

CAB432 – Assignment 1 API Mashup Report

Supervised by

Jim Hogan

Kevin Nguyen, n9463933

14/09/2019

Queensland University of Technology, Semester 2 2019

Contents

1.0	Introduction.....	3
1.1	Services Utilised.....	3
1.1.1	Zomato API	3
1.1.2	Google Maps Javascript API.....	3
1.1.3	Geocoding API.....	3
1.1.4	Distance Matrix API	4
2.0	Use Cases.....	4
2.1	Use Case A	4
2.2	Use Case B	5
3.0	Technical Documentation	5
3.1	Bootstrap	6
3.2	Express.....	7
3.3	Axios.....	7
3.4	Difficulties.....	7
4.0	Utilisation of Docker	7
5.0	Testing Plan	8
6.0	Extensions	9
7.0	References.....	9
8.0	User Guide.....	9

1.0 Introduction

UrbanBites aims to provide foodies with a convenient method to see the best reviewed restaurants around a specified location. Currently, most food review services only provide users with a list of restaurants from a location query. This makes planning holidays or choosing a meal extremely difficult in a foreign place as the daily activities can dictate where you eat. Hence, UrbanBites allows users to view the best rated restaurants near a given location based on a query.

Users can search for restaurants via providing a location, as well as the category (breakfast, lunch or dinner), the number of restaurants to display, and the ability to sort by restaurant's distance from your current location. With the restaurants obtained via the API mashup, a map with the appropriate markers will be populated for the user to view the restaurant details including reviews, rating and travel time.

1.1 Services Utilised

1.1.1 Zomato API

<https://developers.zomato.com/api>

Zomato, developed in India, is an Indian restaurant aggregator start-up providing users with a method to examine restaurants with their crowd-sourced review and rating system. With these services, Zomato also provides a set of 'callable' methods to retrieve information from their API endpoints. UrbanBites initially utilised Zomato's categories API endpoint to retrieve the important categories of food to allow users to choose from. However, for the web application itself, the search endpoint was utilised where a location, category of food, number of restaurants and sort by method was supplied for Zomato to return information about the surrounding restaurants.

1.1.2 Google Maps Javascript API

<https://developers.google.com/maps/documentation/javascript/tutorial>

The Google Cloud Platform provide a large suite of cloud services in conjunction with their large library of API endpoints. One of the main APIs is the Google Maps Javascript API, where Google's interactive map can be included in web applications with abundant options of customization. This API endpoint provides the basis of the map for the UrbanBites application.

1.1.3 Geocoding API

<https://developers.google.com/maps/documentation/geocoding/intro>

In addition to the API mentioned above, google also supplies a geocoding API endpoint to allow requests to convert location queries into geographical coordinates, more specifically latitude and longitude values which is utilised upon the location query given in UrbanBites.

1.1.4 Distance Matrix API

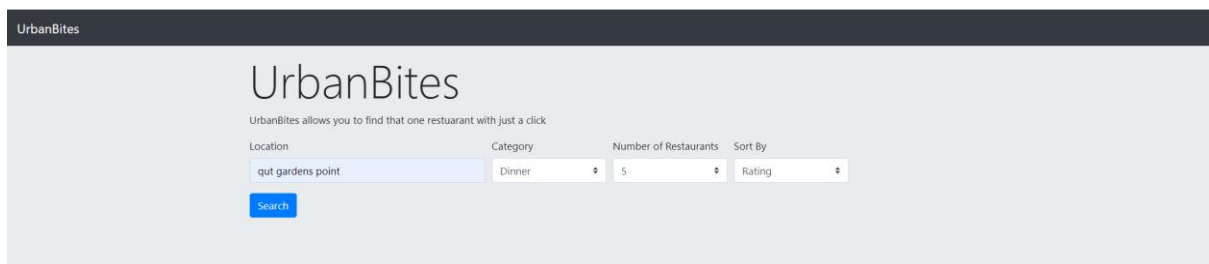
The Distance Matrix API provided by Google provides an endpoint for users to request the travelling details from an origin point to a destination. It provides information such as the travelling time and distance. This will be utilised in UrbanBites for the second use case where user requires to see if the restaurant is within a reasonable travel distance.

2.0 Use Cases

2.1 Use Case A

As a user travelling to a new location, I want to be able to visualise the restaurants around me so I can decide what to eat given local reviews and ratings.

The simplicity of the web application allows only one view for the user to navigate. The user only needs to provide their location, the category of food (optional), the number of restaurants to display and how to sort the restaurants. An example query is given below.

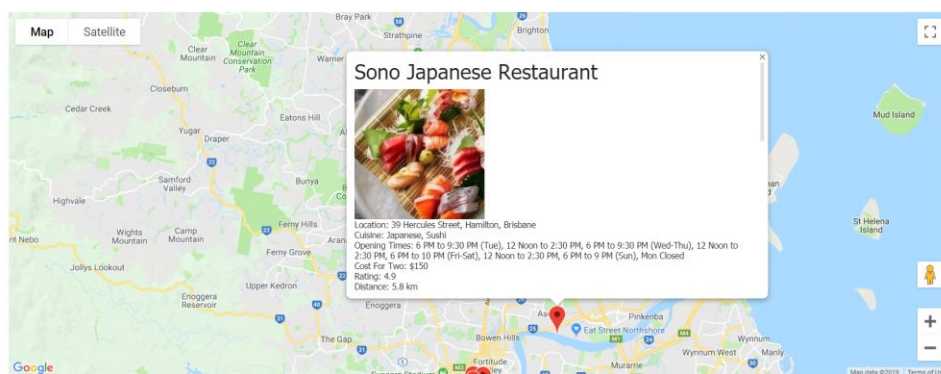


The screenshot shows the UrbanBites web application interface. At the top, the 'UrbanBites' logo is displayed. Below it, a search form is visible with the following fields: 'Location' (containing 'qut gardens point'), 'Category' (a dropdown menu set to 'Dinner'), 'Number of Restaurants' (a dropdown menu set to '5'), and 'Sort By' (a dropdown menu set to 'Rating'). A blue 'Search' button is located below the 'Location' field.

The API calls are performed upon submitting the form and a map below appears with the surrounding restaurants with the highest ratings. By clicking on one of the markers, the reviews and ratings of the restaurant can be viewed.

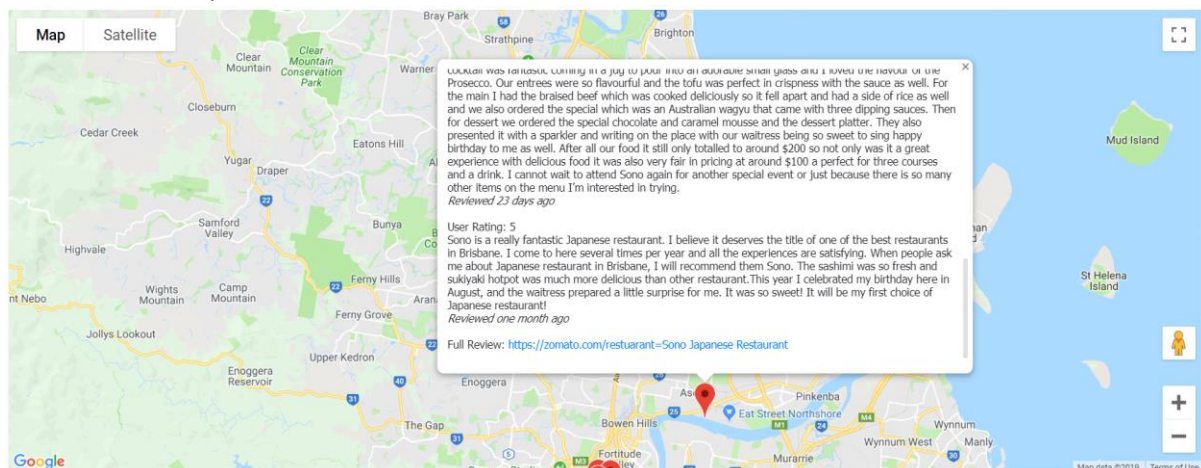


This screenshot shows the same UrbanBites search form as above, with the 'Location' field set to 'qut gardens point', 'Category' set to 'Dinner', 'Number of Restaurants' set to '5', and 'Sort By' set to 'Rating'. The 'Search' button is visible below the 'Location' field.



Search More »

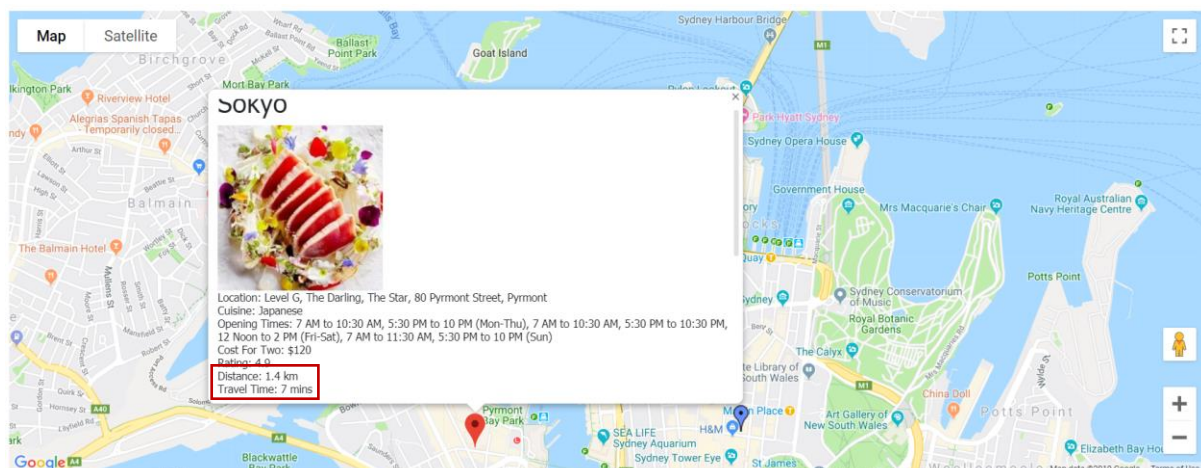
The reviews and ratings of each restaurant can be viewed by scrolling down the custom popup of the marker. An example is shown below.



2.2 Use Case B

As a user travelling to a new location, I want to easily see the restaurants around me and discern which one is feasible to travel to.

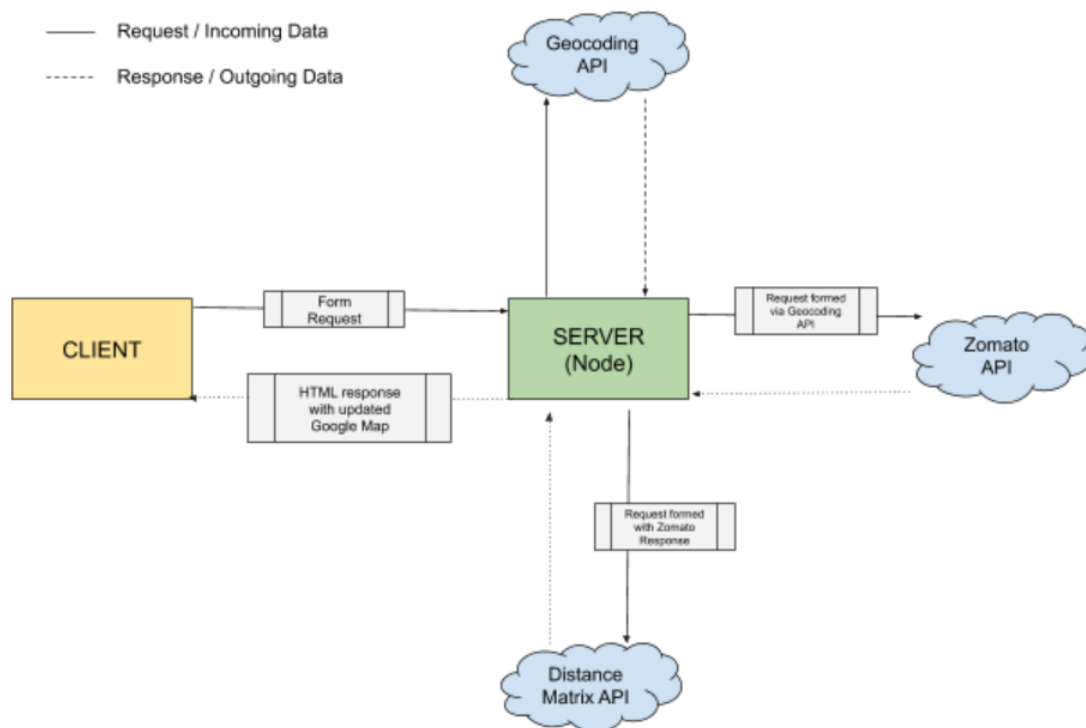
Ratings and reviews of a restaurant do play a major role in the deciding factor of choosing a restaurant to dine out, however it must be feasible to walk to. Thus, via the use of Google's Distance Matrix API the travel time and distance are also given in the popups of each restaurant marker as shown in the screenshot below. This saves users from constantly putting each highly regarded restaurant into Google Maps to see if the travel time is feasible. Note, that the blue marker shown is the location that the user has queried at.



3.0 Technical Documentation

The general goal of this project was to completely minimise the client-side processing performed. Thus, all the API endpoint calls along with the JSON processing occurs in the server side of the web application. This resulted in the decision of the server sending a complete HTML file to the client upon the submission of the form. This was decided at the limitation of requiring a page refresh as the DOM manipulation is quite significant due to the method of serving the interactive Google Map for the user.

An overall representation of the data flow is shown below.



Initially, the client initiates a request to the server, where the client will be served the view of the web application. With the displayed html form, the user can then make a post request where they will be redirected to an alternative URL with their requested information displayed.

The Node server only has the responsibility of dealing with the route management for the incoming client requests and outgoing information. An additional ZomatoMaps library was created to handle the requests to the external APIs.

The process can simply be followed in the following section. The location in the post request performed by the client is filtered into a request to Google's Geocoding API endpoint to retrieve the geographical coordinates of the requested location. This is then combined with the restaurant category, number of restaurants and sort by method to retrieve the restaurants around the given location via the Zomato API search endpoint. Consequently, the origin of the request and the location of all the restaurants are combined to retrieve the travel time and distance from Google's Distance Matrix API. Thus, the appropriate markers are then added to the HTML with the necessary information and sent to the client. Note that every time information was retrieved from an API endpoint, the JSON response was required to be parsed for the necessary information to be utilised.

3.1 Bootstrap

Bootstrap is an open source toolkit for developing HTML, CSS and JS and allows for the quick and professional development of the front-end of web applications. Thus, it was utilised in the web application to provide a professional look.

3.2 Express

Express is a minimal and flexible Node.js web application framework that provides developers with a boilerplate to begin the development of their web application. Its simplicity and ease of use in the separation of route management and API request libraries provided an easier experience in the development of this web application.

3.3 Axios

The axios library greatly simplifies the task of performing HTTP requests within node, relying on a chain of then clauses. This allowed for the chaining of API calls to be performed simply for UrbanBites.

3.4 Difficulties

Originally, the idea of the API mashup consisted of utilising Zomato's API endpoints to also allow the user to specify the radius of the query and cuisine of choice. Upon finishing the implementation of this concept, it was discovered that the API endpoints were not working as intended where the radius and cuisine specified were not utilised at all. Thus, these features were removed and simplified to the current application. In addition to this, the Google Maps Javascript API was called in the HTML view sent to the client. Thus, it was difficult for development and as a result for this prototype, the node server consistently sends back full HTML pages and requires a refresh instead of DOM manipulation.

4.0 Utilisation of Docker

Docker was utilised to encapsulate the application into a container for easy download and access. The figure below depicts the Dockerfile utilised for the containerisation of UrbanBites. The build is derived from the latest stable Docker node image. The source code is then copied into the base directory of the image along with setting the work directory for that source code. The *RUN* command is then performed to install the required Node packages for the Docker image. Finally, port 3000 is exposed for the Node server and 'npm start' is utilised to begin the application.

```

# File Author
MAINTAINER Kevin Nguyen

# Set the base image
FROM node:10

# Copy App source
COPY . /app

# Set the work directory to /app
WORKDIR /app

# Install app dependencies
RUN npm install

# Expose port
EXPOSE 3000

# Start Command as per package.json
CMD ["npm", "start"]

```

5.0 Testing Plan

The following basic test plan was initialised with the intention to have a smoother sailing development process of the web application.

Task	Expected Result	Result
Basic Google Maps Display	Maps Displayed with no information	PASS
Google Maps Marker Test	Display simple arbitrary marker on map	PASS
Client POST request	Ensure communication of POST request to node server.	PASS
Geocoding API Test	Obtain geographical coordinates from location given	PASS
Chaining of API calls	Use axios to chain API calls	PASS
Zomato Search API	Get restaurant information based on query and print onto console	FAIL
Zomato Search API Radius	Get restaurant information based on radius and print onto console.	FAIL
Zomato Search API Cuisine	Get restaurant info based on cuisine	FAIL
Zomato Search API with category and sortby	Get restaurant info based on category and sortby method and print onto console.	PASS
Distance Matrix API with restaurants	Get the travel details and print onto console between origin and restaurants.	PASS
HTML for restaurant markers	Generate the correct HTML for each restaurant marker	PASS
Response to Client	Generate the correct HTML to send to the client with the right markers and information.	PASS

As stated previously, the radius and cuisine API endpoints of Zomato were not working as intended and were thus replaced with category and sort by method.

6.0 Extensions

In the process of developing this web application, there were many discernible features already considered for future development. However, many of these features would require the Zomato API endpoints to be patched as originally this web application had the radius and cuisine type specified for the user to input, allowing for more customisation of the query. A lot of the Zomato endpoints when inputted simply, do not work. Upon the completion of these endpoints being patched the further features to be implemented are listed as:

1. Allowing for the user to specify the radius and cuisine.
2. Integration with Google Maps Directions API to allow the user when they've selected a restaurant, they can immediately get the directions from it.
3. Allow a search of categories and cuisine from a database as the query input, such as "brunch", "pizza" and so on.
4. Support for travel time for walking as right now, the app is only displaying travel times for cars.

7.0 References

<https://developers.zomato.com/api>

<https://developers.google.com/maps/documentation/javascript/tutorial>

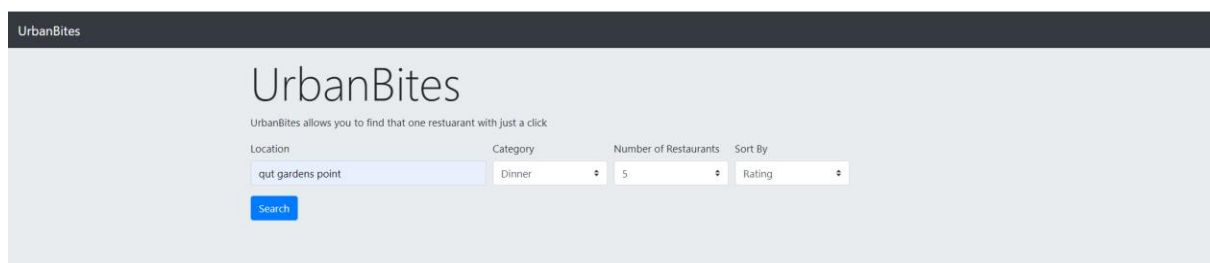
<https://developers.google.com/maps/documentation/geocoding/intro>

https://developer.mozilla.org/en-US/docs/Learn/HTML/Forms/Sending_and_retrieving_form_data

<https://nodejs.org/de/docs/guides/nodejs-docker-webapp/>

8.0 User Guide

The single view of this web application allows for the user to simply acquire the necessary information via a simple form submission at the top of the web application.


The screenshot shows the 'UrbanBites' web application interface. At the top, there is a dark header with the 'UrbanBites' logo. Below the header, the main content area has a light blue background. The title 'UrbanBites' is prominently displayed in a large, dark font. Underneath the title, a subtitle reads 'UrbanBites allows you to find that one restaurant with just a click'. Below this, there is a search form with four input fields: 'Location' (containing 'qut gardens point'), 'Category' (a dropdown menu showing 'Dinner'), 'Number of Restaurants' (a dropdown menu showing '5'), and 'Sort By' (a dropdown menu showing 'Rating'). A blue 'Search' button is positioned below the 'Location' field.

Then upon the submission of the HTML form, the user is displayed with the appropriate restaurants, around their blue marker indicating their location specified. Upon clicking any of the red markers, the restaurant's details, rating, reviews and travel information are displayed for the user to read.

Location	Category	Number of Restaurants	Sort By
qut gardens point	Dinner	5	Rating
Search			

MapSatellite

Sono Japanese Restaurant



Location: 39 Hercules Street, Hamilton, Brisbane

Cuisine: Japanese, Sushi

Opening Times: 6 PM to 9:30 PM (Tue), 12 Noon to 2:30 PM, 6 PM to 9:30 PM (Wed-Thu), 12 Noon to 2:30 PM, 6 PM to 10 PM (Fri-Sat), 12 Noon to 2:30 PM, 6 PM to 9 PM (Sun), Mon Closed

Cost For Two: \$150

Rating: 4.9

Distance: 5.8 km

Google

Map data ©2019 Terms of Use

[Search More »](#)