Food Diary

Team 45

Student 1	Student 2	Student 3
Gun Woo Park	Wei Kang Tan	Terry Williams

April 23, 2018

Department of Computer Science

Unversity College London

ABSTRACT

The application that our team (Team 45) has developed is a food diary based on the Microsoft Cognitive services as per the requirements specified by our client from Microsoft.

In the initial research phase, my team has found that some people like to or need to write a food diary due to their allergies or interests. There were many applications on the market which records total calories intake or manually editable food diary, but they required manual updating of the diary. This comes as a hassle and inconvenience for users which result in them forgetting or even just too lazy to update their diary.

To solve the issue, we developed a web application which largely simplifies the process of updating a diary by automating things. Our web application works by retrieving an image of the food and an image of the user's face from any of our supported cloud storage providers. It then uses artificial intelligence provided by Microsoft Cognitive API to analyse the photos – the face is analysed for the emotion while the food is identified. From there, a diary entry including star ratings of the food (generated based on the emotion) and a description of the food is generated and displayed on the user's account. Also, the application can also suggest restaurants that the user might like based on the star ratings.

By doing this, our solution reduces the amount of information that user needs to input manually, which means the diary can be written and updated more conveniently. This would result in a higher usage rate overall.

TABLE OF CONTENTS

LIST OF FIGURES

- Figures used for the appendix have been omitted from the list.

LIST OF TABLES

1 Introduction

1.1 Problem definition and project goal

Background and context

The clients and the team have noticed that some people have specific dietary requirements or preferences and they like to record their food eating history. The members of the team have also recognised that by using Microsoft Cognitive Services [1] (a set of machine learning APIs), analyzing and generating tags for the images is possible, which means identifying different types of food is possible. The team and the client have also observed that majority of mobile phone users upload their images to the cloud storage.

In terms of the format of application, there were needs for running core information processing engine and DB connector to run on the server side, instead of the client side, which means there was a need for using Node.js framework for the development. The team have also agreed that in order to provide better compatibility with users' device, it is better to develop a web app.

Problem statement

The current problem is that although people with the special dietary requirements/preferences need to record their eating history, they are often busy that they forget to do so or find it too much of a hassle to do it manually all the time. Additionally, there are food recognition applications already available, but most applications are focused on the measurement of calories [2] or recipes [3] of the food that there are no convenient services for the recording of eating history in particular. As such, people with specific dietary requirements do not have an easy platform for doing their recordings.

Scope and relevance

This project focuses on the development of the food diary instead of a general-purpose diary. Due to the inherent limitations of Microsoft Cognitive Services, the scope of the automatically generated text will be limited to the title of the images and tags with an

emotion recognition result of the human face (this will be used for the evaluation of the food satisfaction). In order to ameliorate the potential problem of inaccurately generated tags or an inability to recognise allergens, the web app supports manual editing functionality, therefore the nature of this web app-based food diary will be in between one of a conventional food diary and a totally automated diary. The web app also supports suggestion function - based on the existing diary entries, the web app will suggest other food and restaurants that the user is likely to be satisfied with.

This web app will provide a better recording platform for those who need to record their eating history as well as to give some suggestions in case people want to discover a restaurant in a distant region or an unvisited local restaurant. This means the web app will be very relevant as a solution to the problem.

Goals

The aim of this project as per the client's requirement is to show the ability of the Microsoft Cognitive service, by providing a diary application that solves the aforementioned problem. The application will solve the problem by recognising the food and the emotion of the user to generate cards for each food-face image pair. This means users can record their food eating history, thereby avoiding food that is not suitable for their dietary requirements/preferences while being provided with some suggestions for the eating. (For the list of allergens in the web app, we follow the Annex II of the Regulation (EU) No. 1169/2011 (FIC) [4].) It is our eventual wish that our web app serves as a better diary and food suggestion platform than existing ones to provide a better eating experience for our users.

1.2 Development Team

None of our team members had particular experiences with JavaScript or other web app technologies. Therefore, the development field often overlapped as members learnt on the job.

For the testing, all three members have been assigned a role as a tester.

Gun Woo Park (Team leader)

Gun Woo had programming experiences with various languages including Objective-C and Swift. For web development, Gun Woo had only one experience with PHP and simple JavaScript on the C coursework for COMP101P.

Gun Woo was responsible for the server-side web app (front and back-end) programming, UI design (i.e. client-side front-end) and programming, client liaison and report editor.

Wei Kang Tan

Wei had programming experiences with only Python and limited experiences in web development.

Wei was responsible for the client-side web app backend research and programming. Since he was showing significant progress at the end, he was also assigned to some server-side backend function programming tasks and implementations of auxiliary features. In addition, he worked as a copy editor for the reports.

Terry Williams

Terry had programming experiences with C, Python and Java. However, his web development experiences were limited as well.

Terry was responsible for the client-side web app research and programming as well as the copy editor for the reports. He was also the video editor.

1.3 Original Gantt Chart

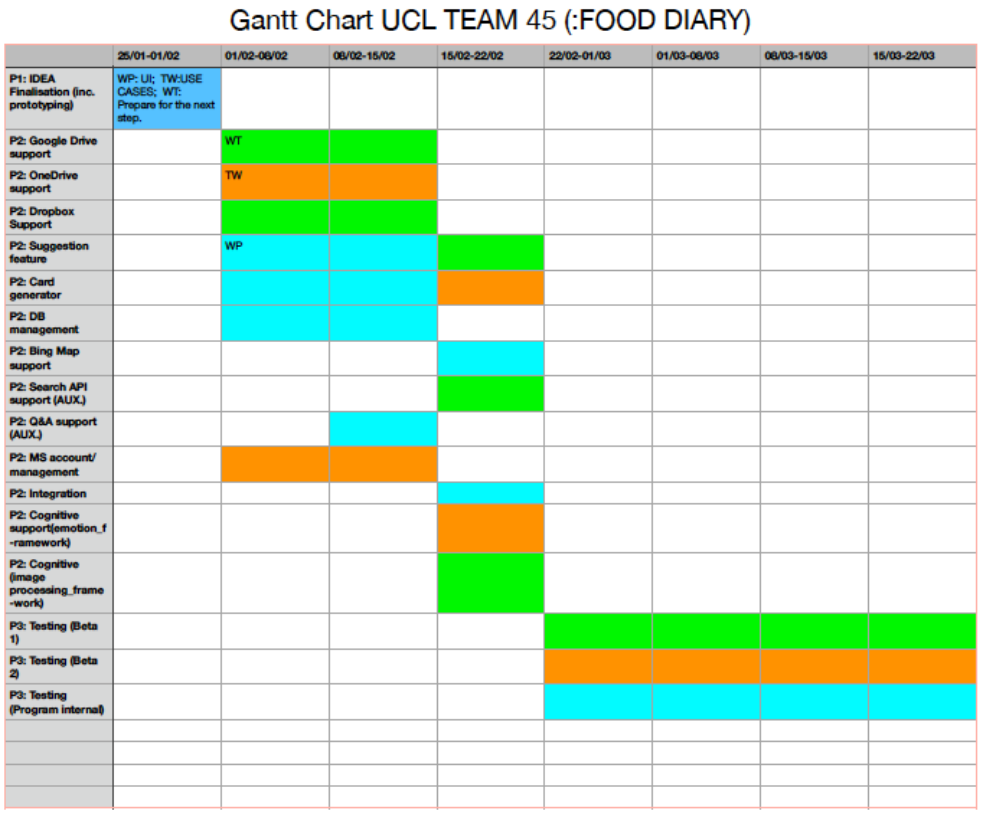


Figure 1.1: The original Gantt chart

*WP means Gun Woo Park, WT means Wei Kang Tan, and TW means Terry Williams. This table shows the tasks that had been originally assigned to each team member.

2 Requirement

2.1 Persona

This web app will be used by people who have a need or habit of recording their eating history due to their own special dietary requirements/preferences and following properties:

- Do not like writing textual content on either a mobile phone or a sheet of paper
- Have a busy schedule
- Like to take pictures of the food and selfies
- Use cloud storage that syncs with the photo album automatically.
- Openminded towards the data processed by artificial intelligence [5]

2.2 MoSCoW Requirement List

Must have

- Food recognition using the Microsoft Cognitive Services API
- Emotion recognition using the Microsoft Cognitive Services API
- Image retrieval from cloud storages
- Database for the generated data recording and account details
- Microsoft account API for sign in
- Simple suggestion platform by using the tags generated

Should have

- Show images on the map by using the location of the image capture (geotagging)
- Suggestions based on a specific location or a food category
- Dietary requirement fields for the cards
- Microsoft Cognitive API assisted Q&A [5]
- Manipulation of tags to show tags in a more visible way
- Automatic image retrieval from cloud storage [6]

Could have

- Advanced suggestion (Context recognition by using another Microsoft API [6])
- Sentence generator
- Showing feeling trends in the graph
- Advanced automatic image retrieval i.e. if the user had logged-in to the cloud storage, the web app should be able to retrieve images without any user action

Won't have

- Cross-checking of tags by using AI APIs from other sources
- Personalised restaurant advertisements
- Filtering functionality for the images that are likely to be recognised as a different object e.g. fried chicken recognised as a cup of coffee

3 Research

The only technologies the team were explicitly instrucuted by the client to use were Node.js and Microsoft Cognitive Services. However, due to the complexity of the project, additional technologies were employed and will be explained in the following paragraphs.

3.1 Data storage

Microsoft Azure Storage was a service recommended by the client which enables the blobs to be stored in the Azure data storage. This technology enables uploading of image data to be done easily, which was the reason the team decided to use this technology.

3.2 Indexing

There are two possible solutions, and two members of the team are working on each solution to compare the ability of each option.

MySQL

This is a popular SQL solution. It enables searching of indexes of the data to happen easily and quickly. This technology is also very stable and easy to use especially for the indexing of the data. Gun Woo Park has researched and implemented this solution.

Azure Search

This is a search service provided by the Microsoft Azure. The search speed of this solution is slower, but it does not require an additional server connection to be created. As Azure search is a service provided by the Microsoft, it has been expected that this solution will integrate with other Azure services well. Terry Williams have researched this solution.

As the final solution, we have chosen to use MySQL for indexing.

3.3 Web Content (Client-side)

HTML5

HTML 5 will be used in combination with the UIKIT 3 [7] to show the contents of the web app to the users.

CSS 3

CSS 3 will be used to define a style of each component in the web page.

JavaScript with JQuery (Especially the AJAX method)

Using JavaScript language with AJAX, HTTP requests will be sent and the received data would alter the web page shown [8] [9]. JavaScript methods will also retrieve images from cloud service providers. As the web app supports three different cloud services, it will have three different authentication methods and retrieval methods.

JADE

Jade uses simpler syntax in comparison to the HTML while providing the same functionality [9]. This solution typically works with the Node.js. However, since the client-side web app will not use the Node.js, this solution has been rejected for the client-side. In contrast, the server web app uses Jade.

3.4 Artificial Intelligence based Decision Maker API

We had to use Microsoft Cognitive services since the client was Microsoft. We had utilised three different Microsoft APIs each with their own unique functionality.

Vision API

Vision API detects objects included in the image and outputs the names of each object [10]. Since the Microsoft Cognitive services are not focused on object recognition in general (they are specialised in face recognition instead, according to the client), the generated tags were often inappropriate. However, since some tags generated from this API were true, and the

description text generated from API was appropriate, we have decided to use this API. This API is used for the food images only.

Face API

As Face API is the API that Microsoft is focused on, the analysis produced by the API was highly accurate. For the application, we have used the most probable feeling with the probabilities for each feeling in order to detect user's feeling as well as generating the star ratings for the food.

QnA Maker API

QnA Maker helps to make an interactive Q&A page similar to that of a chat bot. This API was being used for the main Q&A page and suggests an appropriate answer to the question searched even if the user does not search the exact question in the dataset.

4 Design and Implementation

4.1 System Architecture Diagram

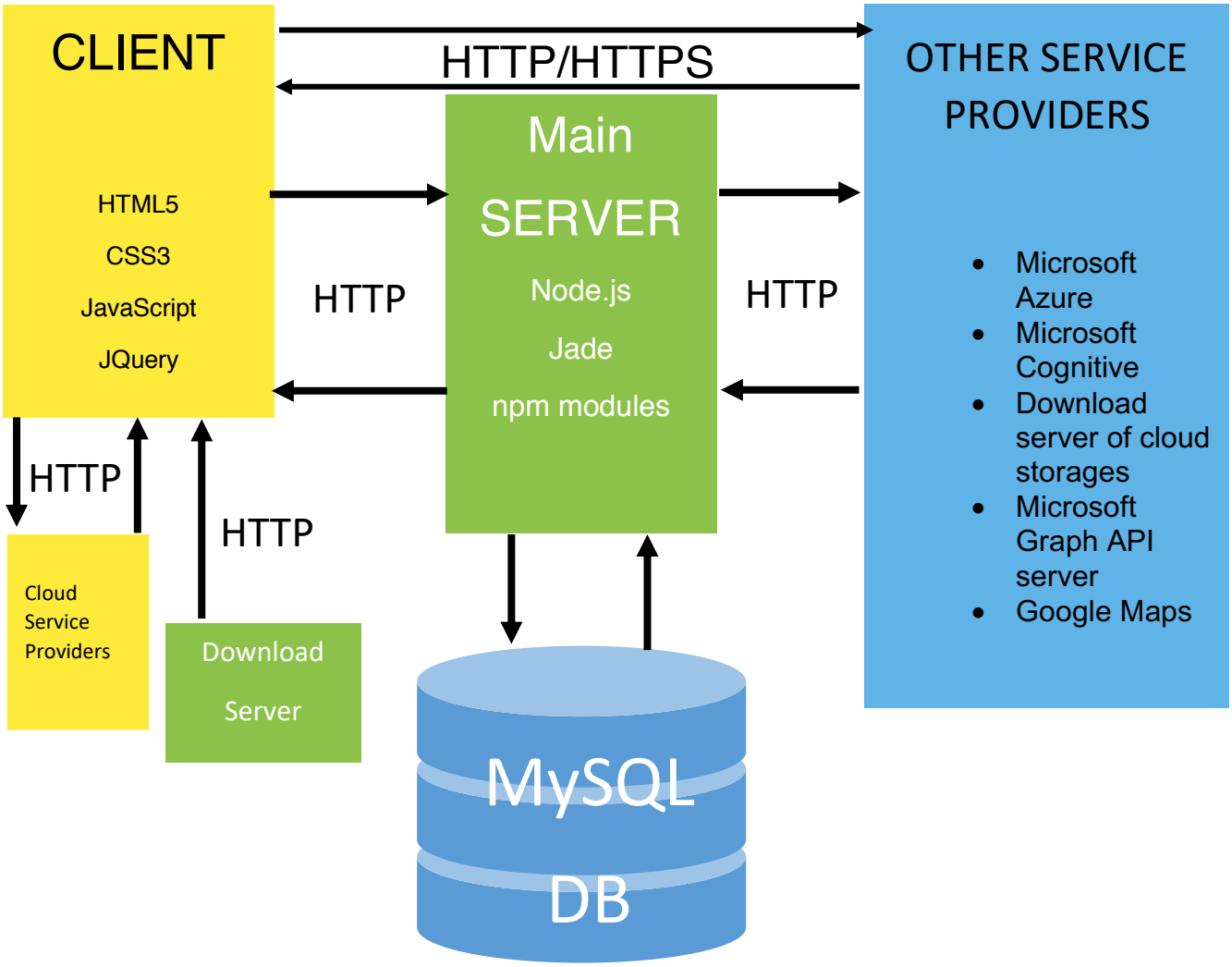


Figure 4.1: System Architecture Diagram

Download Server

This server is a virtual server (Azure web app service) that only fetches the resources e.g. images and HTML, JavaScript framework files, to the client requiring them. The resources are given by using HTTP.

Client

The client downloads HTML pages and JavaScript resources from the download server. The client also obtains the tokens and file ids from the cloud storage service providers based on user authentication and file selection. It sends appropriate HTTP requests (including search, delete and edit) to the main server and displays the data retrieved from the server. Based on the data retrieved, the client can also retrieve cards and images stored in the Azure storage by using a SAS token. Several other features including map viewing, Q&A, user sign-ins and about page are also supported by the client.

Cloud Service Providers

The cloud service providers allow users to authenticate their accounts and enables the client side of our web application to retrieve the access token. They also enable downloading of images, file name and file IDs.

Main Server

The main server (also a virtual server) is the place where all the data is processed and enables suggestion, search, edit, delete, and creation of cards. By separating the server that utilises a lot of computational resources, it makes dealing with the resource related problem easier - even if the main server is down and unable to serve, the download server could still display notices or at least show the server not responding message on the client-side web app.

MySQL DB

MySQL DB stores indexes of each card. It stores essential data per each entry that need to be searched and therefore enables searching of the cards to happen efficiently and effectively.

Other Service Providers

This includes Microsoft Cognitive service servers, Microsoft Graph API server for the user authentication (user sign-in), Google Maps for maps and geocoding, Microsoft Azure Storage, and download servers of the cloud storage providers. The client can only connect to the limited number of providers in order to provide better security since the majority of

the API keys and secrets will not be revealed to the client who can also be a hacker. For the connection with the Azure Storage, clients use SAS key instead of a normal key to enable the developer to control the access permission of the users, since SAS keys in the client-side web app will be visible to the users by using the developer mode of the browsers.

All of the connections between each system components will be using HTTP except the connection between the Microsoft Graph API which uses HTTPS.

4.2 Site Map

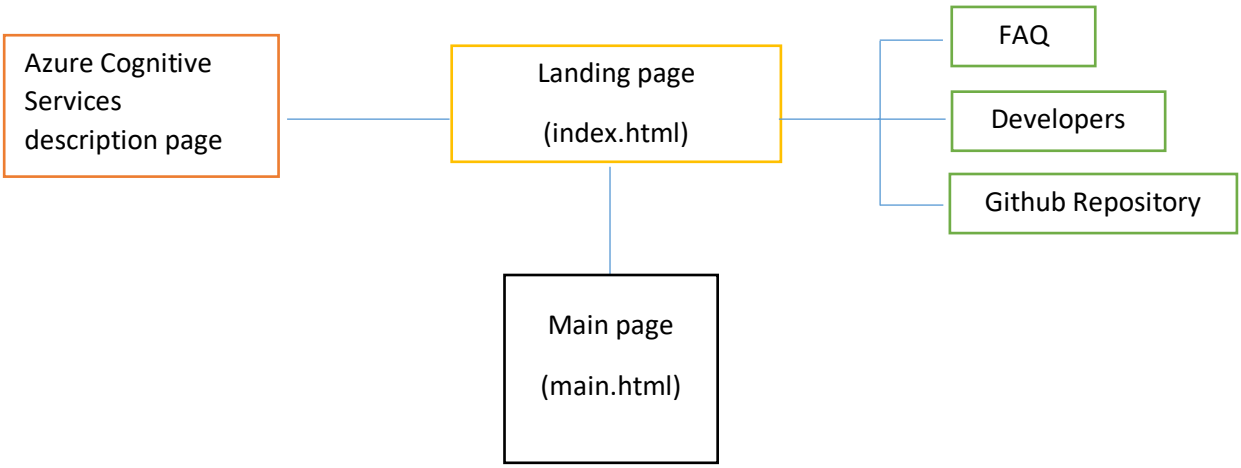


Figure 4.2: Site map

4.3 Page Flow

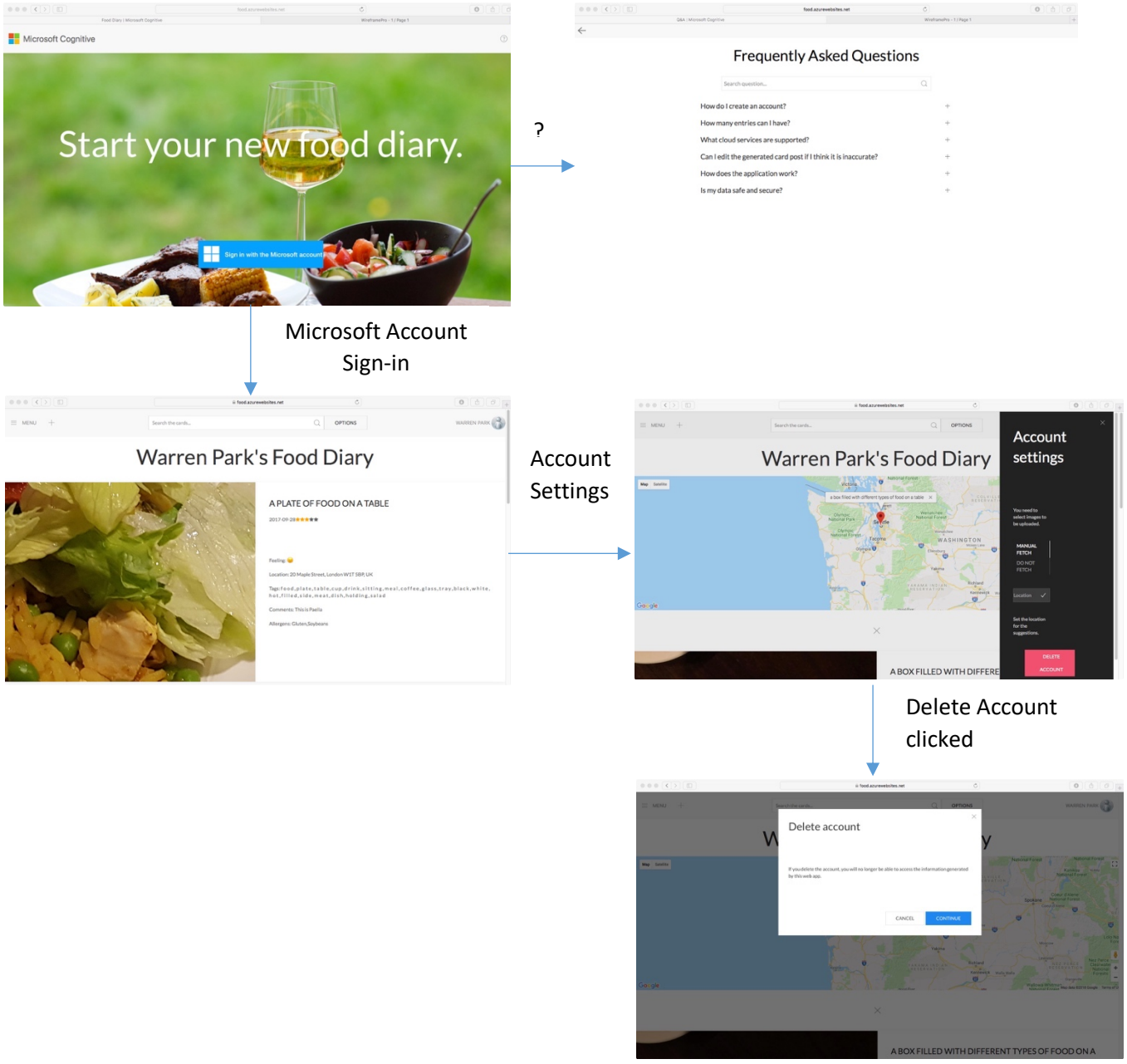
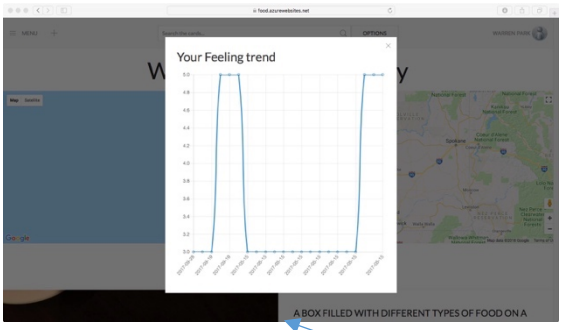
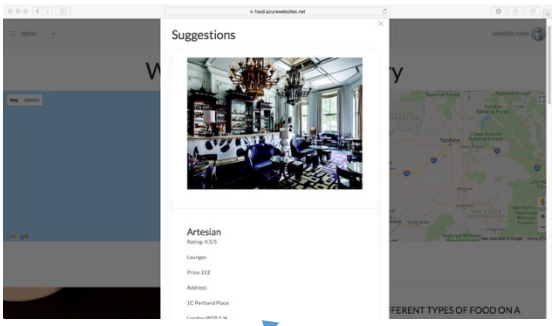


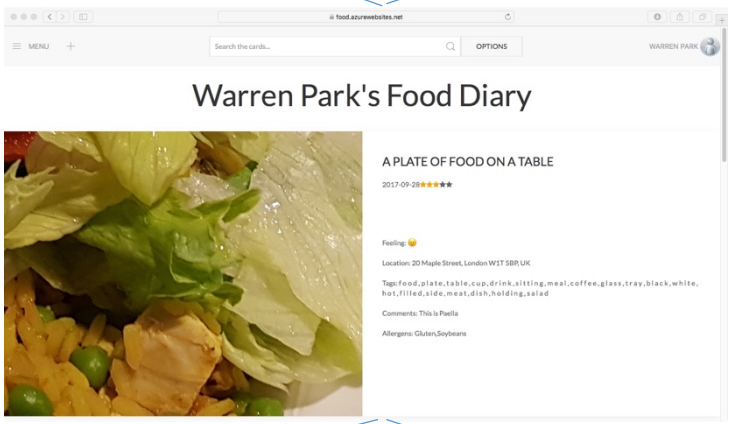
Figure 4.3: Page flow diagram (Overall)





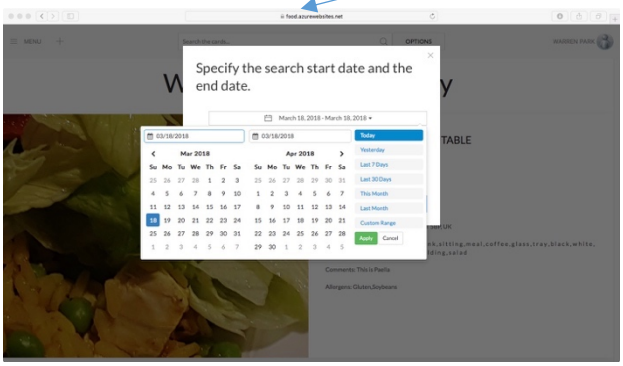
Trend graph button clicked

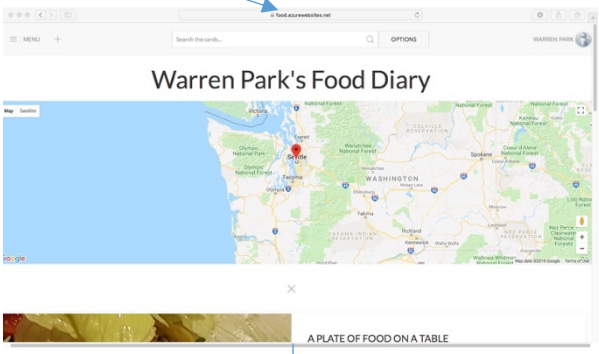
Suggestion button clicked



Options search (Date)

Options search (Map)





Pin Clicked

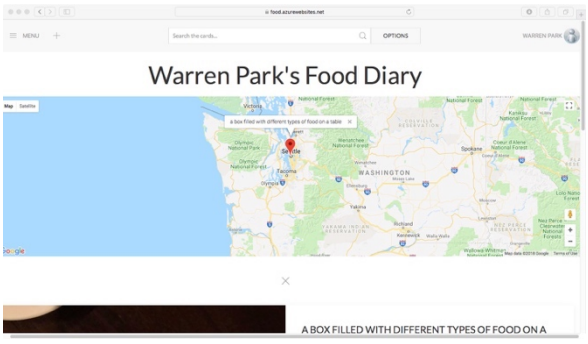


Figure 4.4: Page flow diagram (main page. Copy, delete, and edit function have been omitted.)

4.4 Database ER diagram

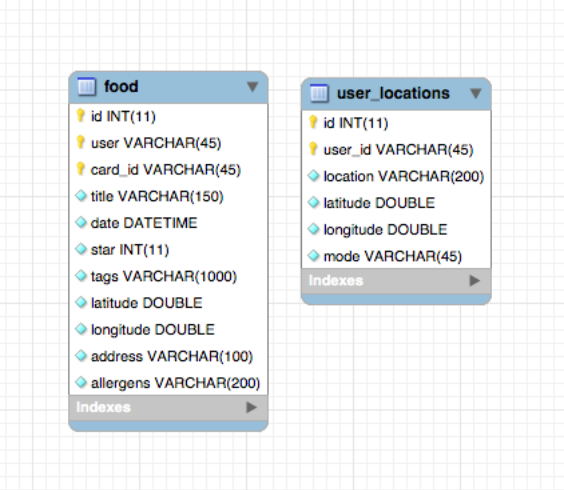


Figure 4.5: Database ER diagram for the MySQL DB

4.5 Development Tools

- Communication: We have used Facebook, Slack and UCL email for the communication. Slack and UCL email were used as the primary form of communication between the team and the client. Facebook, on the other hand, was used only within the team.

- File Sharing: For the sharing of files, Facebook and Github repository were used. Facebook was normally used for document sharing, whereas Github was mainly used for the code sharing.

- UI design: We have used UIKit 3 and Bootstrap in combination as a UI framework. Based on that, standard PowerPoint and Keynote application were used for the actual UI design.

- Version Control: We have actively utilised Github. All the updated files have been uploaded to Github repository frequently, and we used the version controlling feature of the Github in order to use the deleted code when the newer code happened to have a problem.

- IDE: Multiples of IDE were used for the development. For the front-end development, Atom was utilised but for the backend development, Microsoft Visual Studio Code or AWS Cloud 9 were used.

- Post editing tool: This project did not require a post editing tool.

- Video editing: The only tool used was the narration creator built into Microsoft PowerPoint since the slides were made using PowerPoint as well.

4.6 Front-end Technologies

HTML 5

HTML has been used to display information generated by JavaScript.

UIKit 3

UIKit 3 was been utilised in order to provide user-friendly UI. This framework has provided form, notifications, cards, and modal windows which were essential UI components for the web app. This framework was able to support responsive UI component by default.

Daterangepicker

Daterangepicker was used to enable users to input date range for the search easier. This framework was based on BootStrap [11] and provided calendar view as well as direct range input e.g. 7days before today.

CSS 3

CSS was used to define the style sheet that the web app was using. Since UIKit covered most of the design definitions, the use of CSS was limited. However, CSS was used usefully to define new fonts for the UI components or the z-index.

JavaScript

JavaScript enabled the web app to be interactive. It was the most important technology for our application and with jQuery and AJAX, it made the web pages updatable without reloading.

BootStrap

BootStrap was being used as a backend technology for the Daterangepicker and some other design components.

jQuery

jQuery enabled access to each UI component to happen easily and efficiently. One of the feature AJAX was utilised for making website updatable while not requiring the user to refresh the website. Especially, AJAX XMLHttpRequest object was used for the connection between the client and the server.

Chart.js

Chart.js enabled plotting a stylish line graph to be possible. This framework was used for the feeling trend data visualisation.

Jade

Jade is an HTML alternative that has the same function, but with less coding. Jade was used for the main server to display the server status.

Google Maps API

Google Maps API was used for the project to show the locations where the image has been taken on the map. This API provided an access to the high-quality map that supports touch gestures, without a lot of coding.

Google Drive File Picker API

Google Drive File Picker API was used to show the native Google file picker to the Google Drive users. Since it shows similar interfaces that the users used to be using, it makes the web app to be more user-friendly while providing a stable access to the files stored in the cloud storage.

Dropbox Chooser API

Dropbox Chooser API is a file picker for Dropbox. It also provides native Dropbox UI, so it makes the web app to be more user-friendly.

OneDrive File Picker API with Graph API

OneDrive File Picker API is a file picker for OneDrive and provides a user-friendly interface that is native to the Microsoft OneDrive. It required Graph API to be used for the Microsoft Sign-in.

MSAL

MSAL (Microsoft Authentication Library) enabled Microsoft sign-in to happen easily since the programmer does not have to consider each different stage in the authentication process. This library enables the obtaining of user ID, token and user's full name.

4.7 Back-end technologies

Node.js

Node.js is a JavaScript run-time environment for the server-side. It enables asynchronous execution of the program to happen efficiently that it reduces the number of computational resources required to process the HTTP request and provides stability for the execution.

MySQL

MySQL is a typical DB solution. As mentioned before, this technology was required to enable web app to search the card efficiently.

Azure Storage

Azure Storage services were used to store images in the data centre. It provides an easier interface to the developers in terms of using the services. This web app has used Blob Storage service which is a part of the Azure-storage. Blob Storage enables storing files without having to consider about encoding or decoding. Microsoft provides an npm module which can be used to access the storage with many options.

Microsoft Cognitive

As mentioned before, Vision API and Face API were used to generate the textual contents of each card. QnA Maker API was also used for the FAQ of the web app.

Exif (npm module)

Exif metadata is a data stored with an image that contains various information about images [12]. Among the information, the Exif metadata has coordinates of the location that user has taken that picture or the date that the picture has been taken. Therefore, this web app utilised Exif metadata to determine the date and location that need to be included on the card. For this, Exif, which is the name of an npm module [13], had to be used.

Yelp-fusion (npm module)

Yelp-fusion npm module [14] provided an access to the Yelp-fusion API [15].

Google-drive (npm module)

Google-drive npm module is used for generating a file download URL from by using the token and the file ID.

Node-geocoder (npm module)

Node-geocoder uses geocoding service of the Google and other providers (in this web app, we have used Google's service) [16]. This module converts the address into coordinates or coordinates into an address.

Cognitive-services (npm module)

Cognitive-services npm module was used for the Face API. Although accessing directly to the Cognitive service is possible, we used this npm module to provide stability as the module has been tested.

Microsoft-computer-vision (npm module)

Microsoft-computer-vision npm module is used for the Vision API. As like Cognitive-services module, this npm module is used to waive the testing process for using Microsoft API.

sharp (npm module)

This npm module is used to meet the requirement of the Microsoft Cognitive Vision API. This module detects the size of the image and resizes if needed while keeping the aspect ratio [17]. The developer is claiming that this module can work faster than other modules using "ImageMagick" [18], so provided that the claim is true, it will show a great performance. Due to the compatibility issue (this module does not work properly on the x86 Windows environment), we have also developed a version of the server-side web app that uses another framework.

resizer-stream (npm module)

This npm module is used as a substitute for the sharp. Although it has slow processing speed and there exists a file size limit (due to the jpg-stream module), this module can work in a various server environment. Therefore, a version that uses this module has also been developed.

jpg-stream (npm module)

This npm module converts jpg file into a stream that is generic. Therefore, this module was required in order to use resizer-stream module since the resizer-stream only accepts a file stream. This module, therefore, decodes the image file and also encodes the file stream output from the resizer-stream module.

4.8 Web Hosting

For the web hosting, as the client is Microsoft, Azure has been used. In order to run the web app, we had to establish two web app services for the web app hosting, a MySQL database, and an Azure Storage. Heroku has also been used to test the server-side web app that uses sharp since Azure did not support sharp. A version with sharp worked as intended on Heroku.

4.9 Implementation of key functionalities

Card creation

In order to create a card, the client-side web app collects the file ID and token or download URL of the image that has been selected by the user. Then, the web app sends an HTTP request to the server, and the server-side web app processes the data. Processing data steps are given below:

1. Download requested images (create file streams)
2. Send the image to the Azure Storage to save
3. Send a food image to the Microsoft by using HTTP request in order to use Vision API
4. Send a face image to the Microsoft by using HTTP request in order to use Face API
5. Determine the Exif metadata of an image
6. Reverse geocode[1] the coordinates stored in the Exif metadata.
7. Receive HTTP response from Vision API

[1] Reverse geocoding coverts coordinates data into a human-readable address [18].

8. Receive HTTP response from Face API
9. Generate star rating from the Face API request
10. Write a card HTML content (Title from Vision API, date from Exif metadata, star rating from generated star rating, feelings from Face API, address from reverse geocoding and tags from Vision API)
11. Send a generated HTML content for the card to the Azure Storage
12. Create an index for the card on the MySQL
13. Delete images (unlink file streams)

Since the execution is asynchronous, it was difficult to make some functions to execute sequentially. Therefore, nesting functions were required to ensure all the information processes as intended even though the external server is responding slowly. Using this process, the web app generates card.

For generating stars, we determine the probability for the happiness and 5 stars are given when the user's dominant feeling is happiness and the probability is high (greater than 80%). 4 stars are given when the dominant feeling is happiness, but the probability is low. 3 stars are given if the dominant feeling is neutral, 2 stars for anger and sadness, 1 star for contempt and fear and 0 stars for disgust.

Personalised Suggestions

The web app can suggest a restaurant registered on Yelp based on user's tendency. In order to generate personalised suggestions, it first determines the default location set by the user. Then, it gets the list of user card indexes aligned by stars in a descending order. After that, the web app selects the card index from the top of the SQL response and gets the tags from that index. Using tags and location, the web app sends a request to the Yelp and gets the list of restaurants corresponding to the given input. The web app then sorts the information from the searched result and displays the top three results from the response.

Therefore, the web app generates personalised suggestions for users.

5 Testing

5.1 Browser compatibility testing

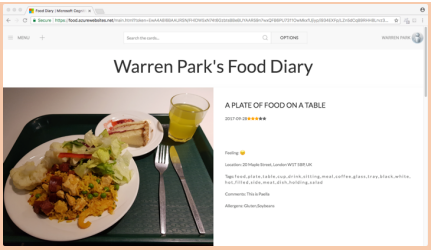
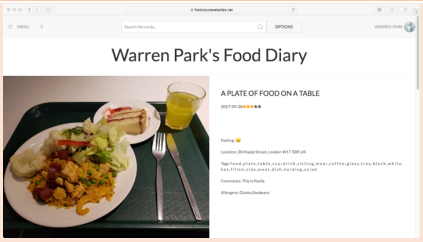
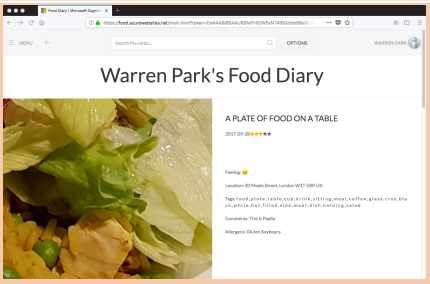
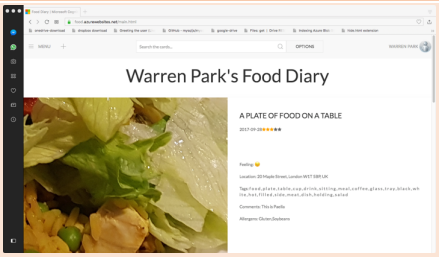
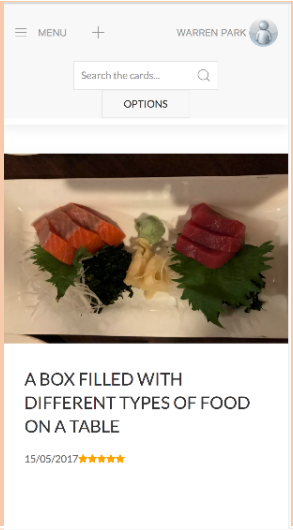
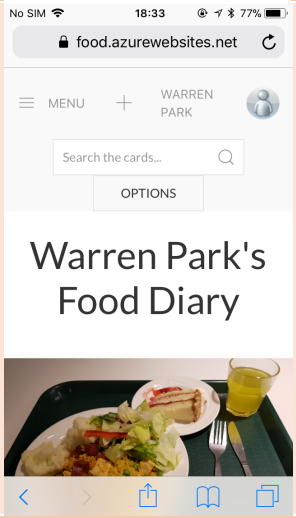
Browser	Version No.	Result	Screenshot
Chrome	65.0.3325.162	Passed All components displayed as intended.	
Safari	11.0.3	Passed All components displayed as intended.	
Firefox	58.0.2	Passed All components displayed as intended.	
Opera	51.0	Passed All components displayed as intended.	

Table 5.1: Browser compatibility testing

5.2 Device compatibility testing

OS	Device	Result	Screenshot
iOS	Apple iPhone 7	Passed All components displayed as intended.	
iOS	Apple iPhone 5S	Passed All components displayed as intended.	
Android	Samsung Galaxy S8	Passed All components displayed as intended.	

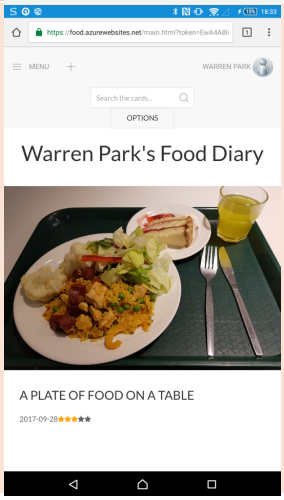
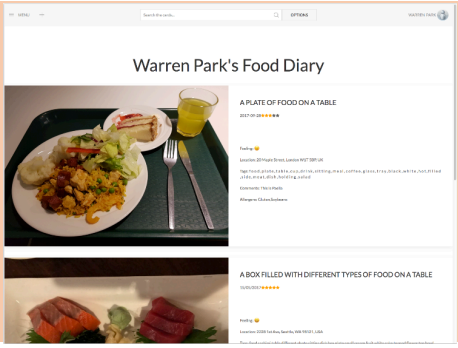
			
Android	Sony Xperia Z Ultra	Passed All components displayed as intended.	

Table 5.2: Device compatibility testing results

5.3 Responsive design testing

*Tested using Safari responsive design mode except for the iPad mini.

Screen size	Result	Screenshot
Desktop 1600x1200	Passed All components displayed as intended.	

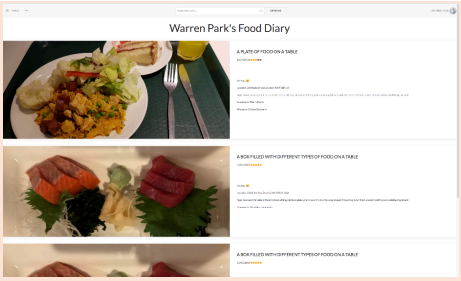
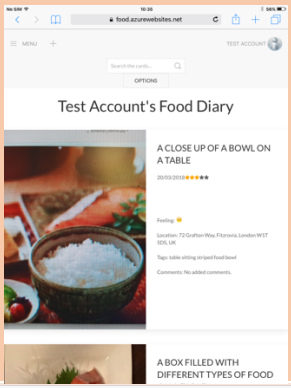
Desktop 2732x1536	Passed All components displayed as intended.	
Apple iPad mini 768x1024	Passed All components displayed as intended.	
Apple iPhone 8 Plus	Passed All components displayed as intended.	

Table 5.3: Responsive design testing results

Since the client has requested that this application need to be publicly available, there are no specific screen sizes that are used most.

5.4 User Acceptance Testing

List of test tasks

1. Create a card from any cloud storage that the tester prefers.
2. Search specific card by typing a term in the search bar
3. Set the default location for the suggestion
4. Try to search the card by using a date range
5. Try to search the card by using the map
6. Try to access the personalised suggestion
7. Try to access the feeling trend graph
8. Try to edit the card
9. Try to delete the card
10. Try to delete the account

Tester ID	Category	Used cloud storage	Feedback
Tester 1	A family member	Google Drive	The user interface was simple, but it was difficult to understand the functionality of the application at first. Although the application works properly, due to the lack of features, it would be good to be used as a framework, but not as a standalone application.

Tester 2	Electrical Engineering PhD student	OneDrive	Fairly simple and intuitive user interface. Might need to come with some initial instructions. I am not sure I would use this product however as I do not like keeping a record of my eating habits. I also like to read online reviews of restaurants although does not always solidify my decision to eat somewhere.
Client 1	Client (Microsoft)	OneDrive	I have been impressed with the outcomes of the project, the student have achieved a great implementation of the project they have gained experience of using Microsoft Azure as a platform and developed a prototype application for interaction of their services.
Client 2	Client (Microsoft)	OneDrive	As a frequent traveller and a coeliac – having a severe and painful reaction to food that contains gluten, it is important for me to keep track of what and where I eat. An app like the one built by this team would help me keep a diary of food, correlated to the location. I see an opportunity for

			expansion, by adding a “reaction” column to it, ability to rate the food itself and share the ratings with other people. Automatic recognition of the food is already an incredible benefit as it helps me figure out what type of food I’ve eaten the most, and where, etc.

6 Evaluation

6.1 Achievement Table

ID	Requirements	Priority	State	Contributors
1	Food recognition by using the Microsoft Cognitive service API	Must	✓	Gun Woo
2	Emotion recognition by using the Microsoft Cognitive service API	Must	✓	Gun Woo
3	Image retrieval from cloud storages	Must	✓	All
4	Database for the generated data recording and account details	Must	✓	Gun Woo
5	Microsoft account API for sign in	Must	✓	Gun Woo, Terry
6	Simple suggestion platform by using the tags generated	Must	✓	Gun Woo
7	Show images on the map by using the location of the image capture (geotagging)	Should	✓	Wei
8	Suggestions based on a specific location or a food category	Should	✓	Gun Woo

9	Dietary requirement fields for the cards	Should	✓	Gun Woo
10	Microsoft Cognitive API assisted Q&A	Should	✓	Wei
11	Manipulation of tags to show tags in a more visible way.	Should	✓	Gun Woo
12	Automatic (In terms of token handling and authentications) image retrieval from cloud storage	Should	✓	Gun Woo
13	Advanced suggestion (Context recognition by using another Microsoft API) *	Could	**X**	None
14	Sentence generator	Could	✓	Gun Woo
15	Showing feeling trends in the graph	Could	✓	Gun Woo
16	Advanced automatic image retrieval i.e. if the user had logged-in to the cloud storage, the web app should be able to retrieve images while user unnoticed. **	Could	**X**	None
17	Tags cross-checking by using multiple AI APIs	Won't	**X**	None
18	Personalised restaurant advertisements	Won't	**X**	None
19	Filtering functionality for the images that are likely to be recognised as a different object e.g. fried chicken recognised as a cup of coffee	Won't	**X**	None
Key Functionalities (must have and should have)		**100%** completed		
Optional Functionalities (could have)		**50%** completed		

Table 6.1: Achivement table

*This function has not been implemented since we decided to use Yelp API. Since the Yelp API gives a descent suggestions data, there was no need to implement a context recognition by using another Microsoft API.

**Advanced automatic image retrieval feature had to be removed due to the privacy and security concern. As our TA has also mentioned, without alerting users, using the data should not be done due to the user privacy. Additionally, having a refresh token stored in a

server means that the server is highly likely to be attacked by a hacker that can enable hackers to access users' cloud storage contents.

It is possible to say that we have met all the requirements since we have met all of the mandatory requirements and all of the optional requirements except the requirements that were unnecessary or should not be implemented.

6.2 A list of known bugs

ID	Bug Description	Priority
1	Google Drive Picker sometimes reopens even though the user has selected the file.	High
2	The sever web app that uses sharp has a compatibility issue with Azure. (works as intended on Heroku)	Normal
3	The jpg-stream module has file size limit that the server web app does not use sharp does not work for bigger files.	Low

Table 6.2: A list of known bugs

6.3 Individual contribution table

Work packages	Gun Woo	Wei	Terry
Client liaison	44%	21%	35%
Requirement analysis	34%	34%	32%
Research	34%	34%	32%

UI Design	35%	35%	30%
Programming	35%	34%	31%
Testing	34%	34%	32%
Progress Report	34%	33%	33%
Technical Report	55%	25%	20%
Poster Design	0	50%	50%
Video Editing	33%	33%	34%
Overall contribution	**34%(33.8%)**	**33%(33.3%)**	**33%(32.9%)**
Roles	**Researcher, Report Editor, Front/back End Developer, Tester**	**Researcher, Back End Developer, Copy Editor, Tester**	**Researcher, Back End Developer, Video Editor, Tester**

Table 6.3: Individual contribution table

6.4 Critical evaluation of the project

User Interface/User Experience

In terms of design, while we believe the UI design is largely inituitive, we recognise that it may be quite difficult for users to find some hidden buttons on our website in the event they are not used to such a design. In terms of user experience, it would not make sense to users why cloud storages need to be used for uploading of files unless the explanation is provided to the user. The reason why cloud storage is being used is because the client has mentioned that he uses cloud storage that automatically syncs with the camera album of his phone and the project was aimed at supporting cloud storage from the beginning as the initial title of the project was “Microsoft Cognitive Node.js app with cloud storage support”.

Functionality

It is true that functionality of the web app is quite limited. It does not have efficient share feature and only supports food images. It also does require a selfie of the person to be taken after taking the image of the food whereas the application does not sync with the cloud

storage automatically. Therefore, it is unlikely that users might want to take a selfie every time when they eat something, and even more, would not want to manually upload pictures on the web app.

Compatibility

While the web app has generally good compatibility, we acknowledge some areas of improvement. For smaller mobile devices, the images are displayed too largely that not much of the other information can be shown. On the other hand, for larger screens, the page could be seen a bit vacuous.

Therefore, in order to enhance the application, the web app needs to have a compromise between larger images to support larger screens and smaller images to support smaller screens.

Project Management

All of the team members including the team leader had no experience with web app development. So, the team leader researched the requirements and assigned tasks to each team member every week. However, although the plan seemed fine, it did not work nicely.

The work allocation was not ideal as there were often times where we would find too much work focused on a particular person while another had too little work.

Therefore, it is the consensus that the project management scheme used this time is not ideal as we did not take into account more carefully the web development competency of each member and split the tasks in a more productive manner.

However, the team was able to finish the project on time, which implies the team was well on track anyway.

6.5 Future work

If the team had another 5 weeks, the team might have implemented an application that syncs with the cloud storage without having users to add the images with enhanced security. Additionally, adding error handlings, and refactoring need to be done if the team had more time. The ability to process multiple images, secure SQL handling and better security features are also the things that can be done.

Overall, if the team had 5 weeks more, the team would have been able to improve the web app that can be used as a commercial application or better framework for existing food social networking services such as “Miil [17]”.

Bibliography

[1] Microsoft, “Microsoft Azure Documentation: AI + Machine Learning,” [Online]. Available: https://docs.microsoft.com/en-gb/azure/#pivot=products&panel=ai. [Accessed 20 March 2018].

[2] Azumio Inc., “Calorie Mama,” 2017. [Online]. Available: http://www.caloriemama.ai. [Accessed 20 March 2018].

[3] C. Smith, “MIT's new app uses AI to look at food and tell you the recipe,” 20 July 2017. [Online]. Available: http://bgr.com/2017/07/20/mit-food-recognition-app/. [Accessed 20 March 2018].

[4] C. o. t. E. U. European Parliament, “Regulation (EU) No 1169/2011 of the European Parliament and of the Council of 25 October 2011 on the provision of food information to consumers, amending Regulations (EC) No 1924/2006 and (EC) No 1925/2006 of the European Parliament and of the Council, an,” 25 October 2011. [Online]. Available: https://eur-lex.europa.eu/legal-content/EN/ALL/?uri=CELEX%3A32011R1169. [Accessed 20 March 2018].

[5] Microsoft, “QnA Maker API PREVIEW,” [Online]. Available: https://azure.microsoft.com/en-gb/services/cognitive-services/qna-maker/. [Accessed 20 March 2018].

[6] Microsoft, “Bing Autosuggest API,” [Online]. Available: https://azure.microsoft.com/en-gb/services/cognitive-services/autosuggest/. [Accessed 20 March 2018].

[7] UIKit, “UIKit BETA,” [Online]. Available: https://getuikit.com. [Accessed 20 March 2018].

[8] T. j. Foundation, “jQuery API,” [Online]. Available: http://api.jquery.com. [Accessed 20 March 2018].

[9] alubbe, “jade,” 2015. [Online]. Available: https://www.npmjs.com/package/jade. [Accessed 20 March 2018].

[10] Microsoft, “Computer Vision API,” [Online]. Available: https://azure.microsoft.com/en-gb/services/cognitive-services/computer-vision/. [Accessed 20 March 2018].

[11] D. Grossman and Awio Web Services LLC., “Date Range Picker,” 2017. [Online]. Available: http://www.daterangepicker.com. [Accessed 20 March 2018].

[12] PHOTOMETADATA.org, “META Resources - Standards : Exif,” [Online]. Available: https://www.photometadata.org/META-Resources-metadata-types-standards-Exif. [Accessed 20 March 2018].

[13] oeuillot, “exif,” 2016. [Online]. Available: https://www.npmjs.com/package/exif. [Accessed 21 March 2018].

[14] tonybadguy, “yelp-fusion,” 2017. [Online]. Available: https://www.npmjs.com/package/yelp-fusion. [Accessed 21 March 2018].

[15] Y. Inc., “Get started with the Yelp Fusion API,” [Online]. Available: https://www.yelp.com/developers/documentation/v3/get_started. [Accessed 21 March 2018].

[16] nchaulet, “node-geocoder,” February 2018. [Online]. Available: https://www.npmjs.com/package/node-geocoder. [Accessed 21 March 2018].

[17] m. inc., 15 February 2018. [Online]. Available: https://itunes.apple.com/us/app/miil-food-camera-sns/id472973118?mt=8. [Accessed 21 March 2018].

[18] Google, “Google Maps APIs Documentation,” [Online]. Available: https://developers.google.com/maps/documentation/javascript/geocoding#ReverseGeocoding. [Accessed 20 March 2018].

[19] patosai, “grunt-image-autoresize,” 2017. [Online]. Available: https://www.npmjs.com/package/grunt-image-autoresize. [Accessed 21 March 2018].

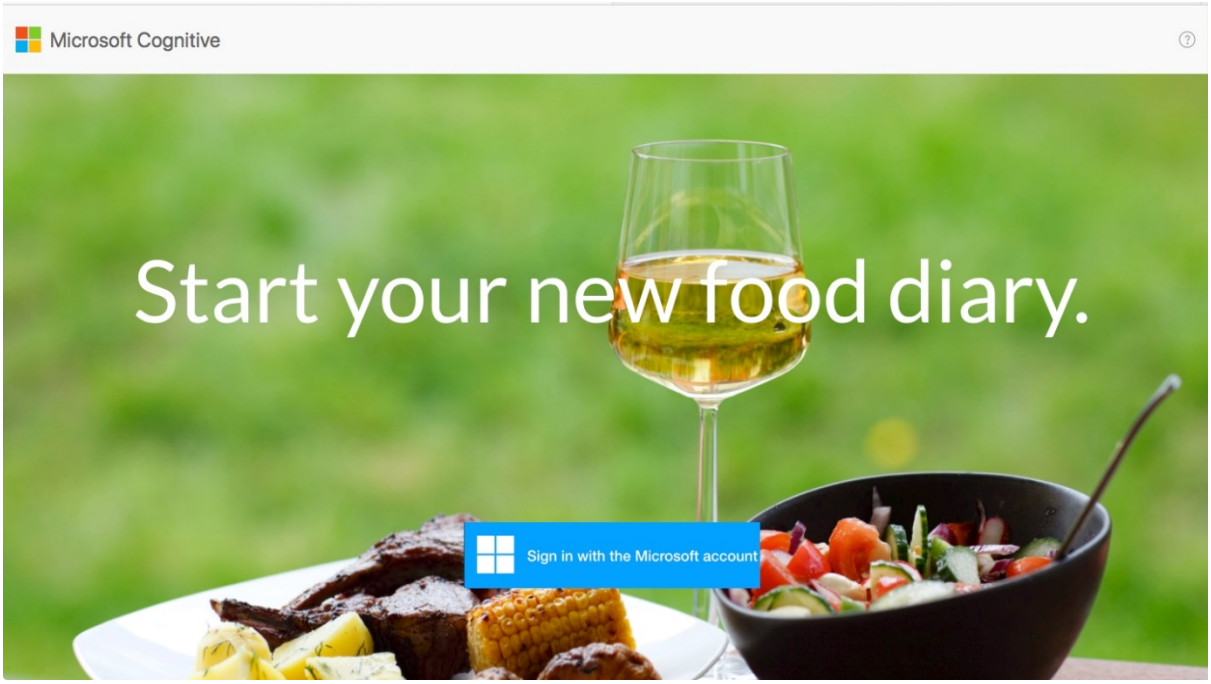
[20] I. S. LLC, “ImageMagick,” [Online]. Available: https://www.imagemagick.org/script/index.php. [Accessed 21 March 2018].

Appendix

- Web app URL: https://food.azurewebsites.net
- Sign-in credentials: Use any standard Microsoft account that has a domain of hotmail.com or outlook.com.

User manual

What is Food Diary?



Food Diary is an intuitive way of recording your food eating history. Food Diary combines artificial intelligence enabled automatic card generation functionality and support for the cloud storage providers that lets you focus on your food eating experiences. Generated cards would include the information about the food and the feeling of the user based on a food image and a face image pair.

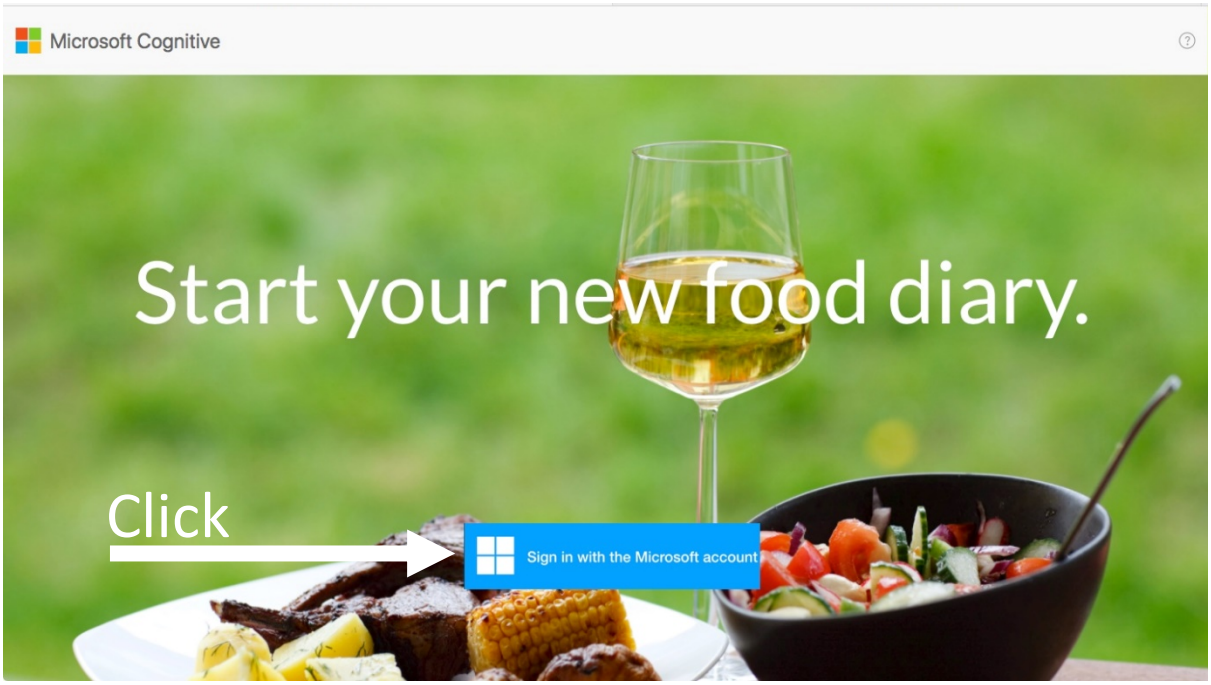
This application supports mobile devices as well.

Food Diary supports various features including:

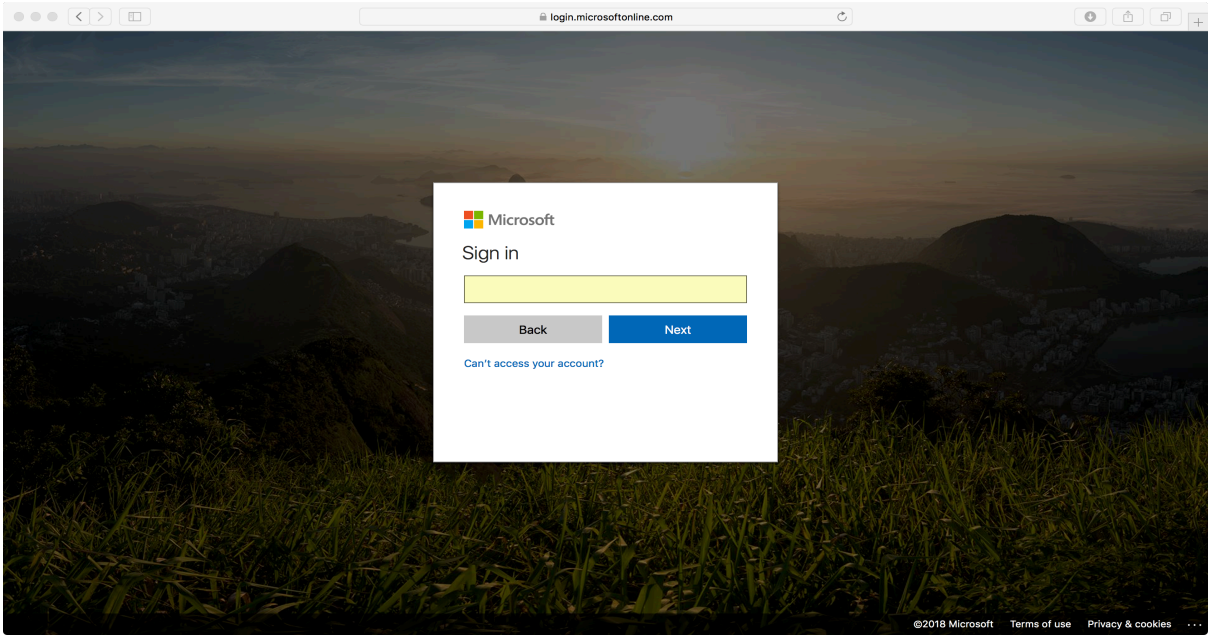
- Automatic card (diary entry) generation using Microsoft Cognitive APIs
- Manual allergens recording capability
- Image retrieval from OneDrive, Google Drive or Dropbox
- Personalised restaurant suggestions
- Feeling trend graph based on your food eating history
- Fast search through cards with various options
- Intelligent FAQ

Sign-in to the application

To sign-in, you need a standard Microsoft account. If you want to make an account or want to sign-in, click “Sign in with the Microsoft account” button.



Then, you would be redirected to the Microsoft sign-in page. Proceed the sign-in from Microsoft.



After you sign-in to Microsoft, you are ready to use the application, and the main page of the application will be shown.

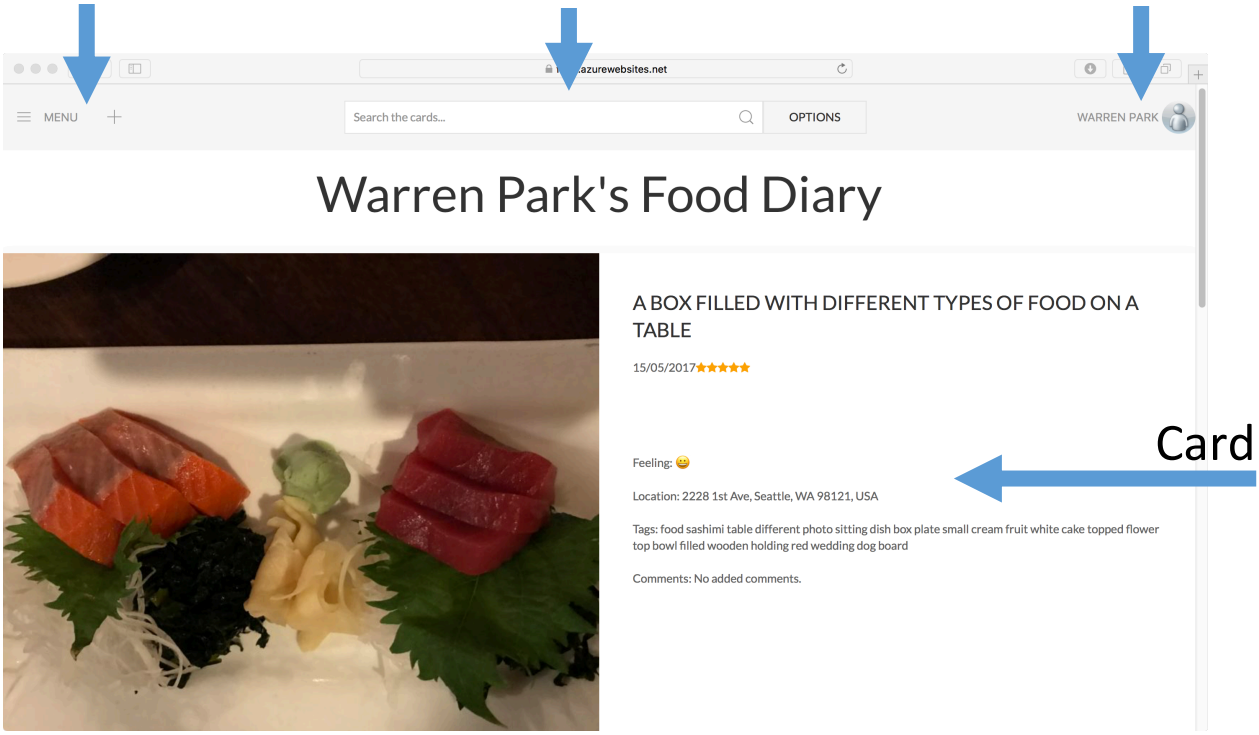
Food Diary interface overview

The Food Diary main page has three main controller areas.

Menu: you can use key features from here.

Search: you can search through your cards here.

Account: you can alter your account settings here.



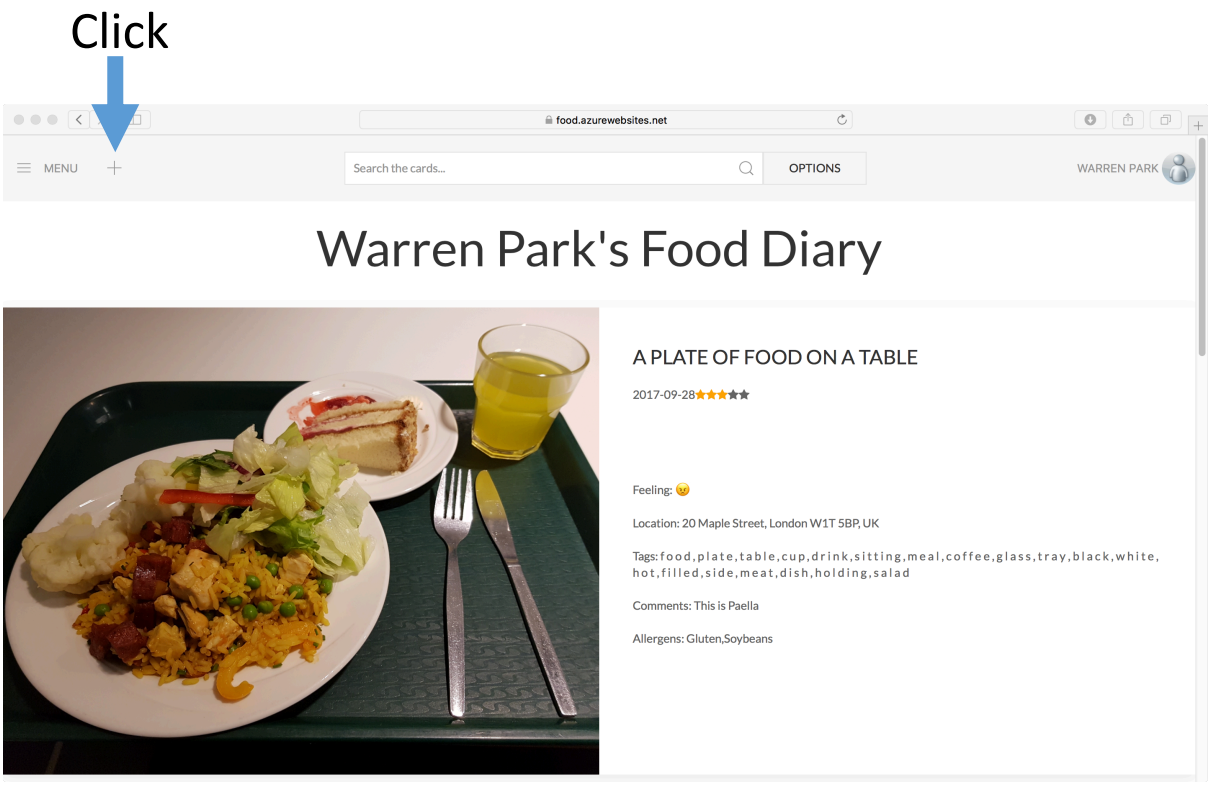
All the controls that can be done from the main page will be shown if you hover your mouse cursor or touch the control area. Each card also has a control area at the bottom, which you can use to edit, copy and delete the card.

Create a card

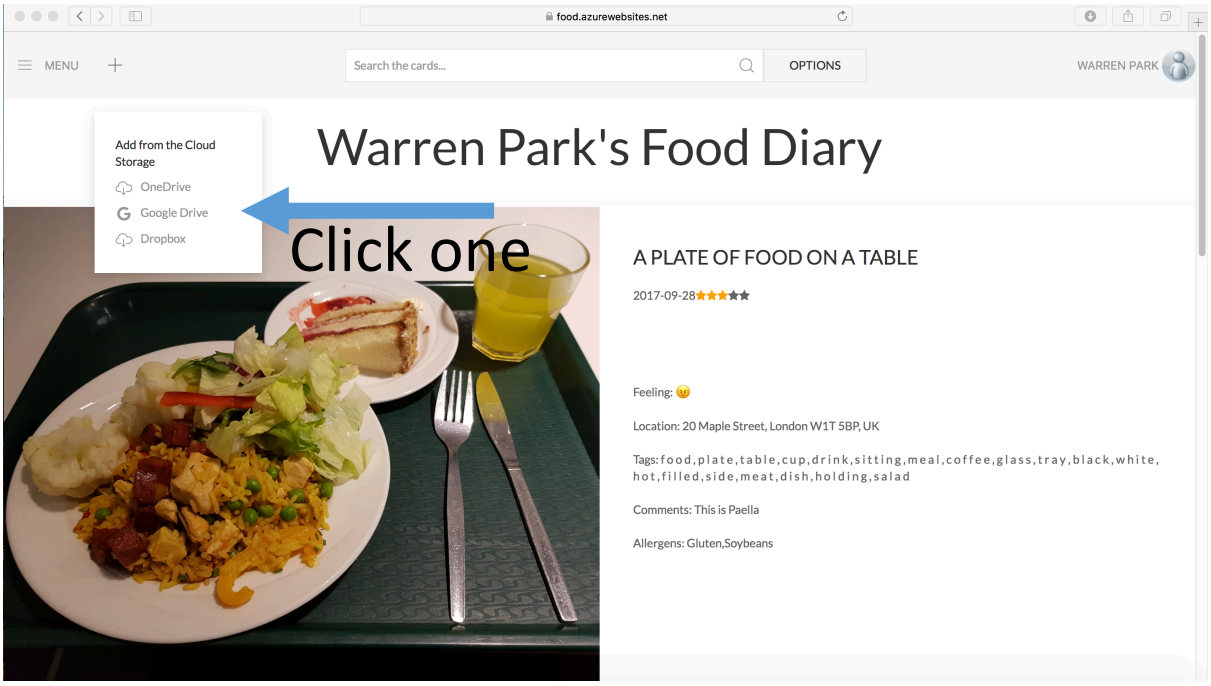
A card is like a diary entry – it is an area where you can record individual food eating history. Each card will include an image of the food and the analysis about the food.

To create a card, you need a cloud storage account from either Microsoft OneDrive, Google Drive or Dropbox. Your cloud storage account should have a food image and a selfie that you have taken after you have eaten the food.

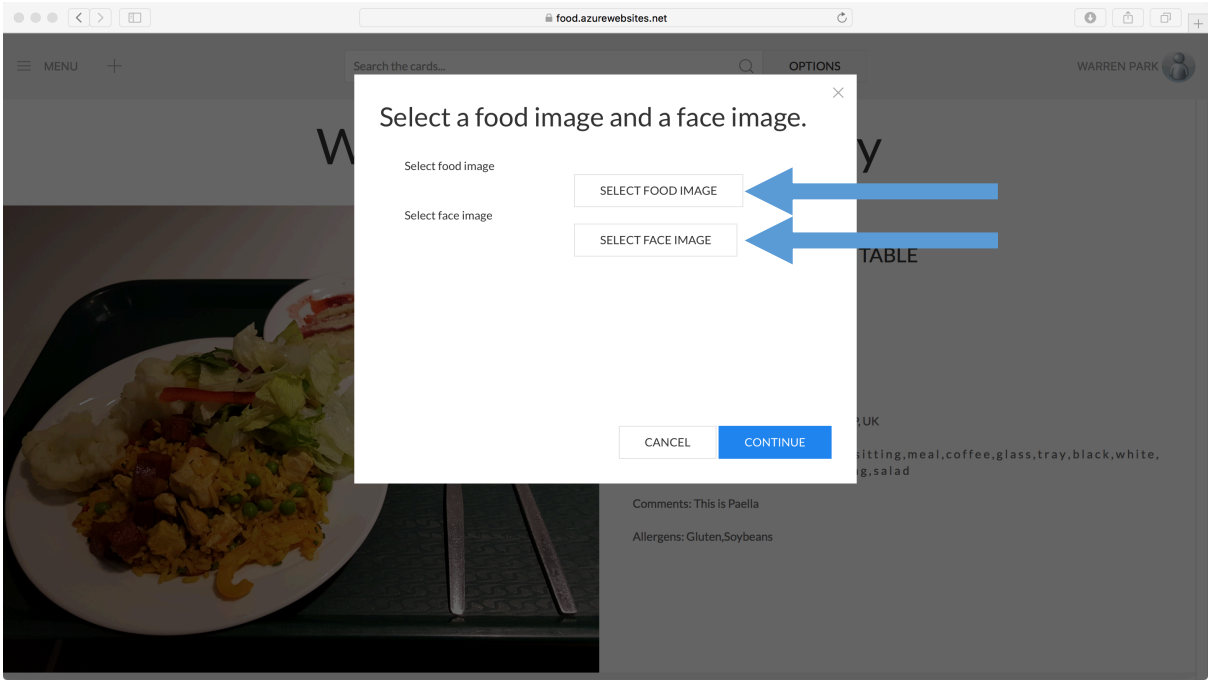
When you are ready with above requirement, click “+” button in the **Menu** area.



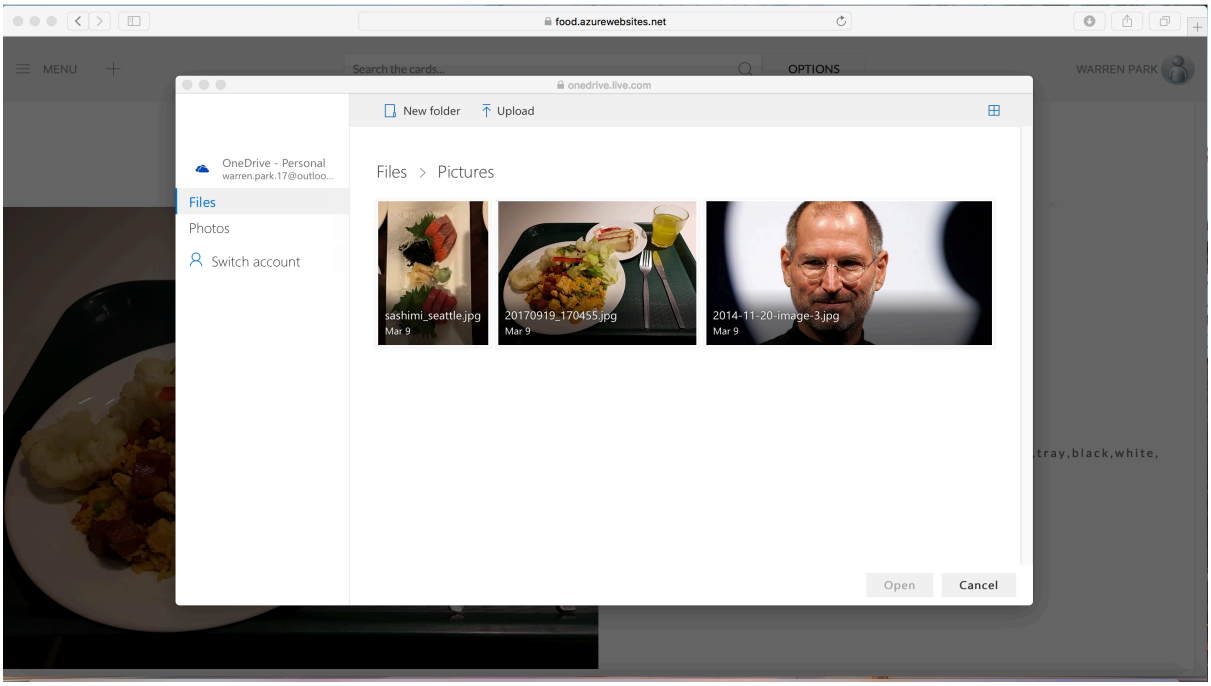
Then, a dropdown menu will be shown. Choose the cloud storage account that the required files are stored and click the corresponding cloud storage button.



After you click one button, you can now upload images to the application from your cloud storage. Please upload a food image by clicking “SELECT FOOD IMAGE” button and a face image by clicking “SELECT FACE IMAGE”.

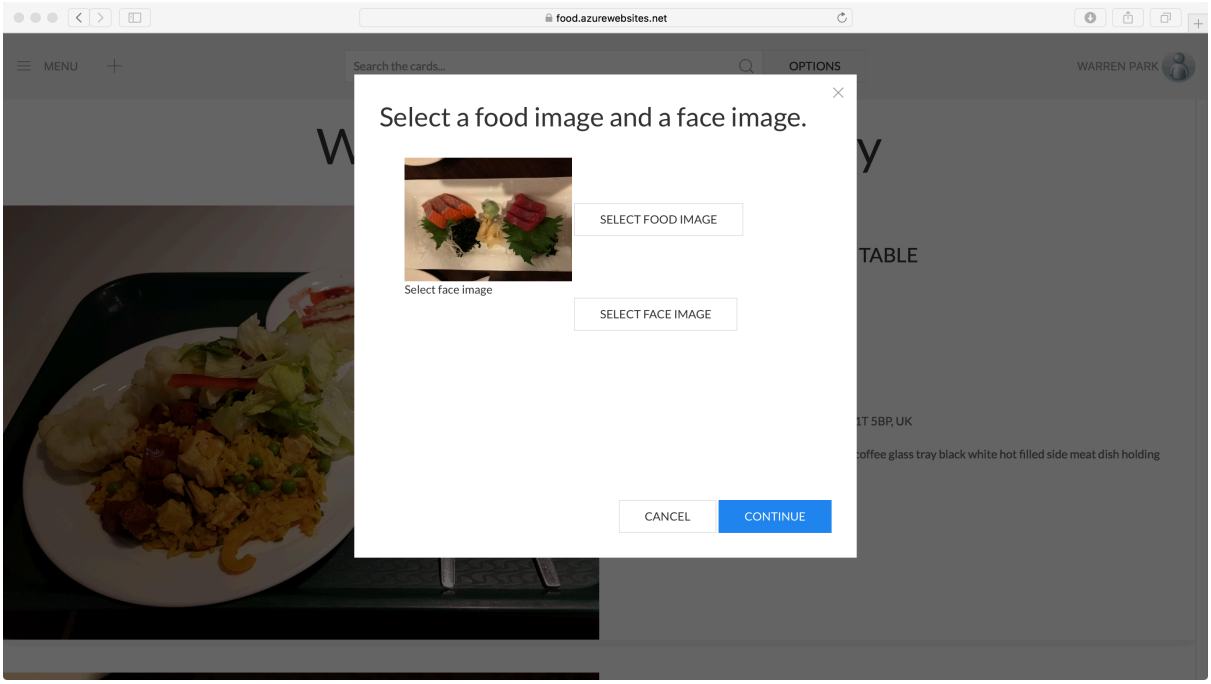


If you click any of those two buttons, a file picker from selected cloud storage will be shown like below:

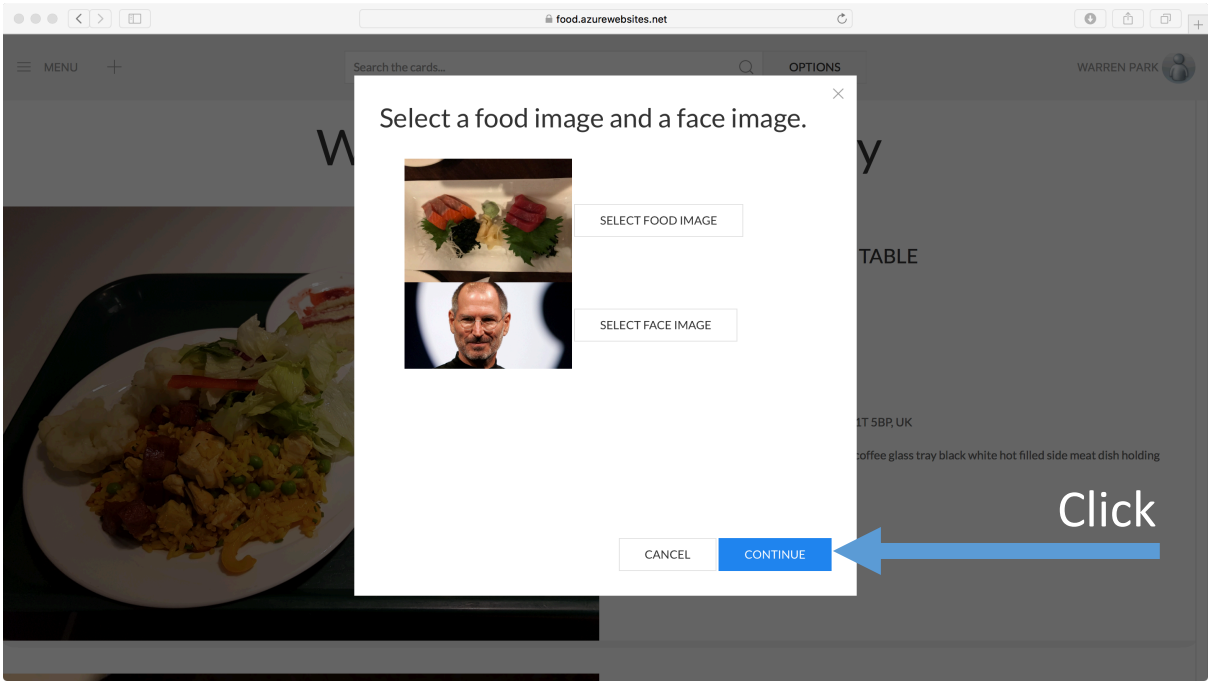


(OneDrive File Picker shown)

If you choose one food image, the food image will be uploaded to the application.

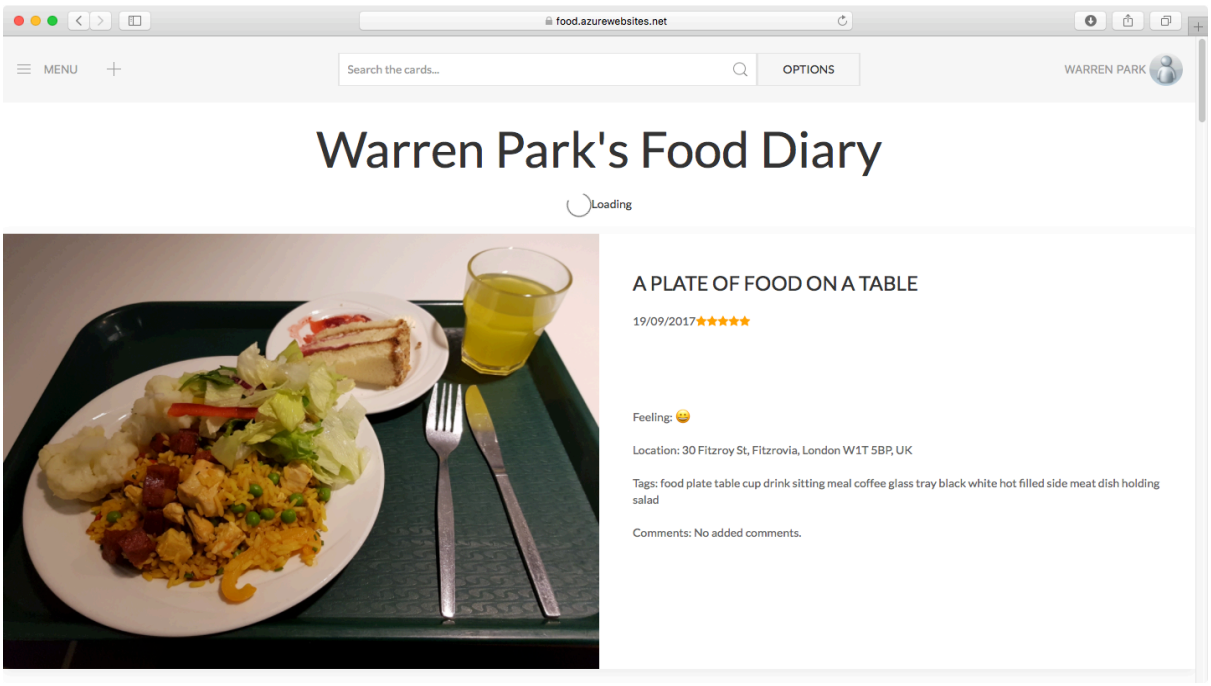


Similarly, if you upload a face image, the face image will be uploaded.

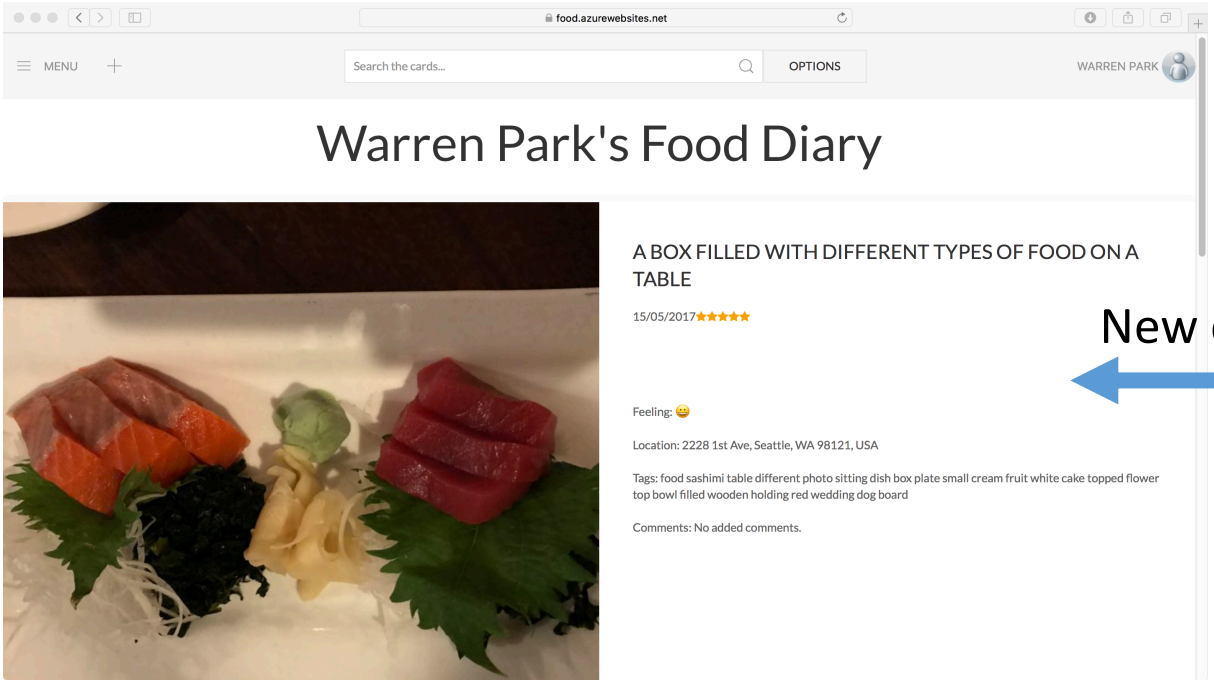


If you have uploaded a pair of images, you can click “CONTINUE” to submit your request. Unless there is a pair of images uploaded, the request cannot be processed, and in that case, you will see an error message.

After you click the “CONTINUE” button, the request will be submitted to the server. Until the program receives a submission confirmation from the server, it will show “Loading” on the screen.

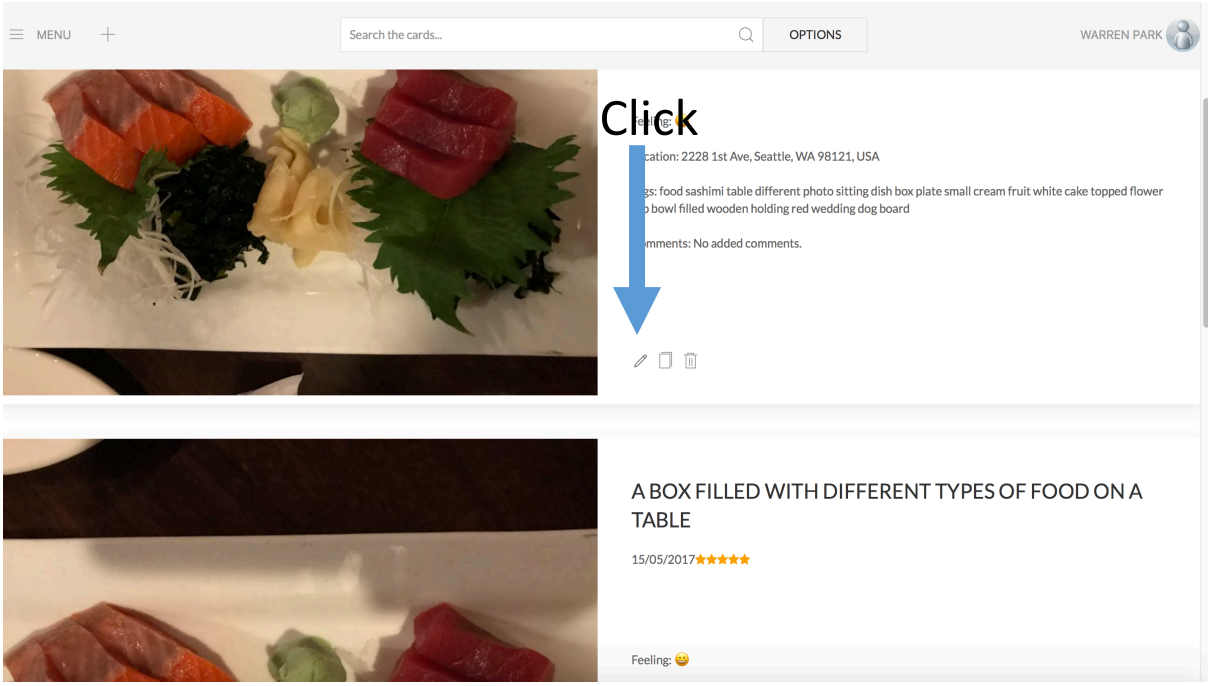


When the web app receives a response from the server, it will update the cards and you will be able to see the uploaded card.

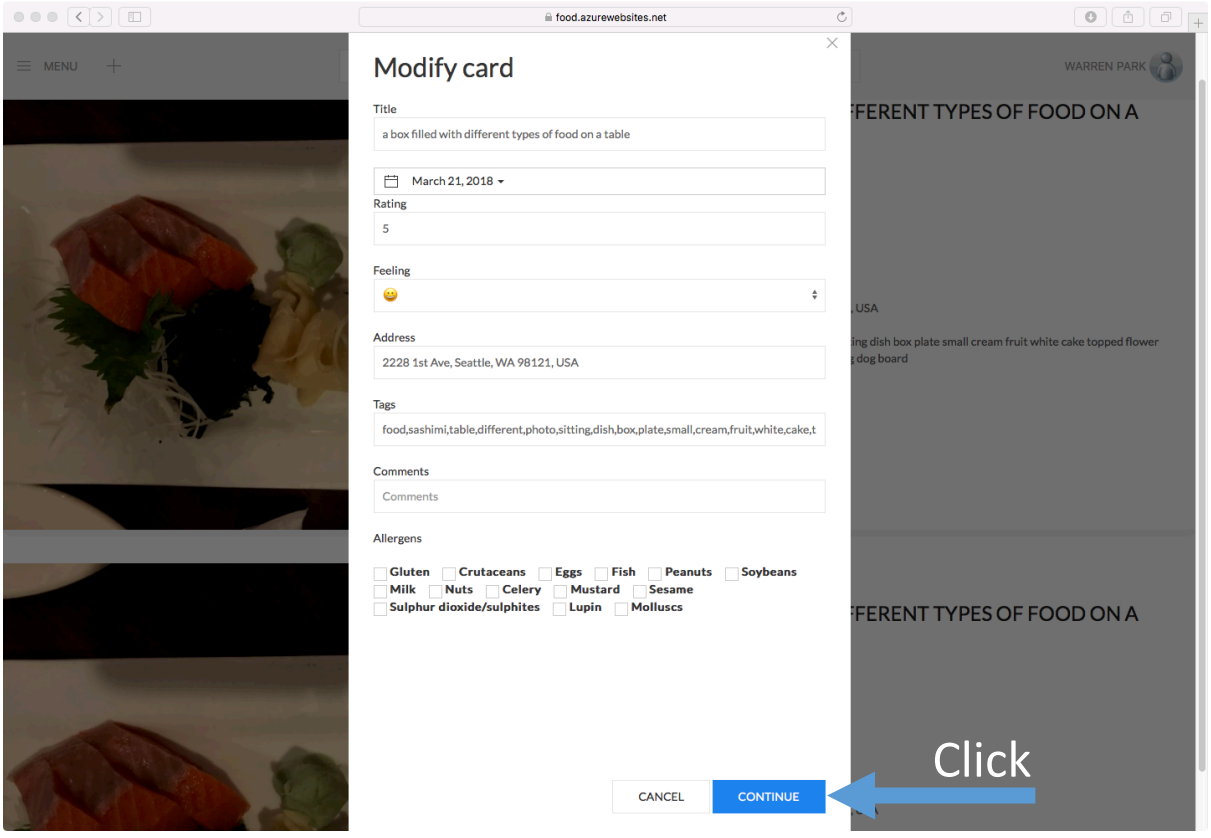


Edit a card

To edit the card, hover the cursor on the card that you want to edit. If you are using a mobile device, please touch the card instead of hovering. If you hover or touch the card, hidden buttons will be shown. For the editing, please click the edit button.



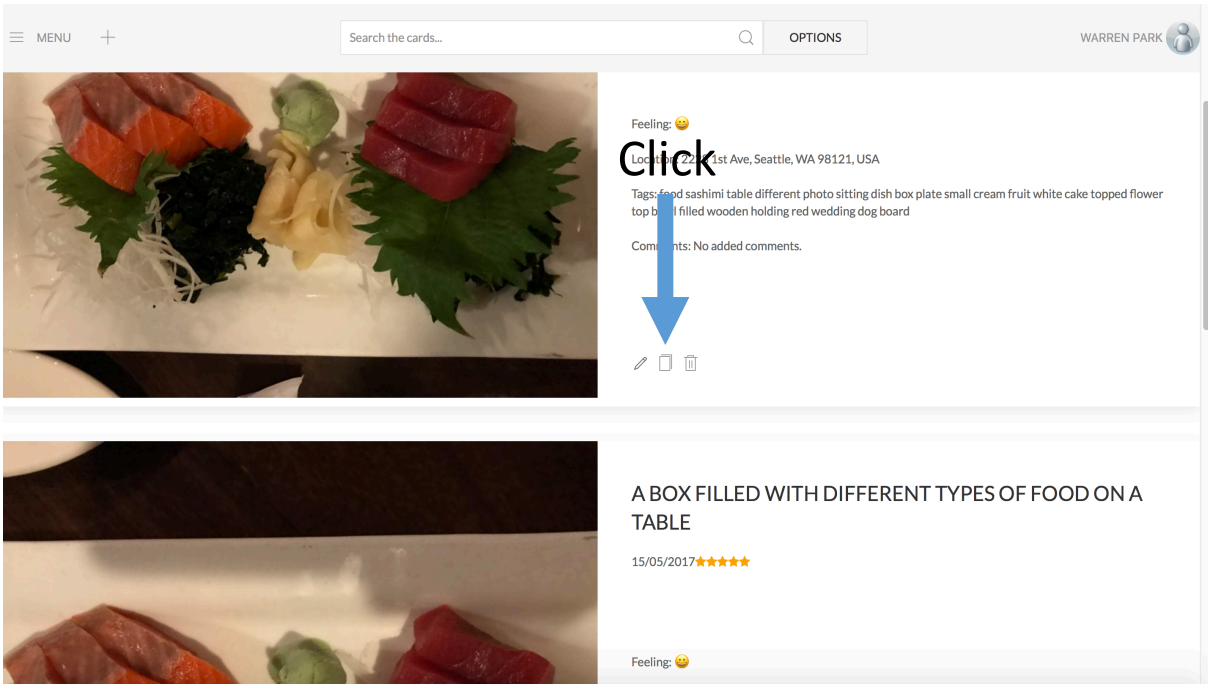
If you click the edit button, edit window will be shown. Edit the card as you like and click “CONTINUE” button at the bottom of the window.



You can also add comments and allergens here. Address that you type should be a valid human-readable address instead of coordinates.

Copy a card content

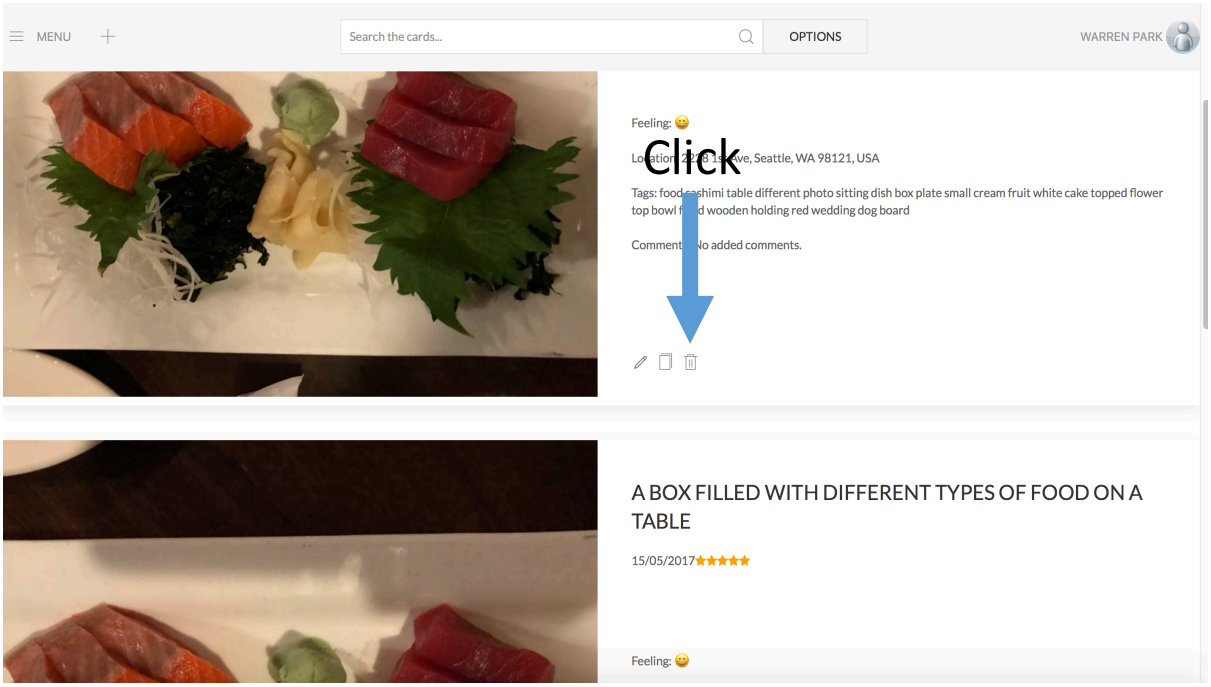
To copy the textual content of the card in order to share, click the copy button. Then a modal window that you can copy the contents from will be shown.



(NOTE: some browsers might not support copying due to the security reasons.)

Delete a card

You can also easily delete a card by clicking the delete button.

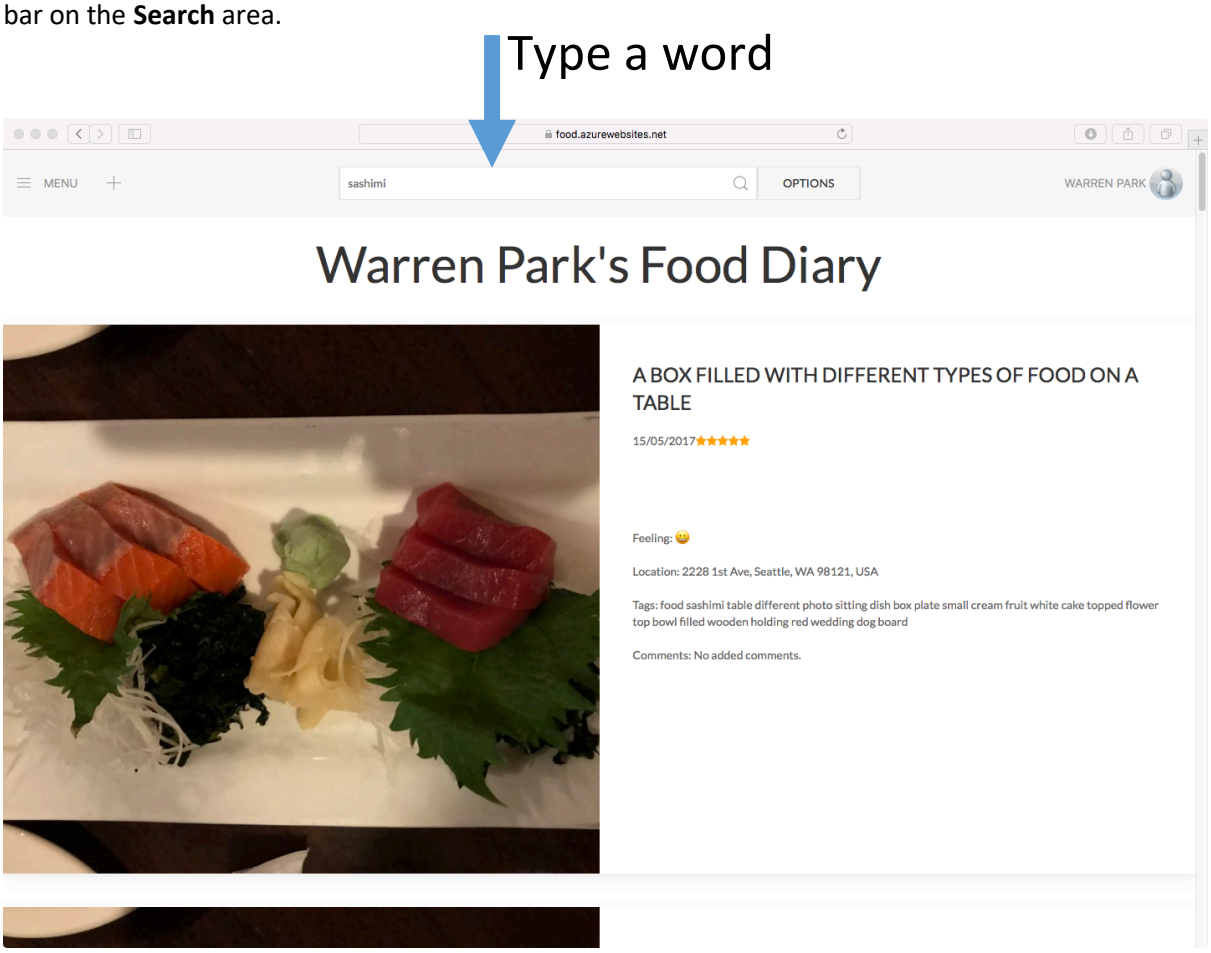


Search cards

For searching, we have three options. If you want to search the textual content on the card, you need to use the search bar. If you want to see cards for the images taken during the certain period of time, date search would be ideal. You can also view the cards generated based on images taken in certain places on the map as well by using map search.

General search

In order to search any cards that includes a word, you can search them by typing them on the search bar on the **Search** area.

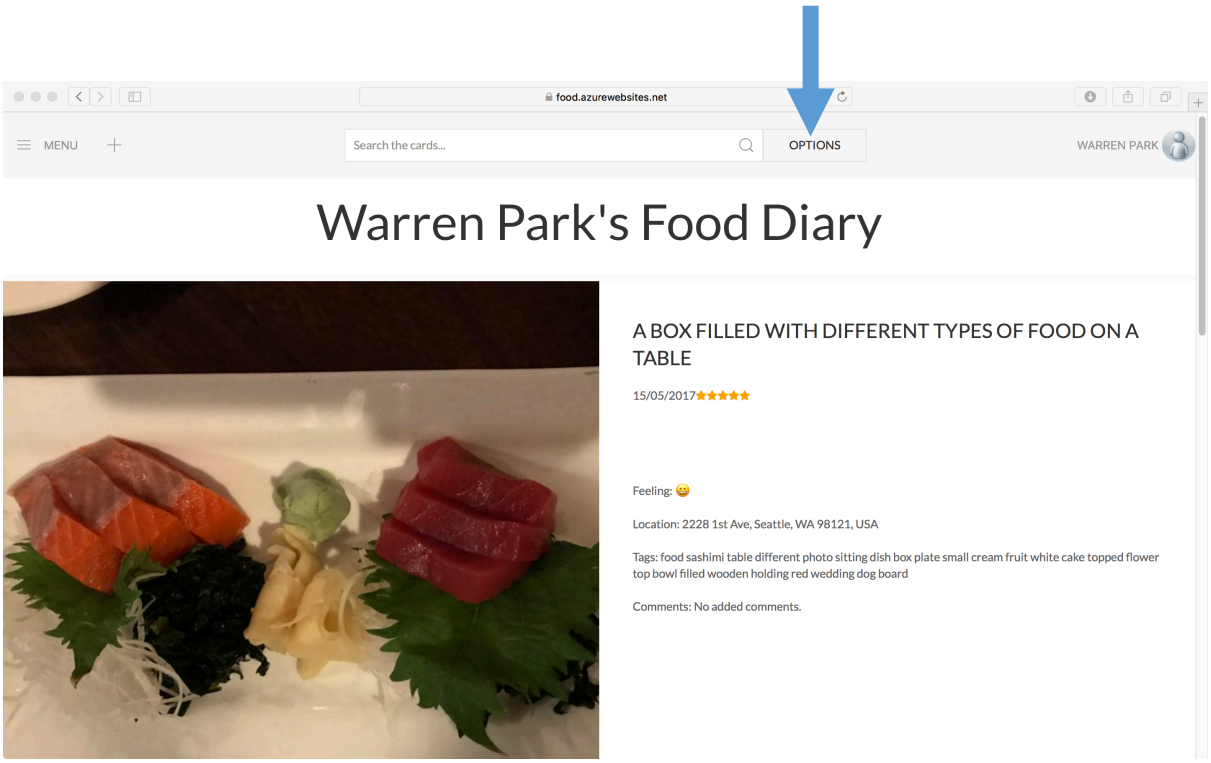


Our search is fast and real-time, so you do not need to press enter or search button in order to search. The program monitors what you have typed and carries out searching automatically.

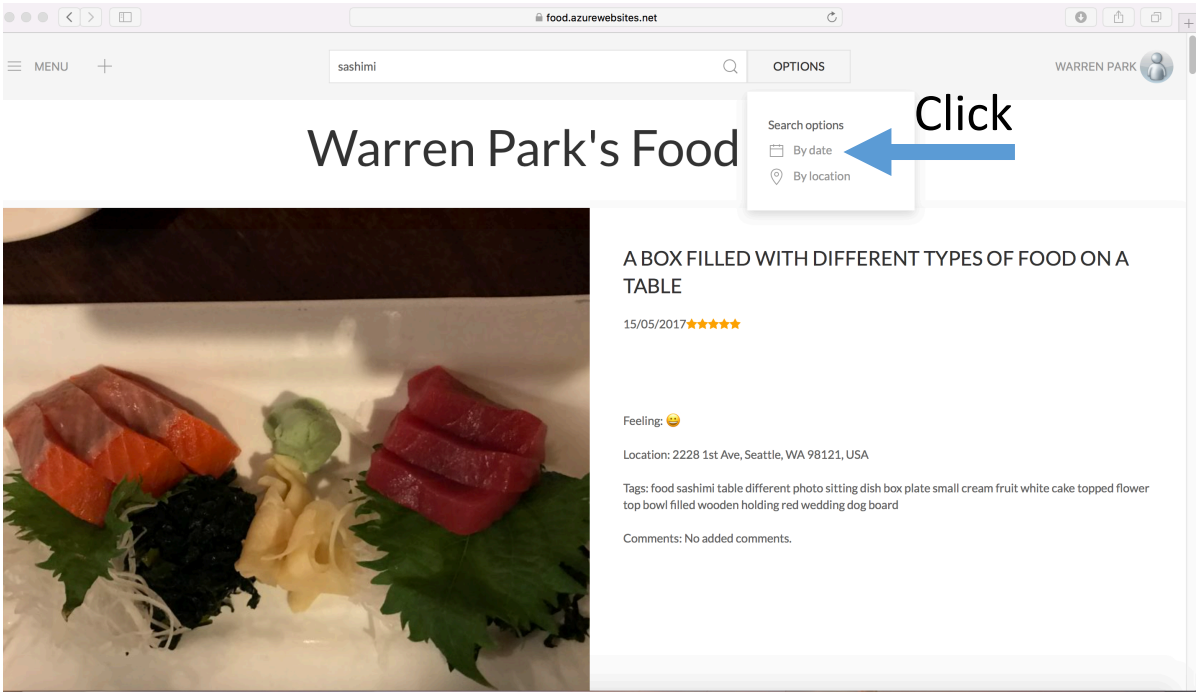
Date search

Date search enables you to search cards that are within the date range specified base on the date that the image used for the card has been taken.

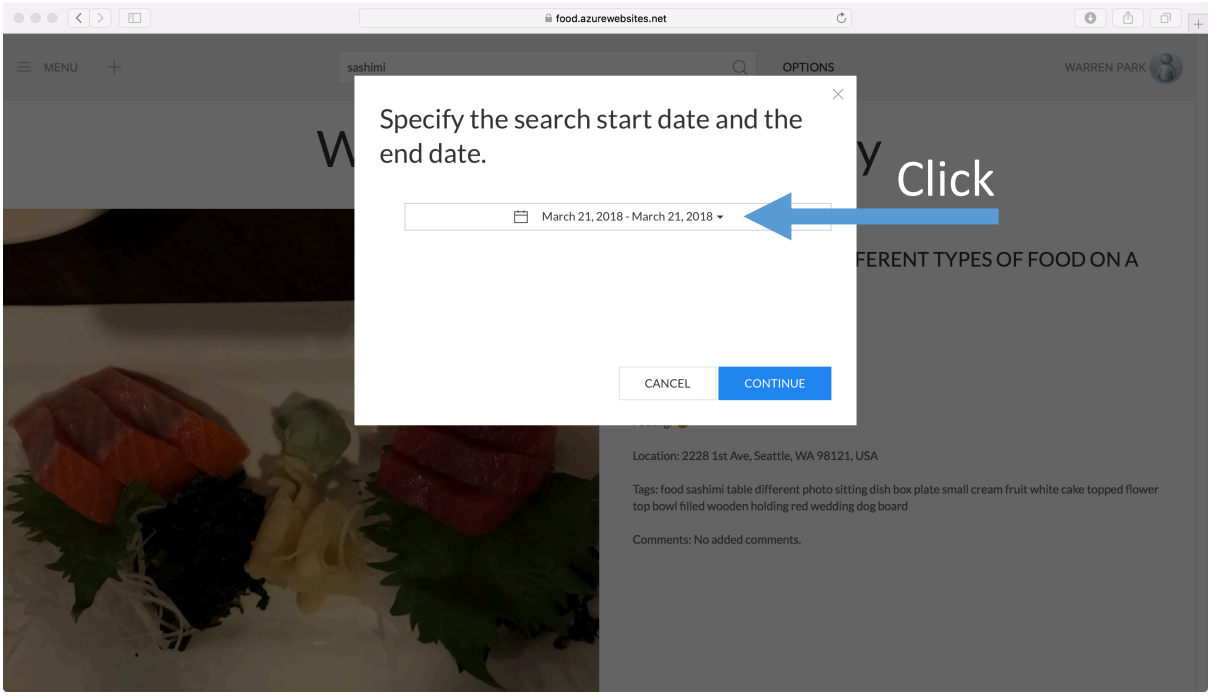
For the date search, hover the mouse cursor on or touch the “OPTIONS” button.



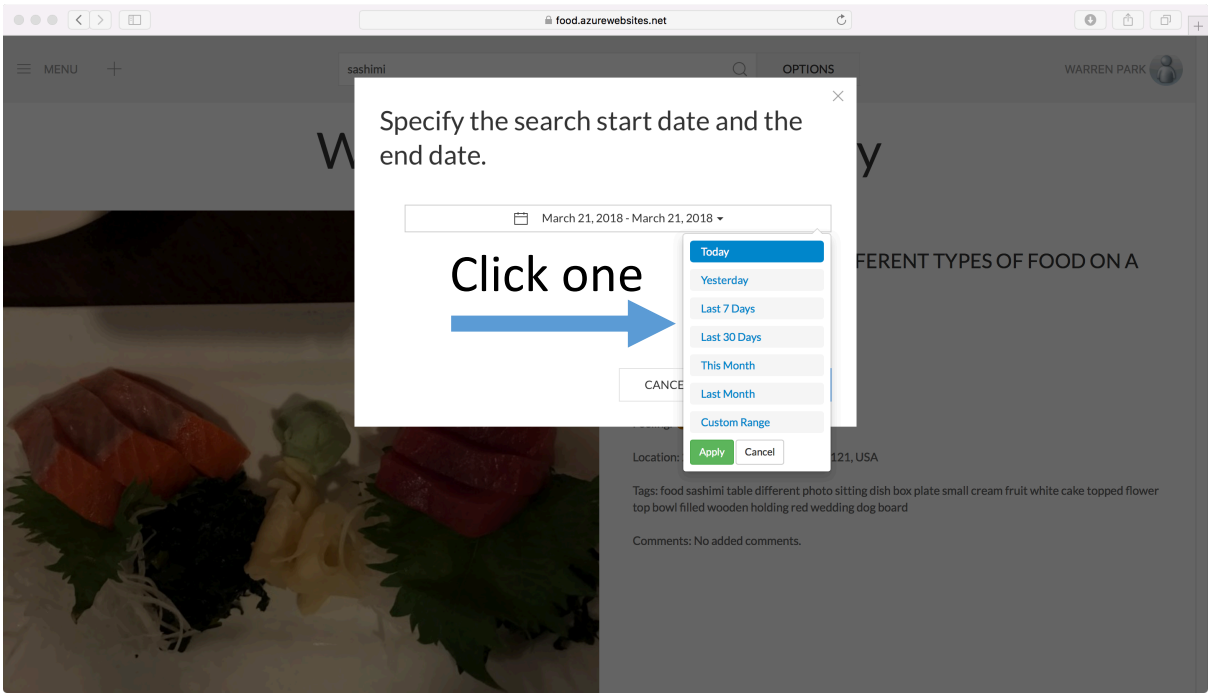
Then, you can select the search option. Click “By date” option.



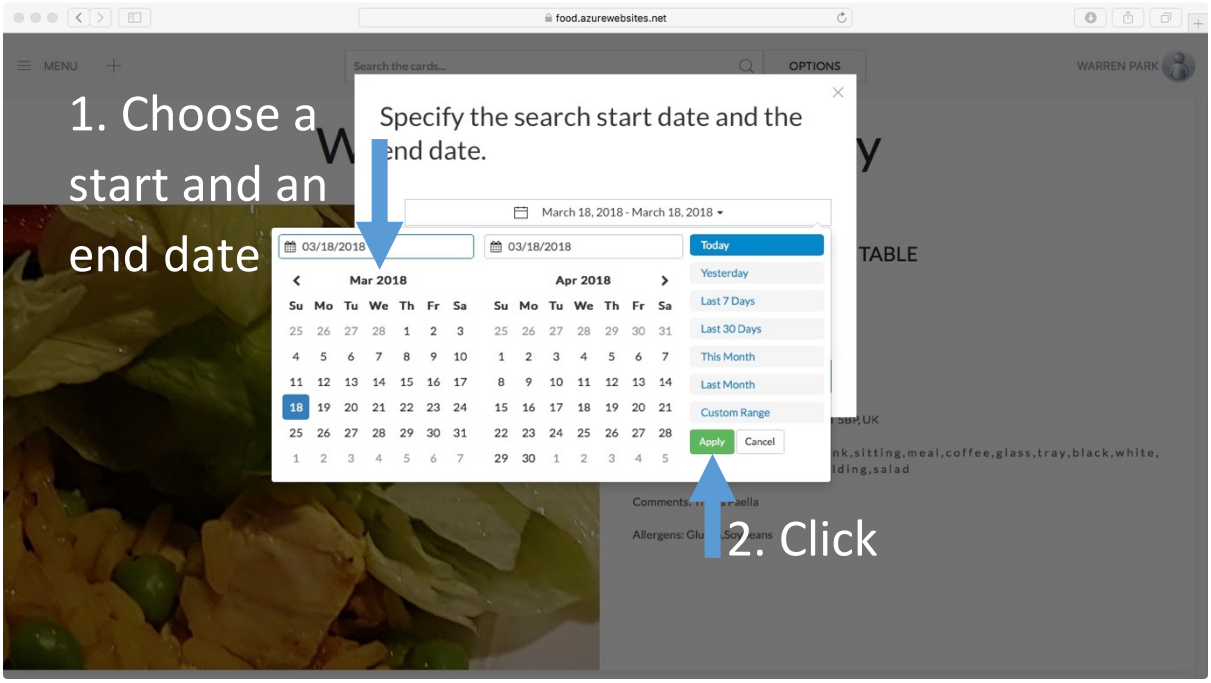
After that, you will see the date range box. Click the box to specify the search range.



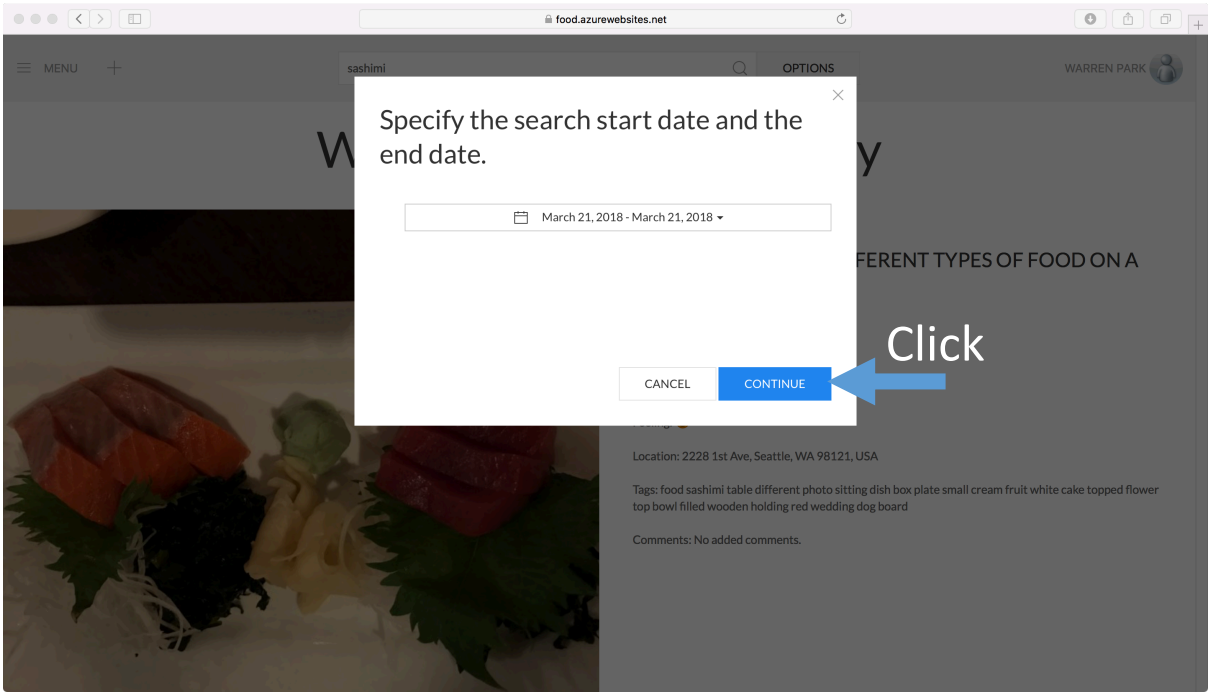
If you click the box, you will see the range specifier like below. Choose the range that you want to search for.



Alternatively, you can specify a wider range by using the calendar. In that case, click “Custom Range” button. Then calendar will be shown. Choose a start date and an end date for the search range and click the “Apply” button.



After you choose the search range, click “CONTINUE” to search.

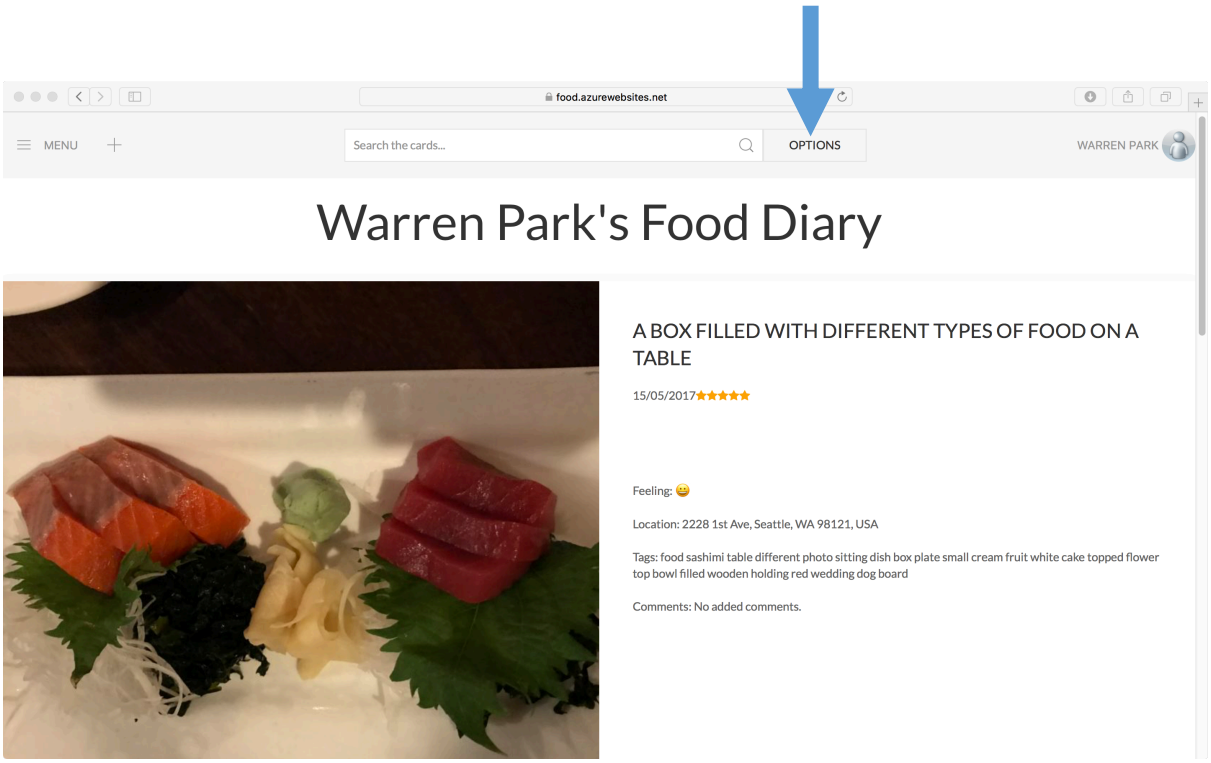


After clicking “CONTINUE” button, the search result will be given.

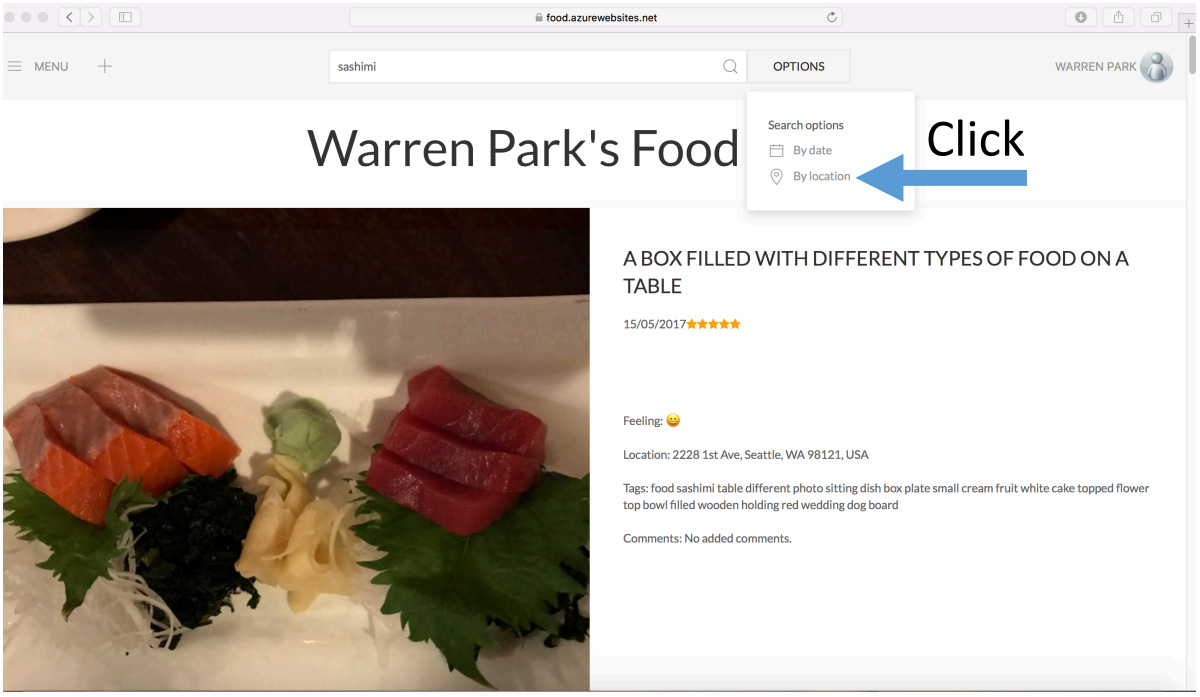
Map search

Map search uses Google Maps and enables you to see the cards represented as a pin on the map based on the address of the place that the image has been taken.

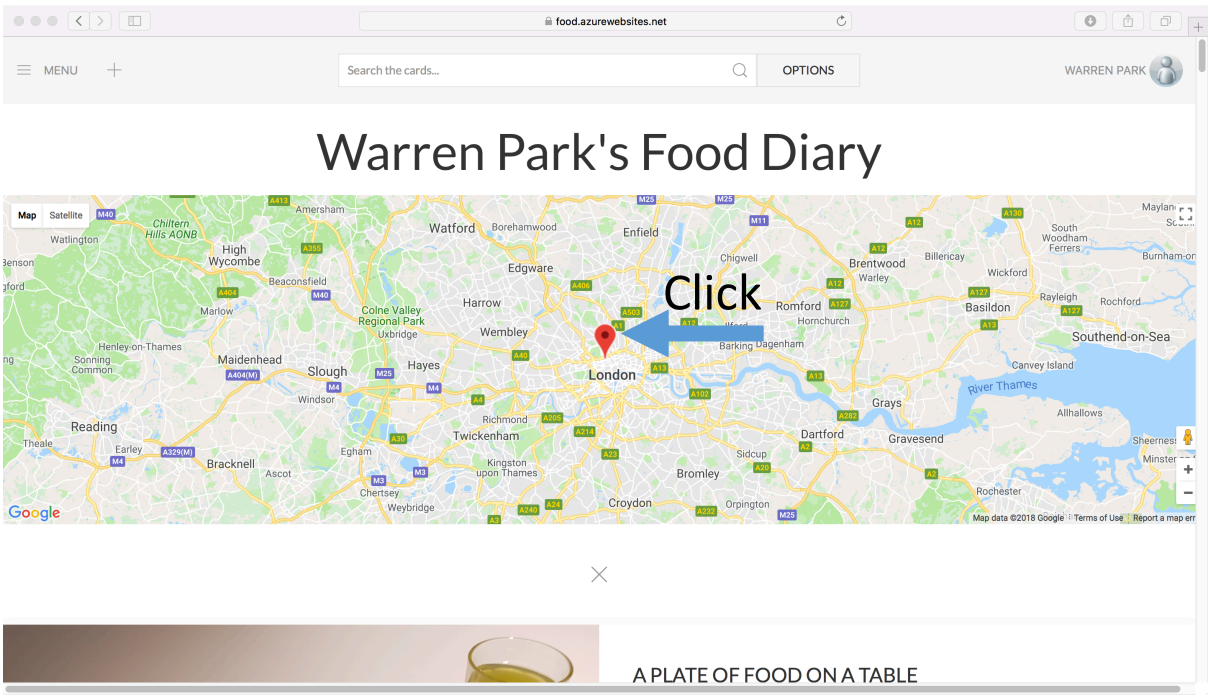
To use map search, hover the mouse cursor on or touch the “OPTIONS” button.



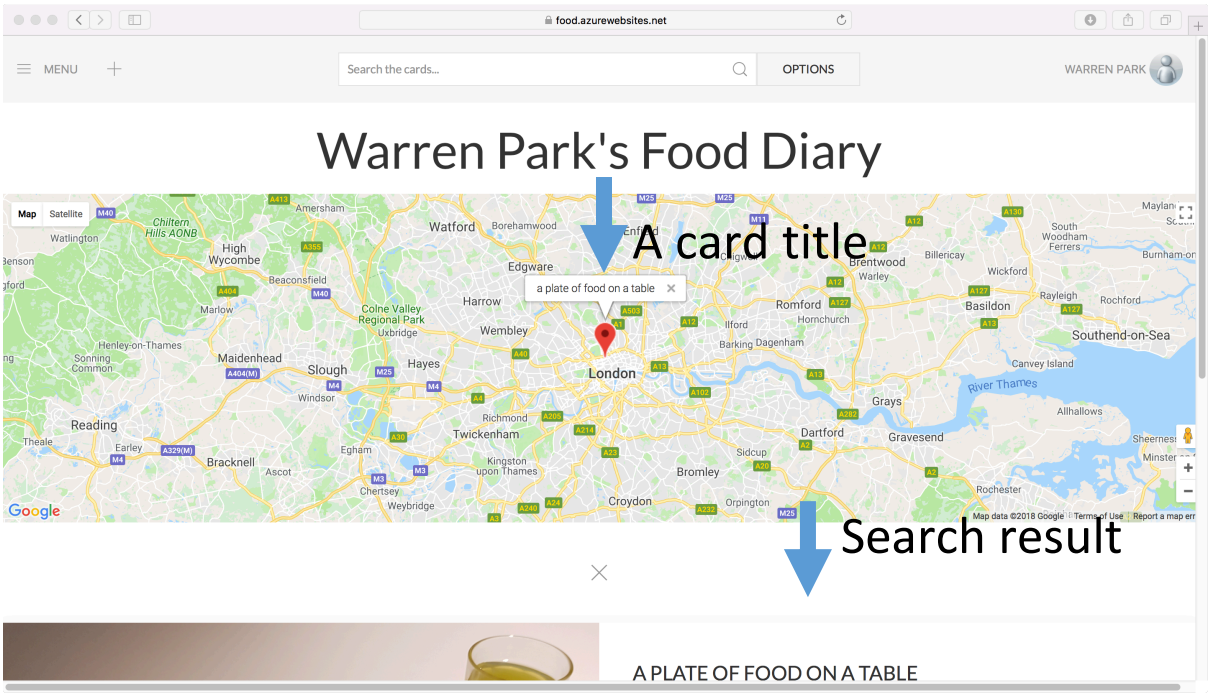
Then, you can select the search option. Click “By location” option.



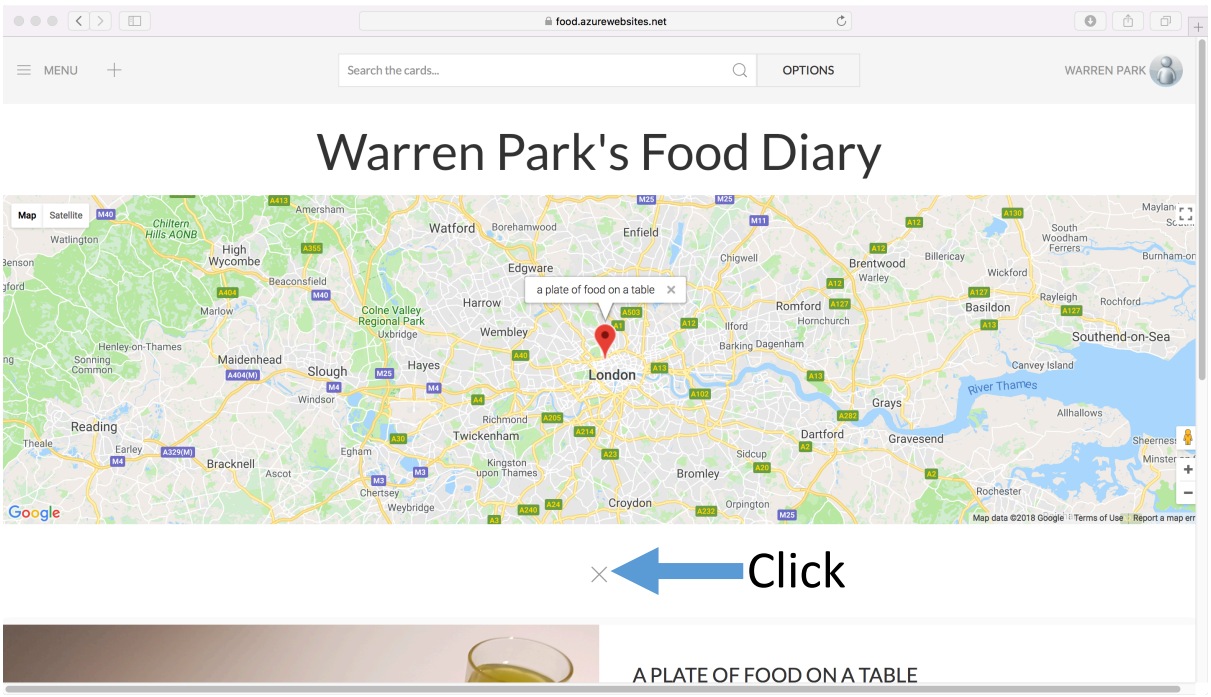
After clicking the button, the map with the pins on will be shown. Click the pins to see the cards correspond to that location.



If the pin has been clicked, a title of the card will be displayed above the pin and the cards correspond to that location will be shown below the map.

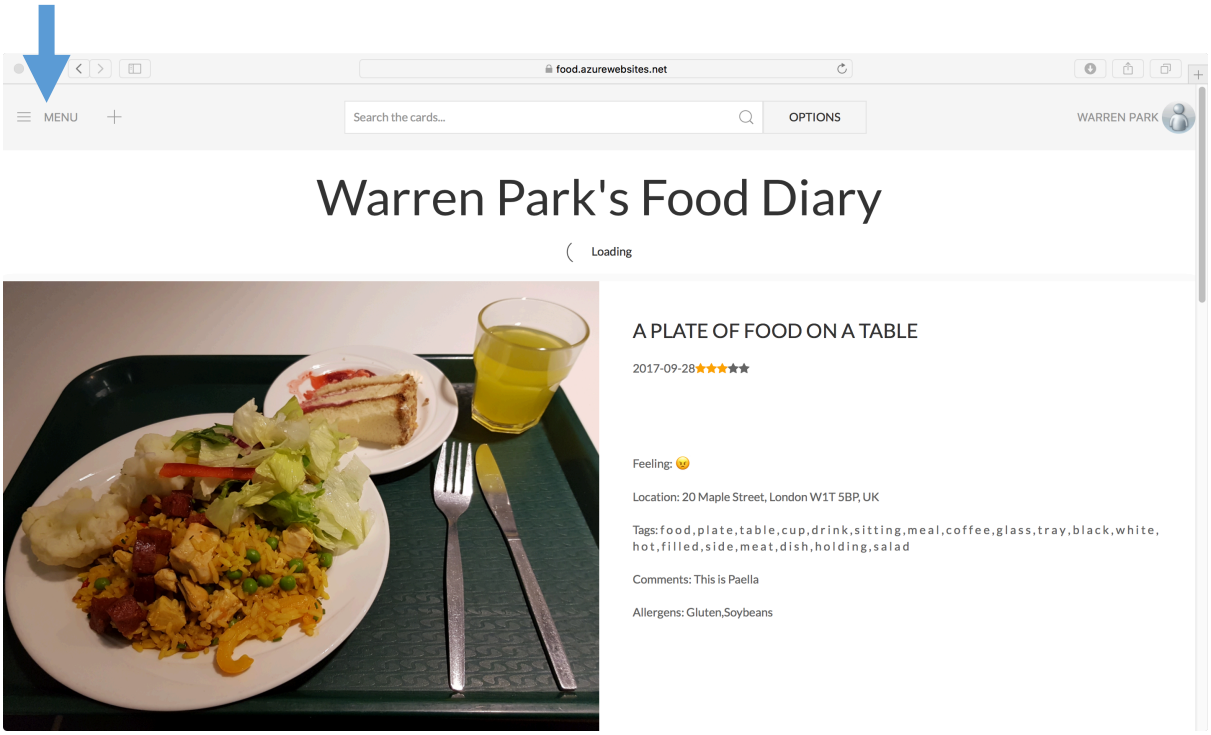


If you want to close the map, click “X” button below the map.

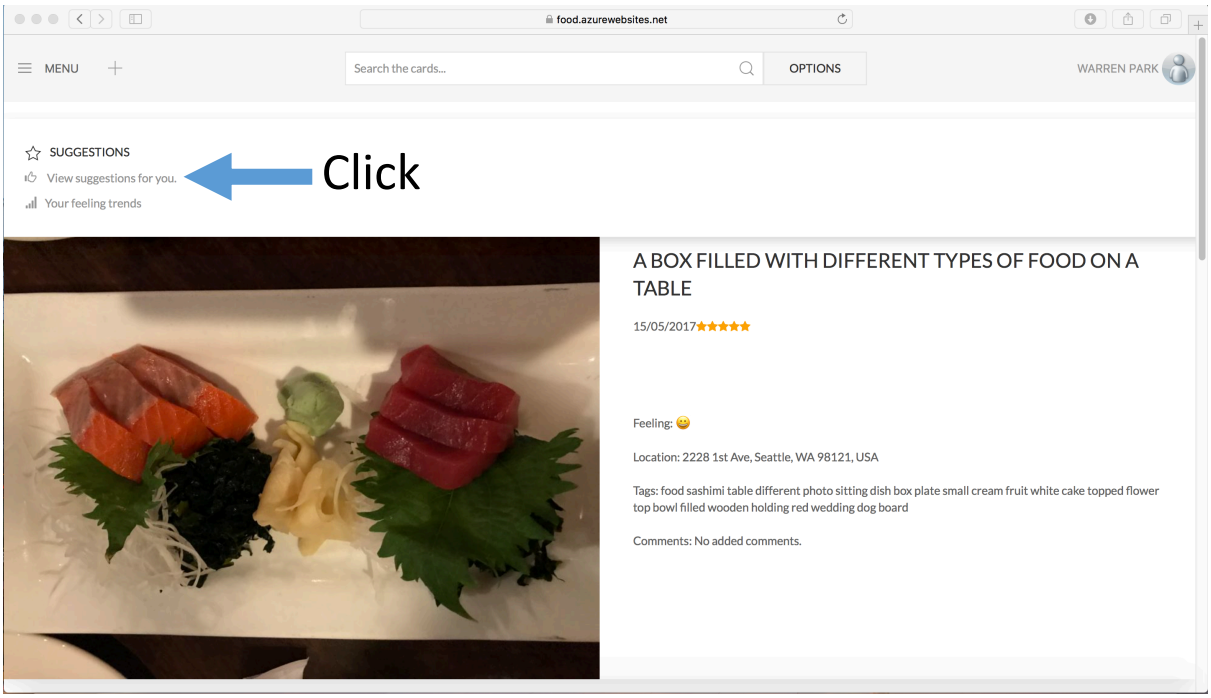


View personalised suggestions

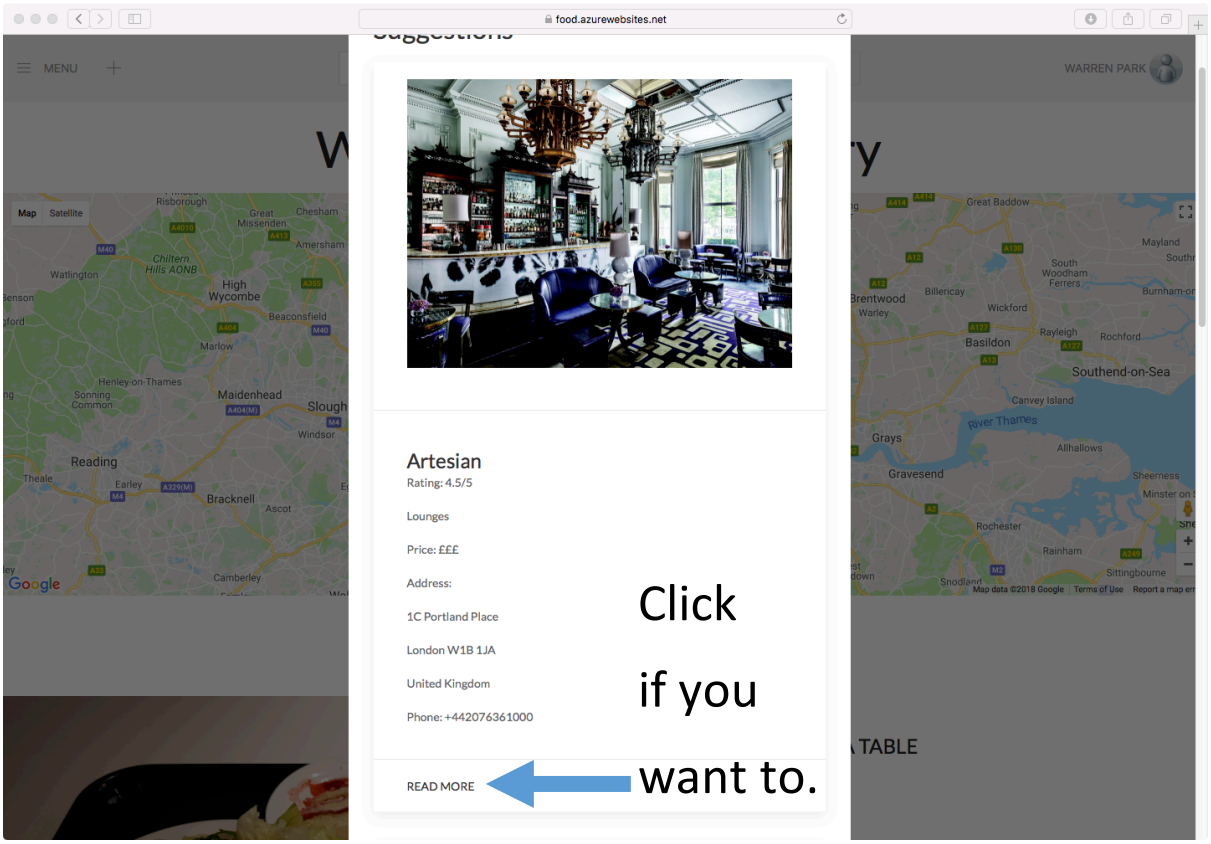
Personalised suggestions can be accessed from **Menu** area. Hover the cursor of the mouse on or touch the “Menu” button.



Then click the “View suggestions for you” button on the drop-down menu.

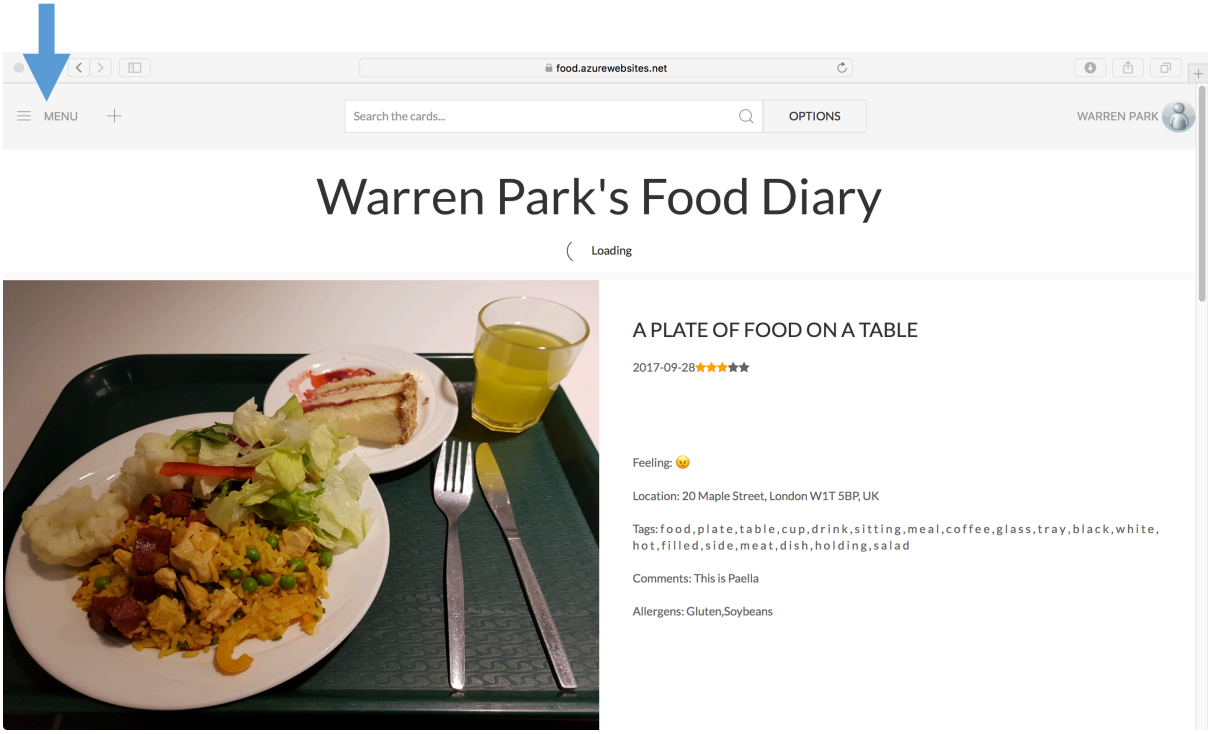


After clicking the button, you would be able to see the restaurant suggestions for you. You can read the contents more by clicking “READ MORE” button.

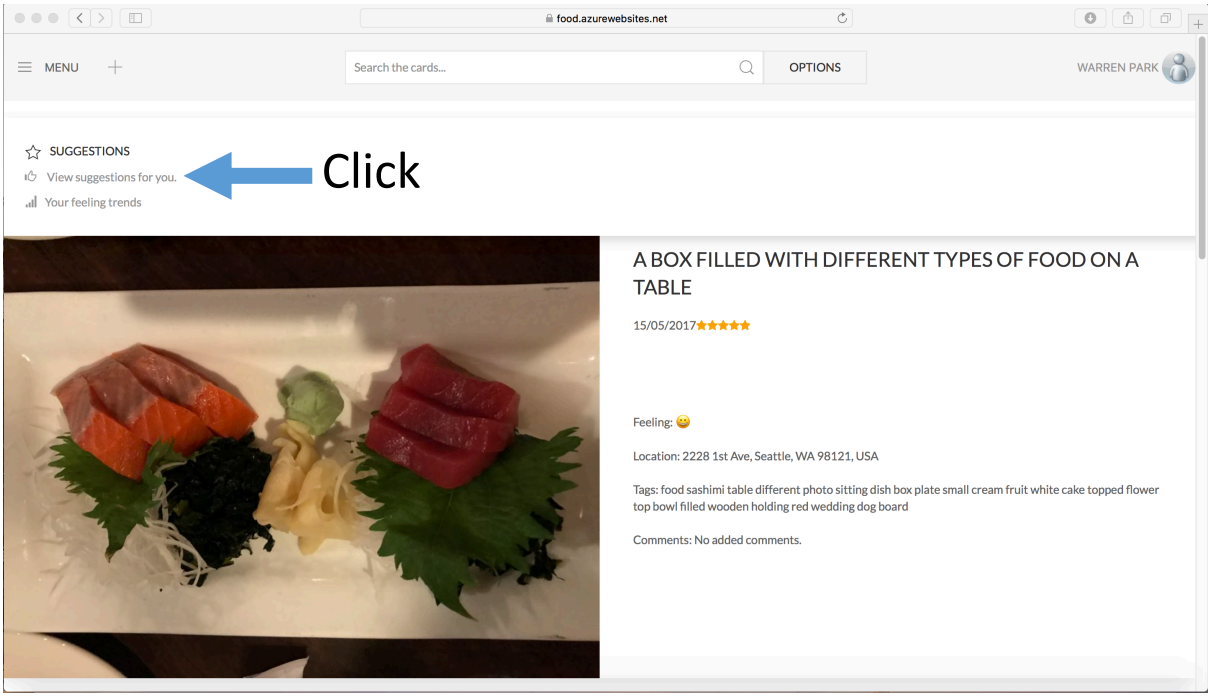


View feeling trend graph

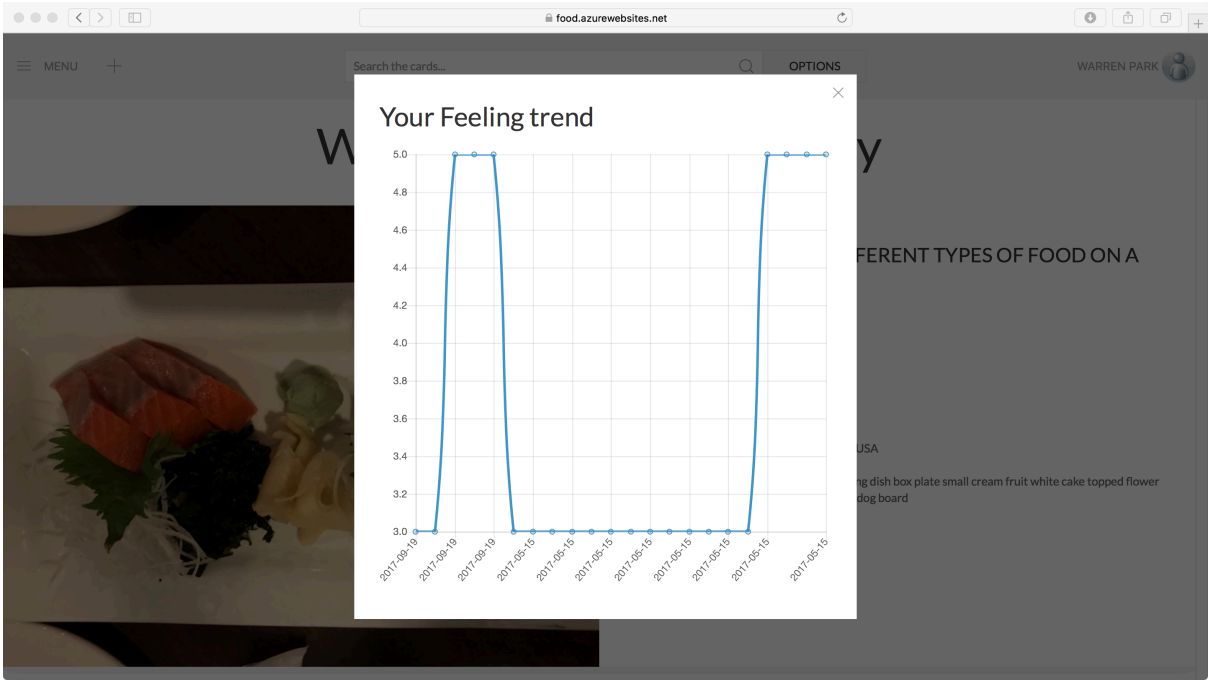
Feeling trend graph can be accessed from **Menu** area. Hover the cursor of the mouse on or touch the “Menu” button.



Then click the “Your feeling trends” button on the drop-down menu.

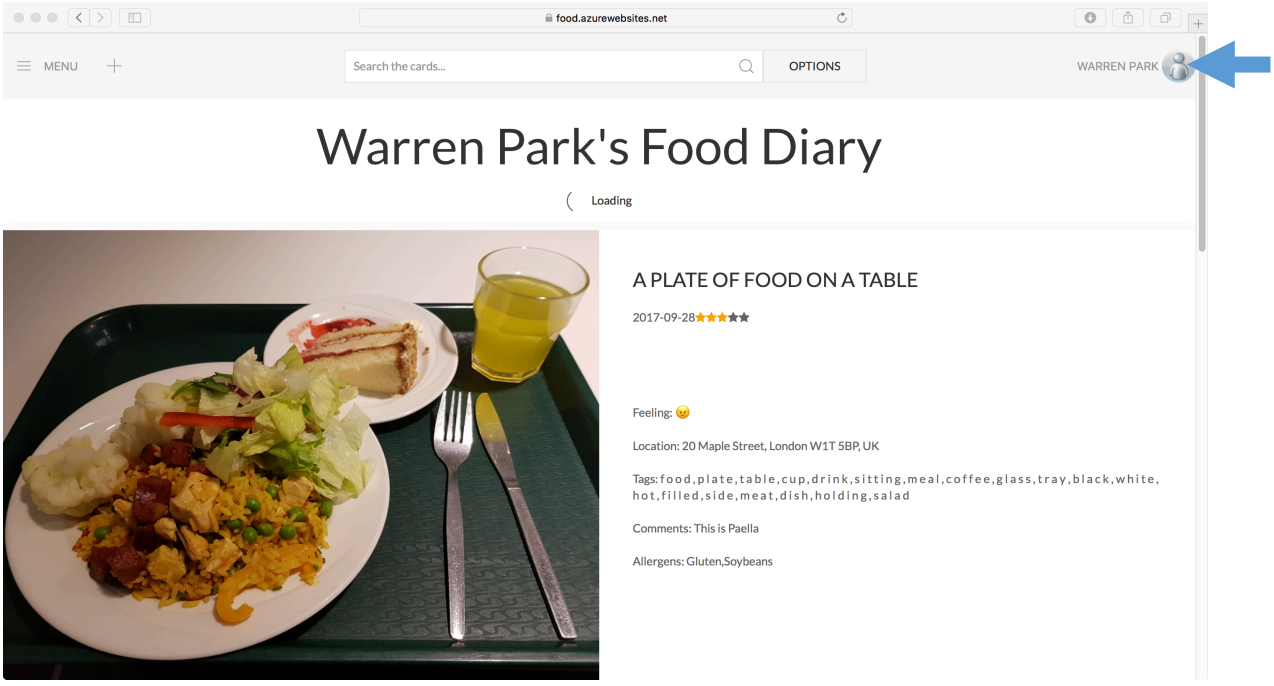


After clicking the button, you would be able to see the graph of your feeling trend.

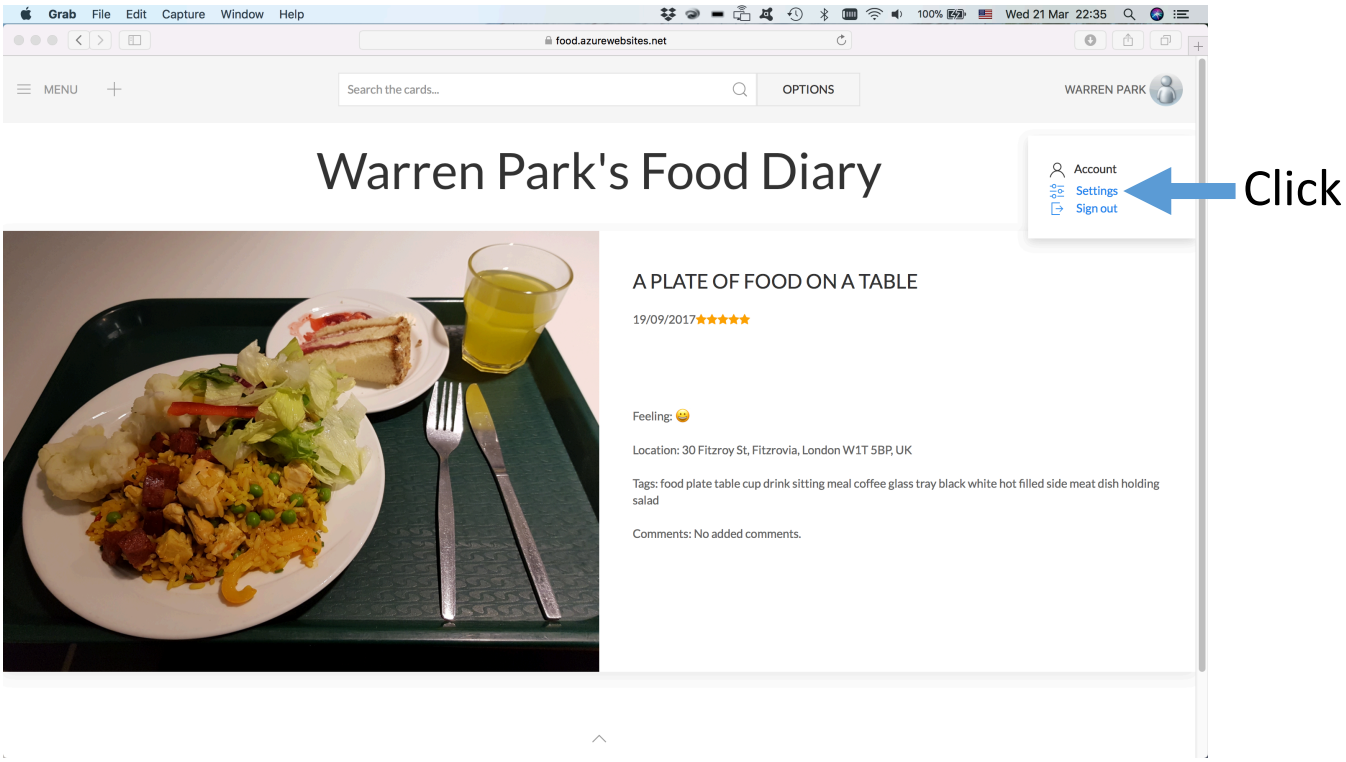


Set the default location

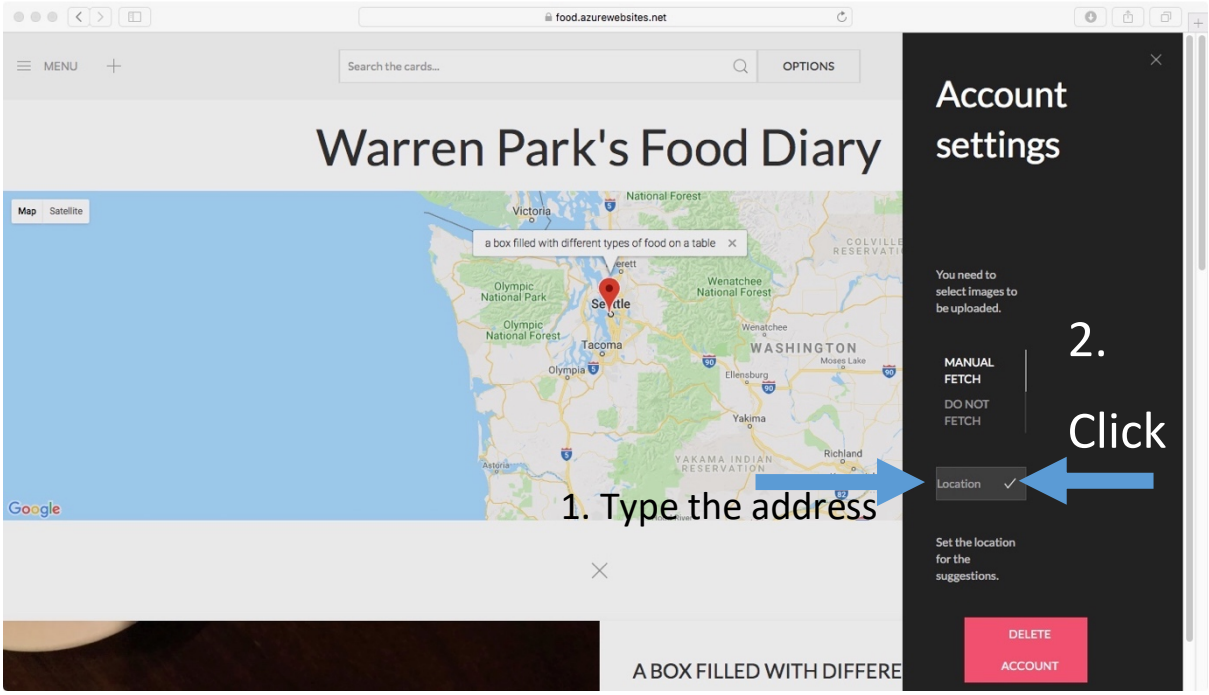
To set the default location, hover the mouse cursor on or touch your name at the **Account** area.



Then the drop-down menu will be shown. Click the “Settings” button.



After clicking the button, a sidebar will appear. Type the address that you want to set as the default for the suggestions and click the "✓" button.

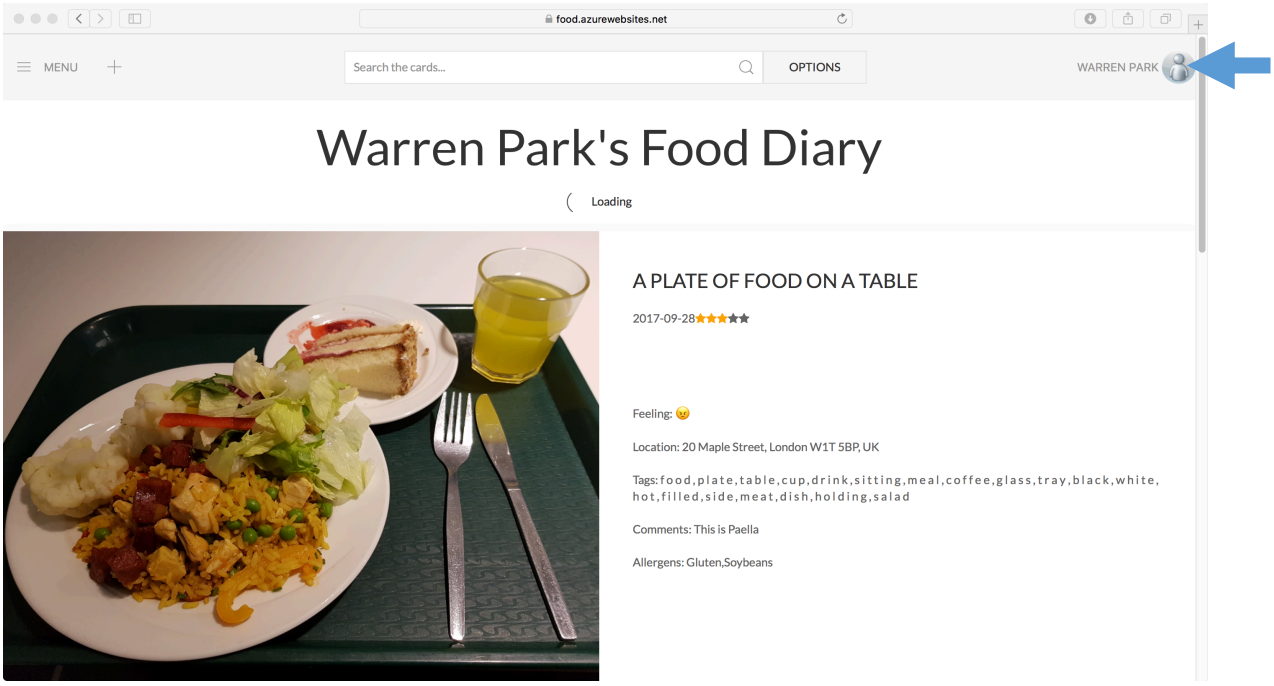


If you click the button, the default address would be changed.

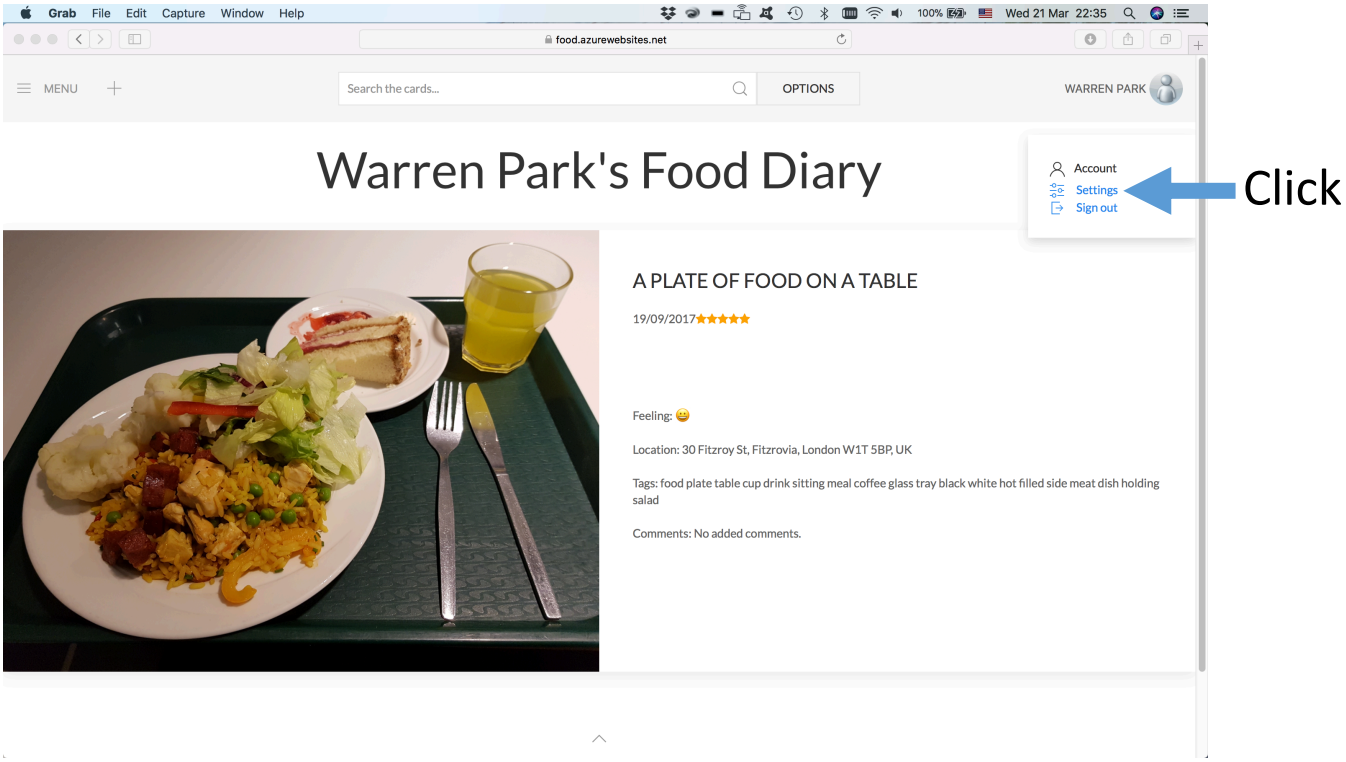
Change the application mode

There are two modes for the application. The first one is “Manual fetch” which means you can create cards, copy, delete, or edit the card as well as searching the card. However, the second one is “Do not fetch” which means you can search or copy the card, but you cannot create, delete or edit the card.

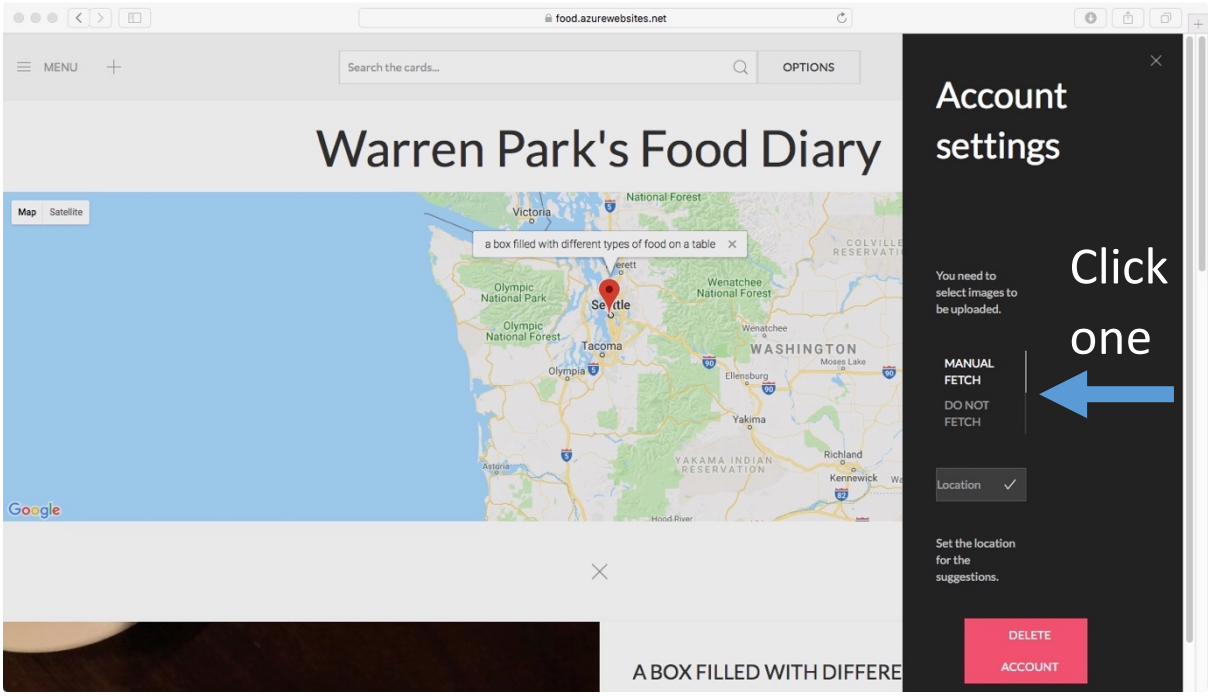
To set the application mode, hover the mouse cursor on or touch your name at the **Account** area.



Then the drop-down menu will be shown. Click the “Settings” button.



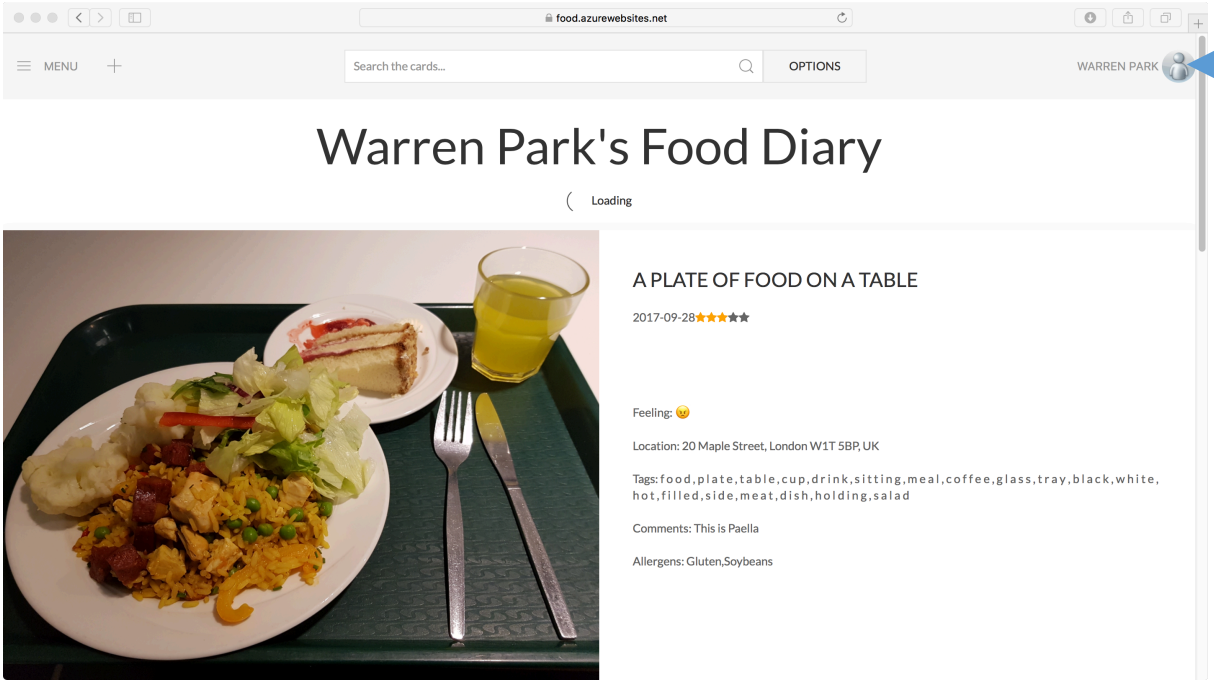
After clicking the button, a sidebar will appear. Click the mode that you want to set the application to. Then the application will set the mode.



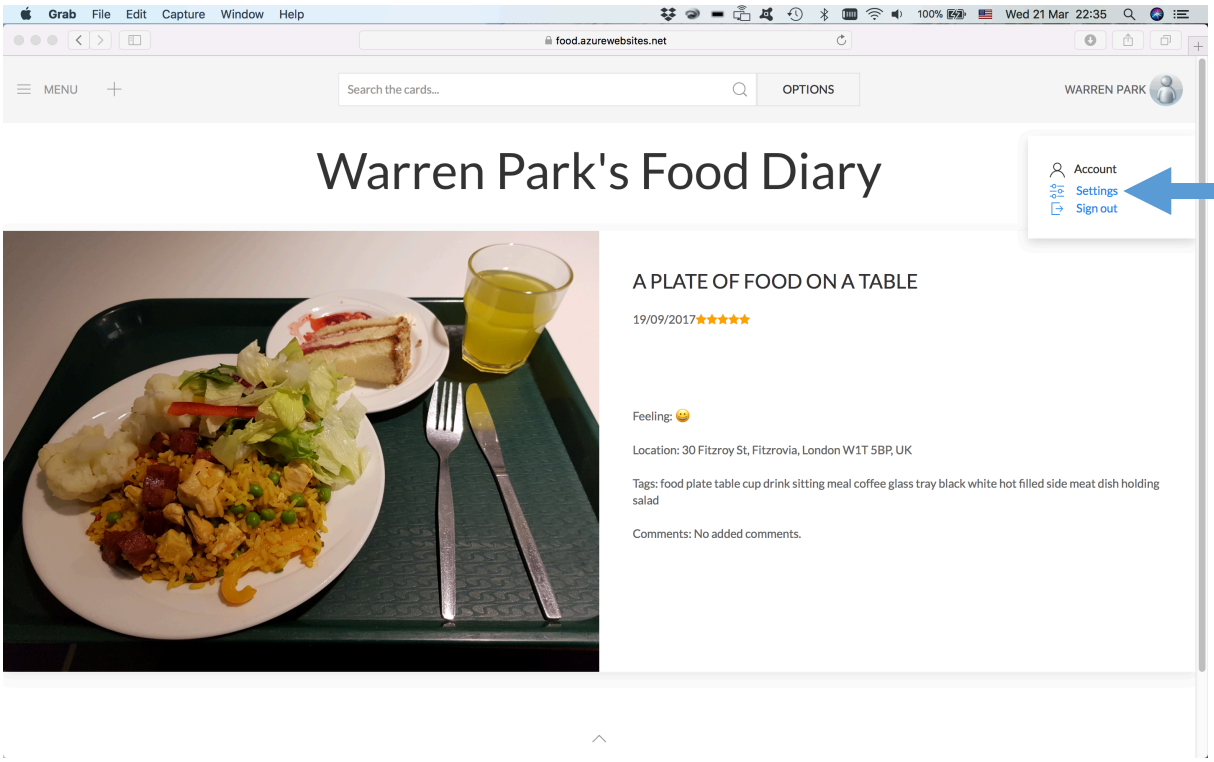
How to delete account?

⚠Deleting the account means that all of your data will be permanently lost.

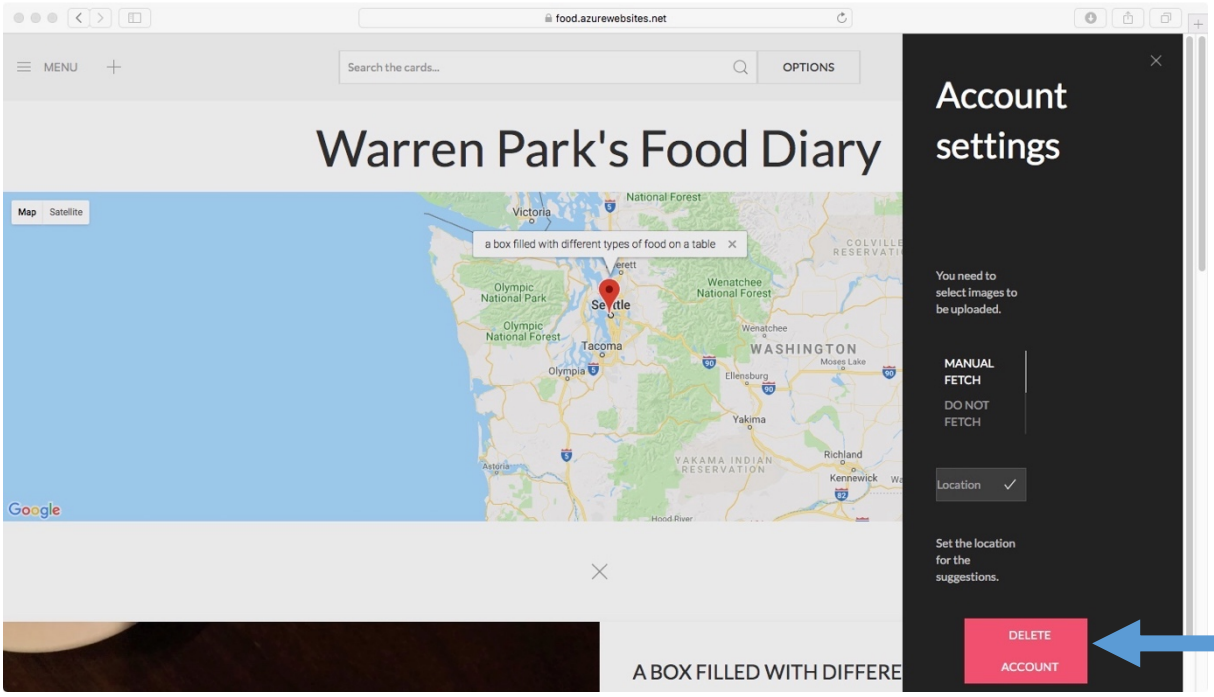
To delete the account, hover the mouse cursor on or touch your name at the **Account** area.



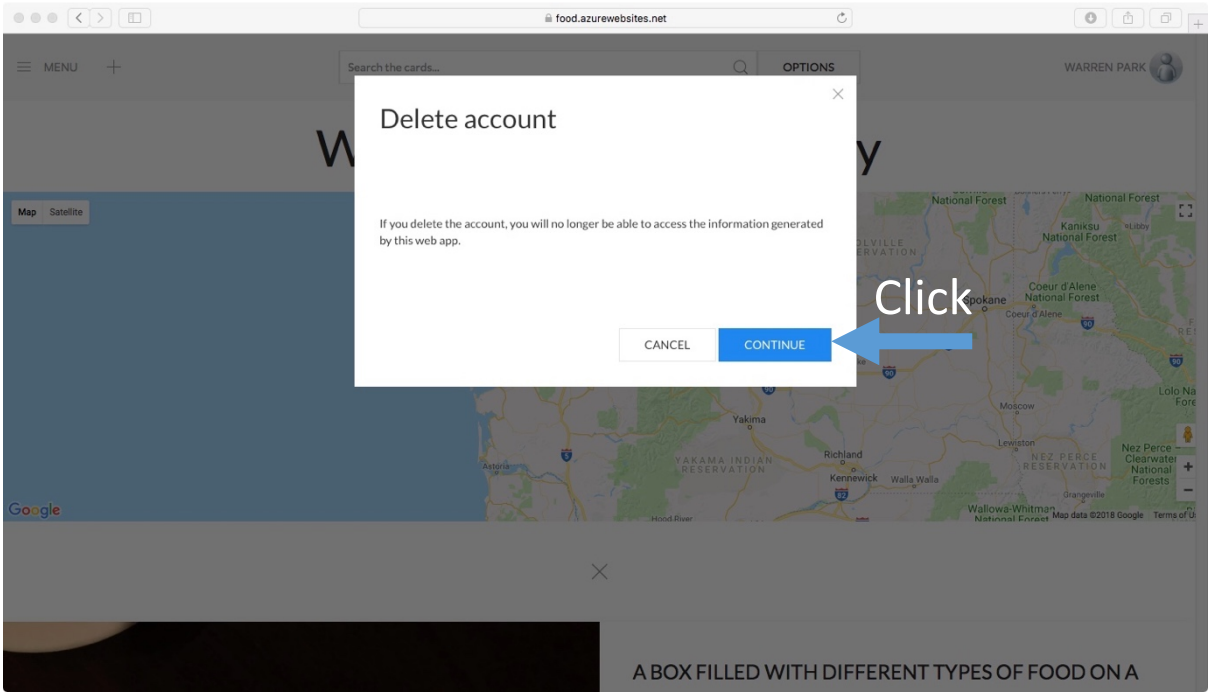
Then the drop-down menu will be shown. Click the “Settings” button.



After clicking the button, a sidebar will appear. Click “DELETE ACCOUNT” button.



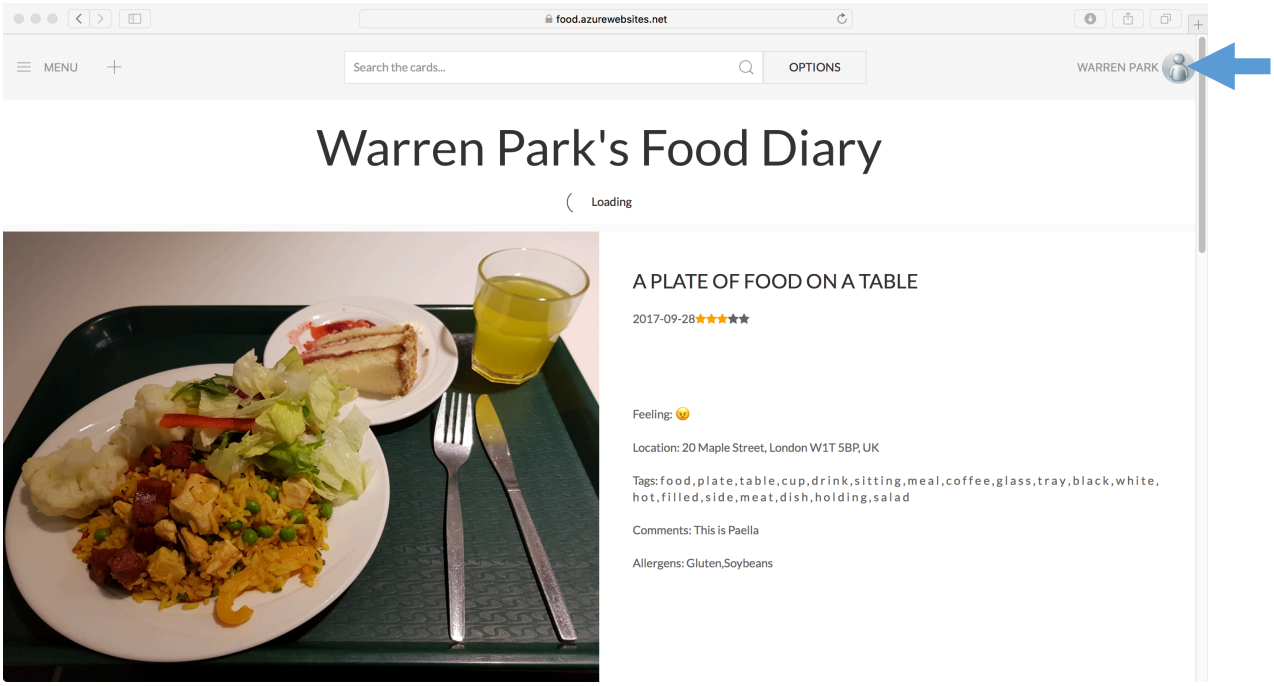
If you click the button, a window will be shown. Click “CONTINUE” to delete the account.



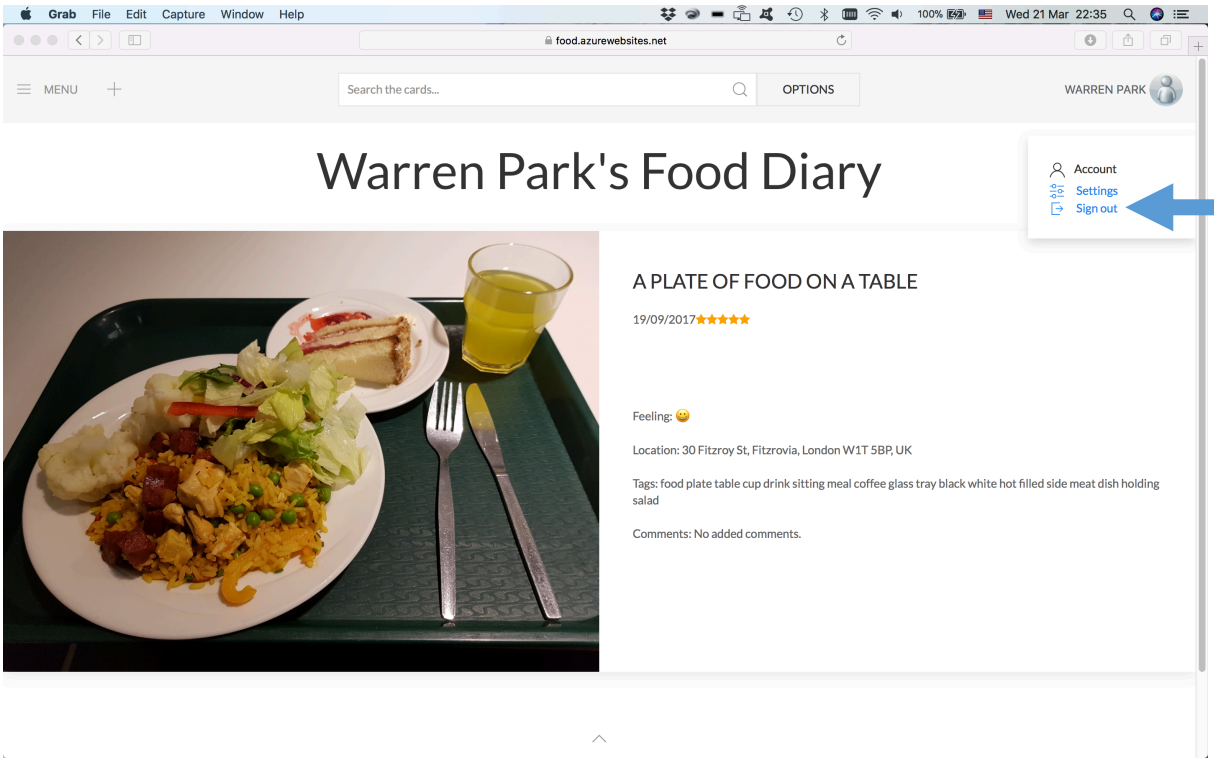
Then, your account would be permanently deleted.

Sign-out

To sign-out, hover the mouse cursor on or touch your name at the **Account** area.



Then the drop-down menu will be shown. Click the “Sign out” button.

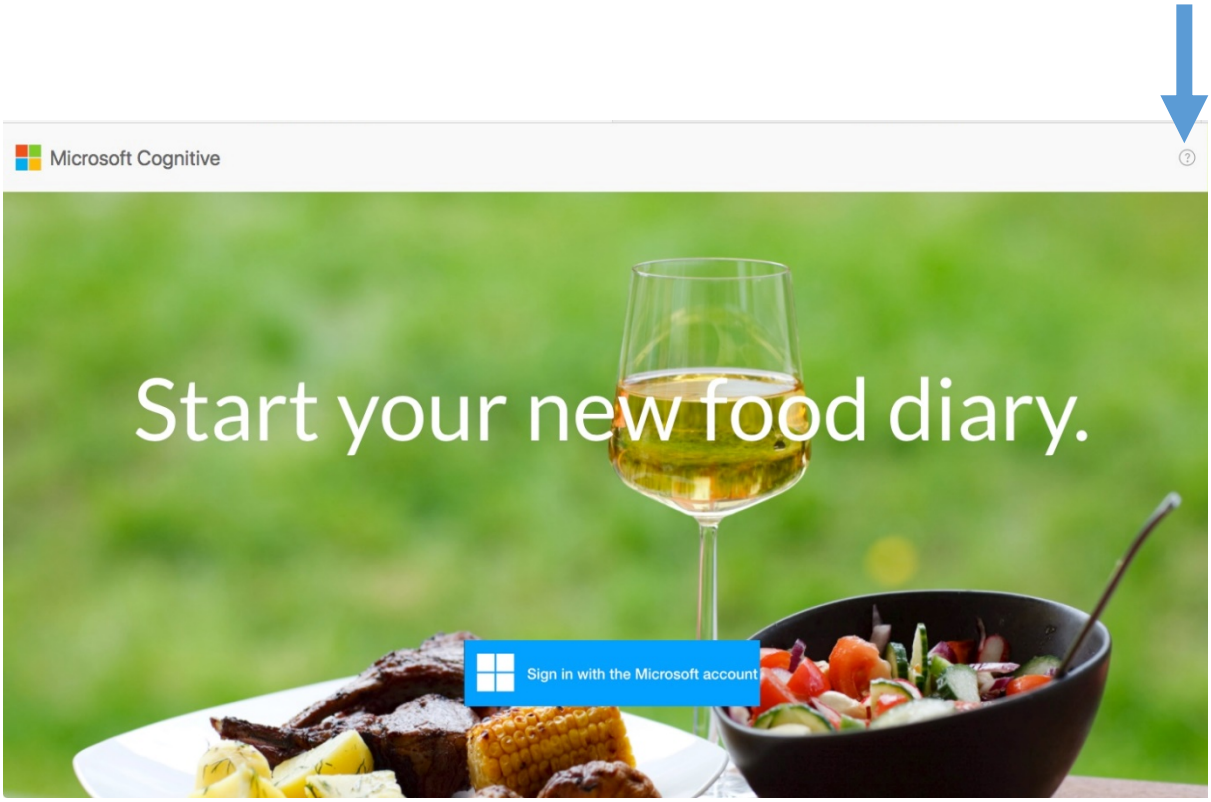


After clicking the button, you would be signed-out from the application.

If you have more questions…

If you have more questions about our web application, you can always ask your question on the artificial intelligence enabled FAQ page.

To access the page, please return to the home page of our application, and click “?” button.



Then FAQ page will be shown. On the search bar, type any questions that you want to ask and then press enter key on your keyboard or the search button. You can also find a relevant answer to your question from the questions list.

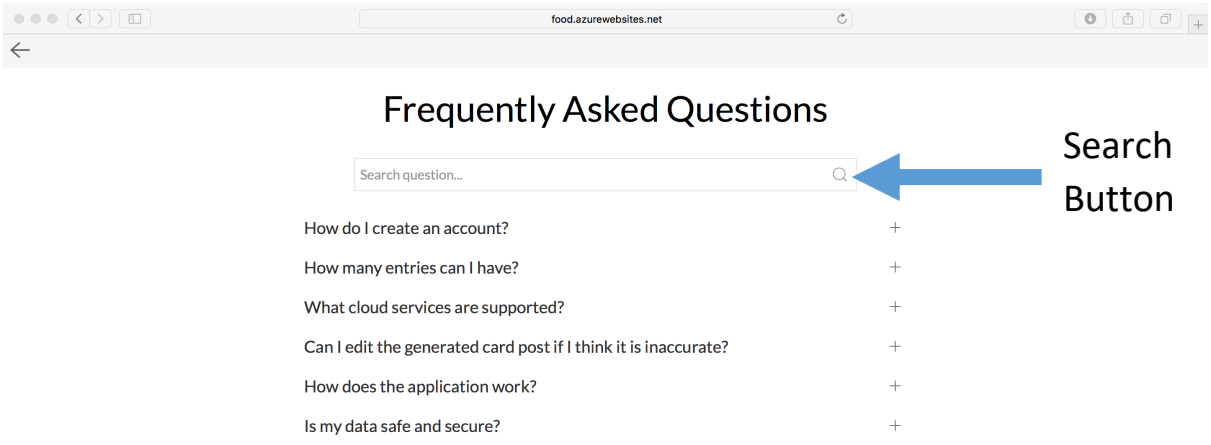


Image copyrights:

Deployment manual

*main.html is in client folder and app.js is in the server folder.

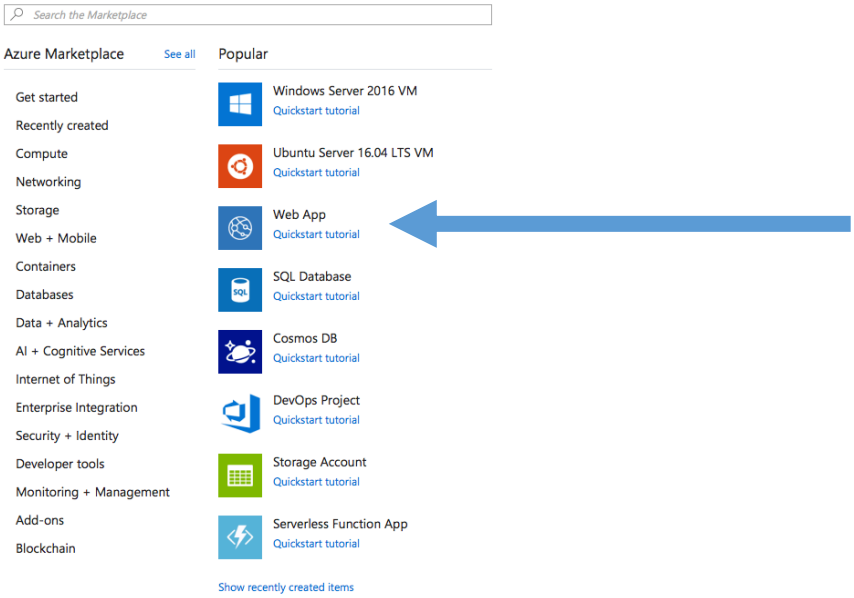
1. Create two separate GitHub repositories and copy the whole contents of the “client” folder of the data in one repository and contents of “server” folder on the other repository.
2. If you want to use your own API key for the Google Maps API, please replace <script async defer src="https://maps.googleapis.com/maps/api/js?key=AIzaSyCBUfeweOkyjviCYBNlYCmQ-yR-1jHkfVg&callback=initMap"> to <script async defer src="https://maps.googleapis.com/maps/api/js?key=(YOUR-KEY) &callback=initMap"> </script> </script> in main.html
   (YOUR-KEY needs to be replaced with the newly generated API key.)
3. If you want to use your API key for Microsoft sign-in, initialise variable “CLIENT_ID” in client/index/signin.html to newly generated key and redirect Uri which would need to be initialised with (YOUR WEB APP URL)/index/signin.html .
4. If you want to use your own API keys for cloud file pickers, get the keys and then initialise following variables with your keys:

   - For Dropbox, Line 33 of the main.html variable “data-app-key”

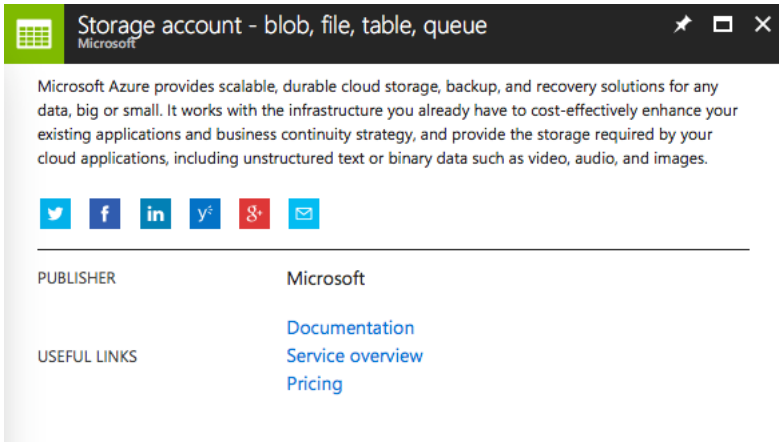
   - For OneDrive, change the “clientId” of the “option” parameter, in the function “launchOneDrivePicker”.

   - For Google Drive, initialise the variables “developerKey”, “clientId”, and “appId”.

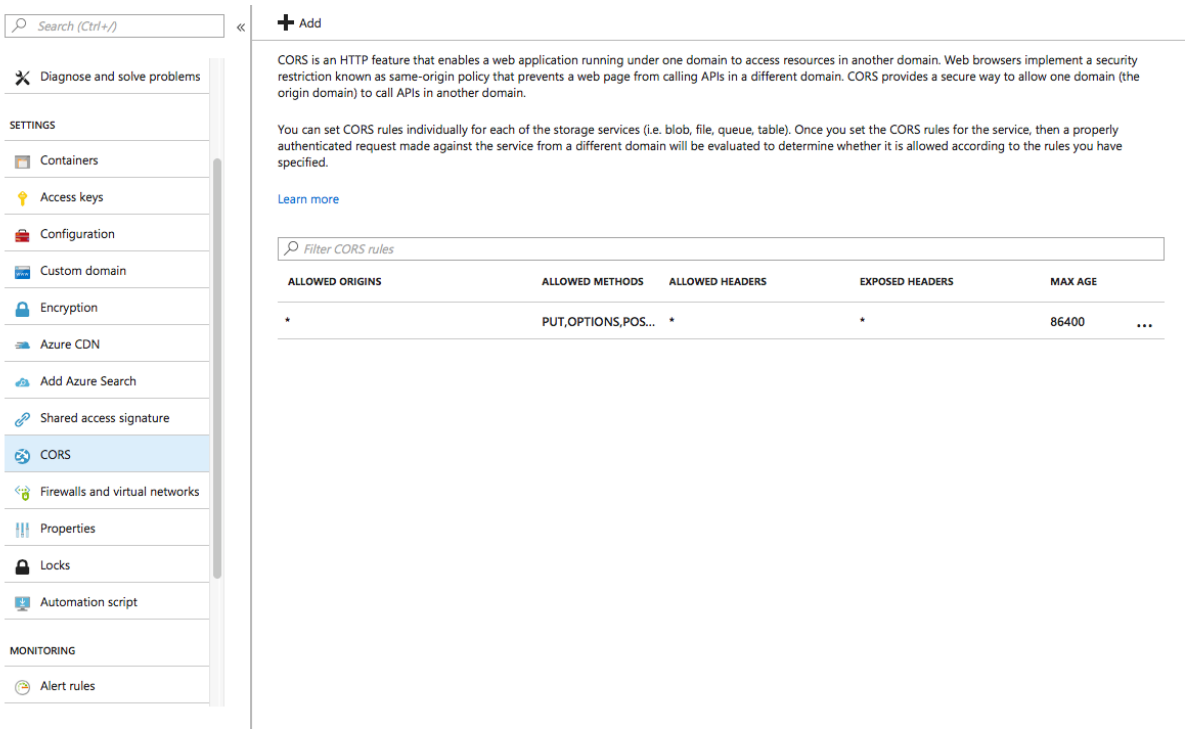
5. Create a Web App in Microsoft Azure:



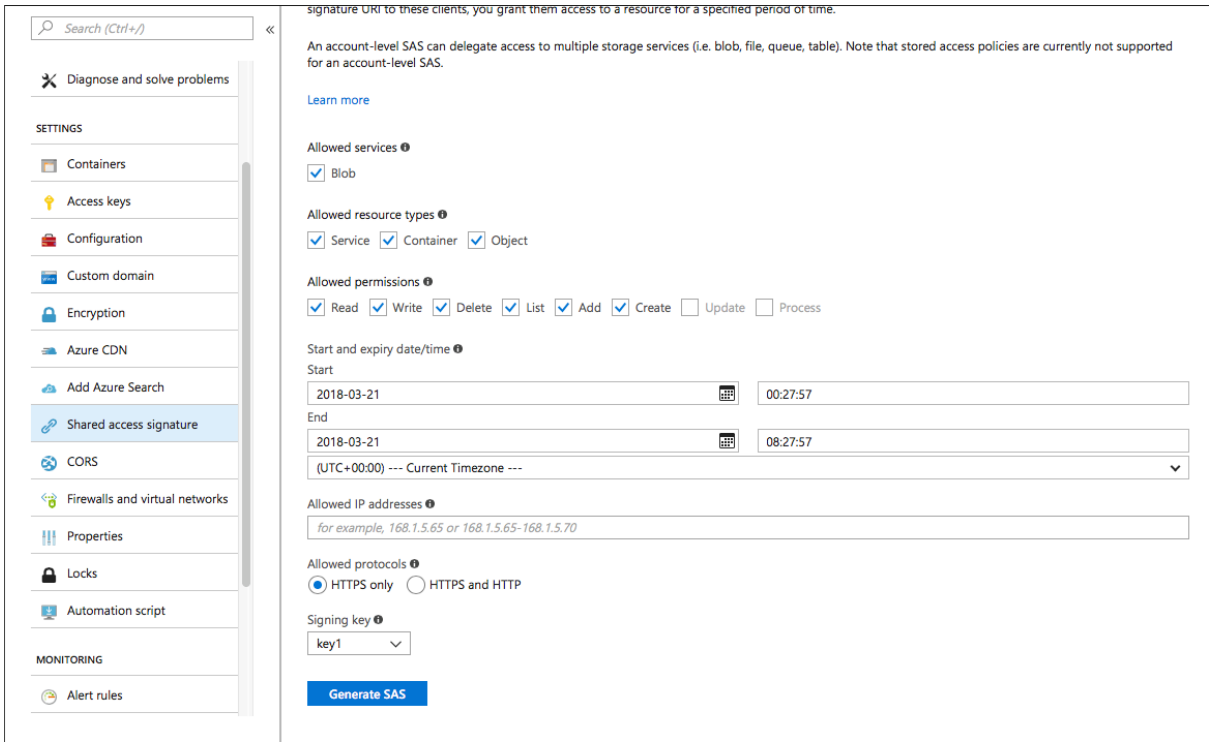
6. Create Azure Storage and configure as Blob Storage from account kind option:



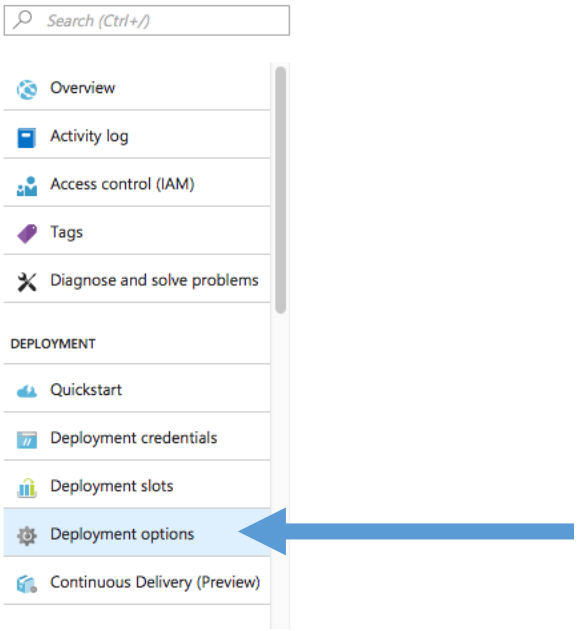
7. Set the CORS rule as below:



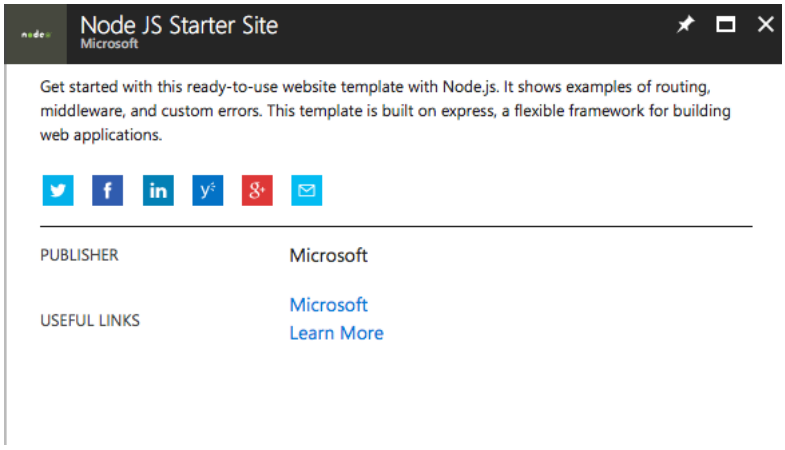
8. Generate SAS.



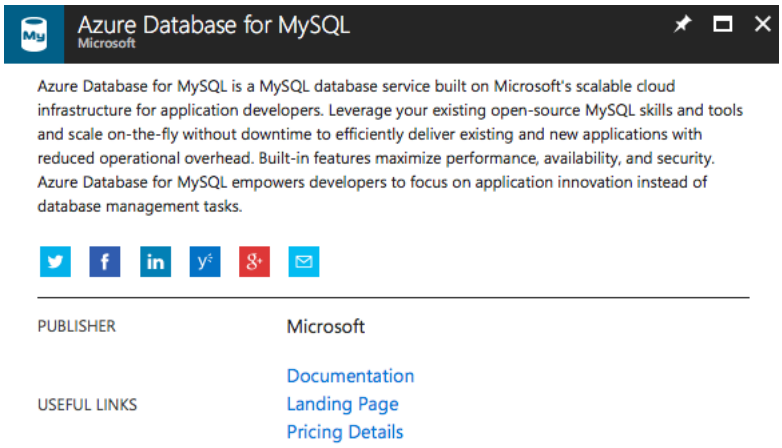
9. Initialise the variable “SAS_TOKEN” in the main.html with the newly generated SAS and also initialise the variable “blobUri” with the newly generated Blob storage Uri.
10. Return to the “Web App” service that has been created, and set the “Deployment options” to the Github and configure the deployment option with the Github repository that includes “client” folder contents.



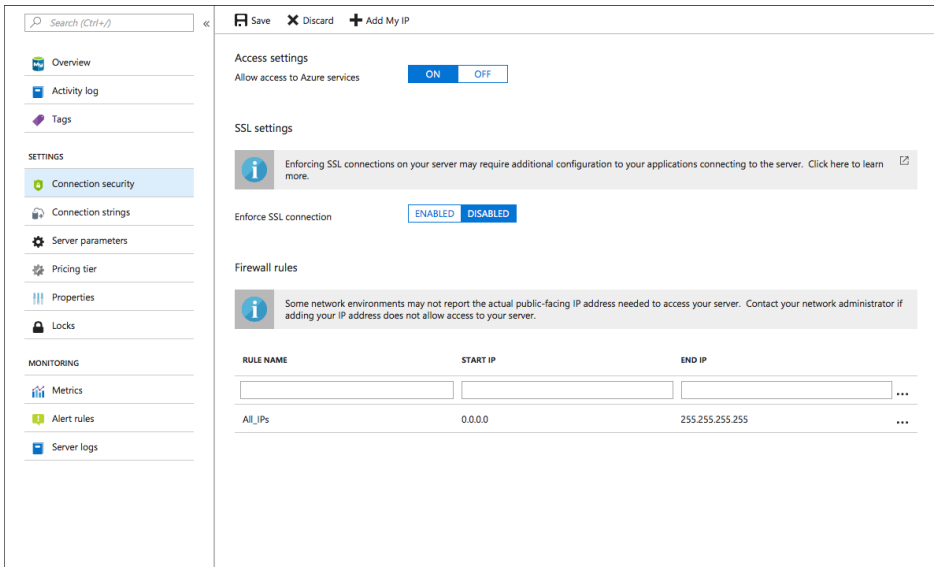
11. Set up another web app called “Node JS Starter Site”.



12. Again for this web app, set up the deployment options, but this time, with the Github repository that includes “server” folder contents.
13. Initialise the variable named “SERVER_URL” with the newly created Node.js web app URL.
14. Create MySQL DB.



15. Configure the connection security of the database as below:



16. On the app.js file of the server folder, initialise variable “connection” with the correct connection string i.e. replace host, user, password and database name as configured on the MySQL DB.
17. Configure MySQL tables as shown in the ER diagram (database name should be food).
18. Initialise the variable blobSvc in line 24 with azure.createBlobService(“YOUR-AZURE-STORAGE-KEY”); (app.js)
19. If you want to use your Yelp API key, please initialise the variable “yelp_client” to yelp.client(“YOUR API KEY”); (app.js)
20. If you want to use your Google geocoding API key, please replace “apiKey” value of the variable “options” with your API key. (app.js)
21. If you want to use your API key for the Microsoft Cognitive Vision API, please initialise variables “subscriptionKey” and “uriBase” of the “azure_image” function with the newly generated key. (app.js)

22. If you want to use your API key for the Microsoft Emotion API, please initialise variables “subscriptionKey” and “uriBase” of the “azure_emotion” function with the newly generated key. (app.js)
23. Initialise QnA Maker key and base Uri from client/index/qna.js post function.
24. git push the changed files
25. Then web app would be deployed.

*In some cases, you might need to npm install azure-storage manually in the console of the server web app. Execute the shell script npm_install.sh to install all of them.

Code citation

No.	Function Name	Code File Name	Source
1	*	client/uikit-3	This folder is adapted from https://getuikit.com
2	*	client/index/msal	This folder is adapted from https://github.com/AzureAD/microsoft-authentication-library-for-js
3	callGraphApi	client/index/signin.html	This function is adapted from https://github.com/AzureAD/microsoft-authentication-library-for-js
4	showError	client/index/signin.html	This function is adapted from https://github.com/AzureAD/microsoft-authentication-library-for-js
5	callWebApiWithToken and index.js	client/index/signin.html and client/index/index.js	This function is adapted from https://github.com/AzureAD/microsoft-authentication-library-for-js
6	initMap	client/main.html	This function is partially adaped from https://developers.google.com/maps/documentation/javascript/tutorial
7	launchOneDrivePicker	client/main.html	This function is partially adapted from https://docs.microsoft.com/en-

			us/onedrive/developer/controls/file-pickers/js-v72/open-file
8	onApiLoad	client/main.html	This function is partially adapted from https://developers.google.com/picker/docs/#gdata
9	onAuthApiLoad	client/main.html	This function is partially adapted from https://developers.google.com/picker/docs/#gdata
10	onPickerApiLoad	client/main.html	This function is partially adapted from https://developers.google.com/picker/docs/#gdata
11	createPicker	client/main.html	This function is partially adapted from https://developers.google.com/picker/docs/#gdata
12	handleAuthResult	client/main.html	This function is partially adapted from https://developers.google.com/picker/docs/#gdata
13	pickerCallback	client/main.html	This function is partially adapted from https://developers.google.com/picker/docs/#gdata
14	*	server/node_modules/azure-storage	This folder is adapted from (npm module) https://www.npmjs.com/package/azure-storage
15	*	server/node_modules/microsoft-computer-vision	This folder is adapted from (npm module) https://www.npmjs.com/package/microsoft-computer-vision
16	*	server/node_modules/cognitive-services	This folder is adapted from (npm module) https://www.npmjs.com/package/cognitive-services
17	*	server/node_modules/google-drive	This folder is adapted from (npm module) https://www.npmjs.com/package/google-drive
18	*	server/node_modules/mysql2	This folder is adapted from (npm module) https://www.npmjs.com/package/mysql2

19	*	server/node_modules/exif	This folder is adapted from (npm module) https://www.npmjs.com/package/exif
20	*	server/node_modules/node-geocoder	This folder is adapted from (npm module) https://www.npmjs.com/package/node-geocoder
21	*	server/node_modules/yelp-fusion	This folder is adapted from (npm module) https://www.npmjs.com/package/yelp-fusion
21	*	server_with_sharp/node_modules/sharp	This folder is adapted from (npm module) https://www.npmjs.com/package/sharp
22	*	client/index/qna.js and client/qna_maker	Those files and folders are based on the web site instructions: https://qnamaker.ai/Documentation/ApiReference
23	*	server/node_modules/resizer-stream	This folder is adapted from (npm module) https://www.npmjs.com/package/resizer-stream
24	*	server/node_modules/jpg-stream	This folder is adapted from (npm module) https://www.npmjs.com/package/jpg-stream

*server folder and server_with_sharp folder are the same except the fact that server_with_sharp includes the web app source code that uses sharp.