CS304 Phase 2
Sharon Zhang and Wendy Wen
Boston Foodie
Design for a concise restaurants and food items recommender

Operations and Different Behaviors by Users:

Boston foodie mainly allows two operations: searching for ideal restaurants and updating the restaurants and dishes databases. Different users might want to search for restaurants by setting different personal preferences, or inputting different dishes. A single user, beside searching, might want to comment on the restaurant, or recommending a dish.

(We assume now each restaurant is unique and is not a chain store)

Searching: In the home page, any user with the link, without logging into the web app, is allowed to search in two ways: general search and dish search

a. *General search:* Users should select at least one of their preferred type of cuisine (Japanese, Italian, etc), preferred location, or preferred restaurant type (cafe or meal), and click "general search" to search for the restaurants. The one or two not being selected will be set to default, which is any type or location. If users do not select all three, the page will provide the most popular ten restaurants from the database.
b. *Dish search:* User can input the name of his/her favourite dish, and click "dish search" to search for restaurants with this dish or dishes with similar name being recorded. Only one dish name can be input at one time. (Detected by punctuation)

Updating: After logging in, users could update the databases in three ways: adding new restaurants, adding new dishes, commenting on a restaurant and liking a dish.

a. *Add restaurants*: Users should enter a restaurant name, a location, then select a cuisine flavor and a type, and click "add restaurant" to add the restaurant. If any data is missing, the restaurant could not be added, and an error message will pop up. If the restaurant exists, the database will update the cuisine flavor of this restaurant, if the selected cuisine flavor is not included already. Otherwise, nothing will change, and users will be notified that the restaurant is in the database.
b. *Add dishes*: Users should enter a restaurant name and a dish name to add the dish. If the dish is existed, user will be notified. If the restaurant is not existed, user will be notified and invited to add a restaurant first.
c. *Commenting*: Within each restaurant page, users are allow to add comment within 50 character length, and click the "add comment" to the restaurant.

Users are encouraged to comment in several adjectives instead of long sentences.

d. *Like dishes*: Within each restaurant page, users can like or dislike a dish by clicking the like button after each dish listed. Each time the button is click, its corresponding value will increase by one. Each user can only like a dish once. The user can later on unlike his/her previous "like". If the same user intend to like the same dish twice, he/she will be notified, and the database will not be updated.

How pages and forms work specifically:

Searching

For searching restaurants, after user enter their preference and click the button, the general search form will be submitted. If users simply click the button without choosing any preference, the page will provide ten most popular restaurants from the database, each with a hyperlink leading to more details.
If user click any of the hyperlink, the section that display popular restaurants will now display the detail of that restaurant.
For searching dishes, after user enter dish and click the button, the dish search form will be submitted. If users simply click the search button without any dish detail entered, nothing will be displayed.
Each time when user re-submit either form, the display area will display the most recent result getting from the submitted form.

Updating

User could click the login hyperlink in the home page to login. The link will direct user to a new login page. If users try to click "add restaurant", "add dish", or "like" button before login, they will be notified and direct to login page, and no update will be processed.
After login, user can like the dishes he/she hasn't liked before. By clicking the like after the dish, the credit table will insert a new row with the user's id and the dish's id. On the restaurant's page, the like number of that dish will increase by 1. If the user re-click the like, the row will be removed from the database, and the displayed number of like will decrease by 1. (We will be cautious on the foreign key restriction for sure)
If user adds new restaurants or dishes, by clicking the "Add" button, the update form will be submitted. If succeeded, either the restaurant table or the dish table will be updated. If restaurant exists, a message will display, and the database won't be updated.
If user adds comment to a restaurant, the restaurant table will be updated. On the restaurant display area, the most recent three comments will be displayed.

Prioritized list of features:

Draft: we intend to

- Create a test file including some self-created restaurant and dishes data (This will be helpful for us to test our following functions)
- Create the search page for general users with search section on it
- Create two forms used to search by general and by dishes respectively
- Start the cgi code of the form and functionalize the search button
- Create the login page for login users. This looks similar to the search page, but has the additional update section
- Create the update form that allow users to add/update restaurant and dishes
- Create the template of the section that display details of each restaurant

Alpha: we intend to

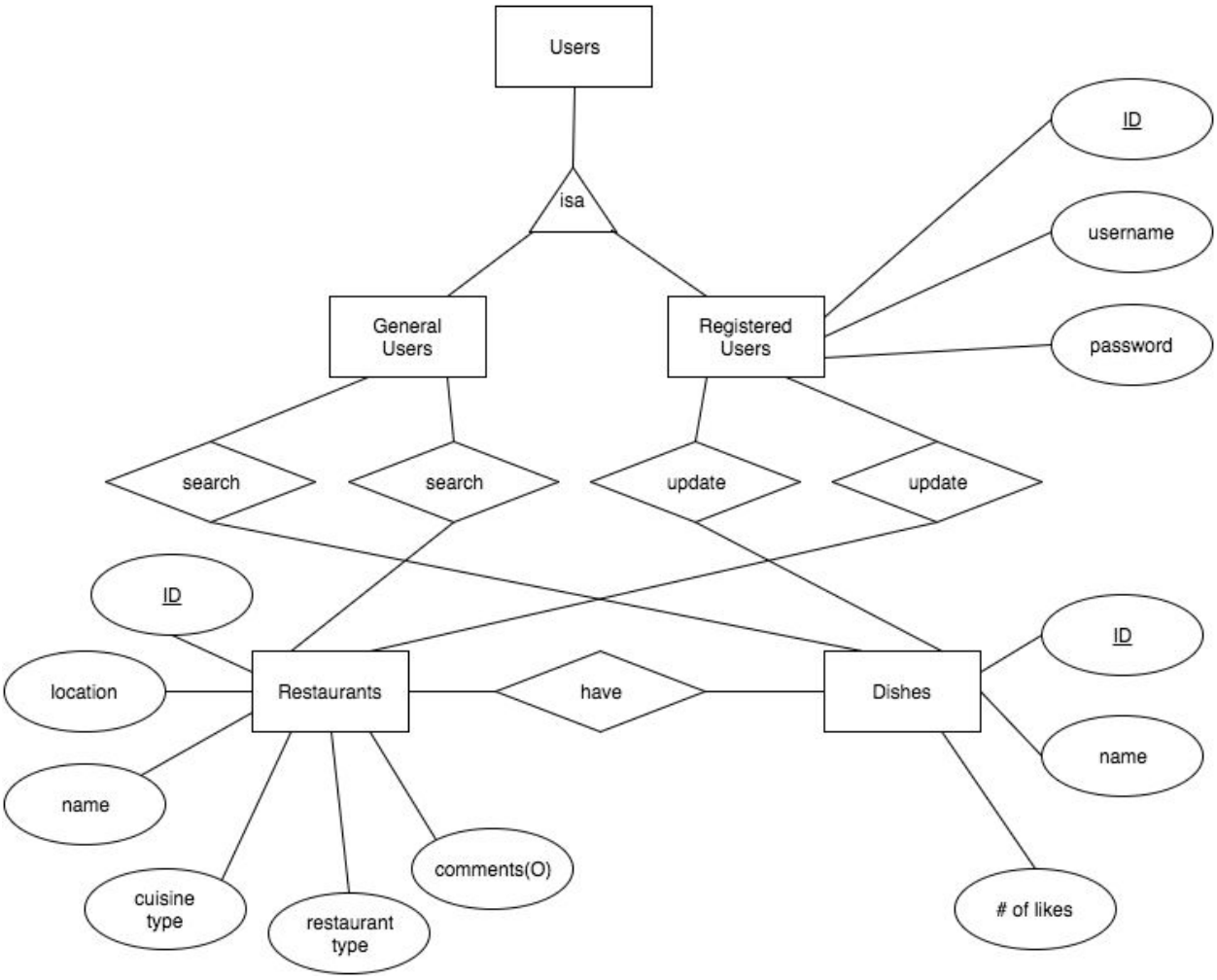
- Establish the database that includes the restaurants table and the dishes table
- Finish the code that allows the users to add new restaurants into database and check if the restaurant exists
- Finish the code that allows the users to add new dishes through update form
- Connect the cgi file to the database

Beta: we intend to

- Create the login feature and create the corresponding people table in the database
- Enable the feature that the users can like the dish (only once) and create the corresponding credit table in the database
- Create the feature that allows the users to leave comments to restaurants

Sharon will do the red part, and Wendy will do the blue part. We will finish the black part together.

ER Diagram



DDL statements that create the tables

User Table

```
CREATE TABLE users (
        id int NOT NULL AUTO_INCREMENT PRIMARY KEY,
        username varchar(50) NOT NULL,
        password varchar(50) NOT NULL
) ENGINE = InnoDB;
```

Restaurant Table

```
CREATE TABLE restaurants (
        id int NOT NULL AUTO_INCREMENT PRIMARY KEY,
        name varchar(50) NOT NULL,
        location varchar(50) NOT NULL,
        cuisine_type varchar(50) NOT NULL,
```

```
    res_type enum("cafe", "meal")
) ENGINE = InnoDB;
```

Restaurants sample data:

1. Id: 1
   Name: Alta Strada
   Location: Wellesley
   Cuisine_type: Italian
   Res_type: meal
2. Id: 2
   Name: Lemon Thai
   Location: Wellesley
   Cuisine_type: Thai
   Res_type: meal
3. Id: 3
   Name: Flour
   Location: Cambridge
   Cuisine_type: dessert
   Res_type: cafe

Dish Table

```
CREATE TABLE dishes (
    id int NOT NULL AUTO_INCREMENT PRIMARY KEY,
    name varchar(50) NOT NULL,
    num_of_likes int DEFAULT 0,
    res_id int NOT NULL,
    FOREIGN KEY (res_id) REFERENCES restaurants(id) ON DELETE restrict),
) ENGINE = InnoDB;
```

Sample Data:

```
INSERT INTO tablename VALUES (null,name,num,res_id);
```

1. Id: 1
   Name: Caesar Salad
   Num_of_likes: 0
   Res_id: 1
2. Id: 2
   Name: Roasted Cauliflower
   Num_of_likes: 0
   Res_id: 1
3. Id: 3
   Name: Wellesley Fried Rice
   Num_of_likes: 0
   Res_id: 2
4. Id: 4

Name: Pad See Eaw
Num_of_likes: 0
Res_id: 2

5. Id: 5
   Name: Awesome Almond Croissant
   Num_of_likes: 0
   Res_id: 3
6. Id: 6
   Name: Tasty Latte
   Num_of_likes: 0
   Res_id: 3

Likes Table

```
CREATE TABLE likes (
        user_id int NOT NULL,
        dish_id int NOT NULL,
        FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE restrict,
        FOREIGN KEY (dish_id) REFERENCES dishes(id) ON DELETE restrict
) ENGINE = InnoDB;
```

Comment Table

```
CREATE TABLE comments (
        user_id int NOT NULL,
        res_id int NOT NULL,
        comments varchar(200) NOT NULL,
        FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE restrict,
        FOREIGN KEY (res_id) REFERENCES restaurants(id) ON DELETE restrict
) ENGINE = InnoDB;
```