4/22/2020

TACKZO

Application Documentation

Steven Joyce & Evan Greaney
TACKZO

Table of Contents

Introduction

Definition of Project

Tackzo is a web application designed around the food sector , it is a basic prototype of a future web application that aims to provide customers and users with the ability to search for food recipes based on calorie intake, amount of people to serve and by ingredients.

This is a three tiered web application that contains a front end, local Storage and a global database to store all user input, it uses the API Spoonacular to call Recipes and sends out information on given ingredients based on the users demands.

Team

Evan Greaney G00348451

Coverage of Work done

Front End – Saved Meals

Back End - Store Meals

API calling

Steven Joyce G00362012

Coverage of Work done

Front End – Store Meals

Back End – Saved Meals

Server

Supervisor

Kevin O'Brien

System Requirements

Basic Requirements

- Windows or Linux
- Visual Studio code or IDE that can run Ionic
- Internet Connection

Technology used and why?

Prerequisites Required To Run

Node

- Download Node.js from https://nodejs.org/en/download/
- Choose version subject to your system
- Follow wizard until Finish

Ionic Framework

- npm install Ionic

Angular

- npm install angular

Cors

- npm install cors

Mongoose

- npm install mongoose

Express

- npm install express

The following is needed to enable the running of this project, the above commands must be ran in the command line of the folder that is in use. Install the Prerequisites in the order as seen above.

Why the technologies we used?

We decided on Ionic due to its capabilities of being able to run on both mobile and web with minor changes such as screen size and layout design. It uses a component based system where rather than changing to new pages, it's a single page application that runs multiple components.

It also allows us to see live changes and automatically detect issues and the debug the project.

The ability to use API's and web servers added to the appeal of the framework that allowed us to come to a final conclusion on what technologies to use.

How to Run Web Application

1. As above download Node.js
2. Dowload Zip or Clone Repository from https://github.com/EvanGreaney/TACKZO.git
3. Open the the downloaded/Cloned Folder called TACKZO
4. Run CMD in the file path
5. As above follow the above commands from Ionic Framework to Express
6. Open code in preferred IDE from the command line or the IDE itself
7. Open BackEnd Folder to access mongoConnect.js file
8. Open this File in the terminal
9. Run the following code: node mongoConnect.js

```
PS C:\Users\Steven\Desktop\GroupprojectTackzo\TACKZO\Backend> node mongoConnect.js
Connected to port on http://localhost:4000
Connected to Mongo Database!
```

10. Go back to the command line of the project and enter Ionic Serve

```
C:\Users\Steven\Desktop\GroupprojectTackzo\TACKZO>ionic serve
> ng.cmd run app:serve --host=localhost --port=8100
[ng] i ｢wds｣: Project is running at http://localhost:8100/webpack-dev-server/
[ng] i ｢wds｣: webpack output is served from /
[ng] i ｢wds｣: 404s will fallback to //index.html
```

11. The application should now be running

Architecture of Solution

Front End

This is an IONIC Application in the Tabs style of UI, it allows the user to switch freely between tabs to access various components of the Application by clicking the specific tabs on the Tabs Toolbar.

Home

This is a rather simple page. It hosts the logo of the App and is the first component the User accesses.

Saved Meals

This is a display of all Recipes searched by the user

Search Meals

This is a navigation page to allow the user to choose between three options: Search by Calories, Search by Ingredients or Search by food Item. All of these are buttons which have a click event that sends the user to the relevant page.

All Pages have a form that is used to store user data.

Search by Calories Page

The User Inputs a number that is used to generate a meal that is inside that parameter. The input is saved by pressing the submit form button. The number is the max number of calories that meal can be worth.

Search by Ingredients Page

The user inputs food items that is stored in a form. It is stored by pressing the submit button. The form is dynamic, as the user can add a new ingredient by pressing the add ingredient button. This lets the user choose the size of the ingredient list that is used to generate a meal.

Search by Food

The user inputs a food item that is stored in a form by pressing the submit form button. The input is saved in local storage and used for an API call. The API call then sends the user information about the food item inputted.

Local Storage

Why is it used?

The Ionic Offline Storage is a way to store input form the user in the local scope of the project and be used only in that component scope.

How is it implemented?

When the user populates the form and saves the data. The form value maxCal is used to populate the maxCalories variable in the ts page. The maxCalories variable is a variable that is found in the local storage.

Global Server

What database is used?

With this project, we use the MongoDB Database. It is a NoSQL database.

Why?

This database type is a database I am familiar with from learning React in Year 3 of this course. This database can be accessed using HTTP requests. The data can be converted easily from and to a JSON data type.

API

The API was one of the main features of the Application as it allowed the user to retrieve recipes based on what they wanted whether it was for ingredients, the calorie size or Food types

What API was used?

The API that was used was Spoonacular API.

Why did you chose this?

It is an API that allows us to retrieve recipes based on the calls made whether it's through searching food for different cuisines , searching for recipes with calories in mind or based on what's in your pantry stating what ingredients you got and it will find a recipe based on that.

Design Methodologies

One of the main philosophies that we incorporated in this app was the same one Apple incorporates into their design process and it was that of simplicity , we wanted to make the Application as simple for the user to use as possible.

We included a three-tab system that if ported to a mobile device, it allows the user to simply swipe through the components with ease and access the pages at their own leisure.

The first Tab is simply a Home Page that shows the Logo of the App.

The second tab is where the user can simply find any recipe that previously search for based on the categories provided.

The third tab allows the user to search for Recipes based on the buttons provided, when they click on the buttons, they are redirected to the component that searches particularly to their request.

Search Calories
All the user must do is enter the number of calories they want in the meal and it will return back recipes that is in line with that request.

Search Food
When brought to this component, the user can enter any cuisine type, whether its burgers or pasta and when they press submit, it will bring them back a recipe of that variety.

Search Ingredients
To allow ease of use we designed this section to work with the user in mind, all the user has to do is enter the ingredients they currently have and it will retrieve a recipe with the ingredients they have and what they must get to make this recipe.

Stored Meals
With this section we have three buttons and depending on what they click it will send back the recipes they searched for previously under those headings.

Conclusion
We have tried to design this Application with the user in mind and to keep it as easy to use as possible.

Features of Implementation

The Application is an ionic app that generates API calls and stores user input in a NoSQL Database. It is designed for single user use.

API Implementation

The User will input a parameter that is stored that is stored in a form. The input is used to generate a search parameter for an API call. This input is then sent to the Spoonacular API with a HTTP GET request. The API receives this data and generates a result depending on the input is has received. The result is sent back to the page that the input was sent from and outputted to the user inside an ion card.

Database Implementation

The user will input data into the page. That data is stored in a form. This form is sent to a class inside the application, this class is then sent to a local port using a HTTP POST request. This post request sends the data to the local port created. The local port is 4000. The data on the port is added to a mongoose schema. The mongoose schema translates the data into a JSON data type. The JSON data type is sent to the Mongo Database in the MongoConnect.js page.

Limitations and Known Bugs

Limitations

API

The API is great being one of the few recipes searching API's that is free but limited to amount of calls per day and also gives a decent variety of searches that can be done. All be it being a decent API it does have its setbacks , when you get a search result back , you get ingredients with the search by ingredients but no steps on how to make the recipe, this is a flaw with the Spoonacular API itself and not the Application.

The API returns very basic information on the recipe but never enough to actually make the recipe and it doesn't even offer a link to go to a page that has the recipe.

This is a major limitation of the API that we found while trying to develop this Application.

Ionic Storage

The scope that saving local storage to be accessed is very limited.

The stored data can only be accessed on that page. The stored data is a good way to set and get data that is needed in more than one method in that page. The only drawback is not being able to access the sorted data on other pages.

This limitation led us to have to change the way we output results from the api. We decided to print them on the same page rather than send data to local storage and access on a page called meal choice.

Known Bugs

SearchFood Component and Saved Meals Componet

A known bug in this component is within search-food.page.html , it is in the ion-card segment of the this page where *ngFor = "let recipe of Recipes" seems to not find the Array Recipes, this is a bug as that line is a carbon copy of the same line from the search-calories component and search-ingredients component , both .ts pages of the above were written exactly similar and we know there is no error in the .ts files of the above pages as the API calls correctly and stores the information on the MONGO database, it can be seen working in the console and the results are sent to the database which can be later viewed in the Saved Meals Tab.

This bug limits the view of the search in the page but the results are still retrieved and shown in the console. The issue is with that line but neither of us can see why it is causing this issue.

The same issues does exist on the tab2.ts page. This bug can sometimes stop the user from seeing the data in the MongoDB database on the savedmeals.html page. It has also stopped the application from running, so the error can be fatal. It commented out in the code on line 29 of the tab2.ts page.

Issues

Compliance Issues

Problem

A global pandemic called Covid-19 led to a national shutdown of all colleges and schools. We could also not have face to face meetings due to a government ban on people leaving home for non- essential reasons such as work or shopping for essential items.

Solution

To counteract this issue, we began having meetings on Microsoft teams and text messaging on WhatsApp. This was how we communicated for the past month to complete this project.

Difficulties with development

The initial Plan

Initially our App was going to be a shopping saver app that could read barcodes, get product information based on the barcodes and to act as a Shopping assistant app that would recall what you spent in your last shopping bill and either try find your usual shopping list and show you or what is different compared to your previous shopping trip.

This all changed when we realized the timeframe was not in our favor to even attempt this ambitious project so we instead went back to the drawing board to create a new , not too far from the original theme but that it would feel far more comfortable to complete in our time frame.

Previous Github Repository of the old project:

https://github.com/EvanGreaney/3rdYearGroupProject.git

Issues with GitPod

We tried to be as innovative as possible when it came to working on our project so in the development cycle we attempted to use Gitpod as to develop simultaneously and to be able to work on it from anywhere without have to download the specific programs and files necessary to run. This came with drawbacks for Ionic and the other programs necessary for it to run, it seemed to dislike what were using and failed to run whenever we tried to use GItpod.

Ionic Framework

When we took on this project, we attempted to keep this project as modern as possible by trying to use the latest versions of our frameworks, servers, API's …etc. This came with many drawbacks as because they were so new , documentation and professional knowledge was few and far between , we had confidence in our abilities as we have had some knowledge of

the technologies we were using due to previous modules and projects. We chose Ionic as for this reason but soon came to realize the flaw in our endeavor as with the jump from Ionic 3 to 4 , massive changes were made to Ionic and were used to using Ionic 3.

Unfortunately, Ionic was actually on version 5 and was fairly new with little to no documentation on it except for the documentation on Ionic 4. This really made it difficult for us as now we had to basically go and learn a new Framework as to attempt our project

Issues found by Evan

API Issues

Problem

Limitations of Spoonacular

The Spoonacular API is great in regard to what is available on the current market but because of limited searching power, we were limited in our design scope and had to design the entire app based on what Spoonacular could provide.

Spoonacular itself changes the API key of the user every now and again causing difficulties with keeping the app more robust and causing as little problems as possible. Unfortunately this is an issue we couldn't solve as it was an issue on the API's side

Issues found by Steven

Front End Problems

Problem

The ionic 4 framework changed how to navigate to a different page on the application.

Solution

Now to navigate to new pages you must import the router Module from@angular/router. The app-routing.module.ts page is used to create a array of const.

```
import { PreloadAllModules, RouterModule, Routes } from '@angular/router';

const routes: Routes = [
  {
    path: '',
    loadChildren: () => import('./tabs/tabs.module').then(m => m.TabsPageModule)
  },
```

This array is used to create paths that can be accessed by calling the route name.

```
import { Router } from '@angular/router';

@Component({
  selector: 'app-tab3',
  templateUrl: 'tab3.page.html',
  styleUrls: ['tab3.page.scss']
})
export class Tab3Page {

  constructor(private router: Router) {}

  calories()
  {
    this.router.navigate(['search-calories'])
  }
```

Local Storage

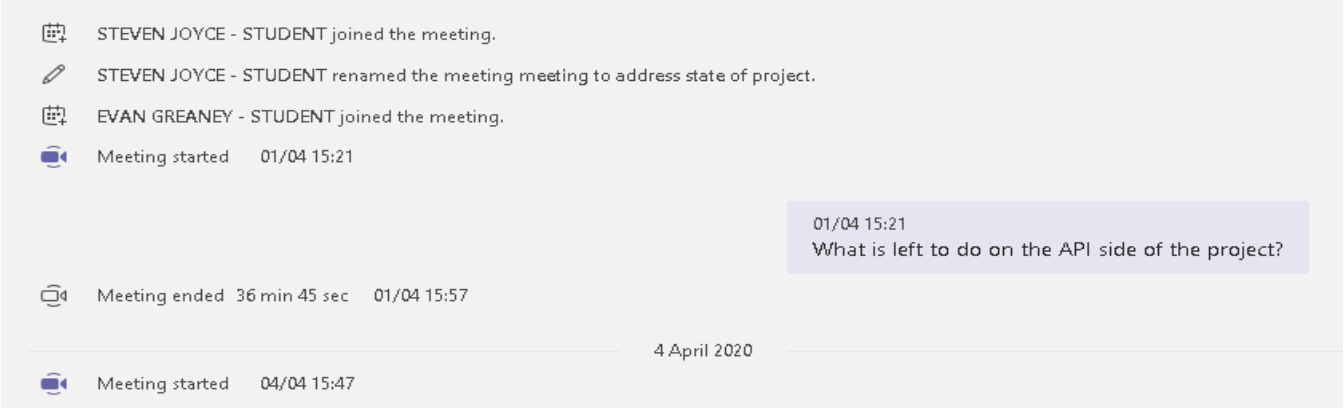
Problem

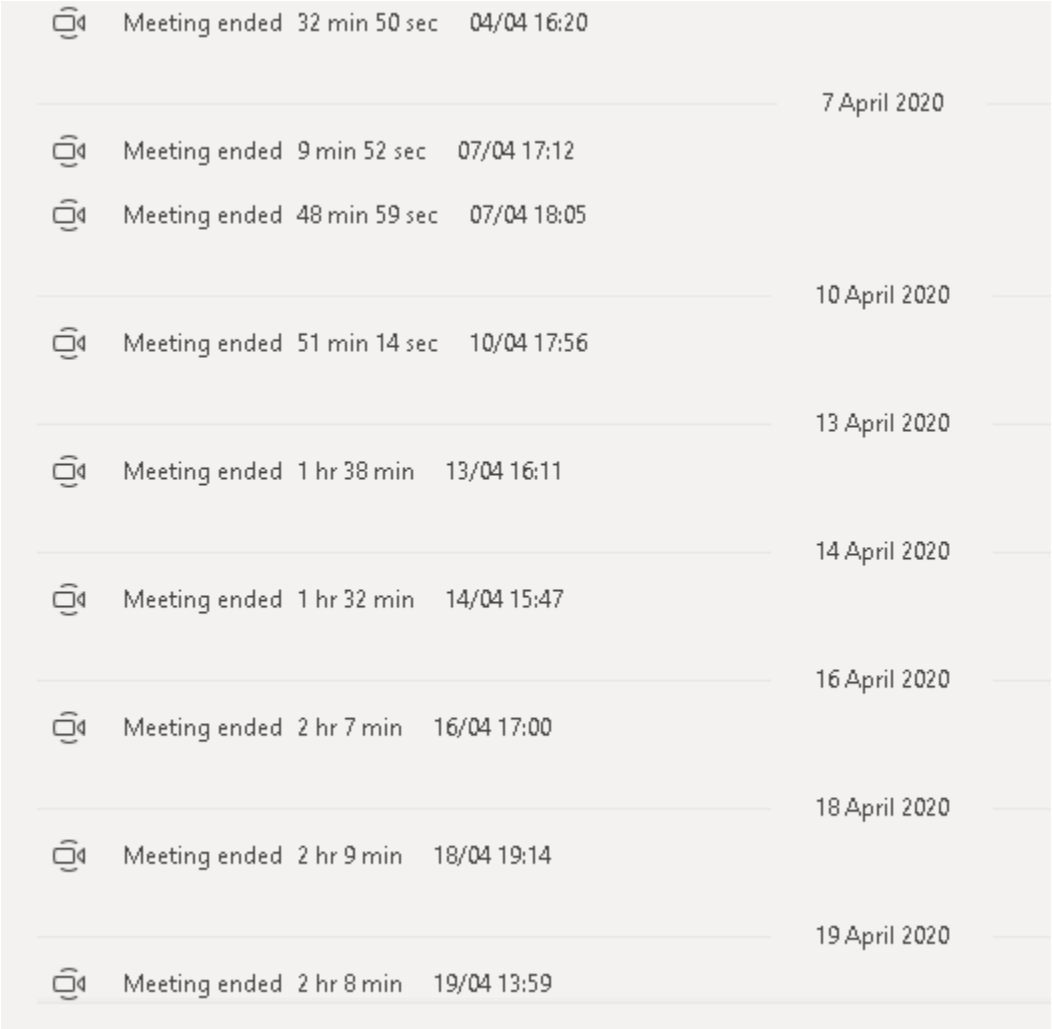
Ionic does not have a direct way to access MongoDB. MongoDB is an open source database management system that is a NoSQL database program.

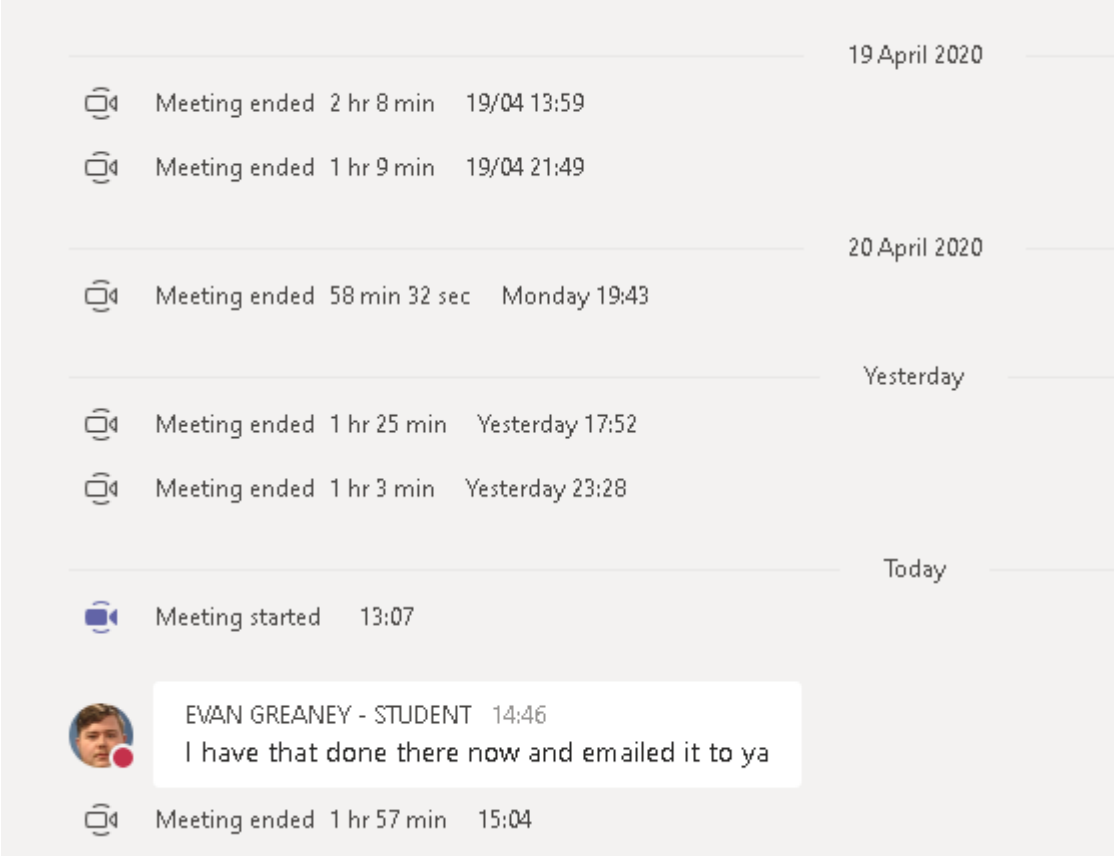
Solution

The references to how to access a database is not easy to find and the main documentation is written with NestJS, a program I am not familiar with. I followed the react concept for connecting to a MongoDB Server. I installed Cors, Express and Mongoose. I created a BackEnd folder and sub folders called Schema and Routes. In the Backend folder a js file is used to connect to the MongoDB database. The routes folder has separate js files that is used for HTTP requests to the MongoDB database that will add to the database and receive data form the database. The Schema folder is used to create mongoose schemas. Mongoose is used to translate data from the application to JSON to populate the database, it also translate JSON to data that can be used in the application.

Documentation of meetings







Meeting Agenda

The meetings we first used to see where we were on the project.

We then solved issues using teams with voice chats.

During the last week we wrote our documentation together on Teams.

Testing Plans

Storing the User Inputted Data

Search by Calories

Test 1

```
max Calories in On Form submit200    search-calories.page.ts:39
Max number of calories per meal:  200  search-calories.page.ts:64
```

Test 2

```
max Calories in On Form submit800    search-calories.page.ts:39
Max number of calories per meal:  800  search-calories.page.ts:64
```

Search by Ingredients

Test 1

```
Ingredients in On Form submitapple,orange,lemon  search-ingredients.page.ts:46
Ingredients of meal: apple,orange,lemon  search-ingredients.page.ts:74
```

Test 2

```
Ingredients in On Form submitchicken,noodles  search-ingredients.page.ts:46
Ingredients of meal: chicken,noodles  search-ingredients.page.ts:74
```

Search by Food

Test 1

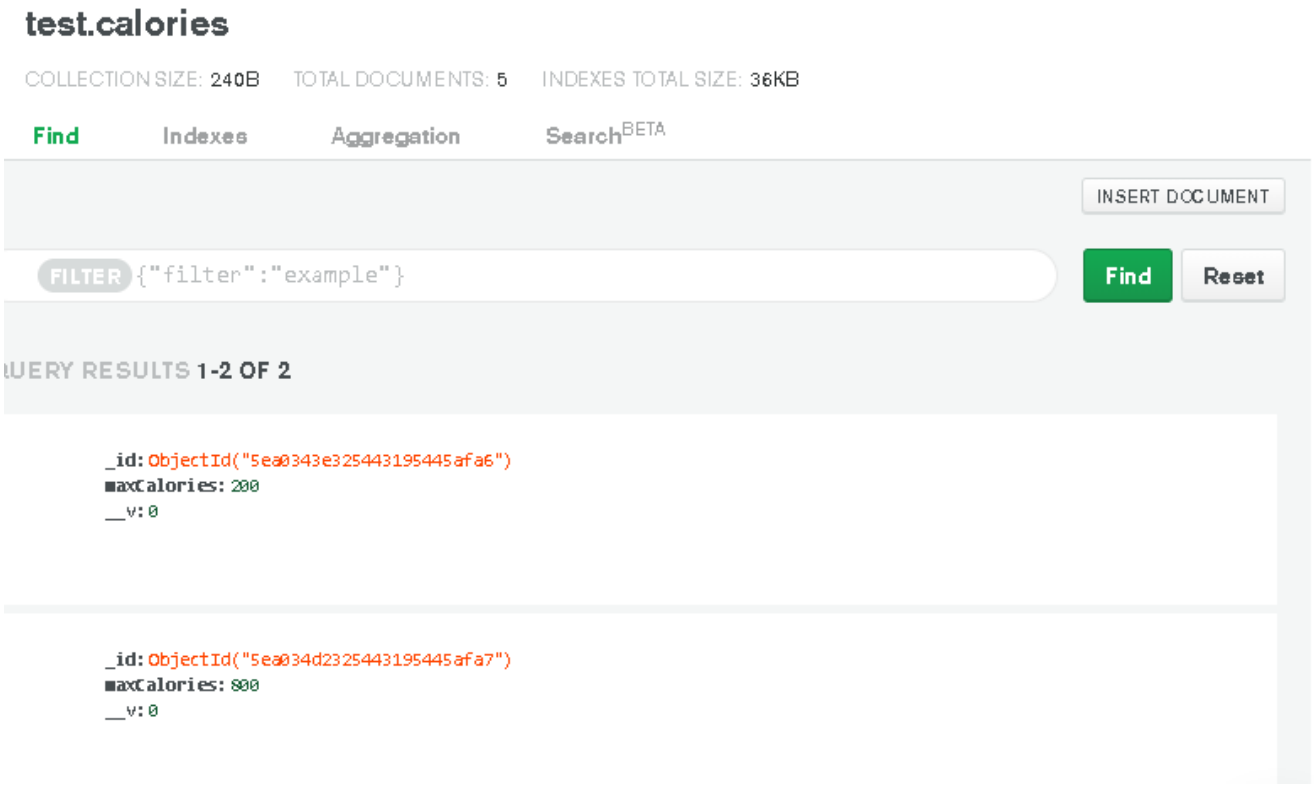
```
food in On Form submit burger    search-food-type.page.ts:39
foodChoice:  burger    search-food-type.page.ts:63
```

Test 2

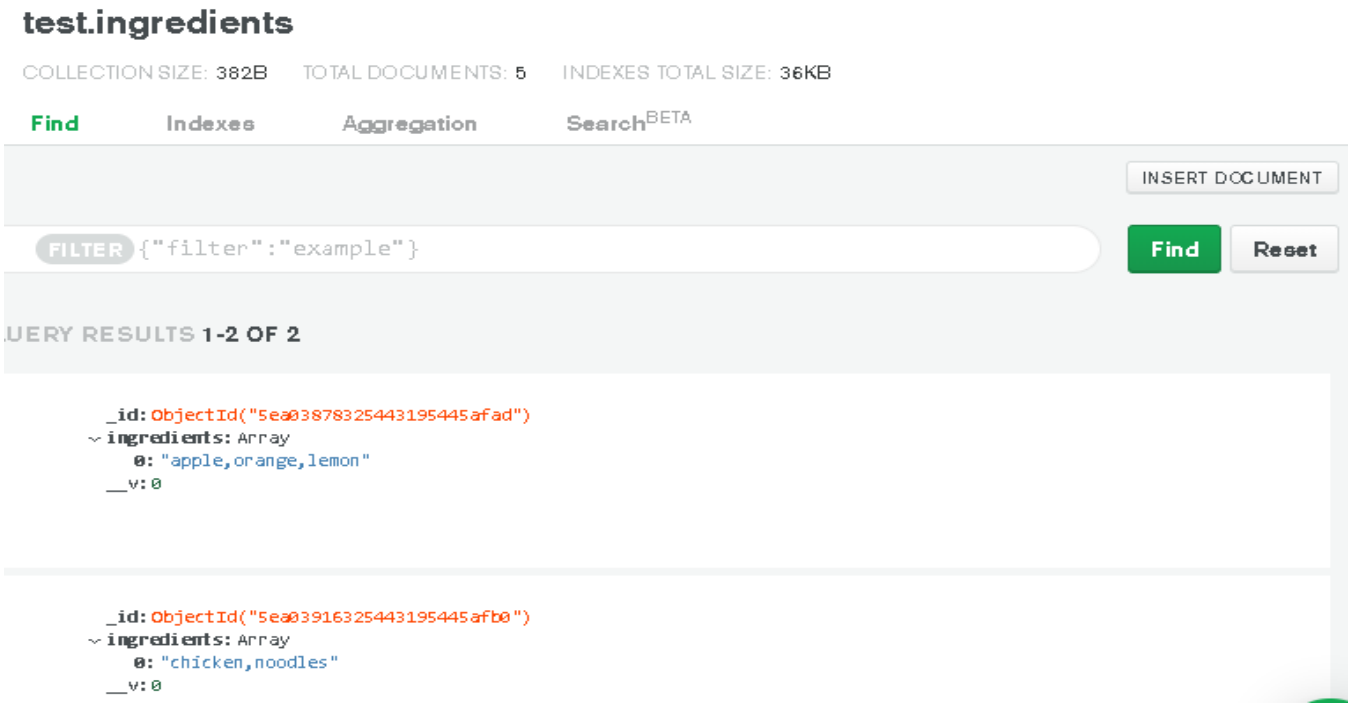
```
food in On Form submitcurry          search-food-type.page.ts:39
foodChoice:  curry                   search-food-type.page.ts:63
```

Storing the User Input in the MongoDB Database

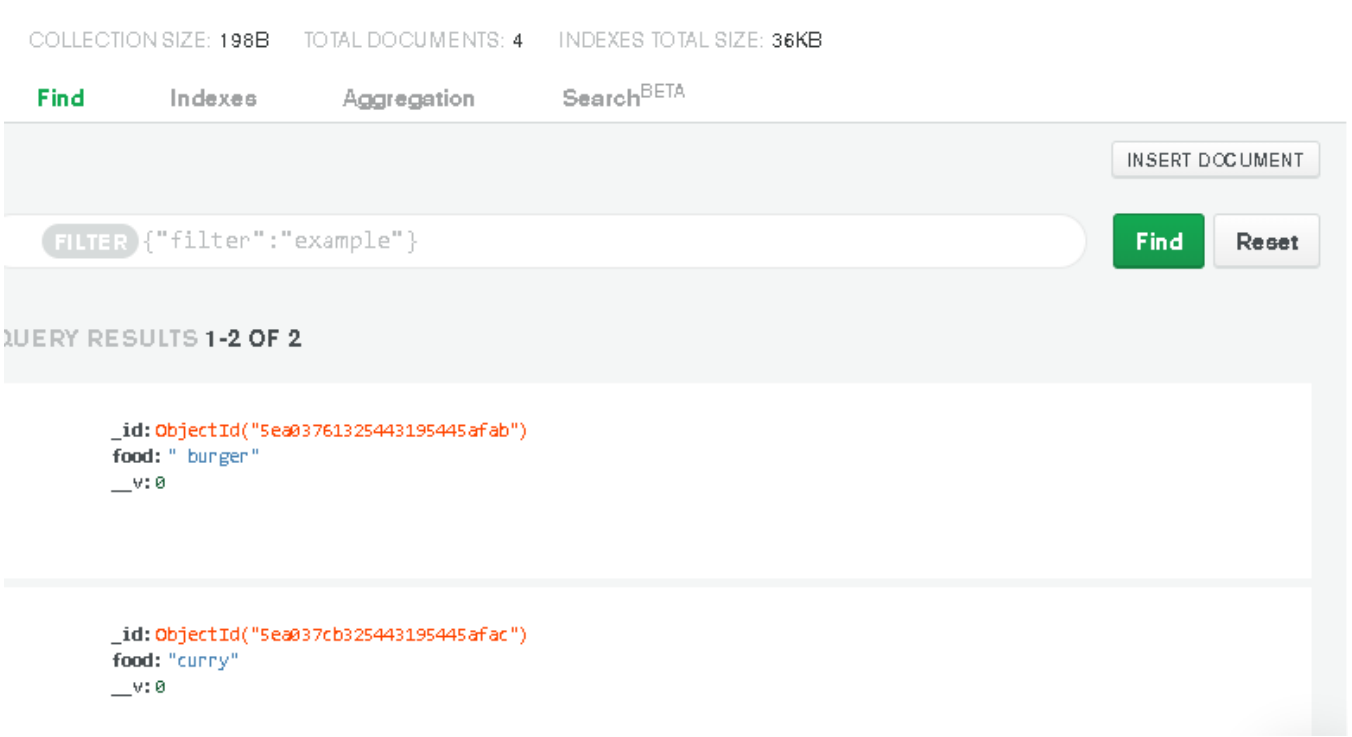
Search by Calories



Search by Ingredients



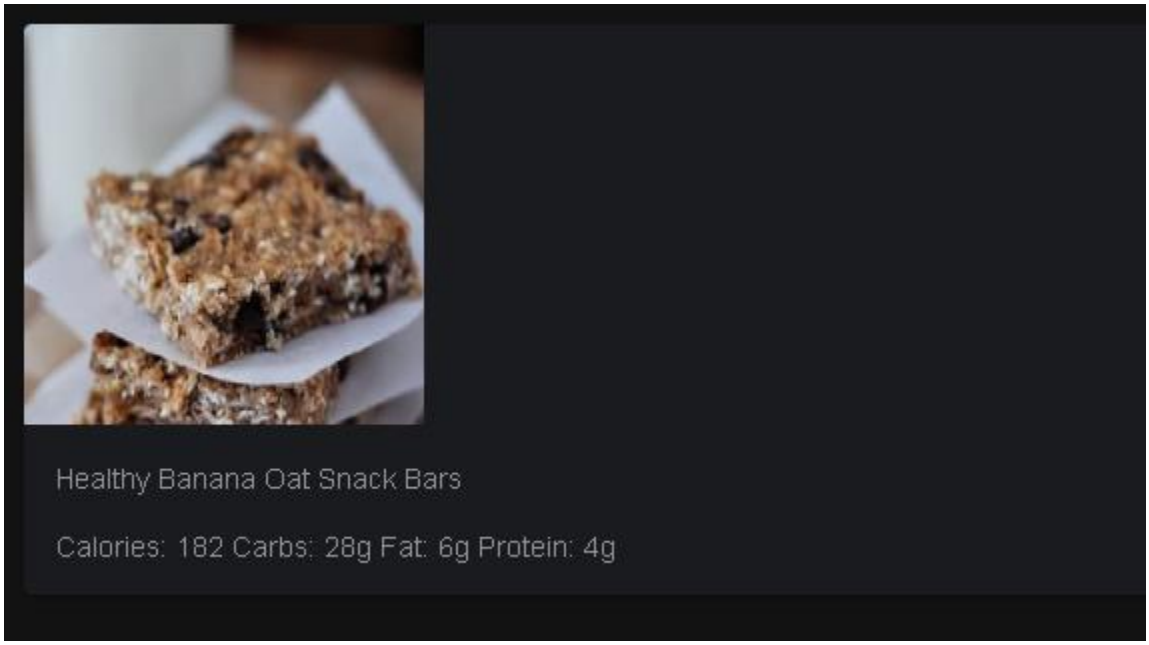
Search by Food



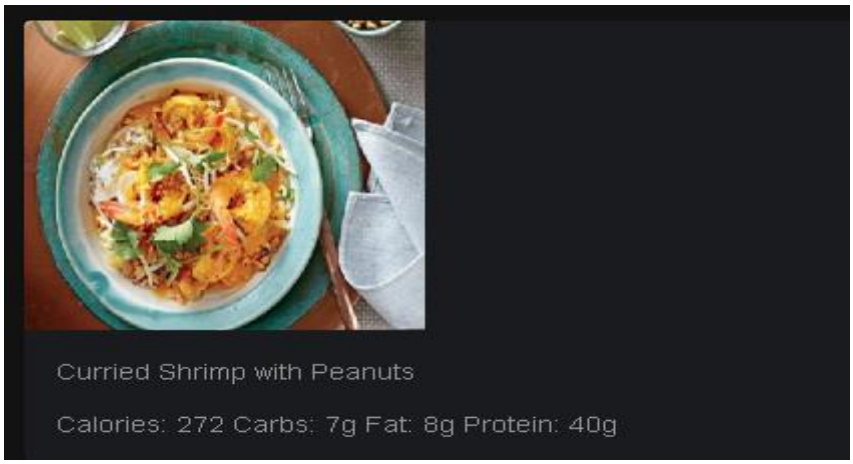
Generating the API Call Results

Search by Calories

Test 1

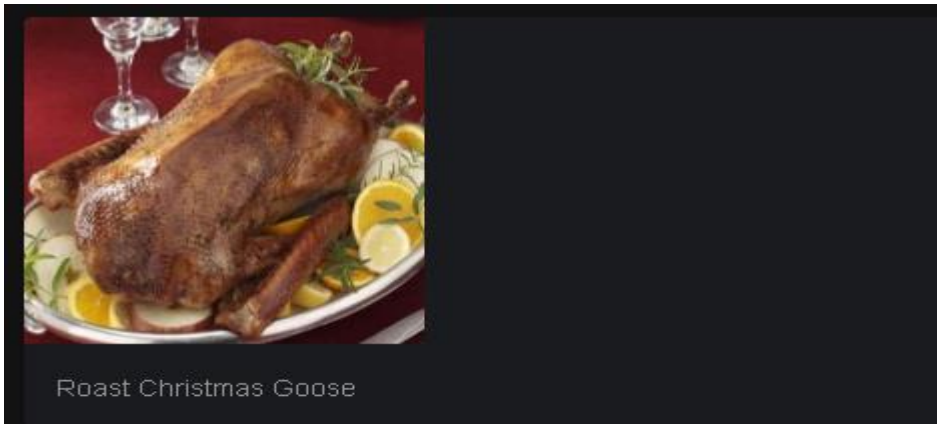


Test 2

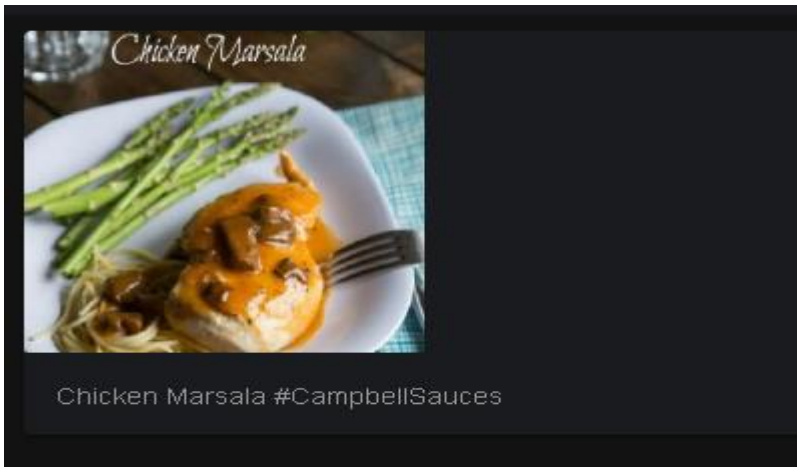


Search by Ingredients

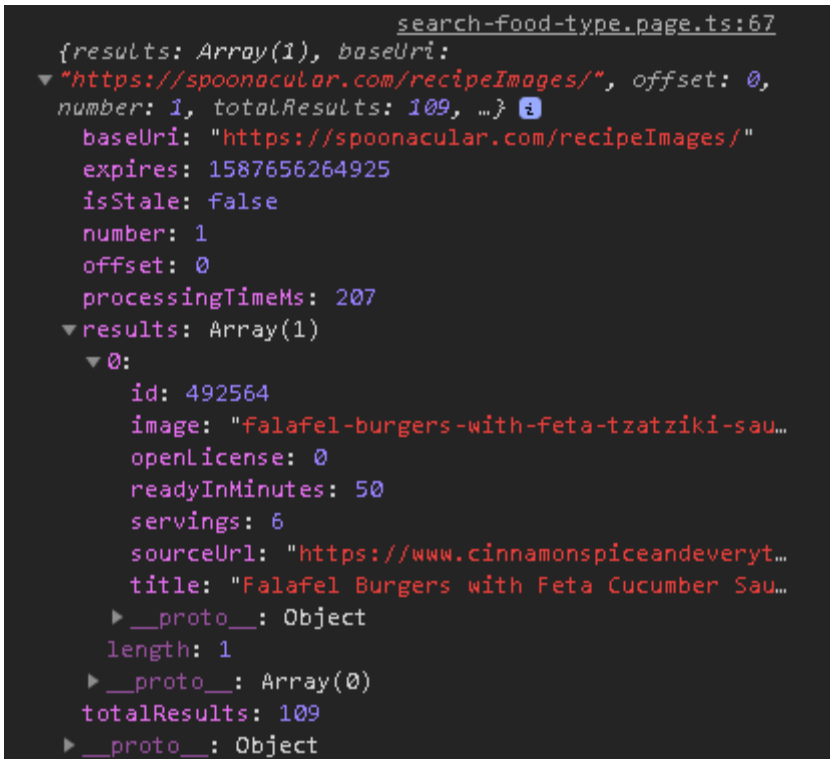
Test 1



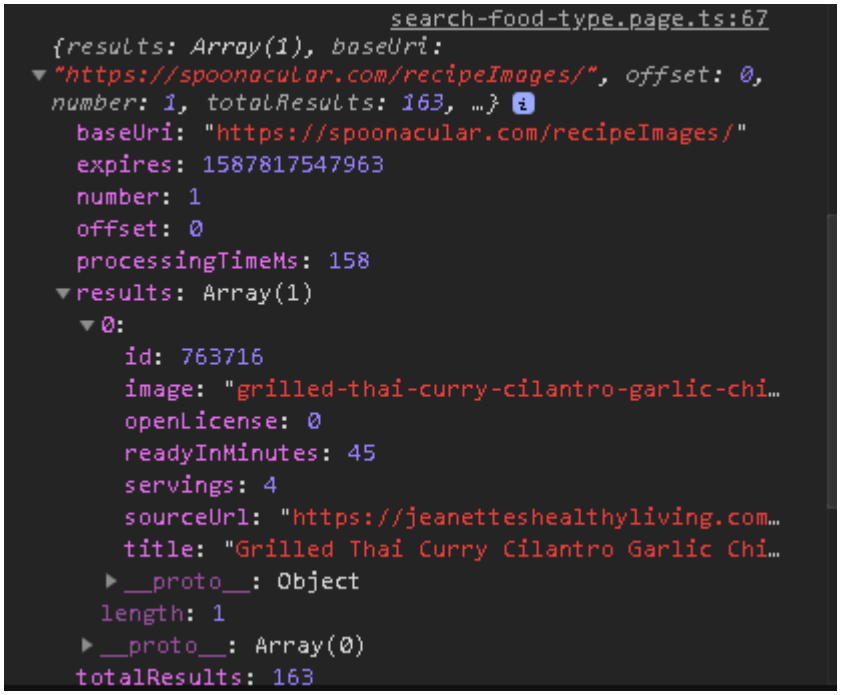
Test 2



Search by Food
Test 1

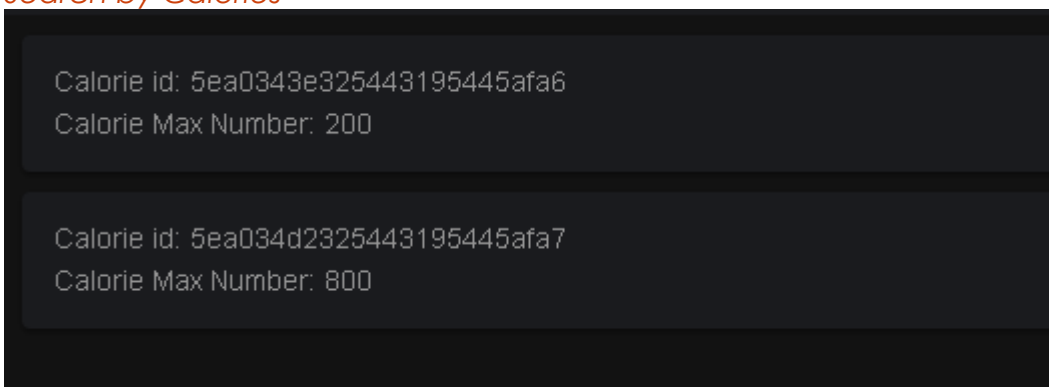

search-food-type.page.ts:67
{results: Array(1), baseUri:
"https://spoonacular.com/recipeImages/", offset: 0,
number: 1, totalResults: 109, …}
baseUri: "https://spoonacular.com/recipeImages/"
expires: 1587656264925
isStale: false
number: 1
offset: 0
processingTimeMs: 207
results: Array(1)
0:
id: 492564
image: "falafel-burgers-with-feta-tzatziki-sau…
openLicense: 0
readyInMinutes: 50
servings: 6
sourceUrl: "https://www.cinnamonspiceandeveryt…
title: "Falafel Burgers with Feta Cucumber Sau…
__proto__: Object
length: 1
__proto__: Array(0)
totalResults: 109
__proto__: Object

Test 2

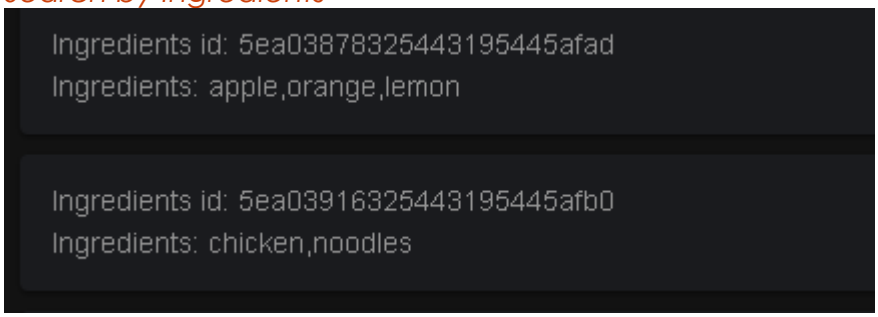

search-food-type.page.ts:67
{results: Array(1), baseUri:
"https://spoonacular.com/recipeImages/", offset: 0,
number: 1, totalResults: 163, …}
baseUri: "https://spoonacular.com/recipeImages/"
expires: 1587817547963
number: 1
offset: 0
processingTimeMs: 158
results: Array(1)
0:
id: 763716
image: "grilled-thai-curry-cilantro-garlic-chi…
openLicense: 0
readyInMinutes: 45
servings: 4
sourceUrl: "https://jeanetteshealthyliving.com…
title: "Grilled Thai Curry Cilantro Garlic Chi…
__proto__: Object
length: 1
__proto__: Array(0)
totalResults: 163

Output the Data stored in the MongoDB Database

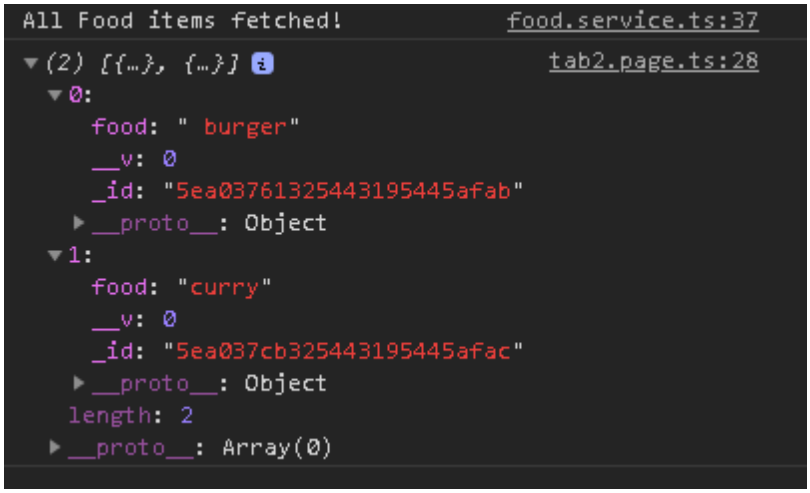
Search by Calories



Search by Ingredients



Search by Food



Recommendations for Future Development

Application Features

User MongoDB Database

Using HTTP Requests, the user can update or delete the data stored in the MongoDB database. With buttons in the saved meals page, the user can access new pages on the application.

Update data Button

The user is navigated to a new page call Update Data. With this page all data stored in the MongoDB Database is called in and modified by the user. When the user clicks an update button which is below each data entry, all changes are added to the MongoDB Database with a HTTP request called PUT.

Delete data Button

A new page is accessed called Delete Data. All data is called into the page from the MongoDB database after each data input a button called delete will be able to be clicked on the page. When this button is clicked, the saved data is deleted from the page and in the Global storage. It will use the HTTP request called DELETE.

Conclusions

This project was a challenging application to code. When we first started out, we thought our previous knowledge of Ionic would make the Ionic side easy to work out. The major change Ionic made from version 3 to version 4 had a massive impact on this project. Trying to code an Ionic 5 application with no documentation for that version was a major issue to try and resolve. We used ionic 4 documentation to help us see what changes was made. The lack of documentation for using Spoonacular API with ionic and creating a server in ionic led us to learn how to implement them as we worked.

With this project we used our base knowledge of Ionic, Angular and React to form a basis of our project and we learned how to connect to the Spoonacular API and the MongoDB database ourselves.

References

Ionic
https://ionicframework.com/docs

Angular
https://angular.io/docs

Spoonacular
https://spoonacular.com/food-api/docs

Tackzo Github Link – main Project
https://github.com/EvanGreaney/TACKZO.git

Previous Github Link – old project
https://github.com/EvanGreaney/3rdYearGroupProject.git

HTTP Requests
https://www.tutorialspoint.com/http/http_requests.htm