**Software Implementation and Testing Document**

**For**

**Group 11**

Version 3.0

**Authors**:

Tristan F

Kiara B

Kaedon H

Jared R

Jeff M

**1.** **Programming Languages**

JavaScript, with the ReactJS and NodeJS framework will be used to design a dynamic User Interface, by manipulating HTML. CSS was used for stylizing. MySQL is used for back-end and communication with the Database, hosted locally.

# **2.** **Platforms, APIs, Databases, and other technologies used**

*NutritionFacts:* This API is NodeJs wrapper of the USDA Food Composition Database API (NDB API), which provides the ability to search a large database of food items, and returns their individual macro values, such as proteins, carbohydrates, and fats.

*ReactJS:* This is a JavaScript library for building the front-end of websites. We opted to use ReactJS because it provides a highly dynamic design to User Interfaces.

*React Router:* API to create easier navigation between various pages in the application.

*Material-UI:* React design framework that uses more grid-based layouts, responsive animations and transitions, padding, and depth effects such as lighting and shadows.

*Node.js*: This is an open-source, cross-platform, JavaScript runtime environment that executes JavaScript code outside of a browser. We will use this to allow for communication between the front-end and back-end.

*MySql*: This an open-source relational database management system that we will be using to to host our database locally. We used Sequel Pro on the Mac to in order to execute the queries needed for database management.

# **3.** **Execution-based Functional Testing**

Workout generation has been implemented and tested. Users input a name and check their desired target muscles for their new workout, then press “Generate Workout”. In the back end, a workout in created with randomly selected exercises matching the muscle groups.

Every generated workout is displayed for the user when they select the “Saved Workouts” button. They are displayed as a list in expandable panels where the name of the workout is showing. When the user clicks on the name, the panel expands to show the list exercises for that workout.

Users are able to navigate back and forth between the various pages of the application. This feature was implemented using React Router.

At the end of the day, the user’s calorie count should be logged and reset to 0. This was tested by switching the date in the database and waiting 24 hours to see if it works.

The Calorie Log was tested by using the above actions and then checking the NutritionLog page to see if the results were there.

Much of the design of the website is subjective and cannot be measured in clear terms; rather, when one member makes changes to the design, layout, or aesthetic choices within the website and the website is accessed, the effect is tested by the rest of the group by their interaction with the website.

# **4.** **Execution-based Non-Functional Testing**

We are able to host our application locally on various browsers, such as Chrome, Firefox, and Safari. Viewing other websites confirms that it is possible to implement features for people with vision issues.

All user data is put into the KitchGym database upon sign-up to decrease the amount of local storage used by the user.

Console log statements and error messages are returned following API calls to ensure that a command was (or wasn’t) successful. The Nutrition Facts API has built in error messages, while our group coded in error messages in query calls to the MySql database.

Material-UI framework was used to give the application a smoother and more modern look. The pages transition in under 3 seconds with the help of the React Router API.

# **5.** **Non-Execution-based Testing**

React documentation states that React application can be supported by most modern browsers, and support for Internet Explorer 9, 10, and 11 requires polyfills.

Reviewing the SQL code is a simple process of making sure the correct type of components were placed in the correct locations. Peer code reviews were conducted when one member pulls another members code from the GitHub, and looks over it to ensure that it will run.

The exercises available in the database we created are backed by our expertise and research we have done on muscle-targeted exercise. This database was checked to ensure that each exercise included the attributes needed to identify the correct one for the workout generation in the website, such as the exercise name and target muscle group.

The US Department of Agriculture’s food database contains information on all types of food and the nutritional value that comes with them. This nutritional information includes calories and macronutrient breakdowns. The API was tested and verified to return the stated information in JSON format via the web browser. We reviewed the documentation to ensure the API would be compatible with Node.js. We eventually found a wrapper, called nutrition-facts, that would allow us to use it with JavaScript.