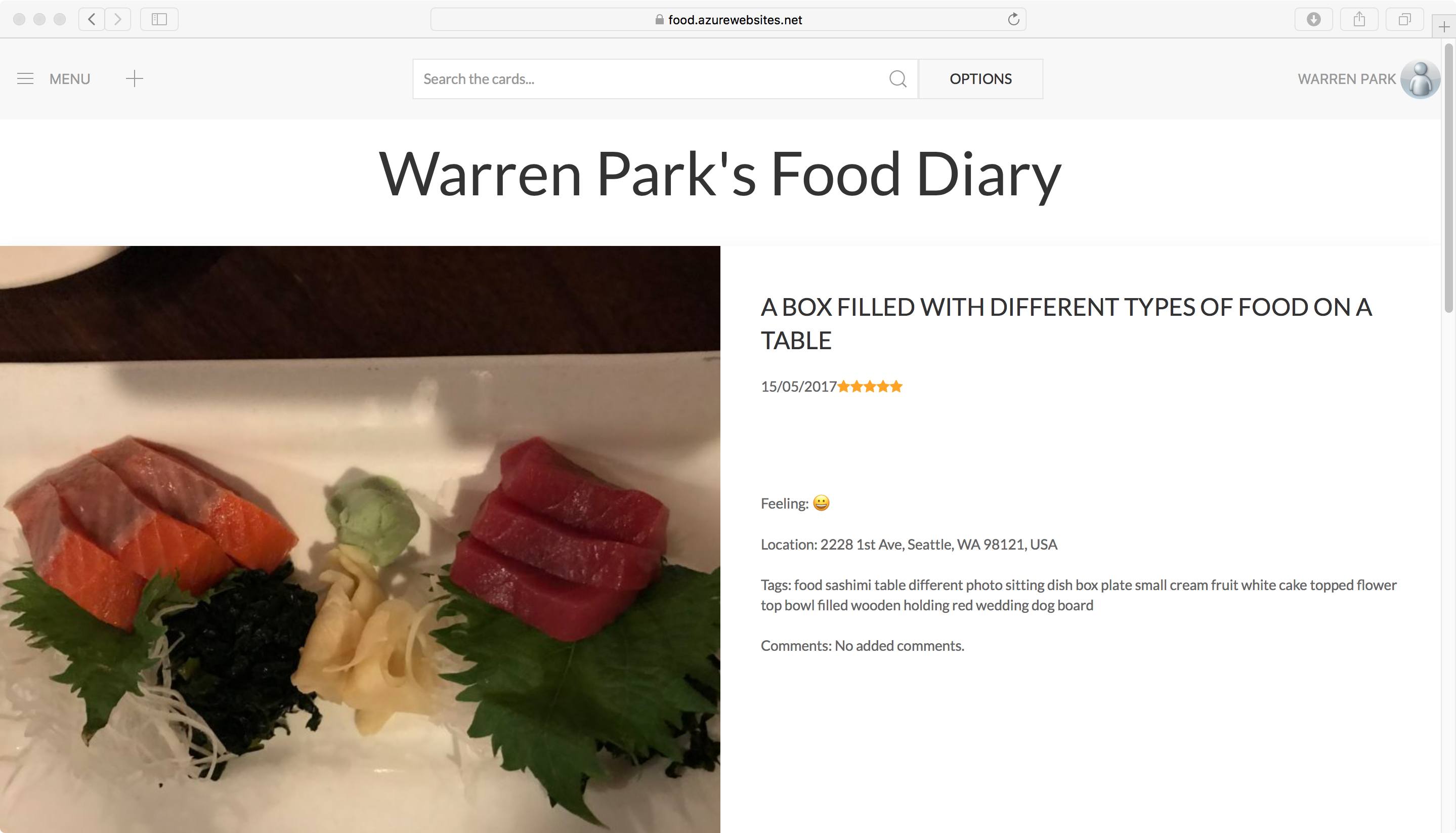
Food Diary web app powered by Microsoft Cognitive services

Warren Park March 28, 2018

I am a first-year student at UCL studying computer science. As part of the course, I was being assigned to a team and had to develop an application for Microsoft.

The title of the project was “Microsoft Cognitive Node.js app with cloud storage support”, and I was the team leader. After having a meeting with the client (Anze Vodovnik from Microsoft), me and my team members have decided to make a food diary web app for people who need to record the food that they have eaten on a day due to their allergies, or for people who are interested in recording the foods that they have eaten daily.

My team has got many ideas for the application, and decided to develop a web app that retrieves a food image and a face image pair from cloud storage providers (OneDrive, Google Drive and Dropbox), analyses the image by using Microsoft Vision API (for generating tags and food title) and Face API (for determining the feeling in the face to determine the degree of satisfaction about the food), and then shows the analysis result on the web app in the form of card like below:

1 An example card generated by the web app.

My team had three members including me, and each person had different roles.

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

Since I was responsible for dealing with the Vision API and Face API, I firstly started from reading documentation from the Microsoft website.

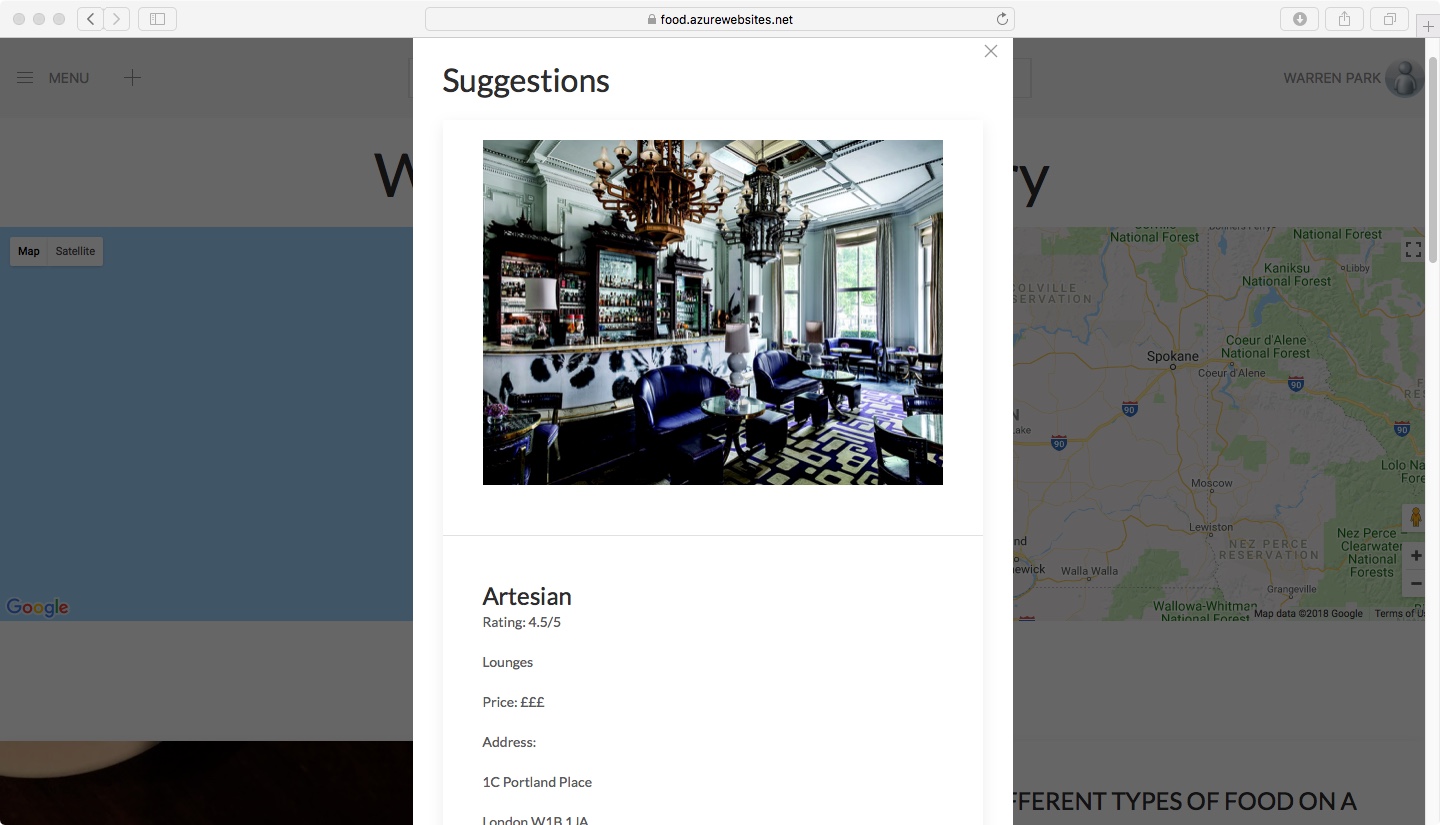
I have read the documentation and have found that the Microsoft Cognitive services are very easy to use since most of them happen in a simple HTTP request. Sending images using octet stream or URL was implemented quickly in the web app.

Then, I have implemented all of the Microsoft Cognitive services needed for the application and found that if I use an npm library called “cognitive-services”(https://www.npmjs.com/package/cognitive-services), implementation can be done even more concisely. The npm library can handle most of the Microsoft Cognitive services and can handle various functionalities of each API separately. I have decided to use this library instead of pure HTTP request and was able to develop the app quicker.

When most of the backend development had been finalised, I had to find a storage provider that can handle images securely and easily. As the staff from Microsoft has recommended, I have tried to utilise Azure storage especially the Blob storage.

Using Blob storage, I was able to upload image files directly without conversions to the storage and also was able to directly download image files from the storage. Since there were no conversion steps were required, it was easier for me to handle images. Microsoft had provided an npm module called “azure-storage”( https://www.npmjs.com/package/azure-storage) to enable developers to use Azure storage services with predefined upload and download functions, so there was no need for me to write separate functions that handle uploading and downloading.

Finally, me and my team members were able to develop a web app that can create a card that includes a single food diary entry with the analysis results from the Microsoft Cognitive services, search through the cards, suggest a restaurant based on the accumulated food eating history and preset default location, and show feeling trends over time.



2 Example restaurant suggestion generated by the application. Suggestion uses Yelp Fusion API.

When the deployable version of the web app has been developed, my team needed a web hosting service that can handle our web application. I have found that Azure provides effective Node.js web hosting service, and I have tried to deploy the web app by using the Github repository.

Using Github repository, Azure web app service automatically set up the application that I did not have to “npm install” from the console. It was able to show “Failed” if there was something wrong with the deployment as well as the log generated by the system, which was helpful for debugging the web app.

The overall architecture of the system had 6 components which are illustrated below:

CLIENT

HTML5

CSS3

JavaScript

JQuery

3 System Architecture Diagram

Main

SERVER

Node.js

Jade

npm modules

OTHER SERVICE PROVIDERS

* Microsoft Azure
* Microsoft Cognitive
* Download server of cloud storages
* Microsoft Graph API server
* Google Maps

Cloud Service Providers

HTTP

HTTP

MySQL

DB

HTTP/HTTPS

Download

Server

HTTP

Firstly, the download 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.

Secondly, 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.

Thirdly, 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.

Fourthly, 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.

Fifthly, 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.

Finally, other service provider 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.

Having the architecture described above, the system was able to provide a framework for the other application developers while ensuring the functionality of the web app.

Overall, this project was a fascinating development experience for my team. I think this development would not have been possible if Azure or Cognitive services was not existing. Thus, I highly recommend Azure services to any person who wants to develop AI enabled services or services that handle images.

\*The web app developed is available on https://food.azurewebsites.net

\*Public Github repository including all the source codes and report is available on https://github.com/Warren-Park/UCL-Cognitive-TEAM45-\_Food\_Diary-