many (other) depreciated functions/code

    a. Solution Steps:
        i. Via research, update all depreciated code and VSCode errors
        ii. In the process, updated depreciated TypeORM function syntax
    b. *it is believed that the depreciated redis code is the reason for the errors but this is unconfirmed as we were unable to fully update this

8. Serverside still does not have expected functionality
    a. Solution Steps:
        i. Discuss hurdles and overall progress with client
        ii. Mutually determine that project has a larger, overall issue with a lack of documentation
        iii. We decide to prioritize documentation of current known issues, and update documentation for future development

Development Steps

(the process, outline for future development. This is a modified version of the text originally found in the readme)

```
## Contributing to ShiftOverflow
Note:
- Most of the instructions below applies to MacOS users, for windows users, please checkout the seperate section for more guides.
- Since Fall2020, COMP110 switched to Python setup, so please make sure you have Node.js and Typescript installed with the correct version as specific in `package.json` .
0. Currently `brew` is recommended for installing dependencies for MacOS.
1. Clone this repository: `git clone https://github.com/comp110/shiftoverflow.com.git`
2. cd into this repository: `cd shiftoverflow.com`
3. Install dependencies: `npm install`
4. Install redis: `brew install redis`
5. Start the local development server: `npm run start`

## Setting up MySQL for ShiftOverflow use
1. Install MySQL, if you have `brew` installed you can do `brew install mysql`.
2. Run MySQL, either `brew services start mysql` or `mysql.service start`
3. In the shiftoverflow.com directory, run: `mysql -uroot`
4. Create the shiftoverflow database: `CREATE DATABASE shiftoverflow;`
5. Create the shiftoverflow user: `CREATE USER 'shiftoverflow'@'localhost' IDENTIFIED WITH mysql_native_password BY 'shiftoverflow';`
6. Grant shiftoverflow permissions: `GRANT ALL PRIVILEGES ON shiftoverflow.* TO 'shiftoverflow'@'localhost';`
7. Exit by typing `quit`
```

```
## Making yourself an admin for developer use *EXPECTED FUNCTIONALITY IF SIGN UP WORKS*
1. Run the local server using `npm run start` after installing, then go through the Sign Up interface to create a user. This email and password will be what you use to sign in for local development. You should use invite code `110rulez`.
2. Go into mysql, run `mysql -u shiftoverflow -p`, the password is `shiftoverflow`
3. Run `use shiftoverflow`
4. Type `UPDATE membership, person set isAdmin=1 WHERE personId=person.id AND person.email='<email>';` where <email> is replaced with the email you signed up with in step 1.

### If you screw up your database
It's okay, it happens to everyone :).
1. Open mysql with `mysql -u shiftoverflow -p`, password `shiftoverflow`
2. Run `DROP DATABASE shiftoverflow;`
3. Run `CREATE DATABASE shiftoverflow;`
4. Exit with `quit`
5. In your terminal, type `npm run populate-mock-data`
6. Good luck!

### Windows Guide
- Instead of `brew` for MacOS, please use `chocolatey`, download [here](https://chocolatey.org/install)
- `choco install redis-64` to install redis
- `choco install mysql` to install MySQL
- To start MySQL, use `"C:\tools\mysql\mysql-8.0.23-winx64\bin\mysql" -uroot`, make sure to replace with the correct directory of MySQL on your laptop.
- In `package.json`, add a new script to start redis server: `start-redis: <route to your local redis>.redis-server.exe` (eg. `/c/tools/Redis-x64-3.0.503/redis-server.exe`)
- In `package.json`, replace `redis-server` in `dev-local` script to `npm run start-redis`
```

Current Known Issues/TODOS

1. npm run populate-mock-data does not work
    a. The file exists but the old comment to run the script doesn’t work, and there is not currently a replacement for it. Since the code exists the only necessary thing is to connect a command to the script.

2. In server/server.ts lines ~190-214 VSCode notes a few errors such as on req.logout(), req.isAuthenticated, req.user, etc…
    a. server/custom-req.d.ts attempts to fix this issue but failed
    b. This issue doesn't actually affect the functionality of the project, and as of now it does reflect the correct usage of the express/passport functions.
3. Development Server is unable to make server side API calls
    a. Usually creating a timeout, uncaught exception, or 404 error when viewed in the browser console.
    b. It is believed to be caused by the redisClient calls using legacy syntax from v3 of the node redis package. (comments mention this on line 63 and 78)
    c. deleteSessionKey and disableAllUserSessions in server/server.ts also had this issue but they're syntax was updated. It is unconfirmed if they were updated correctly so the original legacy syntax remains in a comment.
4. 'https://shiftoverflow.33.33.33.14.nip.io'
    a. If the project is started without the line "REACT_APP_LOCAL_DEV=true" in .env.development then sometimes the app with attempt to make calls here. We don't know what this URL is.
5. Deployment to CloudApps
    a. The older version of the app is deployed on wordpress and should be updated to UNC CloudApps. This requires a namespace that does NOT belong to a student, otherwise it will disappear after graduation. As our team is all seniors and weren't able to get another namespace, we did not attempt deployment.
    b. (to future devs) You probably want to figure out the serverside issues before deploying this.
6. Other Pages of the App are unreachable
    a. Because the calls necessary for signing in to the app do not work, we were unable to verify whether the other pages render correctly. Based on the login page, they probably do, but in case they don't -- these pages may not render correctly. The cause is probably the implementation of motion. There are many ways to use motion and styled, and the current way may not be correct, so check that first.