

# Let us Eat Restaurants Recommendation for Parties

Mentor:  
Taku Takamatsu  
Team member:  
Moxin Xu/mx2237 Xiyuan Zhao/xz2994  
Yuqin Zhao/yz4131 Yutong Chen/yc3993  
Columbia University

## 1 Introduction

This document serves as an example submission. It illustrates the format we expect authors to follow when submitting a paper to ECCV. At the same time, it gives details on various aspects of paper submission, including preservation of anonymity and how to deal with dual submissions, so we advise authors to read this document carefully.

## 2 Architecture

Our architecture shown in figure 1. Each user can store their information into our database using LF0, create and add group and group member using LF1 and get recommendation using LF2.

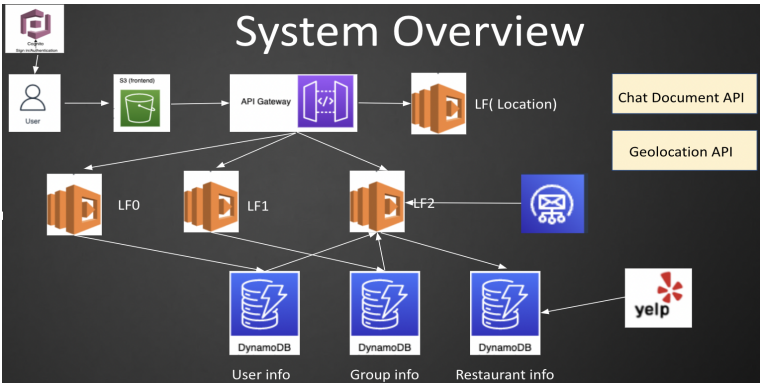


Fig. 1. Group Recommendation

### 3 API Design

1. /sign up  
Parameter: user information: email(unique), location, name, three preference  
Description: User sign up their information in the users DynamoDB table  
Methods supported: POST
2. /login  
Parameter: user's id, password  
Description: Authenticates of the user  
Methods supported: POST
3. /personal/search  
Parameter: user information: location, three preference, mode  
Description: User give back their information to lambda function and get their restaurant recommendation back.  
Methods supported: POST
4. /group  
Parameter: all the group member's information:  
Description: User create a group by all their team members' information and lambda function will send email to rest of the team. Also, a group id will give back to the front end.  
Methods supported: POST
5. /group/search  
Parameter: user information: location, three preference, mode  
Description: User enter their own information and give back their restaurant recommendation.  
Methods supported: POST

### 4 Project Details

#### 4.1 Restaurants Data Collection and Data Store

Restaurants data scraping is necessary and essential for any Restaurant Recommendation System. In our design, Data was collected using Yelp API. We scraped more than 7,000 restaurants' information of different cuisine located in Manhattan, New York. Such information including restaurant id, cuisine, geo-coordinates, rating and so on. All these data fields are essential to our recommendation algorithm, which will be covered in later section.

All restaurants data are stored in AWS Dynamo Database. Figure 1 shows the data model we used in restaurant table.

In order to query related data entry in the table, a secondary index on (type, longitude) was added. We are able to find a specific type of restaurants near a specific location by querying this type of restaurants near the desired location's longitude, then scan through the results to filter out unmatched latitude.

rid	longitude	latitude	rating	type
-----	-----------	----------	--------	------

Fig. 2. Restaurant Table

4.2 Registration

AWS Cognito is implemented to help user register. When a user gets the invitation link from his friend to join the group for dining decision making, he is redirected to log-in page.

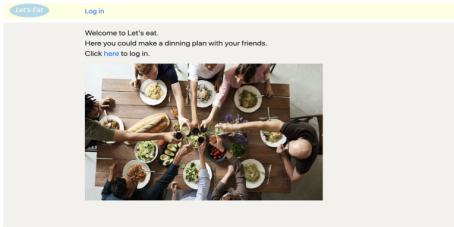


Fig. 3. Login Page

The user could click either of the blue link to the Cognito preset page. It's auto-generated in the control panel. I chose user email and password field needed in the page. And the email verification when signing up. The configuration could be modified by the requirements. The registered user could change their preferences in the sign up page for recommendation.

4.3 Group Creation

4.4 Recommendation Algorithm

Recommendation in our design is separated into 2 parts: personal and group. In personal recommendation, user's restaurants preferences, say 3, along with their dining locations are collected from front-end or queried from user's registration database(depend on whether user is using default information). Related restaurants, which are those whose types match user's choices and coordinates are near user's locations(latitude difference less or equal to 0.002, longitude difference less or equal to 0.002). After matched restaurants are retrieved from database, we sort all results by ratings or the Manhattan distance between user and restaurant, depending on what mode user is on, for example, rating first or

distance first.

Group recommendation is much more tricky. Different group members have different restaurant preferences and this makes recommendation hard to be implemented. To solve this problem, a voting algorithm is used to find the cuisines that most group members like. In this algorithm, different scores are assigned to different cuisines. If a cuisine is picked by a member as his/her first choice, then 3 points are added to this cuisine. If it is chosen as a member's second choice, then 2 points are added. If it is the third choice, only 1 points will be added to that cuisine. This process will eventually find 3 most popular cuisines.

In order to give recommendation to a group, we have to use appropriate locations as well. In other words, a reasonable search area is required. In our design, the search area for a group contains: every group members' locations along with the centroid of all members. In this way, all the restaurants that are recommended will not be too far away from each group members.

#### 4.5 External APIs

We have introduced two external API, geolocation and google chat to our project.

Geolocation API is used for auto-positioning. While filling in preferred locations, we can choose to type in the specific location or get current location from auto-positioning. For the auto-positioning function, we are calling geolocation.navigator to fetch current latitude and longitude, and then send it to backend as the input of prediction model.

Google chat API is used for creating chatroom for recommendation groups. In the chatroom, group member can share their preferences, allergies, favourite dishes, etc.

### 5 Conclusions

Compared to Yelp, our design lacks some functionalities when it comes to personal restaurants suggestions. Our design supports finding restaurants based on my location and preferences, but we cannot make or view others' comments regarding any restaurants. Besides, we cannot automatically navigate to a preferred restaurant. To achieve these, a database to store comments and Google Map API as a navigator is required.

However, our design solves the problem of providing group dining suggestions, which is not implemented by Yelp. Our design is capable of forming dining groups and give suggestions by considering every member's preferences and locations.


Let's Eat

♥ Make a dining plan with your friends ♥

PERSONAL SUGGESTION


GROUP SUGGESTION

LOG INSIGN UP




Amity Hall Uptown

\$\$ American (Traditional) Burgers Pubs




Lion's Head Tavern

\$\$ American (Traditional) Sports Bars



La Piccola Cucina

\$\$ Italian



Tartina

\$\$ Italian

Content

+16469302501

982 Amsterdam Ave

New York, NY 10025

+12128661030

995 Amsterdam Ave

New York, NY 10025

+12128661336

964 Amsterdam Ave

New York, NY 10025

+16465900577

1034 Amsterdam Ave

New York, NY 10025

Fig. 4. Group Recommendation

6 Annex

Project Link: <https://ui-hw13.s3.amazonaws.com/index.html>  
Github Link: <https://github.com/yc3993/Let-s-eat>