Final Documentation for Gym Companion React/Node Application

By Liam Bryant and Aaron Darcy

3rd Year PPIT Module

Supervisor: Joseph Corr

GitHub Repo: https://github.com/Aaron-Darcy/PPIT

Intro

As 3rd year Computing in Software Development students, we were tasked at making an application as our end of year project. The goal of the module is to demonstrate what we have learned as students so far in this course and to put it into practice as you would in a professional working environment. Liam and I decided early on that we would use a React front end for the project and some MySQL for the back end. After discussing some ideas for the project, we decided that it would be between a clothes shop with CRUD elements or a Fitness type application.

We decided that we would do a ReactJS fitness application. We thought out some design elements and what kind of functions/Components that we would have on the web page. Our main objective for the project was to make an easy to use Fitness application with functionality in one place.

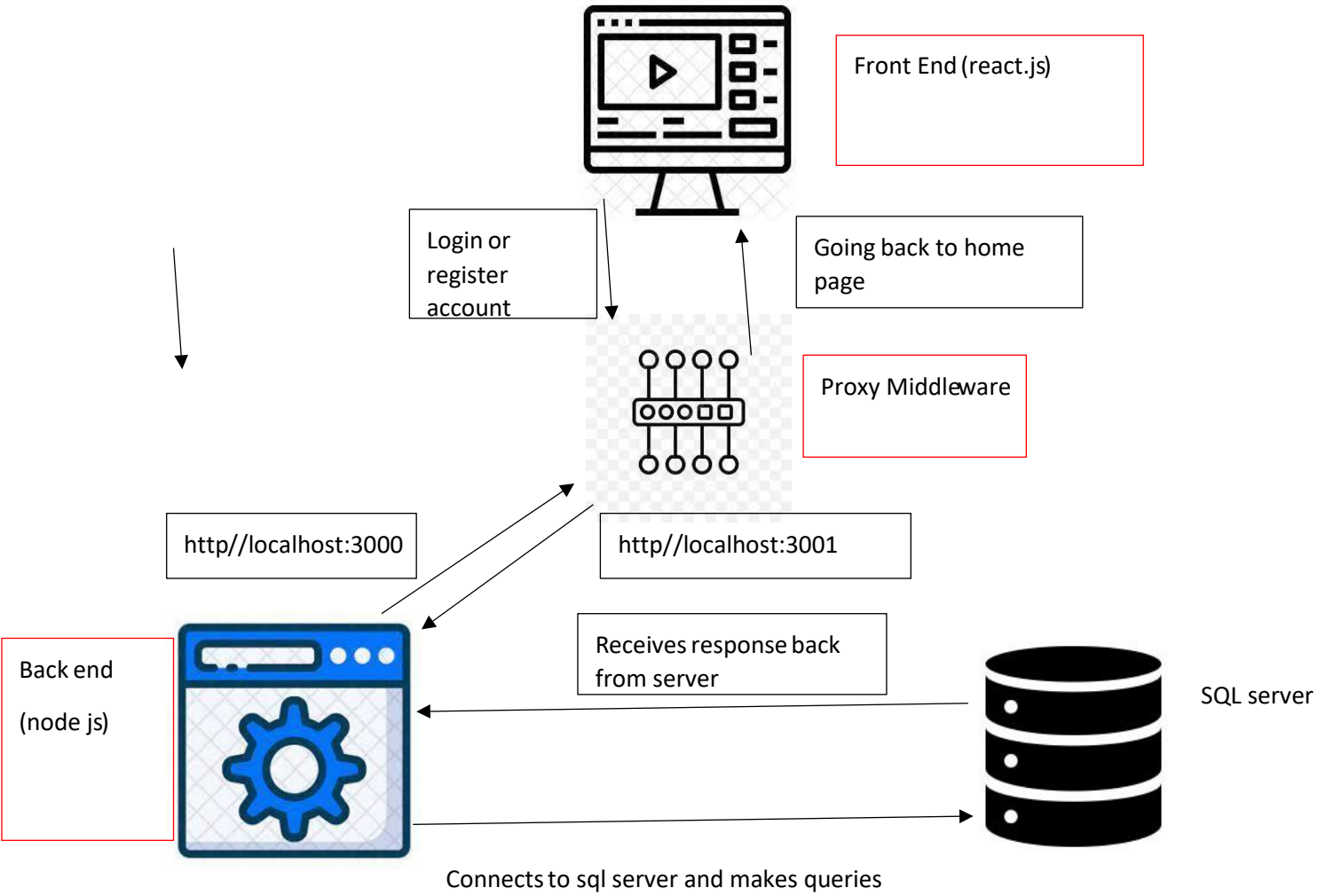
System Requirements

The application is a REACT js front end with a node.js/ejs backend. The user will have to have a pc/mobile device with a browser such as google chrome installed.

Technology Used and Why

We decided that we would use the REACT and Express Framework for our application as it was very fitting to the ideas and functionality that we brainstormed at the start. It was fitting to use REACT also because we had experience with the framework from the previous semester in the module data representation and Querying. In the backend of the application, we used the express framework and that would link up to a MySQL database with some html/views pages. We used these technologies' as it was fitting for the web page/application.

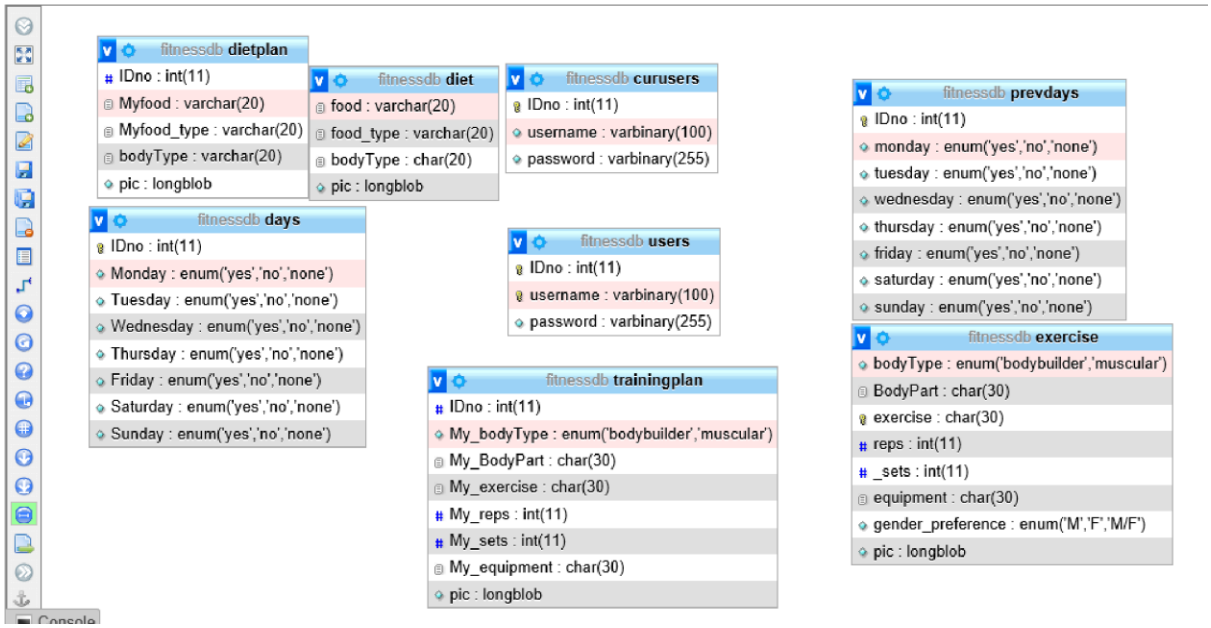
Architecture of The Solution



The solution architecture is as follows.

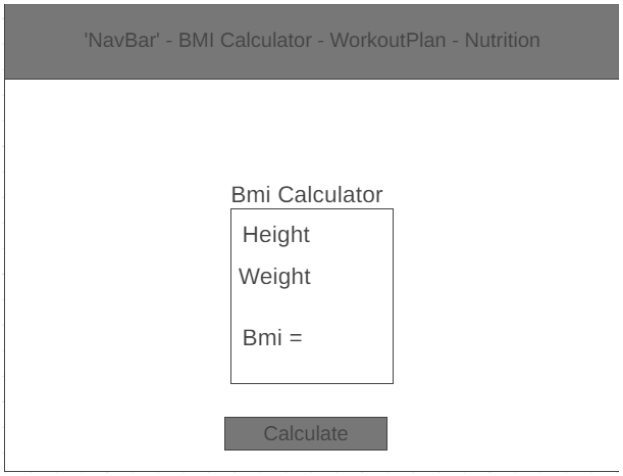
- Front end loads the home page of the website where the user can view what the site is about.
- Proxy middleware communicates with the front and backend of the website. The JavaScript class called setupProxy.js is created on the front end and sets the target link address to the back ends (http//localhost:3001). This means that when the user wants to access a page that isn't found on the front-end, it switches to the backend address and obtains the required page. It also prevents React from blocking back end's address due to it being 'malicious'.
- Back end is where the user's can access their account. Once the logged in or registered, the back end communicates to the sql database with the name of the database it's trying to access and starts creating, reading, updating and deleting data.

- SQL Server is where the user's account is stored. All details relating to the user's workout plan, meal plan and days worked out are all store on this database. The diagram below shows a database schema with the tables shown. Upon Registering, each user is given an ID number, which is auto incremented. When the user is logged in, it's the ID number that gets referenced when searching for details. For example, user 'Liam' has an ID number of 1, so when the user is making a request to get all exercises that Liam is doing, it will find id number beside it, determine who that id number belongs to and show Liam his workout table.

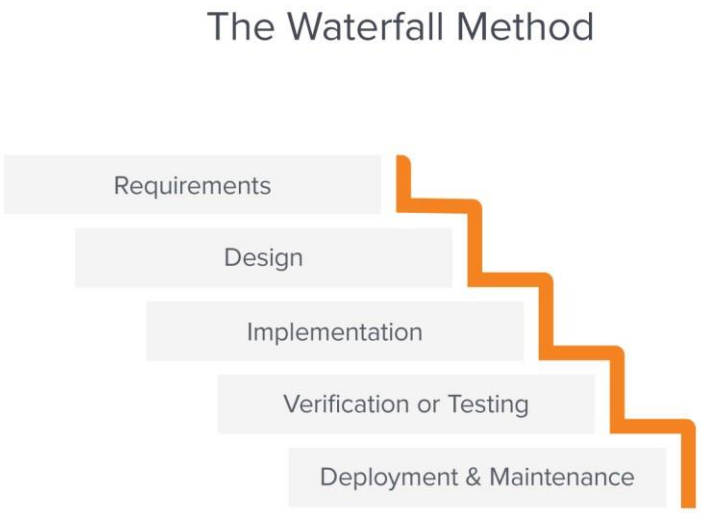


Design Methodology Applied

For the design we wanted to make a clean and simple web page application. We wanted to have the application have a navbar with routing to switch between the pages with ease. We brainstormed some functionality for the pages that we would have on the application and decided on a design and style that was fitting to the objective at hand.



Software Development Life Cycle



Requirements (8th February-16th February):

In this stage, we were given the requirements of the project, which was to develop any kind of project like a unity mobile app, a react js web application or react native app. The main purpose of this project was to recreate a scenario where we (the students) are working in a company and we are expected to work together as groups of twos or threes. We would then create a project within a given deadline and in a professional manner. This would mean that we need to use concepts from previous modules like professional IT skills to show we understand principles like sprints, testing and definition of done.

Design (23rd February-30th February):

In this stage, our group had chosen the intended project to create. A website gym app that tracks your exercises, diet and days worked out in order to obtain the desired body. We design user stories which helps us get into the point of view of the customer and plan out features that would be implemented into the project. The features were then **Implementation (1st March-30th March):**

The implementation are as followed, react js which acts as the front-end of the project and a node js and MySQL database as a backend to create, read, update and delete data. On first loading the website, the front end takes you to the home where the user has the option to use the BMI calculator, BMR calculator or login to their account. The backend is where the users can create their own person account and customise their personal workout routine. The front and back end communicate with one another via proxy middleware. The front end would run on localhost:3000 and the backend would run on localhost:3001. A JavaScript class is created where the back end's port number is set as a target for the front end. This tells the front end that another port number also exists and it won't block it because it is a 'malicious site'.

Verification/Testing (1st April-15th April):

The idea behind the testing of this project is to take the user stories and see if they fulfil the user's need. For example, a login component's main purpose is to log into an existing account created by the user, if it doesn't do that, then it hasn't been programmed properly. The test was done using selenium IDE which is a chrome extension tool that allows users to run unit tests on websites by targeting a specific section of the site and performing an action like inserting values into input boxes. Each component was given a pass or fail based on the expected output. All the failed tests were then further examined by the developers to fix the issue(s) affect the component. The website then goes through a regression test where all components are tested again to make sure that any new changes don't also affect the components that previously passed the unit test.

Deployment and Maintenance (16th April-29th April):

Once all tests all complete, all user stories fulfilled and front end and back end are connected, the project will be deployed on GitHub and Docker where anyone interested in our project can copy it and use it if they wish. The great thing about GitHub is that there is a section for reporting bugs so we can view them and fix them in the new version of our project.

Features of the Implementation

We decided we would have the following Features in our application.

- Let the user log in or register to the web page.
- Allow the user to calculate their body mass index.
- Allow the user to calculate their basal metabolic rate.
- Allow the user to track their calories for the day.
- Allow the user to track the days they worked out for previous weeks.
- Allow the user to switch between functions/components with ease using routing.
- Allows user to view their current diet and add/delete food items from their diet.
- Allows user to view their current Workout and add/delete exercises from their diet.

Limitations and Known Bugs

Connecting the front end and back end: the main problem we were having with the project was connecting the front end to the back end. Upon researching full stack development, We would often come across articles stating that you need to put a “proxy middleware” in the front end of the component. The JavaScript class called setupProxy.js is created on the front end and sets the target link address to the back ends (http//localhost:3001). After adding the middleware, it still didn’t work because I was putting “/Login” instead of “http//Localhost:3001/Login” in the navigation bar on the home page. Now that Localhost:3000 is aware of Localhost:3001, they can communicate with one another when the user wants to login and register.

Another problem we encountered was the input boxes on my backend component weren’t functioning. The only way I was able to insert or search data was to put the details in the search bar of chrome, which would mean that my site would be open to malicious attacks from users if they wanted to. The problem was that when I wanted to add or update details, I was supposed to be using app.post() instead of app.get(). I had to go back and change any functions that require the user to add or update.

Other limitations we came across were that when I tried to set up routing to the pages with the nav-bar it would not work for me as ‘React-Router-DOM’ had been updated to a new version since we last used it. You now had to use the new <Routes> Component as opposed to the older <Switch> component that we previously used in React-Router-DOM.

I also had to use different <button> types for saving the info on the calorie table as it was crashing the application.

The biggest limitation for the project in my opinion is that it is on available as a web page and not as an application on the app store.

Testing Plans

For testing we used Selenium IDE which is a free extension on the chrome web store. We done out test cases on a spread sheet which are based on the user stories we created. We also done them out on a Business Requirement Document.

Recommendations For Future Projects

For future projects we would have done the application in react native or some sort of architecture that allowed the application to be submitted to the app store /google play store. So, the user could use the application while they were in the gym.

I also would have wanted to do the back end with a mongo DB database with axios being used as well to make a smoother link to the back end.

Conclusion

In conclusion we learned how to make a full-Stack web application with some built in components and functionality. The Gym Companion app showed that it was possible to make an efficient and easy to use application with ReactJS, NodeJS and MySQL.