

CAB432 Assignment 1 Report

Semester 2 2019

By Zachary Murray - n9710418

1 CONTENTS

2	Introduction	4
3	Mashup use cases and services	5
3.1	Use case 1: Health.....	5
3.1.1	User story	5
3.1.2	User experience	5
3.1.3	Supporting this use case	8
3.2	Use case 2: Customisation	9
3.2.1	User story	9
3.2.2	User experience	9
3.2.3	Supporting this use case	11
3.3	Use case 3: Ingredient versatility	11
3.3.1	User story	11
3.3.2	User experience	11
3.3.3	Supporting this use case	12
3.4	Use case 4: Unknown ingredients.....	13
3.4.1	User story	13
3.4.2	User experience	13
3.4.3	Supporting this use case	14
4	Technical specifications	15
4.1	Technology Stack	15
4.1.1	Node.JS.....	15
4.1.2	Express.JS	15
4.1.3	EJS	15
4.1.4	JavaScript + jQuery.....	15
4.1.5	HTML + CSS	15
4.2	Architecture	16
4.3	Caching & persistence.....	17
4.4	Security	18
5	Docker	19
6	Test Plan.....	20
6.1	Feature: Cocktail search.....	20
6.1.1	Testing Methodology	20
6.1.2	Acceptance Criteria	20
6.1.3	Results.....	20

6.2	Feature: Cocktail Recipes	20
6.2.1	Testing Methodology	20
6.2.2	Acceptance Criteria	20
6.2.3	Results	20
6.3	Feature: Nutrition information	21
6.3.1	Testing Methodology	21
6.3.2	Acceptance Criteria	21
6.3.3	Results	21
6.4	Feature: Recipe editing	21
6.4.1	Testing Methodology	21
6.4.2	Acceptance Criteria	21
6.4.3	Results	21
6.5	Feature: Ingredient information	21
6.5.1	Testing Methodology	21
6.5.2	Acceptance Criteria	21
6.5.3	Results	21
6.6	Feature: Random cocktail	22
6.6.1	Testing Methodology	22
6.6.2	Acceptance Criteria	22
6.6.3	Results	22
6.7	Feature: Ingredient substitution	22
6.7.1	Testing Methodology	22
6.7.2	Acceptance Criteria	22
6.7.3	Results	22
7	Possible Extensions	22
8	User guide	23
9	References	23
10	Appendix	24
10.1	Full dockerfile	24
10.2	Margarita Recipe	25
10.3	Margarita nutritional information	26

2 INTRODUCTION

“Skinny sip” is an application that combines API’s to produce nutritional information for common cocktail recipes. The user can search for their favourite cocktails and will be presented with recipes and nutritional info laid out in a familiar format that conforms to Australian food packaging laws. This is of importance as Australian law does not require alcoholic beverages to display detailed nutrition info on their packaging list most other food items, this is something that *Skinny Sip* aims to fix. Among other things, *Skinny Sip* also allows the user to temporarily edit cocktail recipes as they see fit whilst updating the nutrition information, and it can also provide ingredient specific details including which cocktails contain each ingredient.

“TheCocktailDB” (<https://www.thecocktaildb.com/>) API is used to search for cocktails, retrieve drink recipes, and list ingredient details. Additionally, the cocktail DB hosts all the cocktail images, and some ingredient images too.

“Nutritionix” (<https://www.nutritionix.com/>) is an API which can determine the nutritional information of thousands of ingredients. It is used to compute the nutritional information for each recipe. It also hosts some ingredient images.

3 MASHUP USE CASES AND SERVICES

This section contains several use cases for the application, and how the underlying API's support such use cases.

3.1 USE CASE 1: HEALTH

3.1.1 User story

As a health-conscious person, I want to know the nutritional information of my favourite cocktail, so that I can modify my diet accordingly and improve my overall health.

3.1.2 User experience

1. The user arrives at the home page and is immediately greeted by a large and clear search bar to find their favourite drink.

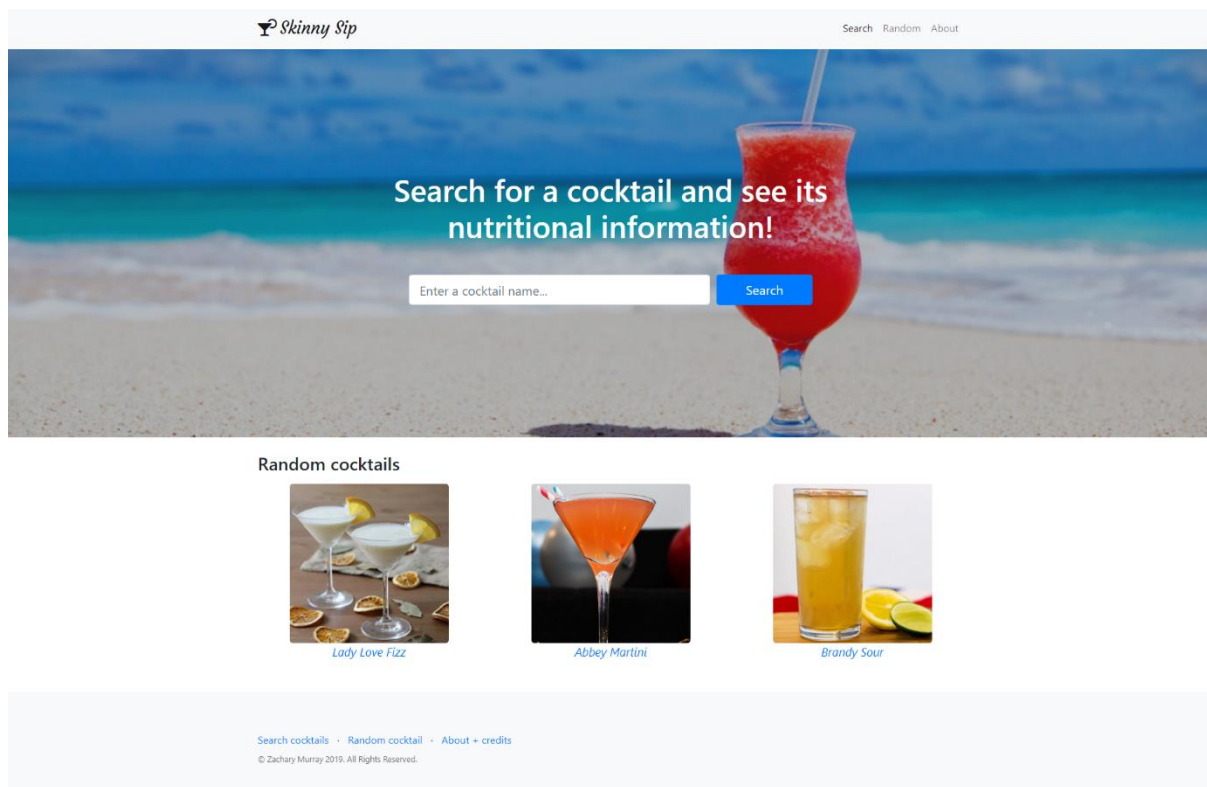


Figure 1 Skinny sip home page

2. They will then be presented with some search results, they can select the drink they are looking for from this list.

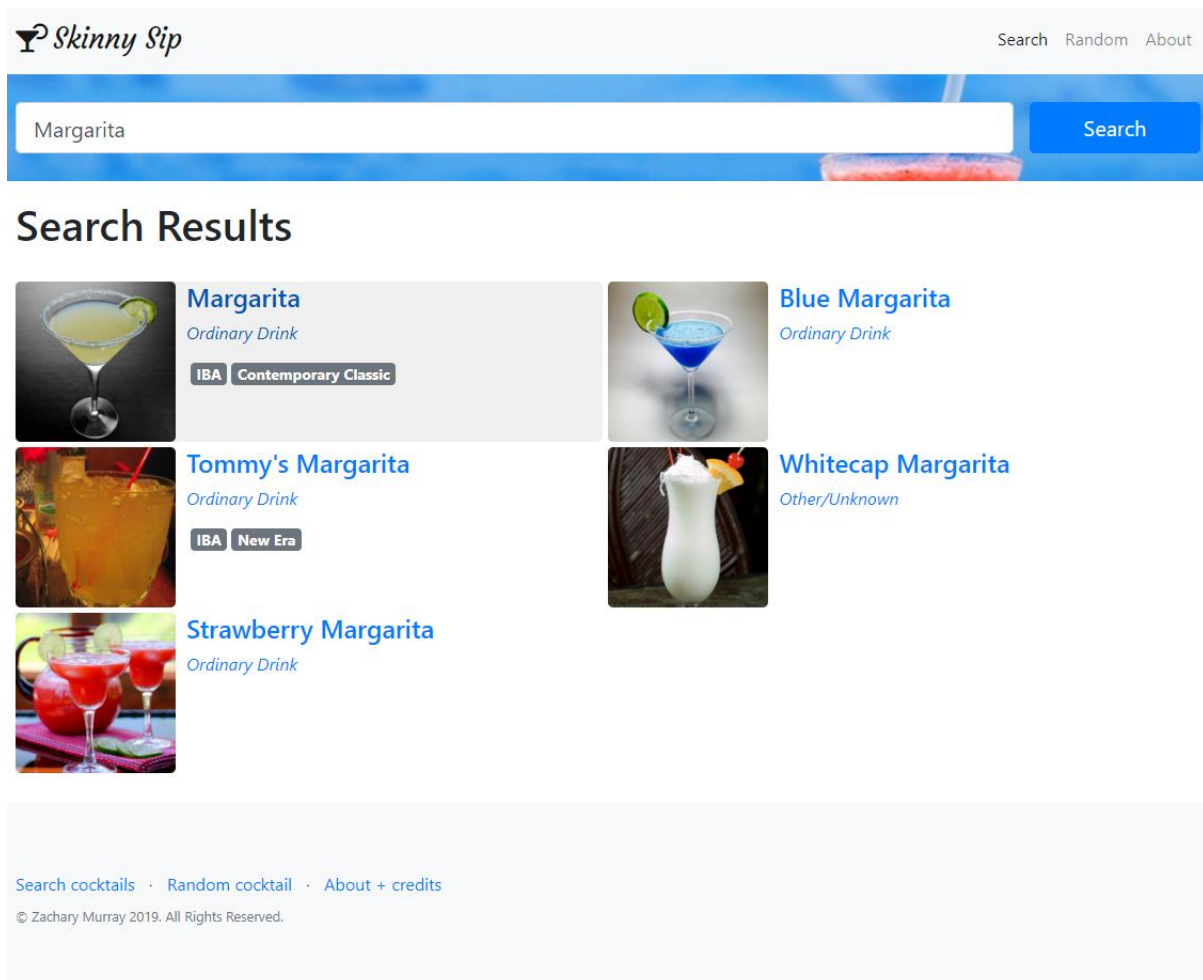
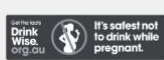


Figure 2 Search results screen

- Once they have selected their drink, the following page will display to show its recipe and nutritional information

Margarita

Ordinary Drink IBA Contemporary Classic



Nutrition Information (AVERAGE)

Servings per package: 1
Serving Size: 90.8g

	quantity per serving	% daily intake ▲ per serving	quantity per 100g
ENERGY	578 kJ	7%	636.6 kJ
PROTEIN	0.1 g	<1%	0.1 g
FAT, TOTAL	0 g	<1%	0 g
- SATURATED	0 g	<1%	0 g
CARBOHYDRATE	6.2 g	2%	6.9 g
- SUGARS	4.1 g	5%	4.6 g
DIETARY FIBRE	0.1 g	<1%	0.1 g
SODIUM	2326.6 mg	101%	2562.4 mg
POTASSIUM	37.6 mg	1%	41.4 mg
CHOLESTROL	0 mg	-	0 mg
		% RDI*	
CALCIUM	5.8 mg	1%	6.4 mg
IRON	0.1 mg	1%	0.1 mg
ALCOHOL, ETHYL	16.8 g	-	18.5 g
ASH	6.1 g	-	6.7 g
BETAINE	0.1 mg	-	0.1 mg
CAROTENE, BETA	9.2 µg	-	10.2 µg
CHOLINE	1.6 mg	-	1.7 mg
FLUORIDE	0.1 µg	-	0.1 µg
FOLATE, TOTAL	3.1 µg	2%	3.4 µg
FRUCTOSE	0.2 g	-	0.2 g
GLUCOSE	0.2 g	-	0.2 g
MAGNESIUM	2.5 mg	1%	2.8 mg
NIACIN	0.1 mg	1%	0.1 mg
PHOSPHORUS	6.3 mg	1%	7 mg
SELENIUM	0.1 µg	<1%	0.1 µg
SUCROSE	3.8 g	-	4.2 g
VITAMIN E	0.1 mg	1%	0.1 mg
VITAMIN A	0.6 µg	<1%	0.7 µg
VITAMIN C	9.2 mg	23%	10.2 mg
VITAMIN K	0.2 µg	<1%	0.2 µg
WATER	61.5 g	3%	67.8 g
ZINC	0.1 mg	<1%	0.1 mg

▲ Percentage daily intakes are based on an average adult diet of 8700kJ.
* Percentage Recommended Dietary Intake (Aust/NZ)

Ingredients

Image	Name	Amount	Energy (kJ)
	Tequila	1.5 oz (44 mL)	403 kJ
	Triple Sec	0.5 oz (15 mL)	143 kJ
	Lime Juice	1 oz (30 mL)	32 kJ
	Salt	1 tsp	0 kJ

Method

Rub the rim of the glass with the lime slice to make the salt stick to it. Take care to moisten only the outer rim and sprinkle the salt on it. The salt should present to the lips of the imbiber and never mix into the cocktail. Shake the other ingredients with ice, then carefully pour into the glass.

Figure 3 Cocktail info screen

The user now can view both the cocktail recipe, and the kilojoules plus other nutritional information. If they decide the energy consumption fits well in to their diet, they can proceed either make it according to the recipe or purchase it.

3.1.3 Supporting this use case

To support this use case the application will make several calls to both TheCocktailDB and Nutritionix, they happen in this order:

1. A call is made to TheCocktailDB to retrieve search results, TheCocktailDB also serves up images for the thumbnails. Once this is complete the user is presented with the screen shown in Figure 2.
2. Once the user has selected their cocktail, another call is made to TheCocktailDB to get the full cocktail recipe, image, description and tags.
3. After the recipe has been received, it is processed on the server into a format understandable by Nutritionix, it is then sent to Nutritionix which responds with a list of objects containing the nutritional information for each ingredient.
4. Further processing on the server happens, then the user is displayed with the screen as shown in Figure 3.

3.2 USE CASE 2: CUSTOMISATION

3.2.1 User story

As a bartender, I want to adjust the recipes I see in Skinny Sip to my own tastes while still retaining the ability to see accurate nutritional information. This is so that I can see how different ingredient ratios will affect the drinks healthiness before I serve it to people.

3.2.2 User experience

1. After the user has found their base cocktail recipe, they will be presented with the page as shown in Figure 3. They notice that beside each ingredient measurement is a small edit icon:

Ingredients



Image	Name	Amount	Energy (kJ)
	Tequila	1.5 oz (44 mL) 	403 kJ

Figure 4 Ingredient edit icon

2. After clicking this icon, the field becomes editable:

Ingredients


Image	Name	Amount	Energy (kJ)
	Tequila	<input type="text" value="1.5"/> <input type="text" value="oz"/>	403 kJ

Figure 5 Ingredient being edited

3. Once they have made the necessary adjustments, they click the “Update Nutrition” button that has appeared at the bottom of the ingredient table. (if they have decided to revert their changes, they can also click the “Cancel Changes” button)



Figure 6 Update/cancel nutrition buttons

- The page will refresh, and all edited ingredients will be highlighted like so:

Ingredients



Image	Name	Amount	Energy (kJ)
	Tequila	3 oz (89 mL)  Edited	806 kJ

Figure 7 Highlighted edited ingredient

- The nutrition info panel on the left now reflects the nutrition values for the edited cocktail:


<div> <div>STANDARD DRINKS 3.1 APPROX.</div> <div> Get the facts Drink Wise. org.au </div> <div>  It's safest not to drink while pregnant. </div> </div>			
Nutrition Information (AVERAGE) Servings per package: 1 Serving Size: 132.5g			
	quantity per serving	% daily intake ▲ per serving	quantity per 100g
ENERGY	981 kJ	11%	740.4 kJ
PROTEIN	0.1 g	<1%	0.1 g
FAT, TOTAL	0 g	<1%	0 g
- SATURATED	0 g	<1%	0 g
CARBOHYDRATE	6.2 g	2%	4.7 g
- SUGARS	4.1 g	5%	3.1 g

Figure 8 Updated nutrition info after edit

- The user can continue to make changes in this manner to as many ingredients as they like. All changes are temporary and affect only the current user, and can be reverted at any time using the “Cancel Changes” button

3.2.3 Supporting this use case

This use case is primarily supported by server-side processing as well as the Nutritionix API.

1. When the user edits an ingredient, the field is pre-populated using the existing measurement that was retrieved from TheCocktailDB.
2. Upon applying the changes, the server will get the recipe from TheCocktailDB like normal, however before processing and sending to Nutritionix, it will change the quantities of each ingredient by the specified amount.
3. Nutritionix is queried as normal and returns the updated nutrition information which is displayed to the user.

Note that this specific use case was what prompted the development of a caching facility for Nutritionix data, this is explained further later in the report.

3.3 USE CASE 3: INGREDIENT VERSATILITY

3.3.1 User story

As a consumer, I want to know what other cocktails contain an ingredient, so that when I buy the ingredient, I can use it for other recipes and use it up faster.

3.3.2 User experience

1. On the drinks screen, the user can click on an ingredient name to see its details

Ingredients



Image	Name	Amount	Energy (kJ)
	Tequila	1.5 oz (44 mL) 	403 kJ

Figure 9 Link to view ingredient details

2. They are then presented with a screen that give more information about the ingredient, as well as which drinks contain it

Tequila

Spirit



Tequila (Spanish pronunciation: [teˈkila] (About this sound listen)) is a regionally specific distilled beverage and type of alcoholic drink made from the blue agave plant, primarily in the area surrounding the city of Tequila, 65 km (40 mi) northwest of Guadalajara, and in the highlands (Los Altos) of the central western Mexican state of Jalisco. Aside from differences in region of origin, tequila is a type of mezcal (and the regions of production of the two drinks are overlapping). The distinction in the method of production is that tequila must use only blue agave plants rather than any type of agave. Tequila is commonly served neat in Mexico and as a shot with salt and lime across the rest of the world. The red volcanic soil in the region around the city of Tequila is particularly well suited to the growing of the blue agave, and more than 300 million of the plants are harvested there each year. Agave grows differently depending on the region. Blue agaves grown in the highlands Los Altos region are larger in size and sweeter in aroma and taste. Agaves harvested in the lowlands, on the other hand, have a more herbaceous fragrance and flavor. Mexican laws state that tequila can only be produced in the state of Jalisco and limited municipalities in the states of Guanajuato, Michoacán, Nayarit, and Tamaulipas. Tequila is recognized as a Mexican designation of origin product in more than 40 countries. It is protected through NAFTA in Canada and the United States,[6] through bilateral agreements with individual countries such as Japan and Israel, and has been a protected designation of origin product in the constituent countries of the European Union since 1997. Tequila contains alcohol (also known formally as ethanol) and is most often made at a 38% alcohol content (76 U.S. proof) for domestic consumption, but can be produced between 31 and 55% alcohol content (62 and 110 U.S. proof). Per U.S law, tequila must contain at least 40% alcohol (80 U.S. proof) to be sold in the United States.

Drinks that contain this ingredient



110 in the shade

110 in the shade



Adam Bomb



3-Mile Long Island Iced Tea

3-Mile Long Island Iced Tea



Amaretto Stone Sour #3

Figure 10 Ingredient information screen

3.3.3 Supporting this use case

This use case is entirely supported by TheCocktailDB API.

1. The ingredient name is run through the search API at TheCocktailDB to determine its ID.
2. Once it's ID is found, TheCocktailDB is queried again to get the category and description for the ingredient.
3. A final call to TheCocktailDB gets a list of drinks that contain the ingredient

This process is necessary as the ingredient ID is not sent down with the drink recipe, only the name hence why it needs to first be searched.

3.4 USE CASE 4: UNKNOWN INGREDIENTS

3.4.1 User story

As a user of *Skinny Sip*, I want the platform to be able to process even the most obscure ingredients so that all recipes are available for nutrition analysis.

3.4.2 User experience

1. The user clicks a link to a cocktail page for a cocktail that contains obscure ingredient(s) (ingredients that are not supported by the Nutritionix API)

Search Results

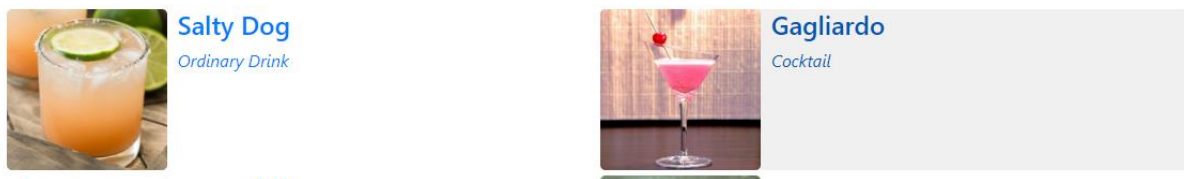


Figure 11 Selecting a cocktail that contains obscure ingredients

2. The system will recognise an unsupported ingredient and redirect the user the following page

Unknown Ingredient

We couldn't find any nutrition info for the ingredient: "peach vodka"

Please enter a similar/generic ingredient below, and we'll use the nutrition information from that instead.


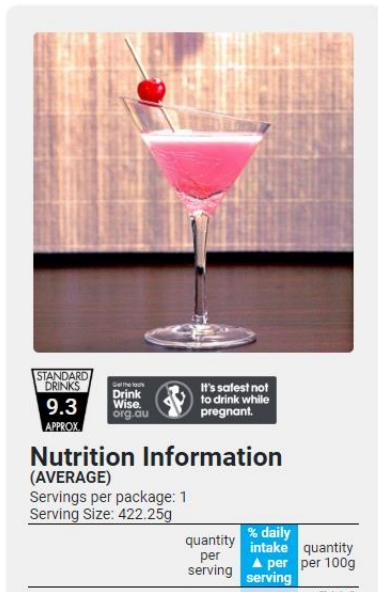
Original:		Nutritional Substitute:
	peach vodka	<div><input type="text" value="apple vodka"/></div> <div><div>apple vodka</div><div>penne vodka</div><div>vanilla vodka</div><div>carmel vodka</div><div>vodka</div></div>
	<div>→ Submit</div>	

Figure 12 Unknown ingredient page

3. As shown in Figure 12, the user is asked to use the search box to find a nutritionally similar food item to the obscure one. In the example shown, clearly “peach vodka” is not known by the nutrition API, however “apple vodka” is known (since it appears in the dropdown). Once the user has found a good substitute, they click the “Submit” button, and the drinks page appears like normal:

Gagliardo

Cocktail



Ingredients







Image	Name	Amount	Energy (kJ)
	Peach Vodka	5 shot (210 mL) 	2059 kJ
	Lemon Juice	3 tsp 	11 kJ
	Galliano	1 shot (1.5 fl oz) 	547 kJ

Figure 13 Drinks page for recipe after nutrition substitution

4. The user now can see the nutrition information for the recipe despite it using an ingredient not supported by the nutrition API. So long as the user picks a suitable substitute, the results should still be accurate. Note that the recipe has not changed, only the underlying nutritional information for the unknown ingredient has.
5. The user can re-visit this page later and will not have to re-do the substitution, it is permanent.

3.4.3 Supporting this use case

1. The recipe of queried from TheCocktailDB like normal and sent to Nutritionix for processing
2. Nutritionix will return an error for unknown ingredients, in this case the server will flag the ingredient for replacement, redirecting the user to the replacement screen.
3. To ensure that the user picks a substitute ingredient that is known by Nutritionix, the client is making AJAX calls to the Nutritionix autocomplete API to populate the dropdown search results as the user types.
4. Upon submission of the replacement form, a final validation of the substitute ingredient is performed with another query of the nutrition API for the substitute ingredient, if this succeeds then the replacement is made permanent.

4 TECHNICAL SPECIFICATIONS

4.1 TECHNOLOGY STACK

4.1.1 Node.JS

Node.JS was used as the server-side technology to host and ultimately run the entire application. Node.JS is a framework used to run JavaScript in a server-side environment – and worked very well for this application thanks to its rich ecosystem of plugins. Additionally, since node uses JavaScript, it increased development efficiency by not having to flick between two different languages when developing the client/server side of the app.

4.1.2 Express.JS

Express.JS is a web framework that runs on top of node and allows us to receive and respond to HTTP calls, and act in the capacity of a web server. Express.JS facilitates the serving of static files to the client, as well as routing requests and sending rendered resources to the client.

4.1.3 EJS

EJS is a templating engine for node which allows us to easily render HTML code ready to send to the client based off EJS templates. It works by reading a template file, then inserting pieces into it based off embedded JavaScript code. Here it was used to insert processed data from the API's into the site templates, then express.js sent those rendered views to the client.

4.1.4 JavaScript + jQuery

For the client side of things, jQuery was used primarily to issue AJAX calls and manipulate the DOM in a more intuitive manner than vanilla JavaScript provides. For most business logic (which there isn't much on the client) vanilla JavaScript was used. While writing client-side JavaScript, ECMAScript 6 was adhered to. While this may not support older browsers, it is intended that *Skinny Sip* be used with modern browsers anyway.

4.1.5 HTML + CSS

The two main building blocks for rendering web pages to the user are HTML and CSS. Use of a HTML and/or CSS transpiler was considered, but ultimately decided against due to extra complexity. Vanilla HTML and CSS was able to get the job done fine.

4.2 ARCHITECTURE

The application structure roughly follows the “Model, view, controller” structure common to many modern web apps. The following table lists the application structure with each file/folders responsibility.

~/Packages.json	<ul style="list-style-type: none">As this is a node application, the natural starting point is the “packages.json” file, this includes a list of the necessary packages to run the app, as well as the command to run to start it.
~/Server.js	<ul style="list-style-type: none">This is the application entry point specified in "packages.json"It is responsible for initialising express.js with the necessary configuration settings such as host and port, and then starts the server
~/Routing.js	<ul style="list-style-type: none">This file uses the express.js router to define links between URL's and either a static file, or an EJS template and associated server-side controllerIt also designates certain folders to be publicly available through the web serverThis file is the entry point for most HTTP requests
~/img/* ~/css/*.css ~/controllers/client/*.js	<ul style="list-style-type: none">These folders have been designated as public within Routing.js, this means all resources within these folders are accessible from the client side.This is because images, css stylesheets, and client-side controllers all need to be downloaded by the browser in addition to the HTML contentThe client side controllers are JS files that control each page
~/api/internal/*.js	<ul style="list-style-type: none">This folder contains the majority of the application's business logic.Each file here exports a function or class which can be used elsewhere in the application to accomplish certain tasks.
~/api/*.js	<ul style="list-style-type: none">This folder contains files that are designed to create an API to serve JSON data to the client.Each file in this folder exports a function which takes the router object as a parameter, when it is called it attaches the necessary API endpoints to the router object with integrated business logic.These files handle all AJAX requests issued from the client-side controllers
~/controllers/server/*.js	<ul style="list-style-type: none">These files are modules that export an object containing all the values necessary to render their associated EJS templateThey are responsible for computing most of the data sent to the client
~/views/*	<ul style="list-style-type: none">These files contain the templates for each route in the applicationEach view has an associated server-side controller which handles getting the values to insert into the template
~/config/*.json	<ul style="list-style-type: none">These files contain configuration information such as API keys.JSON files used for persistence and caching are also located here

Table 1 Application architecture

The following diagram demonstrates this system architecture in relation to incoming HTTP requests. The routes in routing.js will determine whether a request is for a static resource, api or view.

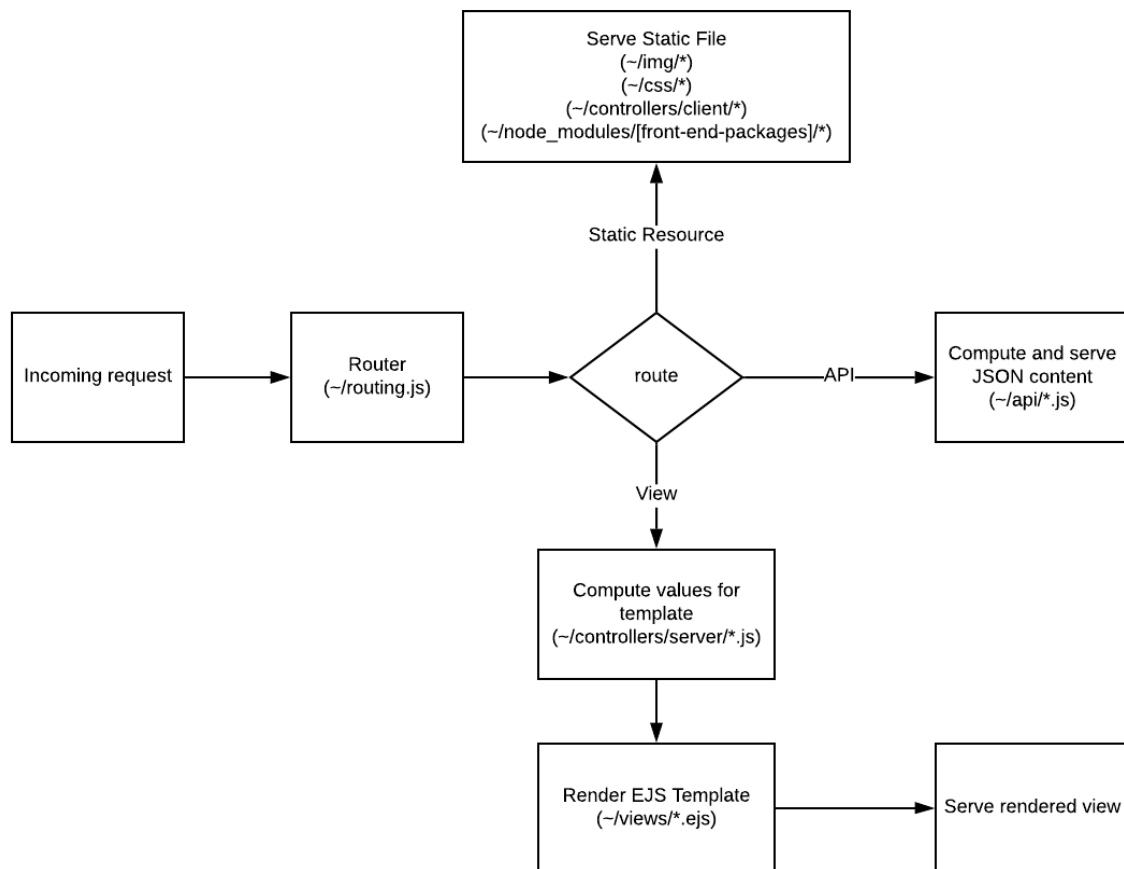


Figure 14 Flowchart of request flow

4.3 CACHING & PERSISTENCE

The Nutritionix API has very low daily usage limits, and the limits were regularly exceeded during testing and development, especially after the introduction of the recipe editing feature. The Nutritionix API permits the caching of their data, so a basic disk cache was implemented to store ingredient results for future use. Every time a request for nutrition data is made, the cache is checked to see if it contains any/all of the ingredients that are part of the requested recipe, only items that are not in the cache are sent to the Nutritionix API. Once the Nutritionix API responds, the results are saved to the cache for future use.

This works by keeping an in-memory copy of the cache and saving this to disk when items are added. The cache is stored on disk as a JSON file, which is read and parsed once at the application start-up. To keep this cache file at a manageable size, the cache is limited to 1000 items. The cached ingredient objects can also be scaled to different unit values, so no additional API call is required when a user modifies an ingredient quantity.

A similar system is used to maintain the ingredient substitutions list.

4.4 SECURITY

The system has been designed with security in mind. The use of EJS and jQuery prevents any attempt of XSS attacks, even if either of the underlying API's are compromised. This is because EJS and jQuery are the only ways for custom data to be rendered to the client – and both do HTML sanitisation by default.

Additionally, SQL injection attacks are not possible due to there being no underlying data base. Each API endpoint was also tested against SQL injection and both seem to have implemented their own security against this.

The ingredient substitution system was also locked down to prevent abuse of it. A security violation will be raised if a substitution attempt is made for an ingredient that has not been flagged as needing substitution. Additionally, substitute ingredients are checked for validity before writing them to disk. This ensures that the mappings table stays valid, and cannot be compromised by malicious users.

5 DOCKER

As per the assignment specification, the application was built into a docker image, ready for deployment as a container on most Linux based systems. The application being distributed as a docker image allows for extremely easy set up as it essentially can run in a completely isolated environment, but without the overhead of running a complete VM.

Since the application was built on top of node.js, the base image that the dockerfile inherits from is the node docker image available on docker hub:

```
FROM node:10.16.3
```

This specific version of node was chosen as it is the current LTS (long term service) release of node. From there, the dockerfile sets up the working directory and copies the package.json and package-lock.json file to it:

```
# Create app directory
WORKDIR /usr/src/app

# Install app dependencies
# A wildcard is used to ensure both package.json AND package-
lock.json are copied
# where available (npm@5+)
COPY ./SkinnySip/package*.json ./
```

To prepare the application to run, all of its npm dependencies need to be installed:

```
RUN npm install
```

Finally, all other files from the application are copied into the image, port 80 (for HTTP traffic) is exposed, and the server is started with the “node server.js” command:

```
COPY ./SkinnySip/. .

EXPOSE 80
CMD [ "node", "server.js" ]
```

With the docker file completed, an image can be built using the following command:

```
Docker build -t protango/skinny-sip
```

With the docker image built, it was pushed to **dockerhub** for easy access:

```
Docker push protango/skinny-sip:latest
```

6 TEST PLAN

6.1 FEATURE: COCKTAIL SEARCH

Display a list of cocktails that contain an arbitrary text string specified by the user

6.1.1 Testing Methodology

Perform a search with at least one term for each of the following situations:

1. Term is the complete name of a cocktail (eg. "Margarita")
2. Term is the partial name of a cocktail (eg. "Long")
3. Single letter (eg. "a")
4. Term that is not part of any cocktail name (random characters) (eg. "jdhf8udb")
5. 0 length search term

6.1.2 Acceptance Criteria

1. Tests 1-3 should return 1 or more result.
 - a. Results should include "Margarita," "Long island tea," and "A1" respectively
2. Test 4-5 should not return any results and display a message conveying this fact
3. Each search result should display an image and category

6.1.3 Results

All tests passed

6.2 FEATURE: COCKTAIL RECIPES

Display cocktail recipes

6.2.1 Testing Methodology

Open a drink page and view its recipe (eg. "Margarita").

6.2.2 Acceptance Criteria

1. A list of ingredients must display
2. Each ingredient must display an image (or a placeholder if none available)
3. Each ingredient must also display its energy content
4. Each ingredient must display its quantity

If using Margarita, recipe must match appendix item 10.2.

6.2.3 Results

All tests pass

6.3 FEATURE: NUTRITION INFORMATION

Display aggregate nutrition information on cocktail recipes

6.3.1 Testing Methodology

Open a drink page and view its recipe and nutrition information (eg. "Margarita").

6.3.2 Acceptance Criteria

If using Margarita, nutrition results must match those in appendix item 10.3.

6.3.3 Results

When all ingredients are known by Nutritionix, all tests pass. Because Nutritionix does not contain every ingredient present in TheCocktailDB, the substitution system was developed to provide estimates of nutritional information by substituting unknown ingredients for known ones.

6.4 FEATURE: RECIPE EDITING

The ability to adjust the quantity of each ingredient in a cocktail recipe and see updated nutrition information

6.4.1 Testing Methodology

1. Open a drink page and view its recipe. Edit one or more of its ingredients to a higher or lower value and change the unit.
2. Open a drink page and view its recipe. Trigger editing on one or more of its ingredients but do not change the value

6.4.2 Acceptance Criteria

For test 1, the nutrition information must update accordingly, and a notification be displayed beside the ingredient indicating it was edited.

For test 2, the nutrition information must not change, a notification should still be displayed beside the ingredient indicating it was edited.

6.4.3 Results

Initial testing resulted in Nutritionix API usage limits being exceeded quickly. After caching was implemented, all tests pass consistently.

6.5 FEATURE: INGREDIENT INFORMATION

The ability to view ingredient information

6.5.1 Testing Methodology

Open a drink page and click one of its ingredients

6.5.2 Acceptance Criteria

A page must open containing the ingredient name, image, category, and a list of drinks that contain that ingredient.

6.5.3 Results

Mostly passes, occasionally no results will show in the "drinks containing ingredient" list when there are actually drinks that contain the ingredient. This issue is unavoidable as it is a problem with TheCocktailDB, and there is no way to test for the issue outside of human intervention.

6.6 FEATURE: RANDOM COCKTAIL

The ability to view a random cocktail

6.6.1 Testing Methodology

Click the random link

6.6.2 Acceptance Criteria

A random drink page must open. It should be seemingly random and usually be different to the last drink that opened.

6.6.3 Results

Passes

6.7 FEATURE: INGREDIENT SUBSTITUTION

The ability to substitute a nutritional equivalent for unknown ingredients

6.7.1 Testing Methodology

Clear the “unknownIngredientsMap.json” file and reload the server. Hit random cocktail until one appears that contains an unknown ingredient.

6.7.2 Acceptance Criteria

A page allowing the user to replace the unknown ingredient should display. It should contain the name and image of the unknown ingredient, along with a search box to find a suitable substitute. The form should reject submission unless the substitution has been validated as a known ingredient.

After all necessary substitutions have been made, the cocktail should load as normal with no changes to the ingredient list. The system should never ask to replace an already replaced ingredient.

6.7.3 Results

Passes

7 POSSIBLE EXTENSIONS



The following is a number of features that were considered, but did not make it into the final product, they could be added later to improve the product:

- The ability to build entirely new cocktails and view the nutrition information for them
- The ability to add/remove ingredients from cocktails
- Nutrition information updates in real time as you edit
- Combining multiple nutrition API's to achieve a wider coverage of ingredients

8 USER GUIDE

1. Open your web browser and navigate to where the site is hosted.
2. Either search for a cocktail using the search bar, or select a random cocktail
3. You will be presented with the recipe and nutritional information of that cocktail
4. From here you can edit the quantities of each ingredient by clicking the edit icon next to each ingredient line:

Ingredients

Image	Name	Amount	Energy (kJ)
	Tequila	1.5 oz (44 mL) 	403 kJ


5. You can also view an ingredient's details and other cocktails by clicking its name

If you arrive at this screen:

Unknown Ingredient

We couldn't find any nutrition info for the ingredient: "peach vodka"
Please enter a similar/generic ingredient below, and we'll use the nutrition information from that instead.

Original:

 peach vodka

→

Submit

Nutritional Substitute:

apple vodka

penne vodka

vanilla vodka

carmel vodka

vodka

Simply enter a nutritionally similar ingredient in the box on the right. It may take a few tries as you need to select an option from the dropdown list.

9 REFERENCES

Docker Inc. (2019). *Docker Documentation*. Retrieved from Docker: <https://docs.docker.com/>

Eernisse, M. (2012). *Docs*. Retrieved from EJS: <https://ejs.co/#docs>

Node.js Foundation. (2017). *4.x API Reference*. Retrieved from Express JS:
<https://expressjs.com/en/api.html>

Node.js Foundation. (2019, January). *Node.js v10.16.3 Documentation*. Retrieved from Node JS:
<https://nodejs.org/dist/latest-v10.x/docs/api/>

10 APPENDIX

10.1 FULL DOCKERFILE

```
FROM node:10.16.3

# Create app directory
WORKDIR /usr/src/app





# Install app dependencies
COPY ./SkinnySip/package*.json ./

RUN npm install
# If you are building your code for production
# RUN npm ci --only=production

# Bundle app source
COPY ./SkinnySip/. .

EXPOSE 80
CMD [ "node", "server.js" ]
```


10.2 MARGARITA RECIPE

Image	Name	Amount	Energy (kJ)
	Tequila	1.5 oz (44 mL)	403 kJ
	Triple Sec	0.5 oz (15 mL)	143 kJ
	Lime Juice	1 oz (30 mL)	32 kJ
	Salt	0.25 tsp	0 kJ

10.3 MARGARITA NUTRITIONAL INFORMATION

	quantity per serving	% daily intake ▲ per serving	quantity per 100g
ENERGY	578 kJ	7%	669.8 kJ
PROTEIN	0.1 g	<1%	0.2 g
FAT, TOTAL	0 g	<1%	0 g
SATURATED	0 g	0%	0 g
CARBOHYDRATE	6.2 g	2%	7.2 g
SUGARS	4.2 g	5%	4.8 g
DIETARY FIBRE	0.1 g	<1%	0.1 g
SODIUM	582.5 mg	25%	675 mg
POTASSIUM	37.2 mg	1%	43.2 mg
CHOLESTROL	0 mg	-	0 mg
		% RDI*	
CALCIUM	4.7 mg	1%	5.5 mg
IRON	0.1 mg	<1%	0.1 mg
ALCOHOL, ETHYL	16.8 g	-	19.5 g
ASH	1.6 g	-	1.9 g
BETAINE	0.1 mg	-	0.1 mg
CAROTENE, BETA	9.2 µg	-	10.7 µg
CHOLINE	1.6 mg	-	1.8 mg
FOLATE, TOTAL	3.1 µg	2%	3.6 µg
FRUCTOSE	0.2 g	-	0.2 g
GLUCOSE	0.2 g	-	0.2 g
MAGNESIUM	2.5 mg	1%	2.9 mg
NIACIN	0.1 mg	1%	0.1 mg
PHOSPHORUS	6.3 mg	1%	7.3 mg
SELENIUM	0.1 µg	<1%	0.1 µg
SUCROSE	3.8 g	-	4.4 g
VITAMIN E	0.1 mg	1%	0.1 mg
VITAMIN A	0.6 µg	<1%	0.7 µg
VITAMIN C	9.2 mg	23%	10.7 mg
VITAMIN K	0.2 µg	<1%	0.2 µg
WATER	61.5 g	3%	71.3 g