Design Document

Team name: FutureGoogleDevs

Team Leader: Hadi Al Lawati

Joey Rice

Malhar Pandya

Michael Ching

Instructor: Arash Saifhashemi

California State University, Long Beach

College of Engineering

CECS 491A, Sec 07 122553,

Fall 2022 September 23, 2022

Objective and Background:

People have been struggling and have been stuck trying to figure out what to eat since the dawn of time. According to a new study that's been published in the journal *Nature Human Behaviour,* it's basically when your brain is encountered with an overwhelming quantity of choices; it stumbles and struggles to make a decision. This overwhelming behavior of people can be helped with our app, "FoodFate."

The objective is to support indecisive people in discovering appropriate restaurants for them by building profiles that are tailored to our user's preferences. We will use that information to suggest, find, and rate food in a user-specified zone, or if they're feeling adventurous, we can find a random restaurant fit for them.

List of classes:

- Entity: Users, Restaurants, Review
- Control: Google maps API, Yelp API, userControl, restaurantControl, routeControl, reviewControl,

- Boundary: loginBoundary, profileBoundary, searchBarBoundary, reviewResturantBoundary, **mapInteractionBoundary,**

UML Diagram:

List of methods: **Only do this for Entity Objects?**

1) User
    a) Method: getUserName
        - Description: returns the name of the user stored in the database
        - Preconditions: account must be in the database
        - Postconditions: user's name is returned to the control object
        - Signature: String getUserName();
    b) Method: getPassword

- Description: returns the password of the user stored in the database
- Preconditions: account must be in the database
- Postconditions: user's password is returned to the control object
- Signature: String getPassword();

c) Method: getEmail
- Description: returns the email of the user stored in the database
- Preconditions: user's email must be in the database
- Postconditions: user's email is returned to the control object
- Signature: string getEmail();

d) Method: setUserName
- Description: sets the name of the user and updates the database
- Preconditions: account must be in the database
- Postconditions: user's name is updated
- Signature: void setUserName();

e) Method: setPassword

- Description: sets the password of the user and updates the database
- Preconditions: user's name must be in the database
- Postconditions: user's password is updated
- Signature: void setPassword();

f) Method: setEmail
- Description: sets the email of the user and updates the database
- Preconditions: user's email must be in the database
- Postconditions: user's email is updated
- Signature: void setEmail();

2) Restaurant

a) getRestaurantName():
   i) Description: returns the name of the restaurant stored in the database
   ii) Preconditions: restaurant must be in the database
   iii) Postconditions: name of the restaurant is returned to the control object
   iv) Signature: String getRestaurantName();

b) getRestuarantAddress():

i) Description: returns the address of the restaurant stored in the database
ii) Preconditions: restaurant must be in the database
iii) Postconditions: address of the restaurant is returned to the control object
iv) Signature: String getRestaurantAddress();

c) getRestuarantRating():
i) Description: returns the rating of the restaurant from yelp
ii) Preconditions: restaurant must be in the yelp database, yelp API must be functional
iii) Postconditions: rating of the restaurant is returned to the control object
iv) Signature: Float getRestaurantRating();

d) setRestaurantName():
i) Description: set the name of the restaurant(could be used in update)
ii) Preconditions: restaurant must be in the database
iii) Postconditions: name of the restaurant is changed in our database
iv) Signature: void setRestaurantName();

e) setRestuarantAddress():

i) Description: set the address of the restaurant(could be used in update)

ii) Preconditions: restaurant must be in the database

iii) Postconditions: address of the restaurant is changed in our database

iv) Signature: void setRestaurantAddress();

3) Reviews:

a) Method: writeReview()

- Description: write the rating of the restaurant
- Preconditions: restaurant must be in the yelp database
- Postconditions: review of the restaurant is updated to database
- Signature: void setRestaurantReview();

b) Method: getReview()

- Description: get the rating of the restaurant
- Preconditions: review of restaurant must be in database
- Postconditions: retrieve the review from the database

c) Method: setRating()

- Description: set the rating of the restaurant

- Preconditions: restaurant must be in the yelp database
- Postconditions: rating of the restaurant is changed in yelp database
- Signature: void setRestaurantRating();

d) editReview()

- Description: change the rating of the restaurant
- Preconditions: review must exist in database
- Postconditions: review of the restaurant is updated in database
- Signature: void editReview();

4) userControl

a) Method: goBack()

- Description: Allows users to go back within the app's constraints.
- Preconditions: Must have registered and logged into the app menu and requires connection to the internet..
- Postconditions: location of user within app is changed.
- Signature : void goBack();

b) Method: SearchResturant()

- Description: Allows users to find and a wanted restaurant.

- Preconditions: Must have registered and logged into the app menu and Also requires connection to internet.
- Postconditions: Finds the restaurant destination and shows the user directions to that specific location.
- Signature : void getSearch();

c) Method: OpenResturantDetails()

- Description: Allows users to look up details about a specific restaurant.
- Preconditions: Must have registered and logged into the app menu and searched a specific restaurant and requires connection to the internet.
- Postconditions: Gives information using Yelp API about the searched resturant.
- Signature : void getDetails;

5) restaurantControl

a) Method: getRandomRestaurant()

i) Description:gets a random restaurant and gives it back to the user.

ii) Preconditions: Must have registered and logged into the app menu. Also requires connection to the internet.

iii) Postconditions: A restaurant is suggested to the user on the map.

iv) Signature : Restaurant getRandomRestaurant();

b) Method: reviewRestaurant()

i) Description: allows the user to give a restaurant a review

ii) Preconditions: Must have registered and logged into the app menu. Also requires connection to the internet. Restaurant must exist.

iii) Postconditions: A review is added to the database, and it is updated.

iv) Signature : void reviewRestaurant();

c) Method: updateRestuarant(String) -

6) loginBoundary:

a) Method: login()

- Description: login into the system

- Preconditions: must be connected to internet, account already exists in database, login information entered correctly
- Postconditions: successfully logged into system under account
- Signature: void login(string, string);

7) profileBoundary:

a) Method: deleteAccount()

- Description: Remove access to account, update system
- Preconditions: must be connected to internet, account already exists in database, login information entered correctly
- Postconditions: user does not have access to account
- Signature: void deleteAccount();

8) searchBarBoundary:

a) Method: searchRestaurant ()

- Description: Searches and retrieves relevant information regarding input restaurant through Yelp API
- Preconditions: must be connected to internet, restaurant exists in yelp's database
- Postconditions: Information is retrieved about searched restaurant

- Signature: arrayList searchResturant();

9) reviewResturantBoundary:

a) Method:

10) mapInteractionBoundary:

a) Method: drawArea()

- Description: Allows user to draw an area on map through Google maps API
- Preconditions: must be connected to internet, map is shown
- Postconditions: A specific area is designated
- Signature: void drawArea();

Sequence Diagram:

Alternatives:

Security:

Privacy:

Reliability:

Scalability: