

FoodJunkies

A Food Recommendation Application: From Design to Implementation

Kah Hau Yap, Ivelisse Morales, Chuk Ho Wu, Szymon Późniewski, Gurjot Singh Nijjar

Platforms: Android and iOS
Programming Languages: Swift, Java, PHP

1. INTRODUCTION

The FoodJunkies application has begun its life as an idea meant to help indecisive, picky eaters or simply people looking to discover new foods quickly decide on what they could eat at the current moment. As creators we've had experiences where we were out exploring the city, whether alone, with a date or family, and a time always came when we became hungry and started looking for something to eat. Going through food sites, such as Yelp, in order to decide on what to eat was always cumbersome as you have to look through many restaurants, make sure that they are in your area and decide on what type of food you're in the mood for. This was especially awkward when you're out on the street and trying to decide where to go next. This problem is often exacerbated by the fact that people can be provided with too many options at once, leading to even harder decision making. The opposite is also possible, where a person may not be aware of enough variety when it comes to cuisines and dishes. The goal of our application, FoodJunkies, is to minimize these problems by providing our users quick and seemingly simple recommendations based on their likes, budget, time, distance while avoiding foods that might go in conflict with their allergies.

The way we went about accomplishing this goal was through careful design process with the experience of our users as our main consideration. The first most basic requirements were that the application had to work on both, iOS and Android, the two most popular platforms for mobile phones. It also had to be intuitive to use plus simple and efficient enough that it can be used on the go. During design process we have come up with basic strategy processes for the ways we implemented our solutions. One example is the way we went about creating the recommendation process:

Goal 1: Provide quick dish recommendations that relate to users tastes

Strategy:

1. Establish what the user cannot eat (Ask about allergies)
2. Establish a base for users tastes/likes (Brief food quiz)
3. Give recommendations based on provided info and filters used
4. Gather feedback on any new food that the user has tried (Rating)
5. Incorporate the new data into user's tastes
6. Give recommendations using updated info and filters used

2. FEATURES

The features of FoodJunkies came down to this:

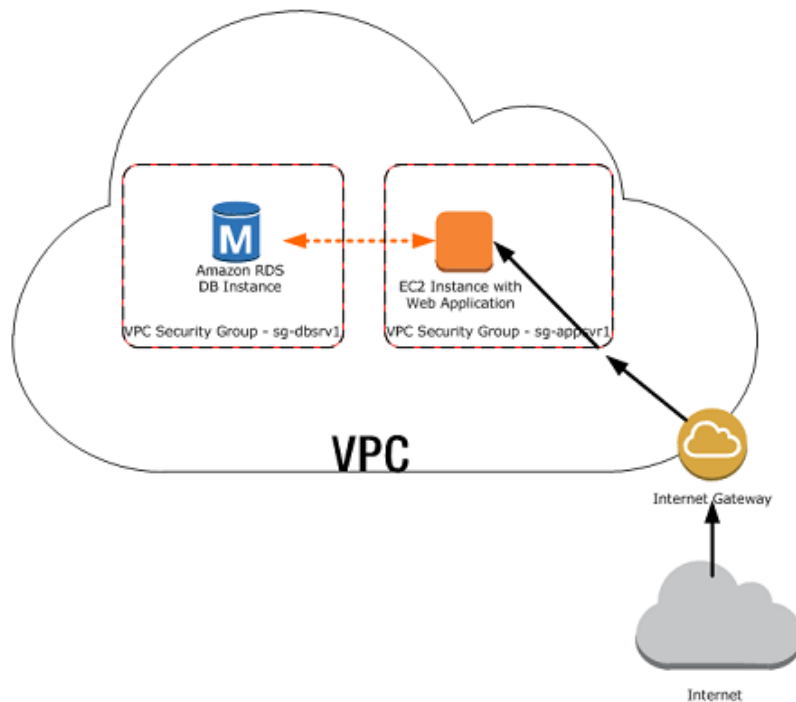
- Way of establishing users tastes and likes, done through a quiz and a rating page
- Dish recommendations based on users tastes and likes while eliminating food that the user cannot consume due to allergies or food they have previously disliked
- Use of constraints such as budget, time to spend on eating, distance to filter results
- Ability to search for specific food and displaying restaurants carrying the searched food
- History of visited restaurants
- Simplicity and ease of use in order to avoid wasting users time

3.1. BACKEND DEVELOPMENT

To create FoodJunkies we needed to first design and create a database that will store the necessary data. After research we have settled for using Amazon Web Services to host our EC2 apache webserver and MySQL database due to its popularity, options and cost.

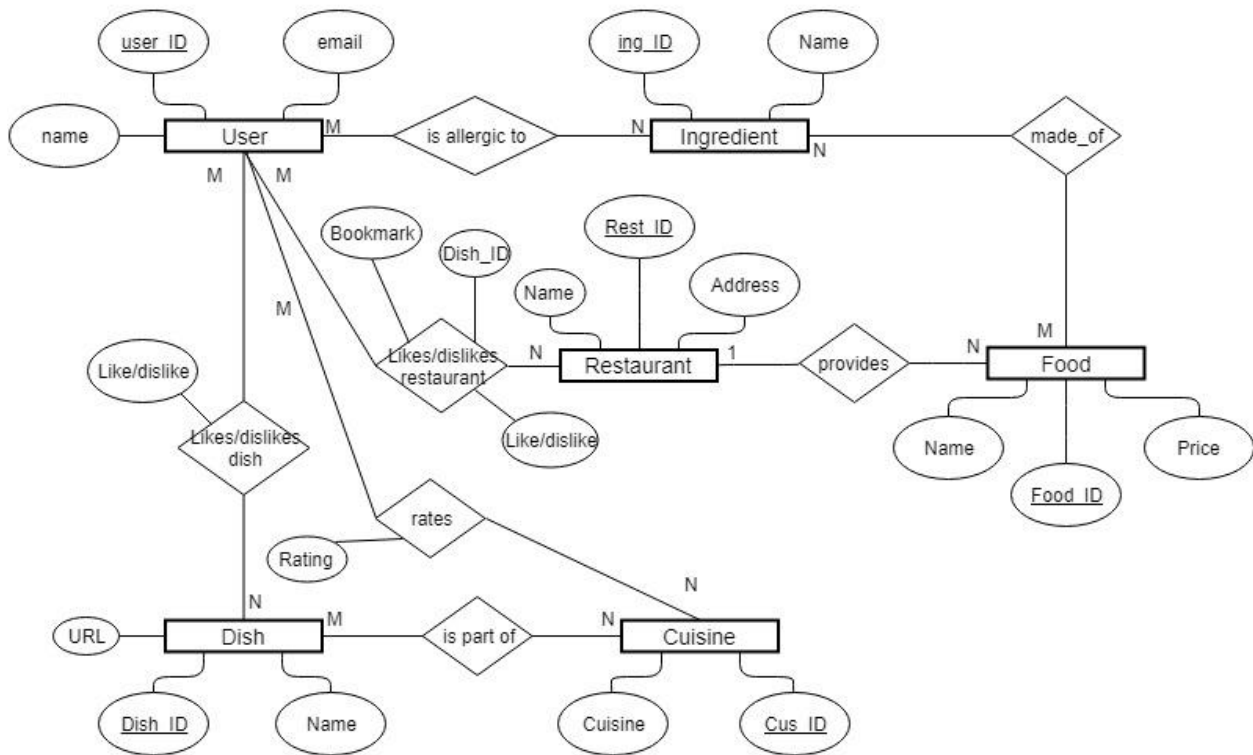
Our application does not connect directly to the MySQL database but the EC2 server which then transfers the connection to the database and also holds our PHP scripts that interact with database.

The way the connection works can be illustrated this way:



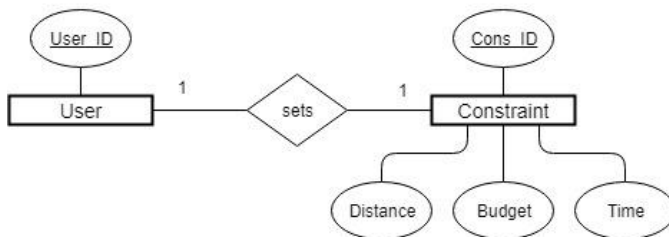
The first stage of database design was the creation of ER Diagram:

Full Database ER Diagram Design v.3F



And a small SQLite for our constraints:

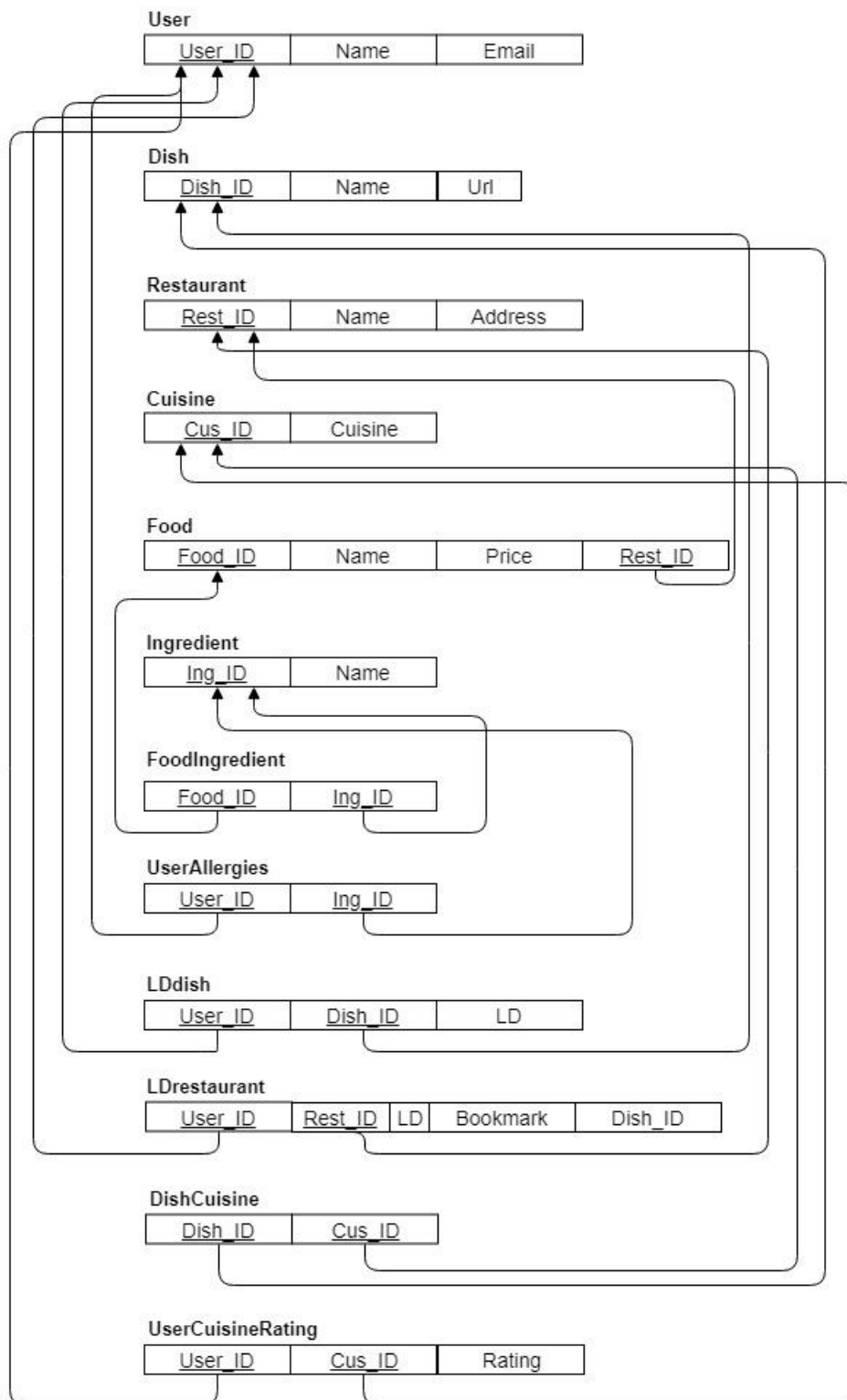
SQLite Constraint Database



<u>User ID</u>	Distance	Budget	Time
----------------	----------	--------	------

From the ER Diagram we have created a Relational Schema:

Full Database Relational Schema v.3F



As you can see our database stores items such as:

- User info
- List of ingredients
 - Used in users allergies
 - Used in food
- Restaurants
- List of general dishes
- Food served by restaurants aka menus
- List of cuisines
- Likes and dislikes of the user
 - Restaurants
 - Dishes
- Users ratings of cuisines

3.2. Table Construction

```
CREATE TABLE User
(User_ID INT(5) NOT NULL AUTO_INCREMENT,
Email VARCHAR(40) NOT NULL,
Password VARCHAR(32) NOT NULL,
PRIMARY KEY (User_ID),
UNIQUE KEY (Email));
```

```
CREATE TABLE Dish
(Dish_ID INT(5) NOT NULL AUTO_INCREMENT,
Name VARCHAR(10) NOT NULL,
Url VARCHAR(100),
PRIMARY KEY (Dish_ID),
UNIQUE KEY (Name));
```

```
CREATE TABLE Restaurant
(Rest_ID INT(5) NOT NULL AUTO_INCREMENT,
Name VARCHAR(200) NOT NULL,
Address VARCHAR(200) NOT NULL,
PRIMARY KEY (Rest_ID),
UNIQUE KEY (Name, Address));
```

```
CREATE TABLE Cuisine
(Cus_ID INT(5) NOT NULL AUTO_INCREMENT,
Cuisine VARCHAR(20) NOT NULL,
PRIMARY KEY (Cus_ID),
UNIQUE KEY (Cuisine));
```

```
CREATE TABLE Food
(Food_ID INT(5) NOT NULL AUTO_INCREMENT,
Name VARCHAR(20) NOT NULL,
Price DECIMAL(5,2) NOT NULL,
Rest_ID INT(5) NOT NULL,
PRIMARY KEY (Food_ID),
FOREIGN KEY (Rest_ID) REFERENCES Restaurant(Rest_ID));
```

```
CREATE TABLE Ingredient
(Ing_ID INT(5) NOT NULL AUTO_INCREMENT,
Name VARCHAR(20) NOT NULL,
PRIMARY KEY (Ing_ID));
```

```
CREATE TABLE FoodIngredient
(Food_ID INT(5) NOT NULL ,
Ing_ID INT(5) NOT NULL,
PRIMARY KEY (Food_ID, Ing_ID),
FOREIGN KEY (Food_ID) REFERENCES Food(Food_ID),
FOREIGN KEY (Ing_ID) REFERENCES Ingredient(Ing_ID));
```

```
CREATE TABLE UserAllergies
(User_ID INT(5) NOT NULL,
Ing_ID INT(5) NOT NULL,
PRIMARY KEY (User_ID, Ing_ID),
FOREIGN KEY (User_ID) REFERENCES User(User_ID),
FOREIGN KEY (Ing_ID) REFERENCES Ingredient(Ing_ID));
```

```
CREATE TABLE LDdish
(User_ID INT(5) NOT NULL,
Dish_ID INT(5) NOT NULL,
LD VARCHAR(1) NOT NULL,
PRIMARY KEY (User_ID, Dish_ID),
FOREIGN KEY (User_ID) REFERENCES User(User_ID),
FOREIGN KEY (Dish_ID) REFERENCES Dish(Dish_ID));
```

```
CREATE TABLE LDrestaurant
(User_ID INT(5) NOT NULL,
Rest_ID INT(5) NOT NULL,
LD VARCHAR(1),
Bookmark VARCHAR(1) NOT NULL,
Dish_ID INT(5),
PRIMARY KEY (User_ID, Rest_ID),
FOREIGN KEY (User_ID) REFERENCES User(User_ID),
FOREIGN KEY (Rest_ID) REFERENCES Restaurant(Rest_ID));
```

```
CREATE TABLE DishCuisine
(Dish_ID INT(5) NOT NULL,
Cus_ID INT(5) NOT NULL,
PRIMARY KEY (Dish_ID, Cus_ID),
FOREIGN KEY (Dish_ID) REFERENCES Dish(Dish_ID),
FOREIGN KEY (Cus_ID) REFERENCES Cuisine(Cus_ID));
```

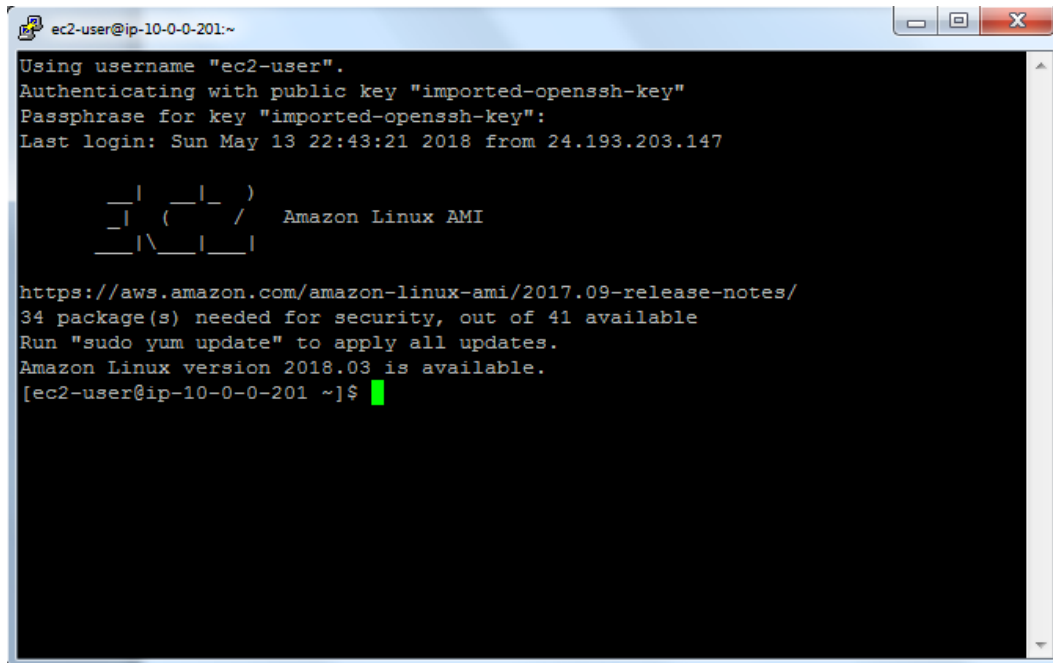
```
CREATE TABLE UserCuisineRating
(User_ID INT(5) NOT NULL,
Cus_ID INT(5) NOT NULL,
Rating INT(2) NOT NULL,
PRIMARY KEY (User_ID, Cus_ID),
FOREIGN KEY (User_ID) REFERENCES User(User_ID),
FOREIGN KEY (Cus_ID) REFERENCES Cuisine(Cus_ID));
```

3.2. PHP, MySQL AND CONNECTIONS

As previously stated our interactions with the server are done through the use of PHP scripts located on our EC2 webserver. Each connection is done by sending a POST message carrying the necessary parameters and values to the servers URL containing a specific PHP file.

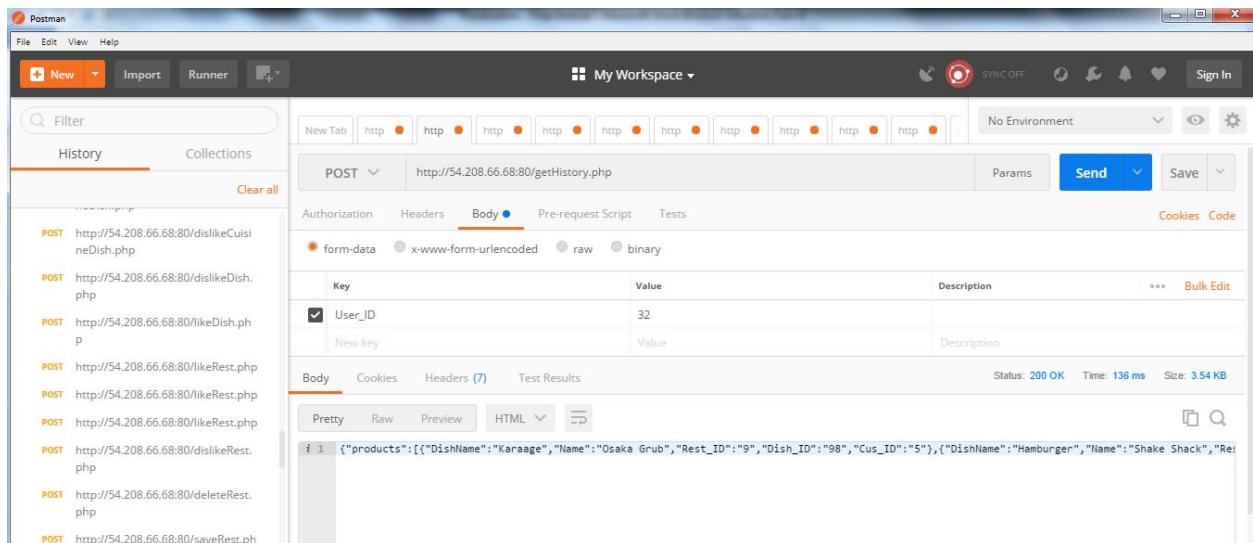
An example of such URL: `http://54.208.66.68:80/getHistory.php`

Our non-application connections to the EC2 apache server have been done using the program Putty, through which we have also inserted all of our PHP files.



```
ec2-user@ip-10-0-0-201:~  
Using username "ec2-user".  
Authenticating with public key "imported-openssh-key"  
Passphrase for key "imported-openssh-key":  
Last login: Sun May 13 22:43:21 2018 from 24.193.203.147  
  
  _ | _ | _ )  
  _ | ( _ /  Amazon Linux AMI  
  _ | \ _ | _ |  
  
https://aws.amazon.com/amazon-linux-ami/2017.09-release-notes/  
34 package(s) needed for security, out of 41 available  
Run "sudo yum update" to apply all updates.  
Amazon Linux version 2018.03 is available.  
[ec2-user@ip-10-0-0-201 ~]$
```

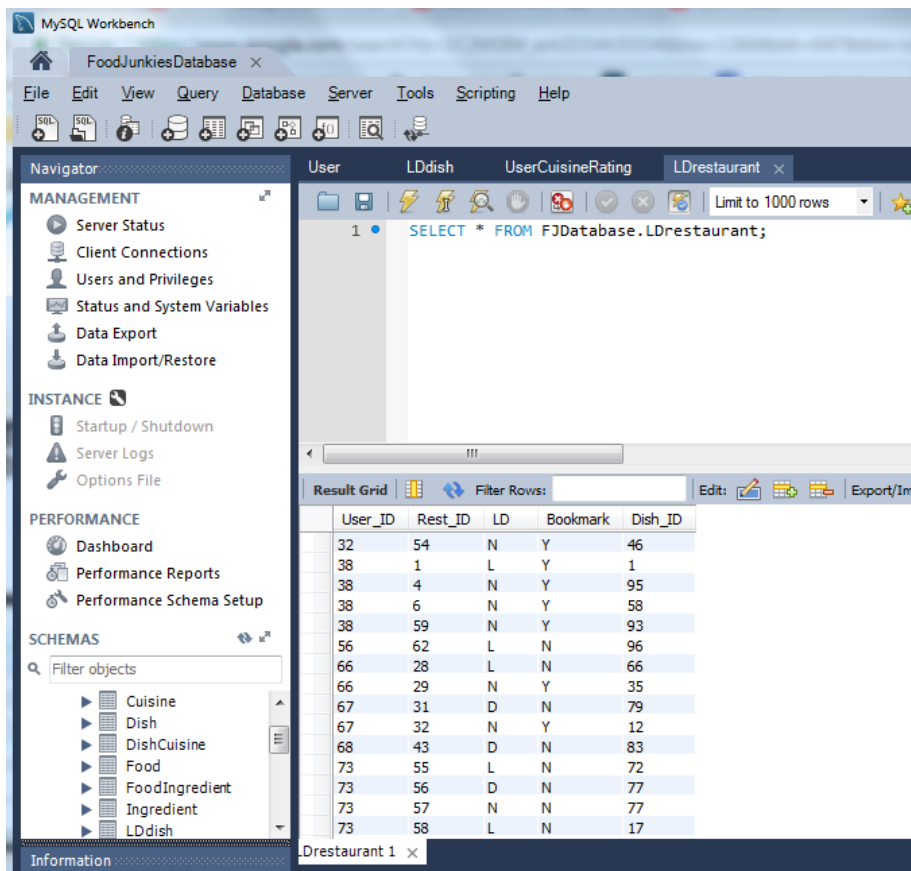
Testing of the connections and the scripts was done through the use of Postman program.



Here is the full list of PHP files created for our database interactions. All of these will also be found on our github:

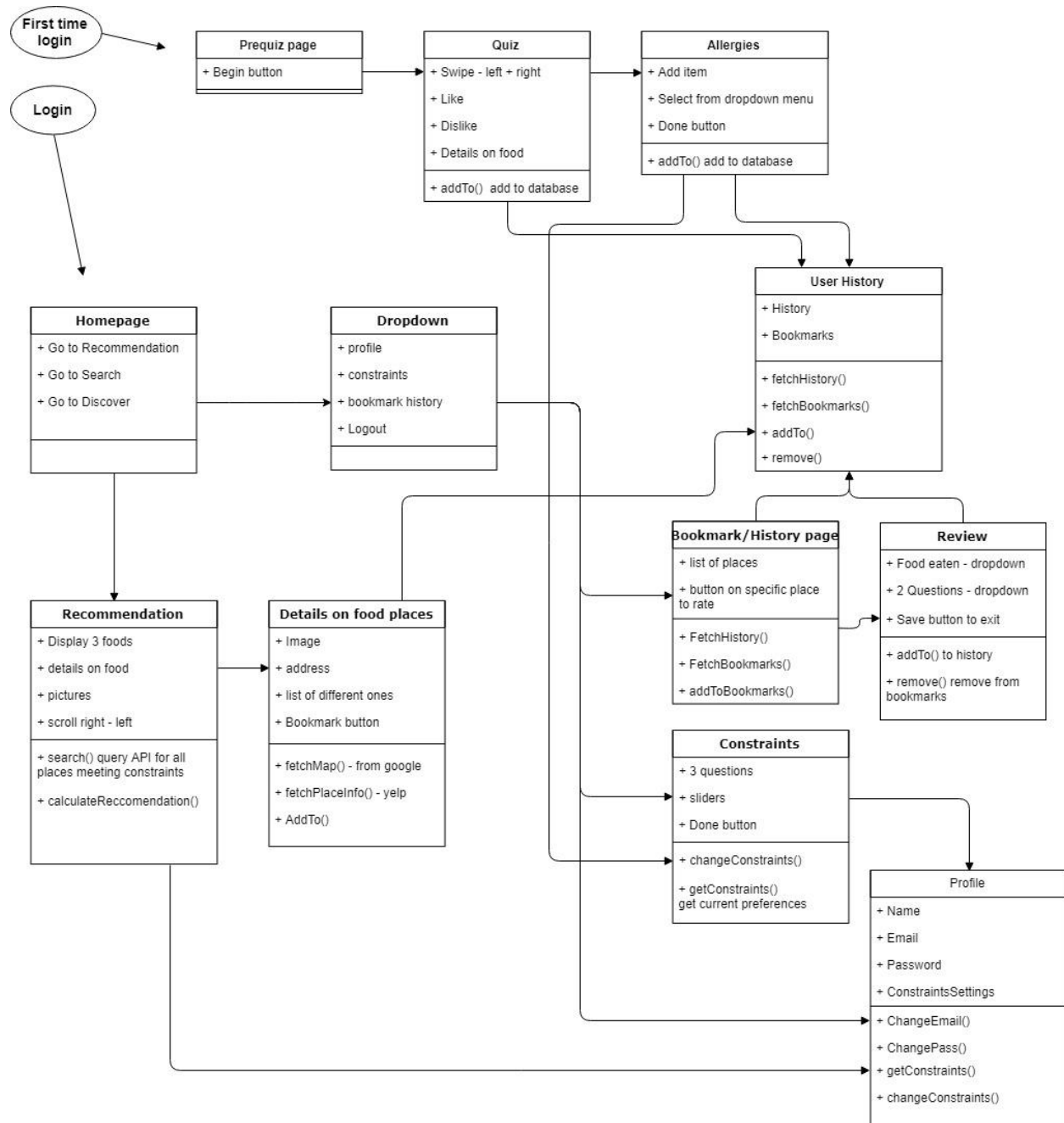
addAllergy.php	
addRestaurant.php	
config.php	
connection.php	
deleteRest.php	
dislikeCuisine.php	
dislikeCuisineDish.php	
dislikeDish.php	
dislikeRest.php	
getAllergies.php	
getBookmarks.php	
getBookmarks2.php	
getDishCuisine.php	
getDishes.php	
getDishName.php	
getHistory.php	
getRatings.php	
getRestID.php	
likeCuisine.php	
likeCuisineDish.php	
	likeDish.php
	likeRest.php
	Login
	recommend
	SamplePage.php
	SamplePage2.php
	saveRest.php
	setDefaultRating.php
	user_control.php
	user_control2.php

For the creation and maintenance of our MySQL database we have used MySQL Workbench program.



4. EARLY STAGE CODE DESIGN AND WIREFRAMES

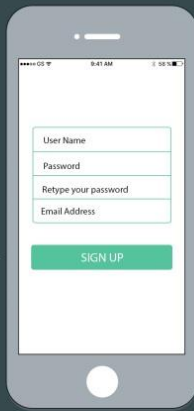
As we worked on the application and learned more about the creation process the code design went through many changes. The following is the first design which ended up quite lacking when we finally started implementing everything so this does not reflect the final application code structure at all but is part of the design process.



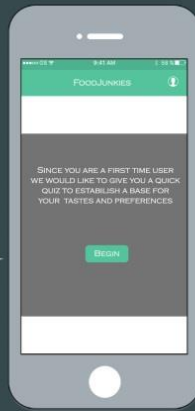
SIGN UP PROCESS



LOGIN SCREEN



SIGN UP SCREEN



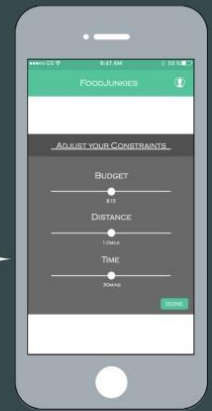
PREQUIZ



PREFERENCES
QUIZ

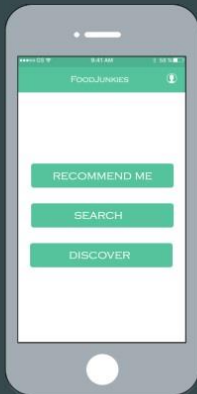


ALLERGIES



RESULT
CONSTRAINTS

RECOMMENDATION PROCESS



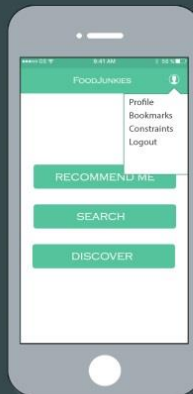
HOMEPAGE



RECOMMENDATIONS



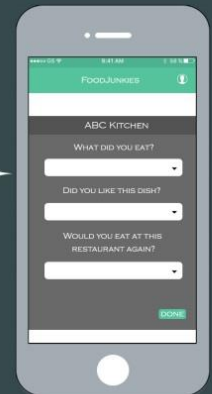
RESTAURANT
INFO



DROPDOWN
MENU

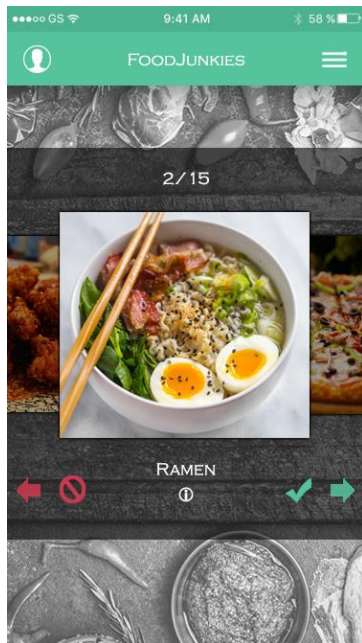


BOOKMARKS



REVIEW

5. EARLY STAGE VISUAL DESIGN



VISUAL DESIGN CONCEPTS



6. QUIZ

During the design of our quiz we ended up with two choices on how to implement it. Each choice came with its own cons and pros.

Choice 1:

Show X number of preselected items from the Dish table in order to provide a specific predefined variety.
Rating the dish adjusts score of all the cuisines it belongs to.

For example: 3 dishes per cuisine for a total of 30 items if there are 10 cuisines.

Pros: Higher precision in rating

Cons: Too many items to rate

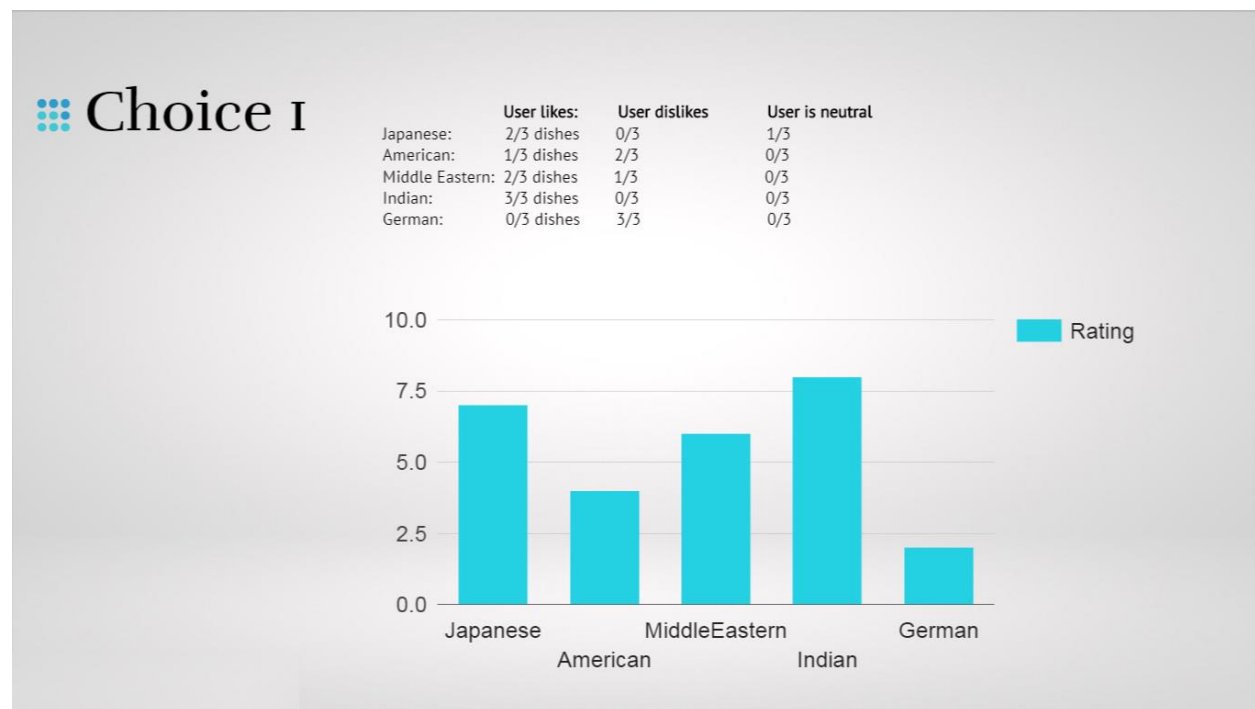
Choice 2:

Show all the cuisines from the Cuisine table for user to rate directly

Total of 10 items shown for 10 cuisines

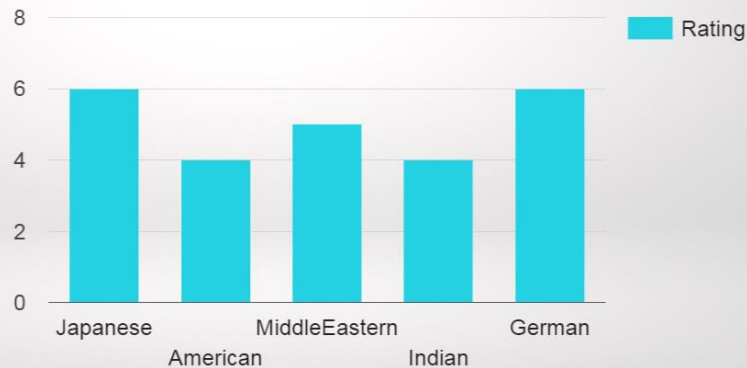
Pros: Quick and simple

Cons: Ratings aren't as precise



Choice II

Japanese: Like
American: Dislike
Middle Eastern: Neutral
Indian: Like
German: Dislike



In the end we went with choice number 2

7. RECOMMENDATION ALGORITHM

Each cuisine gets a rating during the initial one time quiz and later from liked/disliked dishes.

First the algorithm picks a cuisine. The selection is semi random, where the score of the each cuisine affects how likely the cuisine is to be picked.

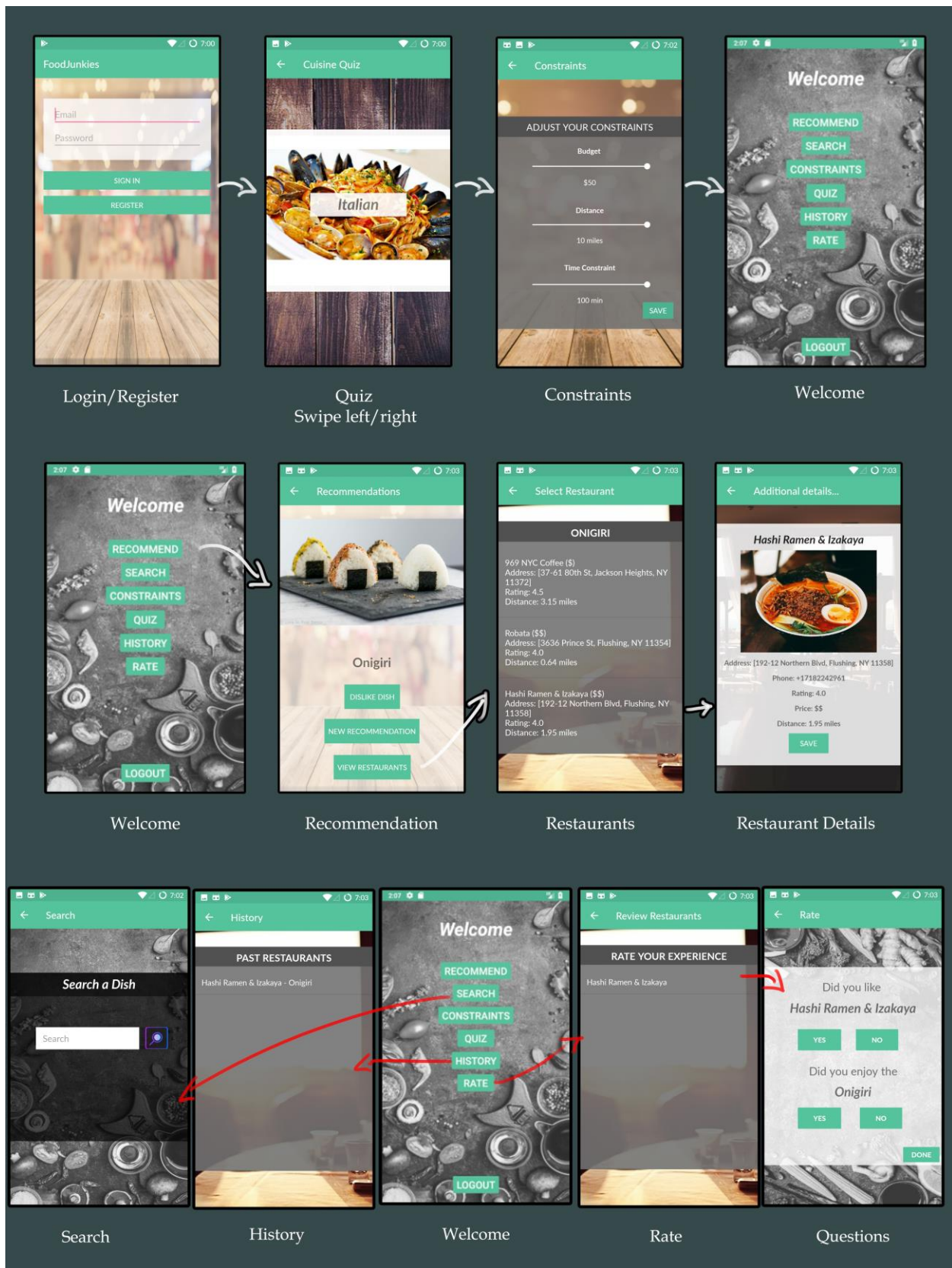
Once a cuisine is picked, the algorithm then selects at random a dish from Dish table.

Algorithm then searches food and restaurant database to find restaurants within specified distance with a food that belongs to this dish type. Any results with ingredients that the user is allergic to are eliminated. Previously disliked foods are eliminated. If no results are found, repeat step two till an acceptable result is found.

8. ALLERGIES

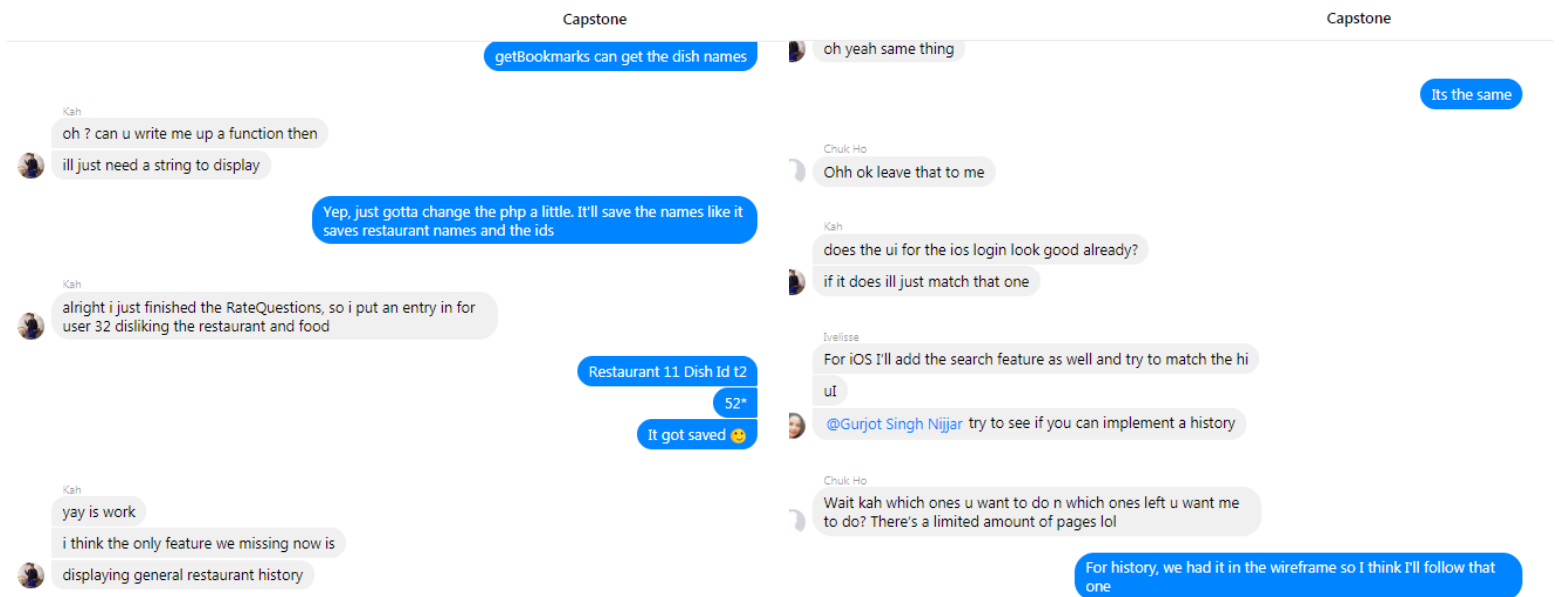
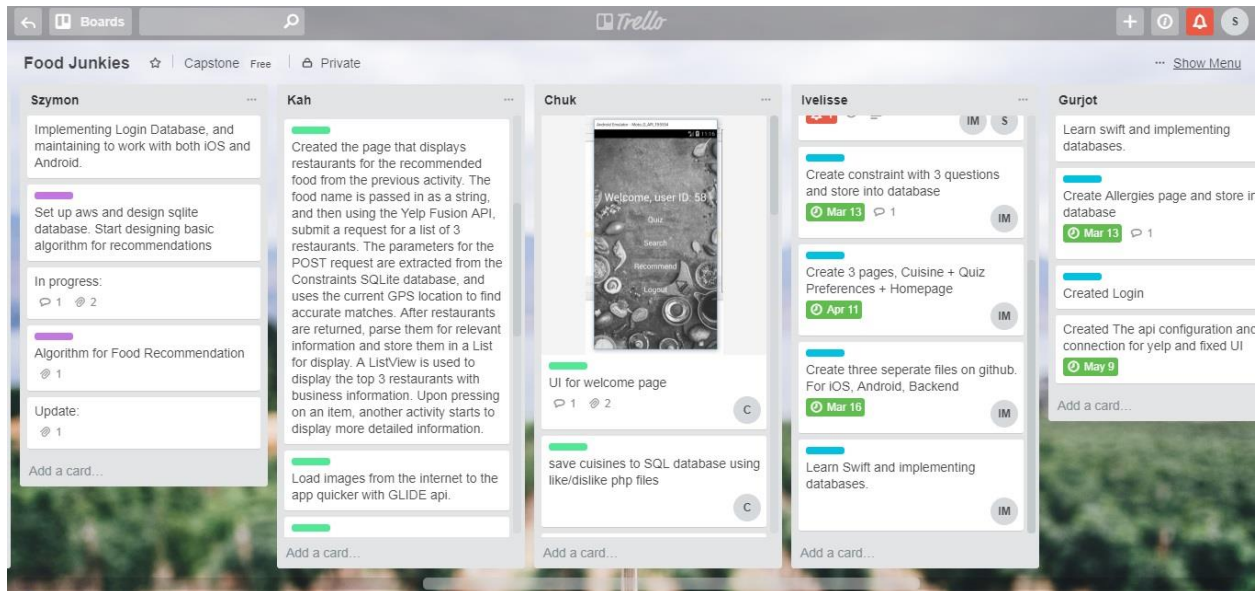
Unfortunately for the final product we had to remove the function of allergies as their implementation would require a monthly subscription of \$500 to the API service. If the subscription is bought then the code can be easily added back.

9. UPDATED WIREFRAME - ACTUAL SCREENSHOTS FROM THE WORKING PRODUCT:



10. TEAMWORK

For our collaboration we have used few websites and applications: Github for sharing code, Trello for setting up tasks and stories, Facebook Messenger and Discord for live communications and screen sharing



2022 - Personal Visual Redesign

