Sources Consulted

ChatGPT

<https://www.w3schools.com/react/react_intro.asp>

<https://www.youtube.com/watch?v=5C1HY1sSHos>

<https://javascript.plainenglish.io/send-emails-for-your-node-js-application-using-nodemailer-express-b12584d999af>

<https://www.w3schools.com/nodejs/nodejs_email.asp>

<https://javascript.works-hub.com/learn/how-to-schedule-emails-using-node-a6bc8>

<https://www.youtube.com/watch?v=-oD50HSBBBI>

<https://www.chartjs.org/>

<https://react.dev/learn/responding-to-events#adding-event-handlers>

<https://medium.com/boca-code/how-to-encrypt-password-in-your-react-app-before-you-send-it-to-the-api-6e10a06f0a8e>

<https://mixedanalytics.com/knowledge-base/api-urls-explained/>

<https://morioh.com/p/9f51203342eb>

<https://www.youtube.com/watch?v=2-6K-TMA-nw>

<https://www.tarascodes.com/convert-timestamp-to-date-in-react>

<https://www.youtube.com/watch?v=l--0JyIS5Ts>

<https://www.dyclassroom.com/chartjs/chartjs-how-to-draw-bar-graph-using-data-from-mysql-table-and-php>

<https://getbootstrap.com/docs/4.0/components/navbar/>

<https://www.youtube.com/watch?v=l2131Rok8XU>

Thoughts on aspect of system:

Mailing system:

* Nodemailer and nodecron make for a useful tool that both sends an email based on a set timing interval
* Using googles own ability to be put into third-party apps, I can send emails from my email to other emails.
* A simple db query allows me to get the emails held in the user table and send them to the users, assuming they have opted in to being emailed, a simple Boolean value of email preference.
* Further info can be pulled from said query, like name and days they are going to gym at
* After a bit of work around, best way to store this info and avoid duplication of info is a map, by setting a map that takes the user email, but keeps track of the username and days gone in separate ways it allows for users who go more than one day to have all their days sent to them
* Simple “,” is added for readability.
* If a user doesn’t have a workout routine they will not appear in the query and therefore will not have an email sent

Leaderboard System

* Accomplished by using map to iterate through values sent to the page, specifically asking for relevant info needed, i.e. points and username.
* Potential security issue, a lot of info due to being a \* query. To consider shortening it to avoid issue of client looking at request and seeing confidential info of user

User Tier System

* Accomplished rather simply, cookies keeps userid which is sent as a query to the api route. This allows the custom SQL query to get the info it needs, namely the title of the user by comparing the amount of points they have to the ranking system list and outputting the title they have.

Bar Chart

* The Bar chart system is accomplished using chart.js, the data is fed into a dataset which has the label, data, colours and width set.
* A loading screen state is set in case it takes some time to fetch the data, and therefore letting the user know the info is loading.
* Once displayed it gives the user the progress they have made. Further improvement is to let the user select the workout they want to look at.
* The userdata is filtered based around a set called unique exercises
* An map of options is fed that goes through the available exercises, which has been set by the set of unique exercises.
* When the exercises are selected on the dropdown menu the handleExerciseChange is called to change the value of the exercise to the selectedExercise variable. This then allows the filteredData to change, as it runs based off the workout name matching the selected exercise. This ensures the exercises shown are always the ones that have been selected from the dropdown.
* The values that compose the chart data are then set on this newly filteredData allowing the chart to differ based on what exercise the user has selected

Deletion of Workout Progress

* Uses a set of the workout names and userworkout ids derived from the response.data gotten from the Api call. The key values are set as the workout name and userworkoutid which are joined with a hyphen
* An array is then taken from that set and used to fill the values of the workout, by splitting the hyphen out which allows the function to assign the values successfully.
* The result is then posted and deletes all progress associated with that user and with that particular exercise.

Removal of Exercise Routine

* Creates two sets which are both defined by a map of the response data from the api, these sets are populated with a map that goes through each row and takes both the day and workoutroutineid.
* When a day is selected the HandleWorkoutRoutineSelection is called, the day is selected by running through the workoutRoutinesAvailable and finding where the rows day matches up with the selectedDay. If found to be true, the workoutRoutineid is set as the workoutroutine id which will take from that particular row.
* The routineExerciseChange is set to blank if there is no value found from the prior function protecting against issues from undefined or null values
* A simple alert is added to prevent user submission of the form without the relevant values being filled in.
* The RoutineExerciseId is helped by a const that filters through each exercise from the api call which goes to find where the exercise.workoutroutine id is equal to the selectedWorkoutRoutineid
* This const is then used as a map which iterates through and gives each value exercise in the map a routinexerciseid and a workout name allowing for user friendliness whilst the actual value is what must be deleted from the db.

Home Page/React Bootstrap

* For ease of use with CSS and HTML in React I am using React-Bootstrap which offers powerful tools for the various elements common to a html page such as navbars, cards, grids etc.
* React-Router-Dom allows me to link to the various pages of my react project like any link in a html page.
* This forms the basis of navigating around the website.

Exercise Completion

* Uses filtering to achieve aims along with find functions.
* The days found are assigned to a set ensuring there is no repetition, which will also be enforced at a database level to only allow a user to have one exercise routine on one day
* The days available are set then as this new set
* The use effect implements a filter over the api response, where it takes every row where the day is equal to the day selected from the dropdown.
* The workoutid, and routineid are set as undefined and null to avoid issues of invalid ids being called into functions related to them
* Once the day has been selected the valid exercises are rendered, and the info is render through find getting the data where the selectedworkoutid and day match up with values in the workoutdata.
* If found this is then set, otherwise it is set to null to disable the input being used
* If the users days have been returned and found another useeffect hook is used to filter the workout data around the exercises available to that day. It finds the exercises through the filteredWorkoutData, based around the day, and then sets the routineexerciseid to the values found at that particular row. If nothing is found it is made null to prevent its use
* The third useeffect populates the select menu of the available workout routines of that day, and changes based on when the days change