TECHNICAL DOCUMENTATION OF “URecipe”

### **Compatibility:**

Devices: PC, MacBook

Browsers: Chrome, Internet Explorer, Microsoft Edge

Additional Hardware: Trackpad, Mouse, Touchscreen, Keyboard

### **Currently NOT Compatible/Untested On:**

Devices: iPhone, Android, other handheld devices, Linux, Ipad

Browsers: Firefox, Opera, interest explorer

Additional Hardware: Bluetooth Keyboard, Stylus/Drawing Pad

# **Files:**

The files use the languages HTML, CSS and JavaScript. The central file (the webpage) is “index.html”. The linked styling is “majorproject.css”, the project also uses JQuery’s with JavaScript, it is in a separate folder called “index.html” linked to my Survey page.

There are three JavaScript files. “majorproject.js” is for all the global functions that also interact with the database. This file are in the “js” folder. My second file is “Wheels.js” which contains the math.random functions in-order for the wheel to spin and coordinate a box to select. My final js folder is titled “survey.js”. It contains buttons for the user to click which link to my RestDB server. It also has many query selectors, The querySelector(".option1") method allows me to select the first element that matches one or more CSS selectors.

There is also an images folder which contains all the images used for the webpage. I have created a fruit banner that reaches across all nav pages, therefore this image is distributed across these folders.

“majorproject.js” is for all variables and functions used mainly on the MealChoices screen. It is formatted with one JavaScript file used for the one index page, which now holds all the index for each page feature. Each section is separated by comments, each section having its own global variable section at the top and functions afterwards.

# **Functions – JS file**

Global variables can be accessed from anywhere in a JavaScript program. There are various “global” functions – generic functions that pass multiple parameters that are used many times.

The function var userProfile = { is used for the signUp page for my project. It contains all the values such as UserName, Password, firstName, lastname, which are text boxes on the page that the user fills in to create a new account. In order for the accounts created to have unity, the user is limited with their password restrictions such as a number an upper case letters must be included, and usernames must be the letter of their first name, followed by their name. These values are then stored in the database, and then the UserName and password values are matched for when the user signIn’s to check if the values entered are correct for them to then have access to the portal. Commenting is also very important on the page, and is used to maintain the code.

# **Click Handler’s**

“$('#SignInLink').click(function()” is a clickhandler to hide one screen and show another and is used every time the sign in button is clicked. “SignUpLink” is another variation that switches screen, then creates a new screen/dialogue if the user decides to signUp to the website, and takes them to a survey page.

The “aboutIndexpage” function is what pops up when the website starts up. It hides all elements except for the main menu screen. Many buttons contain similar functions where, if clicked, everything is hidden except for it’s respective screen. This is the case for “SignIn”, “SignUp”, “About”. The reverse is the implementation of the sideNav bar which send the user back to the SignIn, SignUp screen or the main title screen. The button: “BackButton” is used on the meal Choices page and allows the user to switch between breakfast, lunch and dinner meal choices.

# **Variables**

All variables and elements use camel case.

All global variables are declared at the top of the file (or the top of its section in regards to "css/majorproject.css". However, on the index page, "js/majorproject.js" is a script defined at the end of the page as the page reads the order from top to bottom.

Variables are very important for referencing items on my webpage. For example “Home” is a variable that groups together all the <p> tags on my page. They can be given different id’s to reference in the CSS to change the styling and position on the page. Variables are very useful to differentiate similar items and makes the html page neater and easier to maintain.

# **CSS**

All arrays start with the prefix “arr”. There are multiple relevant arrays in the program.

The css styling in my project is consistent throughout the page. The variables are either recognised with a ‘#’ meaning it is an id, or a ‘.’ Meaning it is a class. All images are given a width and a height depending where about it is on the page. For example, “recImg” is the image on the About page. This has a width of ‘100px’ and a height of ‘100px’. To give my project better styling, a lot of buttons are given a ‘border-radius’. For example, ‘ProfileData’ in the top right corner is given a ‘radius’ of 20px, giving it a sleek look. Colour is important and is given by hex codes in the CSS page. This creates a theme on the page and is different variants of green for me. This is seen through, the ‘header’ with a hex code of ‘#185a37” which gives a deep green colour which compliments the page. Commenting is also very important on the page and is used to maintain the code.

## **Strings**

All string variables start with the prefix “Id” or “class”, and are defined in quotation marks.

The string “SignUp” in the index file holds all the divs and classes that can later be defined and styled in the css file. The SignUp index includes a login form which is presented on the user screen and is multiple text boxes for the user to fill in. It is important that strings are unique for each definition, as problems may arise with multiple strings with same defined names.

The string “ToolTip” is used as the user help guide on the user interface. It has its own class and is used for the user to hover over to find out more information.

## **ID’s /Classes – HTML file**

Classes are used in Javascript to define functions and properties within the index.html file. There are multiple classes within the game. Most classes that are named are usually also given an ID. These classes and ID are utilised for referencing divs, headers, spans and p tags in the CSS files to style and position them on pages through a (.) for classes and (#) for ID’s. By giving classes for items, it makes it clear to differentiate items and makes it easier to find errors in the console.log. some examples of classes within my code include: my SignUp div. This is a class called “form” and holds the sign-up requirements for the page. This class has compiled the login form into a list and groups divs together. Another example of a class includes “option1”. It is a general HTML element that holds a list of buttons for the survey feature of my project.

# **Databases**

The website uses one database called RestDB.io. The server is linked to Visual studio code via variables defined at the top of the JS file, such as through APIKey, which is a specified key for individual accounts. There is also the Recipes URL which is linked to the recipes database created which holds all the recipes for breakfast, lunch and dinner for all the specific diet requirements. The recipe database holds many recipes for each dietary requirement, ensuring the user has choice. There is also an accounts database, which is updated once a profile is created. The user enters in their first name, last name, date of birth, and they also create a username and password. This database is necessary as the username and password created is the key to the signIn page to access their portal.

After completing the survey after the sign Up, the user chooses their diet preference which is then stored in the database, which directly links to what recipes are selected by the user.