Project Report (2nd Sprint)

The online (and easier to read) version of this report is available at http://wiki.nus.edu.sg/x/5ZxRBg.

Table of Contents

The Product Prototype

http://hungrygothere.appspot.com

Vision, Motivation & Product Description

Our project is called *Hungry Go There*.

Our Vision is to bring intelligent and comprehensive "Makan Places" searching to users' fingertips.

Our product is a mobile application that allows the user to find all the well-known/good makan (eating) places in Singapore. But then, you might say, what's new about it? Don't we already have well-established services for such things? Well, our app not only allows the user to find a makan place near some particular point or some particular area, it also allows the user to find makan places along a particular route. So for example, say Louis (a foodie perhaps) needs to travel from his home in Jurong to the airport in Changi, to take a photo of the 787 with his new D7100. So instead of only be able to find a well-known makan place in Jurong and Changi, he's able to use our app to find one along his route. So he would be presented options at Queenstown, Tanjong Pagar, or Bedok (needless to say, but saying anyway, all of which along the way), and, armed with this information, Louis is able take a short detour to and visit any one of the restaurants listed, and hence enjoy a slice of the food paradise Singapore is said to be.

Perhaps watching our project videos might give you a better idea:

- Project Video 1: http://youtu.be/JowbGzuP8Xc (1st sprint)
- Project Video 2: http://youtu.be/-ueixcpMuIM (2nd sprint) ** Please read the video description before watching! **

Features

The following is a list of features for our project:

- Where Can Eat (1st sprint)
- Halfway Eat Where (1st sprint)
- Estimated Time (1st sprint)
- Yummy Places, aka Favourites (2nd sprint)
- Restaurant History (2nd sprint)

Target Audience (Stakeholders)

We are targeting all of the hungry Singaporeans out there, especially those busy foodies who may be interested in finding a nice makan place as they travel between places.

User Stories

Just some user stories to give you an idea who would use our product.

#1 - Where Can Eat (1st sprint)
It's lunchtime, and John asks his colleagues if they want to eat. However, they are very bored of the canteen food, and don't know where they can lunch at! John takes out his phone and uses Hungry Go There!

#2 - Estimated time (1st sprint)
So John is using Hungry Go There and his friends are suspicious, because they only have 1 hour for lunch, and using other apps, they overshot their lunch time! John assures them that Hungry Go There displays estimated queuing and waiting times so they can choose a nice makan place, and be back in time for work!

#3 - Halfway Eat Where (1st sprint)
John is leaving his workplace already and he calls his wife, Jane, who doesn't know what she wants to eat. She is bored of the food around their home, and does not want the food from John's office area because it would be cold by the time he's home! John uses Hungry Go There to find out all the different places he can stop buy to get some food along the way home!

#4 - Favourite Makan Places (2nd sprint)
After seeing how good the app is, John's colleagues have also installed it! However, they find it very stupid and troublesome because everytime they want to go somewhere, and want to check details about it, they have to do a search all over again. John tells them they can actually add restaurants to their favourites, where they will be able to access them easily!

#5 - Restaurant History (2nd sprint)
Lastly, John's colleague was browsing the app previously, and he came across this french restaurant which he dismissed. Coincidentally, his wife is craving for french food for dinner. Unfortunately, he has forgotten where he saw the french restaurant! John tells him to look at his restaurant history. There, he finds the restaurant and has a great dinner with his wife!

Competitors

- Hungry Go Where

Problem Statement

We are going to create a mobile application that allows one to find all the well-known/good eating places either near an arbitrary location, or along the way of a particular route.

Input Requirements

2nd Sprint
We have extracted restaurant data/ratings from Hungry Go *W*here, but limited our area to Kent Ridge and Clementi. For our 2nd sprint, we may continue to expand our database if we have the time (Hungry Go Where recently changed their website format, hence we have to rewrite our spider.) We will continue to use the Google Maps API, and possibly look into using the extended features provided by the API.

1st Sprint
Restaurant data/ratings will be extracted from a certain source (see Competitors). We don't plan to extract everything from our source; instead we will be focusing on just a particular area, most likely Kent Ridge/Clementi (even with the focus, we may not extract every restaurant in the area). Also we don't have our own map data, so we will be using the Google Maps API.

Design

The description of our product technical design has been updated for the 2nd sprint.

Tech Stack (and Justification)

Front-end (User Interface)	Bootstrap, HTML, CSS	We will be using bootstrap for majority of our user interface.
Front-end (Workhorses)	Google Maps, Javascript	We have a map feature, hence it will be largely powered by Google Maps. The remaining interactivity of our webpage will be written in Javascript.
In-Between	JQuery, AJAX, JSON	Our webpages communicates with the server mainly with AJAX (essentially it's like submitting a form without reloading the page); we will use JQuery's support for AJAX to do this. The format of our data that the server will reply in will be in JSON.
Server-end	Google App Engine, Python	GAE is essentially the server, where we will use it to manage our webpages. We will use ~~Python's in-built support for SQLite to communicate with the database~~ the datastore API provided by GAE to communicate with the database.

Database	~~SQLite~~ GAE Datastore	Database used to store all the restaurant data. Google App Engine does not support SQLite, but offers instead a datastore (essentially a database) that we can (reluctantly) use.

Libraries / Plugins

- Google App Engine
- Google Maps API
- Bootstrap
- JQuery

Data Schema

Restaurant data stored in the GAE datastore and communicated to the front-end via JSON.

The current from of our objects in the GAE datastore is as follows:

Object Name: Restaurant			
Name of Property	**Corresponding name in Javascript/JSON**	**Type**	**Description**
uid	-	int	Unique ID for each of the restaurants. Used only for development purposes, to easily identify restaurants. Our app (in 'production' environment) makes use of the **id** auto-generated by the GAE datastore to identify each individual restaurant instead (see next row)
id (auto-generated)	id	int	This id is auto-generated by the GAE datastore. We use this property to identify each restrauant.
title	title	string	Name/Title of the restaurant/makan place
address	address	string	
contact	contact	string	Contact details
rating_overall	*rating*	int	
rating_food	rating_food	int	
rating_ambience	rating_ambience	int	

rating_value	rating_value	int	
rating_service	rating_service	int	
review_count	review_count	int	Number of reviews that gave the above average ratings
latitude	latitude	float	
longitude	longitude	float	
waitingtime_queuing	waitingtime_queuing	int	
waitingtime_serving	waitingtime_serving	int	
url	url	string	The URL of the restaurant's HungryGoWhere.com page (where we extracted the information from).

Object Name: HgtUser		
Name of Property	**Type**	**Description**
user	User	(google.appengine.api.users.User object) This is a wrapper model/class, wrapping around a User object provided by the google.appengine.api.users package. Having this object allows us to save the user details provided by Google Accounts into our own datastore.

Object Name: RecentLink

This object represents a link between the user and a restaurant, where the user recently viewed the restaurant.

```
User       <---->     Recent link     <---->    Restaurant
(HgtUser object)        (RecentLink Object)       (Restaurant Object)
```

So a recent link simply points to one user and one restaurant. With multiple recent links pointing to the same user, we can generate a list of restaurants the user recently viewed.

Also there is a **timeAdded** property which is the date & time the RecentLink object was created. This lets us figure out when the link was created and sort a list of RecentLink objects accordingly.

Name of Property	Type	Description
user	ReferenceProperty / Key	(google.appengine.ext.db.ReferenceProperty / google.appengine.ext.db.Key) A reference to a particular user (HgtUser object).
restaurant	ReferenceProperty / Key	(google.appengine.ext.db.ReferenceProperty / google.appengine.ext.db.Key) A reference to a particular restaurant (Restaurant object).
timeAdded	datetime (Python object)	The time this object was created.

Object Name: FavouriteLink

This object represents a link between the user and a restaurant, where the user favourited a restaurant.

```
User       <---->     Favourite link     <---->    Restaurant
(HgtUser object)        (FavouriteLink Object)       (Restaurant Object)
```

So a favourite link simply points to one user and one restaurant. With multiple recent links pointing to the same user, we can generate the user's list of favourite restaurants.

Also there is a **timeAdded** property which is the date & time the FavouriteLink object was created. This lets us figure out when the link was created and sort a list of FavouriteLink objects accordingly.

Name of Property	Type	Description

user	ReferenceProperty / Key	(google.appengine.ext.db.ReferenceProperty / google.appengine.ext.db.Key) A reference to a particular user (HgtUser object).
restaurant	ReferenceProperty / Key	(google.appengine.ext.db.ReferenceProperty / google.appengine.ext.db.Key) A reference to a particular restaurant (Restaurant object).
timeAdded	datetime (Python object)	The time this object was created.

Roles / Responsibilities

Ian	Backend server stuff. Built the GAE backbone as well as the SQLite database. Also dealing with the communication between the front- and back-end (eg, query formatting). And some Google Maps.
JY	Frontend UI stuff. HTML, CSS, JQuery. Focus is on creating an intuitive interface for mobile users, but also at the same time usable on a normal desktop. Some Google Maps too.

Project Schedule/Plan

	Ian	**JY**
Week 1 (13 May)		
Week 2 (20 May)		
Week 3 (27 May)	Set up Google App Engine, database. Also will extract restaurant data and format it to fit our database.	Decide of the structure of the website, i.e. how many webpages we have (e.g. mainpage.html, inputdetails.html, mapview.html, errorpage.html) or all-in-one approach (divided in DIV tags or something).
Week 4 (3 Jun)	Identify the major user queries (e.g. user sends a longitude-latitude coordinate looking for restaurants in the area) and write the barebones python scripts to handle the queries (and send back the required information).	Implement and come out with a skeleton UI which has functional form input areas, and make sure it looks good.

Week 5 (10 Jun)	Fine tuning of the user queries. Hopefully by now the server-end of things will be more or less OK and workable.	Testing on Mobile Phones, and implementing an Android App if possible.
Week 6 (17 Jun)	Start looking into building a decently usable map based on Google Maps.	More Testing and integration with IAN's part
Week 7 (24 Jun)	More map building. And also fine tuning any stray stuff. Some kind of buffer too for any problem that may crop up.	Buffer time for any problems.
Week 8 (1 Jul)	**Submission Week**	**Submission Week**
Week 9 (8 Jul)	Introduce user login feature. Also create user account DB, and DB for favourites + history.	Add in UI for User Login, Favourites Page and Restaurant History page. Also add in "add to favourites" icon on restaurant details page.
Week 10 (15 Jul)	"Add and Remove" functions for favourites and history (adding is the same just trigger different).	Integrate with IAN by adding functions.
Week 11 (22 Jul)	Clean up for final product.	Try to make it into an android app, using WebView.
Week 12 (29 Jul)	**Submission Week**	**Submission Week**

Testing

We plan to do some user testing by asking our friends for help to test the actual app.

Project Log

See Journal.

About Us

Our team is called Earful, and our team members are (in no particular order, other than alphabetical):

- Ian Leow
- Koh Jun Yang

We are aiming for the Intermediate level.