**Questions:**

* What is the purpose of your application? What does it do?

The program is a text-based social media site akin to Threads or X. Users can write something out and send it to the server, with a choice to make it public or delete it after having posted it. Along with that, users can view all the public tweets of any user updated in real time on the public feed.

* How are you using React?

React is used in the project for any element that could change based on the user or user input. If an element is static like the 404 page or the navigation menu it is just built into handlebars.

* + What components do you have?

Login form

Signup form

Password changing form

Account information display

Tweet writing form

Sponsored tweet

User tweets

Public tweets

* What data are you storing in MongoDB?

User accounts that store: Username, password (hashed), whether they subscribe to premium, and the date they were created.

Tweets that store: text body of the tweet, the id of the account that made them, whether they are public, and the date they were created.

* What went right in the development of this project?

One thing that went well throughout the project was all the various types of database operations. Even operations I hadn’t used before such as delete, I was able to get implemented without the hassle that some of the database assignments presented.

* What went wrong in the development of this project?

A couple times throughout the project I had some async functions that had a chance to be called out of order of one another. Eventually I got them all tracked down and fixed, but it proved a challenge for some of them as they wouldn’t cause an error instead just leaving the output just out of sync with what was expected.

* What did you learn while developing this project?

I learned quite a bit over the course of this project. Getting more experience with sockets helped me better understand the call and response nature of its events. Along with that delving into a mostly react front end for a whole project for the first time taught me a lot about the different functions of react that I had never used before (including in-line event listeners somehow).

* If you were to continue, what would you do to improve your application?

One big thing I would want to continue working on is the premium profit model. Right now, it only serves to give a user a symbol next to their name so that they stand out. While this still serves a purpose as many people on the internet, for various reasons, have shown that they would pay for something just for the recognition of having it, if I had more time, I would have wanted to give more benefits for making the “purchase.”

* If you went above and beyond, how did you do so?

I did a few things to go above and beyond. Firstly, I implemented Socket so that the public feed updates in real time with any changes a user makes to a tweet or their account. Secondly, I used a CSS framework to create a more professionally styled website (I used the same framework as project 1 so it was a little easier to use for this project). Thirdly, I looked at how to send delete calls to the database to allow for users to delete tweets. Lastly, I implemented two different profit models, one ad based, with a sponsored tweet appearing at the top of each tweet feed, the other subscription based, with users able to pay to set themselves out from other users.

* If you used any borrowed code or code fragments, where did you get them from? What do the code fragments do? Where are they in your code?

See “Sources” section.

**Endpoints:**

URL: /getTweets  
Supported Methods: GET   
Middleware: Requires Login  
Query Params: N/A

Body: N/A  
Description: Retrieves all of the current users tweets  
Return Type(s): JSON

URL: /getAllTweets  
Supported Methods: GET  
Middleware: N/A  
Query Params: N/A

Body: N/A  
Description: retrieves all tweets from every user  
Return Type(s): JSON

URL: /login  
Supported Methods: GET POST  
Middleware: requiresSecure, requiresLogout  
Query Params: N/A

Body: { username: username of account trying to log in,

pass: password of account trying to log in }  
Description: get will return the login page, post will log the user in  
Return Type(s): HTML, JSON

URL: /signup  
Supported Methods: POST  
Middleware: requiresSecure, requiresLogout  
Query Params: N/A

Body: { username: username of account trying to create,

pass: password of account trying to create,

pass2: 2nd copy of password }  
Description: Will create a new user account if both passwords match and the username has not been used  
Return Type(s): JSON

URL: /logout  
Supported Methods: GET  
Middleware: requiresLogin  
Query Params: N/A

Body: N/A  
Description: will log the user out and return them to the login page  
Return Type(s): JSON

URL: /changePassword  
Supported Methods: POST  
Middleware: requiresLogin, requiresSecure  
Query Params: N/A

Body: { oldPass: current password,

newPass: new password to use,

newPass2: 2nd copy of new password }  
Description: post will change the password of the user as long as old password is correct and new passwords match  
Return Type(s): JSON

URL: /account  
Supported Methods: GET  
Middleware: requiresLogin  
Query Params: N/A

Body: N/A  
Description: will return the account page  
Return Type(s): HTML

URL: /accountInfo  
Supported Methods: GET  
Middleware: requiresLogin  
Query Params: N/A

Body: N/A  
Description: returns username and creation date of the user  
Return Type(s): JSON

URL: /getPremium  
Supported Methods: POST  
Middleware: requiresLogin  
Query Params: N/A

Body: N/A  
Description: updates the premium bool of the users account to true  
Return Type(s): JSON

URL: /cancelPremium  
Supported Methods: POST  
Middleware: requiresLogin  
Query Params: N/A

Body: N/A  
Description: updates the premium bool of the users account to false  
Return Type(s): JSON

URL: /tweet  
Supported Methods: GET POST  
Middleware: requiresLogin  
Query Params: N/A

Body: { content: text content of tweet }  
Description: get will return the main tweet page post will create a new tweet on the database  
Return Type(s): HTML JSON

URL: /togglePrivacy  
Supported Methods: POST  
Middleware: requiresLogin   
Query Params: N/A

Body: { id: \_id of tweet }  
Description: Will change the privacy bool of a given tweet to whatever it is not currently  
Return Type(s): JSON

URL: /deleteTweet  
Supported Methods: POST  
Middleware: requiresLogin  
Query Params: N/A

Body: { id: \_id of tweet  
Description: will delete the given tweet from the database  
Return Type(s): JSON

URL: /  
Supported Methods: GET  
Middleware: requiresSecure, requiresLogout  
Query Params: N/A

Body: N/A  
Description: will send the user to the login page  
Return Type(s): HTML

URL: /\*

Supported Methods: GET

Middleware: N/A

QueryParams: N/A

Body: N/A

Description: will send user to 404 page if any not supported endpoint is called

Return Type(s): HTML

**Sources:**

* Starting code was taken from domo maker part e
* Gulp implementation was taken from the sass/gulp demo code
* Favicon: <https://fontawesome.com/icons/comment?f=classic&s=solid>
* Premium Check: <https://fontawesome.com/icons/square-check?f=classic&s=solid>
* Searching for javascript functions [W3School](https://www.w3schools.com/) and [MDN Web Docs](https://developer.mozilla.org/en-US/) used these sites to find functions for various different tasks in my code such as reversing the array of tweets in app.jsx
* Making the tweet column scrollable: <https://www.geeksforgeeks.org/making-a-div-vertically-scrollable-using-css/>
* Vertically centering 404 page: <https://stackoverflow.com/questions/44897794/how-to-vertically-center-elements-in-bulma>