**Q&A:**

1. What is the purpose of your application?

The purpose of this application is to allow users to store their different opinions on food and see other people’s likes and dislikes when it comes to food.

1. What data is being stored and delivered by the API?

* Users name
* Array of disliked food
* Array of favorite main dishes
* Array of favorite deserts/sweets
* Array of favorite drinks

1. What went right in the development of this project?

The actual code to send and receive data from the API went smoothly. I was able to get the bare bones up and running in a few hours. After that I was able to get all the checks in place on the client and server without any real hangups.

1. What went wrong in the development of this project?

One thing that held me up for a while was the log in system. When I first went about implementing a system to log in, I was working on the client-side code. However, I realized after having spent a day getting the system to work that since each html file was going to get their own version a shared file that I needed to move everything to the server side. This then led to a bunch of issues with setting up the code to wait for the username so that it wouldn’t try to run anything before it had been set up.

1. What did you learn while developing this project?

I learned quite a lot through this project. I grew my knowledge on how to structure asynchronous code so that you can’t use a variable before it’s set. I learned how to redirect to different pages and use multiple webpack bundles. I feel that I was able to better structure my response code on the server so that the layout makes more sense.

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

There are two things I think I would look to do if I continued. One thing is setting up CSS so that the website is mobile friendly. Another thing I would do is set up the API to be able to add individual items in a user’s different information sections without replacing the entire section.

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

I did two main things to go above and beyond in this project. Firstly, I re-learned how to use Bulma so that my website would look good. Secondly, I setup a suit of tests in postman to run tests on every endpoint of the server/API.

A screenshot of a computer program

Description automatically generated

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

See “Resources” section.

**Endpoints:**

* ‘/’:
  + Method(s): GET
  + Params: N/A
  + Gets the home page
  + Return Type(s): html
* ‘/myPage’:
  + Method(s): GET
  + Params: N/A
  + Gets the only my-page html file
  + Return Type(s): html
* ‘/login’:
  + Method(s): GET
  + Params: N/A
  + Gets the login html file
  + Return Type(s): html
* ‘/allUsers’:
  + Method(s): GET
  + Params: N/A
  + Gets the allUsers html file
  + Return Type(s): html
* ‘/style.css’:
  + Method(s): GET
  + Params: N/A
  + Gets the only CSS file that exists
  + Return Type(s): CSS
* ‘/indexbundle.js’:
  + Method(s): GET
  + Params: N/A
  + Gets the client-side code bundle for the home page
  + Return Type(s): javascript
* ‘/mypagebundle.js’:
  + Method(s): GET
  + Params: N/A
  + Gets the client-side code bundle for the myPage page
  + Return Type(s): javascript
* ‘/loginpagebundle.js’:
  + Method(s): GET
  + Params: N/A
  + Gets the client-side code bundle for the login page
  + Return Type(s): javascript
* ‘/allusersbundle.js’:
  + Method(s): GET
  + Params: N/A
  + Gets the client-side code bundle for the allUsers page
  + Return Type(s): javascript
* ‘/getUsers’:
  + Method(s): GET, HEAD
  + Params (query): name (the name of user being retrieved)
  + Gets the JSON of one or all the users
  + Return Type(s): JSON
* ‘/getUserName’:
  + Method(s): GET, HEAD
  + Params: N/A
  + Gets JSON containing the username of the currently signed in user
  + Return Type(s): JSON
* ‘/addUser’:
  + Method(s): POST
  + Params (body): name, dislikes, mains, sweets, drinks
  + will add (or update) a user to the JSON object (not saved when server stops) if there is a ‘name’ parameter in the body. Will also accept additional listed params with comma separated values.
  + Return Type(s): JSON (creation message)
* ‘/setUserName’:
  + Method(s): POST
  + Params (body): name
  + will set the signed in username to given name and create a new user object if that user doesn’t exist
  + Return Type(s): JSON (creation message)

**Resources:**

* Used API-assignment-ii as starter code.
* Used nodemon-webpack-demo to setup webpack in my project.
* Used W3schools.com for looking up different html components and javascript functionality.
  + Tables as used in /myPage and /allUsers. <https://www.w3schools.com/html/html_tables.asp>
  + Redirecting to another page in javascript, used in index.js and mypage.js to redirect to the login page if no user was signed in. <https://www.w3schools.com/howto/howto_js_redirect_webpage.asp>
* Used bulma for styling the html pages. <https://bulma.io/documentation/>
* <https://www.youtube.com/watch?v=oXW-C2bM0wE&t=103s> learn how to set up tests in postman.