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:

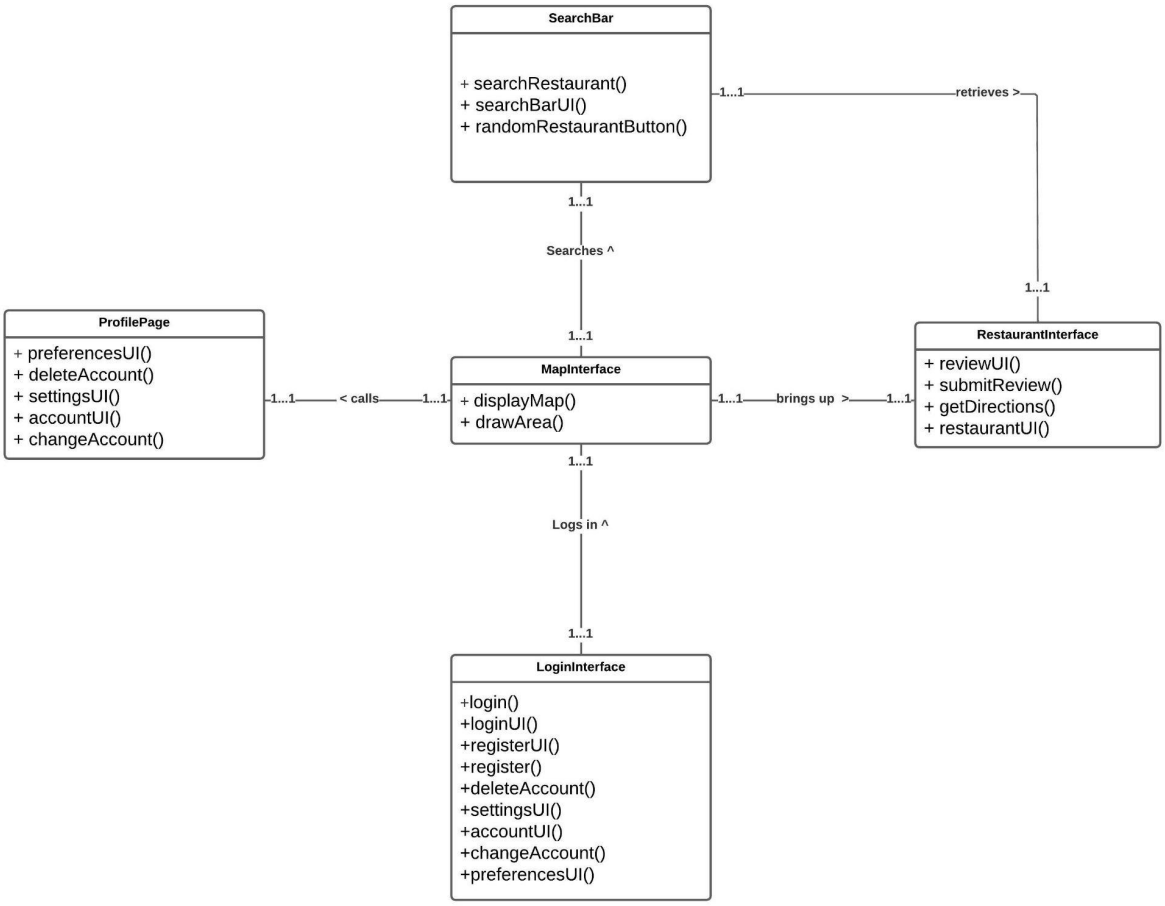
People have been struggling and have been stuck trying to figure out what to eat since the dawn of time. According to a new study that's been published in the journal *Nature Human Behaviour,* it's basically when your brain is encountered with an overwhelming quantity of choices; it stumbles and struggles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

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.

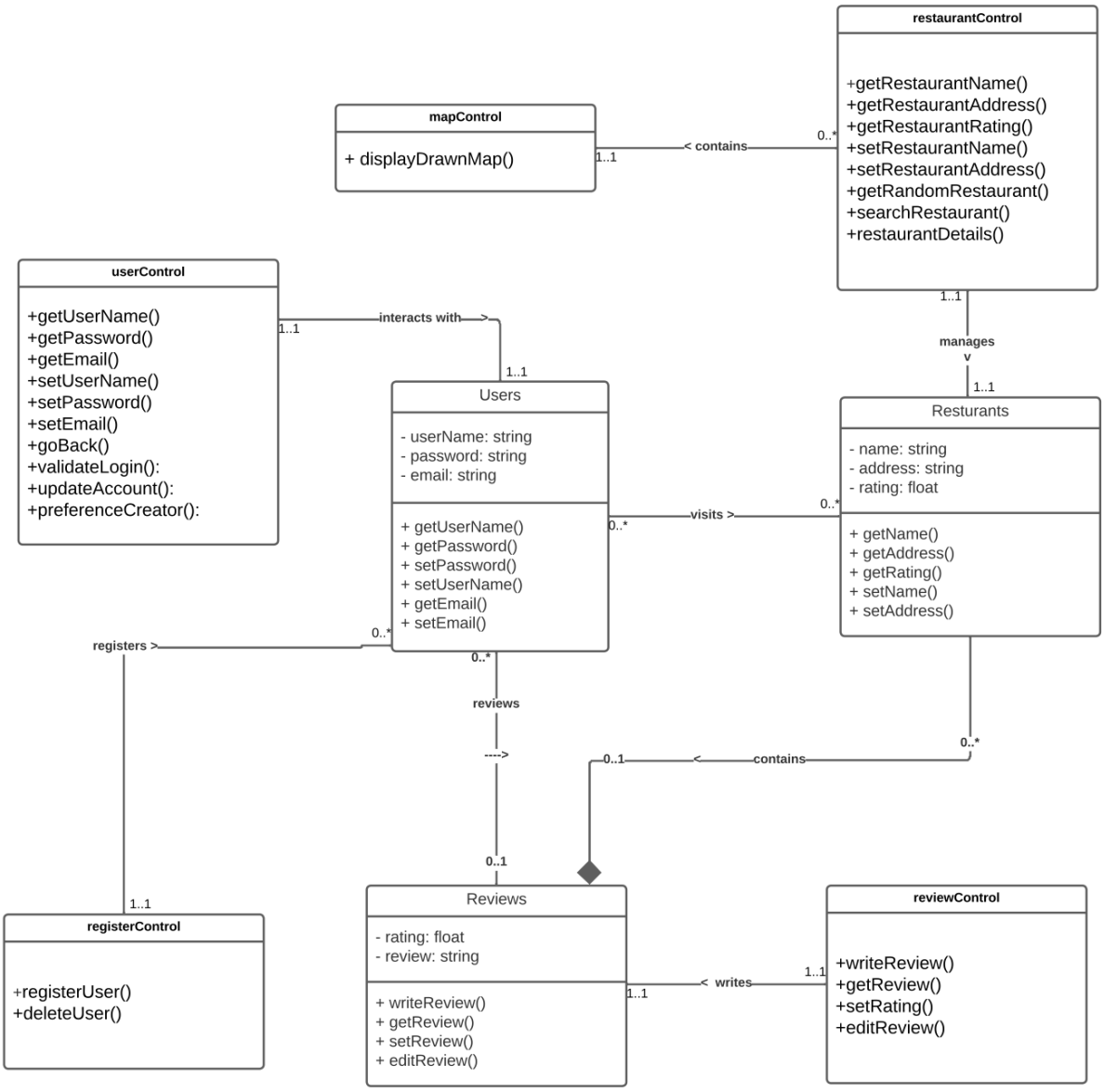
List of classes:

- Entity: Users, Restaurants, Review
- Control: userControl, registerControl, restaurantControl, reviewControl, mapControl
- Boundary: loginInterface, profilePage, searchBar, restaurantInterface, mapInterface

- **UML Diagram:**
    - **Frontend**



- **Backend**


restaurantControl
+getRestaurantName()
+getRestaurantAddress()
+getRestaurantRating()
+setRestaurantName()
+setRestaurantAddress()
+getRandomRestaurant()
+searchRestaurant()
+restaurantDetails()
mapControl
+ displayDrawnMap()
< contains
0..*
1..1
userControl
+getUserName()
+getPassword()
+getEmail()
+setUserName()
+setPassword()
+setEmail()
+goBack()
+validateLogin():
+updateAccount():
+preferenceCreator():
interacts with >
1..1
1..1
manages
v
1..1
Users
- userName: string
- password: string
- email: string
+ getUserName()
+ getPassword()
+ setPassword()
+ setUserName()
+ getEmail()
+ setEmail()
Resturants
- name: string
- address: string
- rating: float
+ getName()
+ getAddress()
+ getRating()
+ setName()
+ setAddress()
visits >
0..*
0..*
registers >
0..*
0..*
reviews
---->
0..*
0..1
< contains
0..1
1..1
registerControl
+registerUser()
+deleteUser()
Reviews
- rating: float
- review: string
+ writeReview()
+ getReview()
+ setReview()
+ editReview()
< writes
1..1
1..1
reviewControl
+writeReview()
+getReview()
+setRating()
+editReview()

List of methods:

Actor → Boundary → Controller → Entities

Entities:

1) User
    a) Method: getUserName()
        - Description: returns the name of the user from the object
        - Preconditions: object must exist , getUserName() must be accessible
        - Postconditions: user's name is returned
        - Signature: String getUserName();
    b) Method: getPassword()
        - Description: returns the password attribute from the object
        - Preconditions: getPassword() must be accessible, the object must exist
        - Postconditions: user's password is returned
        - Signature: String getPassword();
    c) Method: getEmail()
        - Description: returns the email attribute from the object

- Preconditions: getEmail() must be accessible, the object must exist
- Postconditions: user's email is returned
- Signature: string getEmail();

d) Method: setAccessValue()

- Description: set the access value to either true or false
- Precondition: account already exists
- Postcondition: changes the access value
- Signature: void setAccessValue():

e) Method: setUserName()

- Description: sets the name of the user in the object
- Preconditions: account must be in the database, object must exist, setUserName must be accessible
- Postconditions: user's name is updated in object and database
- Signature: void setUserName();

f) Method: setPassword()

- Description: sets the password of the user object
- Preconditions: setPassword() must be accessible, the object must exist

- Postconditions: user's password is updated in the object and database if accessible
- Signature: void setPassword();

g) Method: setEmail()

- Description: sets the email of the user in the object
- Preconditions: user's object must exist, setEmail()
- Postconditions: user's email is updated
- Signature: void setEmail();

2) Restaurant

a) Method: getRestaurantName():

- Description: Returns the name of the restaurant object
- Preconditions: Restaurant object must exist
- Postconditions: Name of the restaurant is returned
- Signature: String getRestaurantName();

b) Method: getRestaurantAddress():

- Description: Returns the address of the restaurant object
- Preconditions: Restaurant object must exist
- Postconditions: Address of the restaurant is returned
- Signature: String getRestaurantAddress();

c) Method: setRestaurantName():

- Description: Set the name of the restaurant object
- Preconditions: Restaurant object must exist
- Postconditions: Name of the restaurant object is updated
- Signature: void setRestaurantName();

d) Method: setRestaurantAddress():

- Description: Set the address of the restaurant object
- Preconditions: Restaurant object must exist
- Postconditions: Address of the restaurant is object is updated
- Signature: void setRestaurantAddress();

3) Reviews

a) Method: writeReview()

- Description: Allows the user to give a restaurant a review
- Preconditions: Must have registered and logged into the app menu. Also requires connection to the internet. Restaurant must exist.
- Postconditions: A review is added to the database, and it is updated.
- Signature: void writeReview();

b) Method: getReview()

- Description: Allows the user to get a review of a restaurant

- Preconditions: Must have registered and logged into the app menu. Review must exist in database. Also requires connection to the internet. Restaurant must exist.
- Postconditions: A review is retrieved from the database.
- Signature: string getReview();

c) Method: setReview()

- Description: Allows the user to set a restaurant a review in float.
- Preconditions: Must have registered and logged into the app menu. Also requires connection to the internet. Restaurant must exist.
- Postconditions: A review is added to the database, and it is updated.
- Signature: void setReview(float);

d) Method: editReview()

- Description: Allows the user to edit a restaurant a review
- Preconditions: Must have registered and logged into the app menu. Must have reviewed the restaurant already. Also requires connection to the internet. Restaurant must exist.
- Postconditions: Review is updated in the database.
- Signature: void editReview();

Controllers:

4) userControl

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: account must be in the database
    - Postconditions: user's email is retrieved

- Signature: void 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();

g) 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();

h) Method : validateLogin():

- Description: Allows users to enter the app after registration and ensures credentials entered by the user are a match.
- Preconditions: Must have registered and have connection to the internet.
- Postconditions: Allows users to access the main page which displays the map.
- Signature: void validateLogin(string, string);

i) Method : updateAccount():

- Description: Allows users to go into settings and change account details.
- Preconditions: Must have registered, have connection to the internet and must be in settingsUI.

- Postconditions: Allows users to update their account details into the database.
- Signature : void updateAccount();

j) Method : preferenceCreator():

- Description: Interacts with the database and sets preferences for where the user wants to eat
- Preconditions: Must have registered, have connection to the internet and must be have filled out preferencesUI question form.
- Postconditions: Allows users to create customized preferences
- Signature: string preferenceCreator();

5) registerControl

a) registerUser():

- Description: makes sure the Username and password are valid, registers the new user into the database
- Preconditions: DB must be accessible, and user registration precondition(unique username, strong password) must be met
- Postconditions: User has an account.
- Signature: boolean registerUser();

b) deleteUser():

- Description: removes user's access from the account
- Preconditions: User must exist in the Database
- Postconditions: user no longer has access to his/her account
- Signature: string deleteUser():

6) restaurantControl

a) getRestaurantName():

- Description: returns the name of the restaurant stored in the database
- Preconditions: restaurant must be in the database
- Postconditions: Name of the restaurant is returned to the control object
- Signature: String getRestaurantName();

b) getRestaurantAddress():

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

c) getRestaurantRating():

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

d) setRestaurantName():

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

e) setRestaurantAddress():

- Description: set the address of the restaurant(could be used in update)
- Preconditions: restaurant must be in the database
- Postconditions: address of the restaurant is changed in our database
- Signature: void setRestaurantAddress();

f) Method: getRandomRestaurant()

- Description:gets a random restaurant and gives it back to the user.
- Preconditions: Must have registered and logged into the app menu. Also requires connection to the internet. User must give location information.
- Postconditions: A restaurant is suggested to the user on the map.
- Signature : Restaurant getRandomRestaurant();

g) Method: searchRestaurant()

- 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 using Google Maps API.
- Signature: void getSearch();

h) Method: restaurantDetails()

- 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 restaurant.
- Signature: string getDetails();

7) reviewControl()

a) Method: writeReview()

- Description: write the rating of the restaurant
- Preconditions: restaurant must be in the yelp database, Must have registered and logged into the app menu. Also requires connection to the internet. Restaurant must exist.
- Postconditions: A review is added to the database, and it is updated.
- 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
- Signature: void getReview();

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();

8) mapControl()

a) Method: displayDrawnMap()

- Description: Displays the area of map
- Preconditions: must be connected to internet, must be logged in, area drawn
- Postconditions: Map is displayed with the drawn area, restaurants retrieved in drawn area
- Signature: list displayDrawnMap();

Boundary

8) loginInterface:

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);

b)Method: loginUI()

- Description: interface to login into the system
- Preconditions: must be connected to internet, account already exists in database
- Postconditions:credentials entered
- Signature: void loginUI(string, string);

c) Method: registerUI()

- Description: Shows User a form to fill out registration information
- Preconditions: must have internet connection
- Postconditions: proper credentials sent to controller object

- Signature: string registerUI()

d) Method: register()

- Description: send the information, from the registerUI form, to the control object
- Preconditions: must be connected to internet
- Postconditions: sends credentials to controller
- Signature: string register(string, string, string, string, string);

8) profilePage:

a) Method: deleteAccount()

- Description: sends a message to register control to remove access to the account
- 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();

b) Method: settingsUI()

- Description: shows a set of buttons: accounts, notifications, appearance, help and support and about us

- Preconditions: must be connected to the internet, account already exists in the database, logged in.
- Postconditions: users gets to view various settings
- Signature: void settings();

c) Method: accountUI()

- Description: shows a set of buttons, options include: change password, delete account, update email, upload picture
- Preconditions: must be logged in, must have network connection, on settingsUI
- Postconditions: User gets to view their account settings while also being able to change password, update email, upload picture and delete account.
- Signature: void accountUI()

d) Method : changeAccount():

- Description: Allows users to go into settings and change account details.
- Preconditions: Must have registered, have connection to the internet and must be in settingsUI.
- Postconditions: Allows users to update their account details into the database.

- Signature: void updateAccount(string, string, string);

e) Method : preferencesUI():

- Description: Allows users to select all the preferences that apply to his profile.
- Preconditions: Must have registered, have connection to the internet.
- Postconditions: Allows users to update and customize their profile which will be updated in the database.
- Signature: void preferencesUI(string);

9) searchBar:

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 searchRestaurant();

b) Method: searchBarUI()

- Description: presents an interface that allows the user to type in a restaurant
- Preconditions: must be connected to internet, author must be able to enter input, search bar must be clicked on
- Postconditions: the filled in information is passed to the searchRestaurant().
- Signature: searchBarUI(string restaurantName)

c) Method: randomRestaurantButton()

- Description: finds a random restaurant in the user's area and lets the user filter different restaurant settings****
    - Discuss what filters to use
    - Discuss what intermediate page between the randomized question and the random search will look like. I.e. questions page and filters page
- Preconditions: must be connected to the internet, the user must be logged in, the restaurant exists in yelp's database, user's location is on, and must be in the restaurant search bar
- Postconditions: the restaurant is selected from the options
- Signature: Restaurant randomRestaurantButton()

10) restaurantInterface:

a) Method: reviewUI()
    - 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: submitReview()
    - Description: pass the review of the restaurant to review control class
    - Preconditions: user fills in fields, restaurant must be in the yelp database
    - Postconditions: review of the restaurant propagated to the review controller
    - Signature: void submitReview();

c) Method: getDirections()
    - Description: Provides a route to the restaurant
    - Preconditions: Restaurant has been chosen, restaurant exists, GoogleMaps API is functional, internet connection
    - Postconditions: A list of directions is returned
    - Signature: List<places> FindRoute (currentLocation, Destination)

d) Method: restaurantUI()

- Description: Presents an interface that allows the user to view restaurants.
- Preconditions: The restaurant must already be in the database, must have an internet connection.
- Postconditions: information and options about a specific restaurant.
- Signature: void restaurantUI()

11) mapInterface:

a) Method: displayMap()

- Description: Displays a map of the users current location
- Preconditions: must be connected to internet, user location is accessible,
- Postconditions: The map is displayed
- Signature: void displayMap();

b) 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();

Alternates:

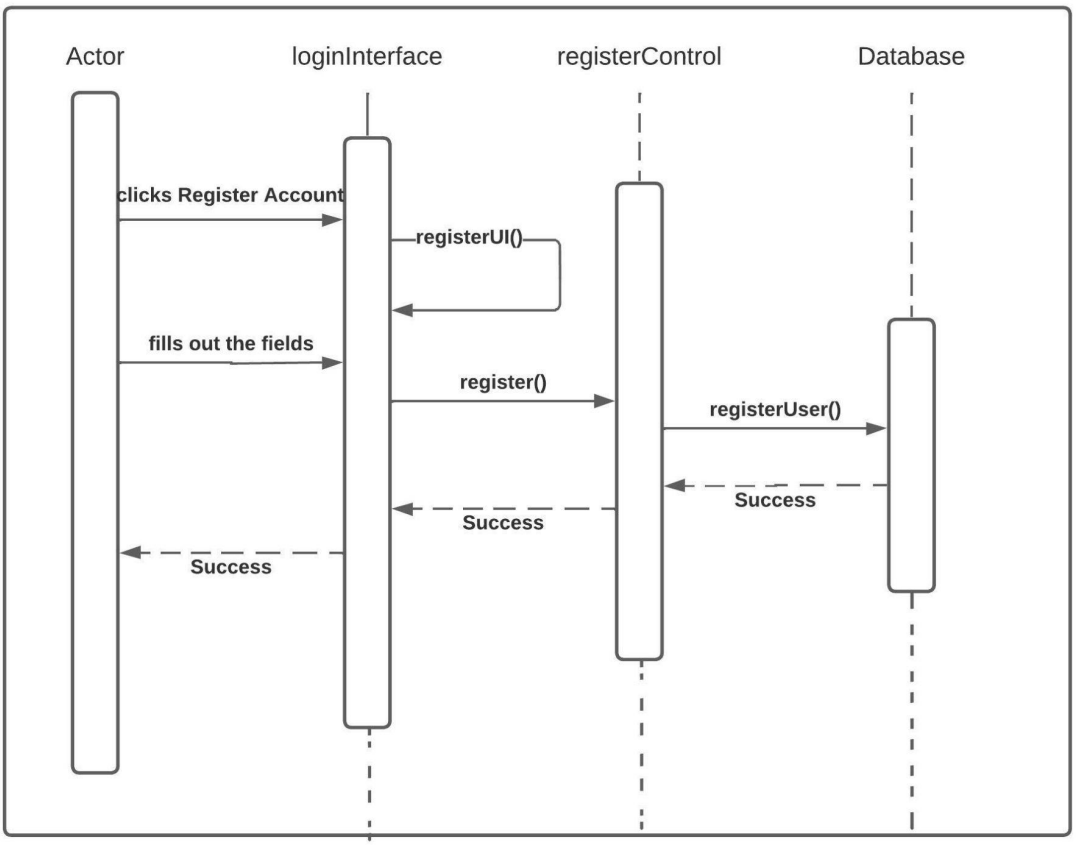
Flutter: We are using Flutter for both frontend and backend because we want to build a mobile app for both Androids and Apple users which Flutter allows us to do.

Firebase: We are using the Firebase database because it is a highly compatible database for Flutter.

Sequence Diagram:

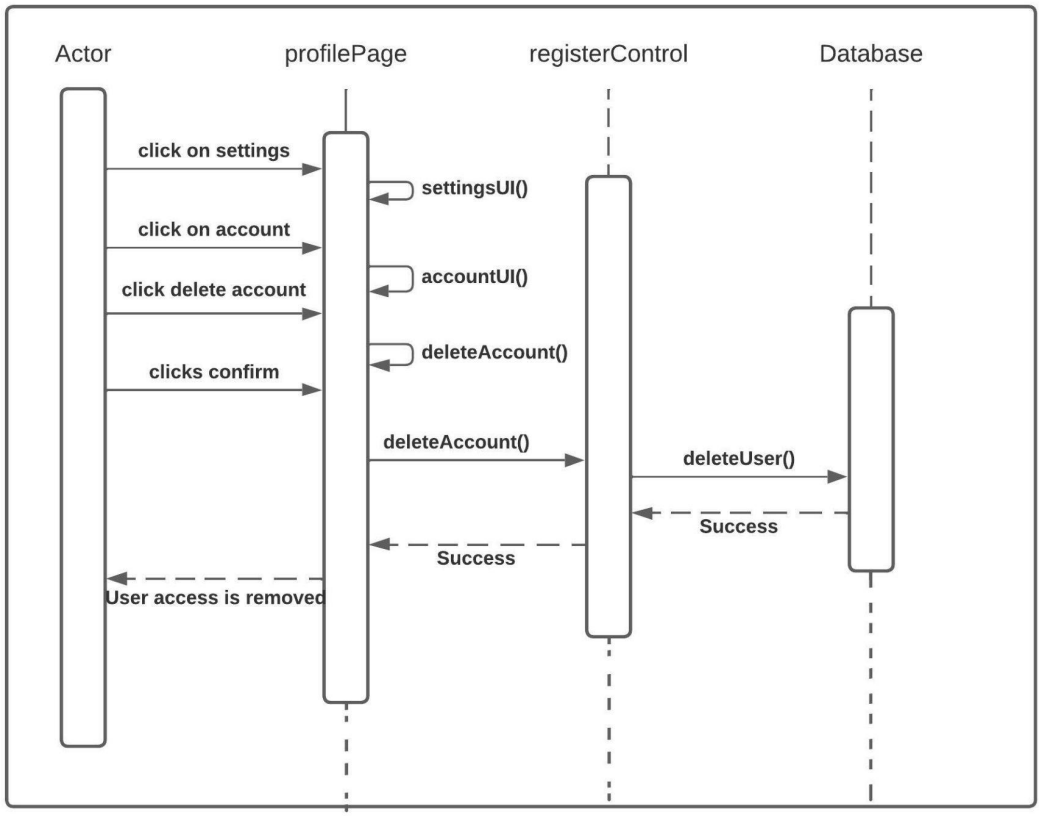
- https://lucid.app/lucidchart/87ff4981-acf2-483b-ae10-7300d81f532d/edit?page=0_0&invitationId=inv_2bc49f12-c1f3-4f8a-abfc-e0f9089df3fd#

Use Case 1: Register Account

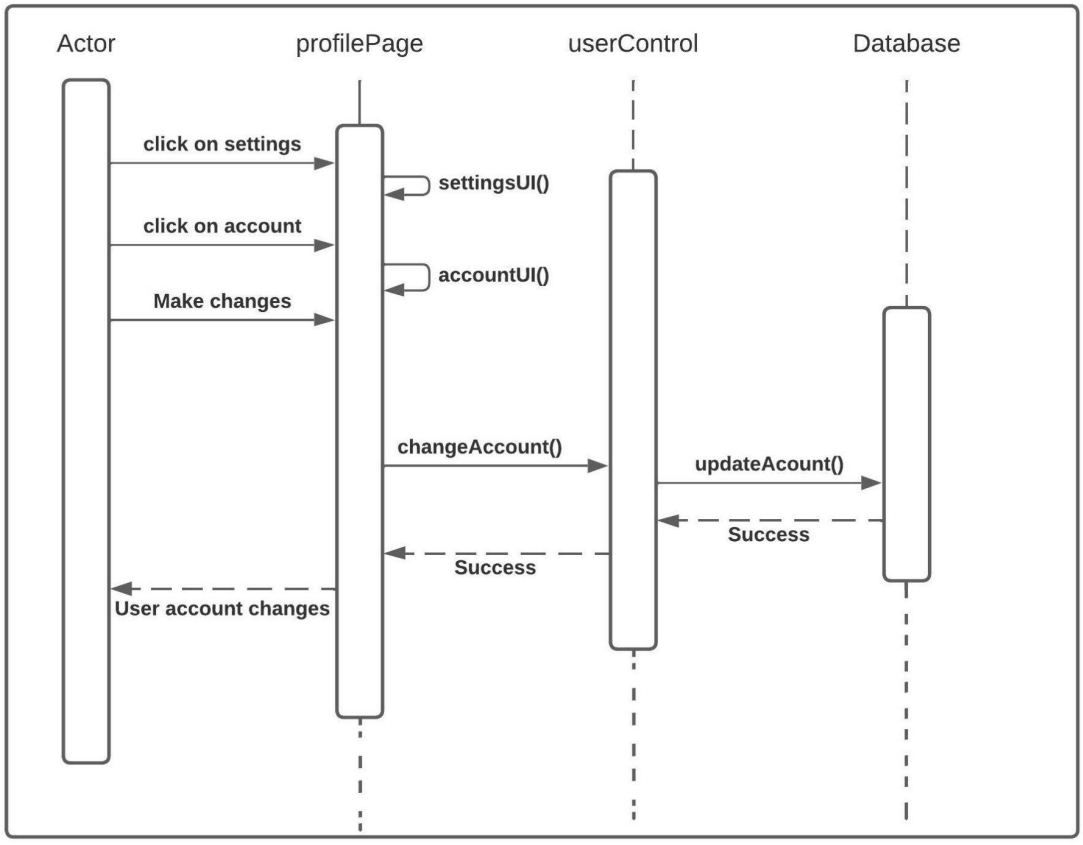


registerUI()
reguster()
registerUser()

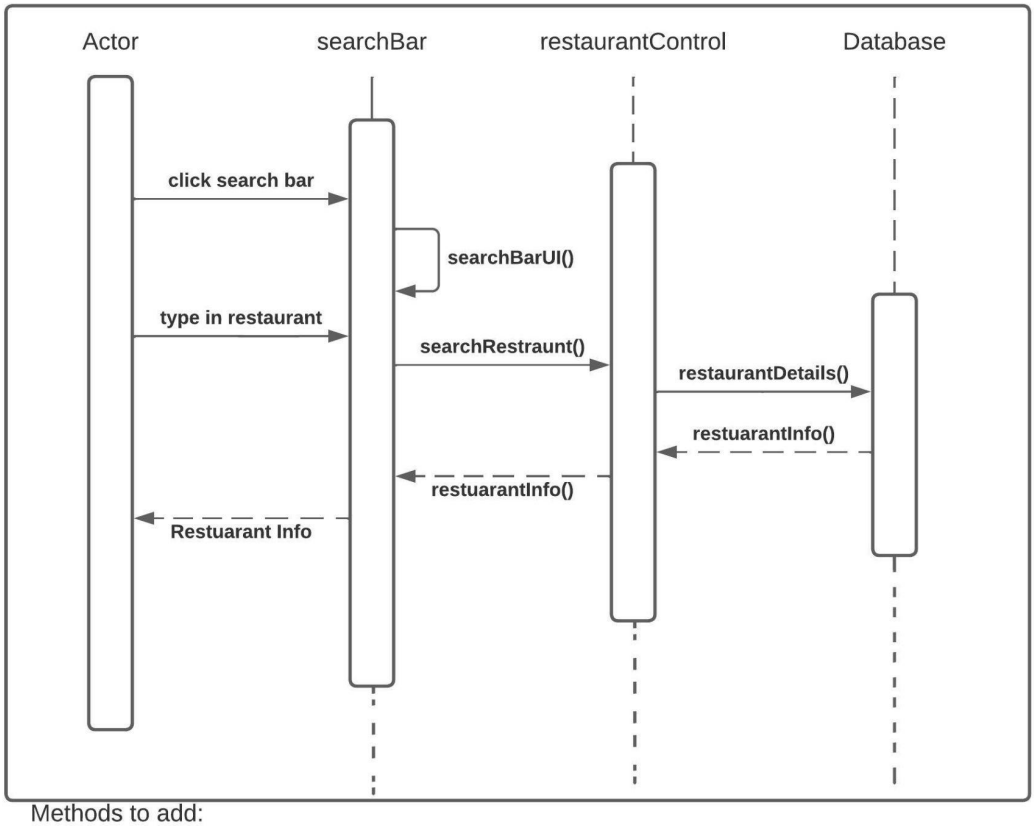
Use Case 2: Delete Account



Use Case 3: Update User information

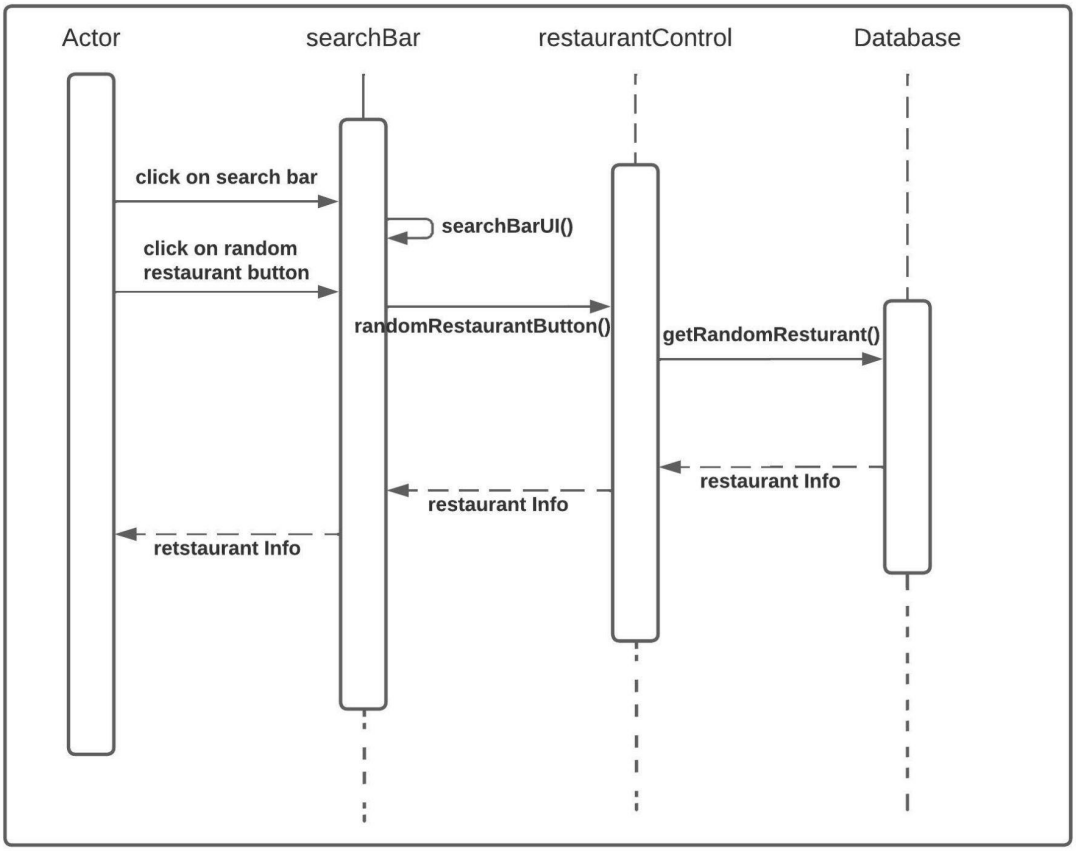


Use Case 4: Search for Restaurant



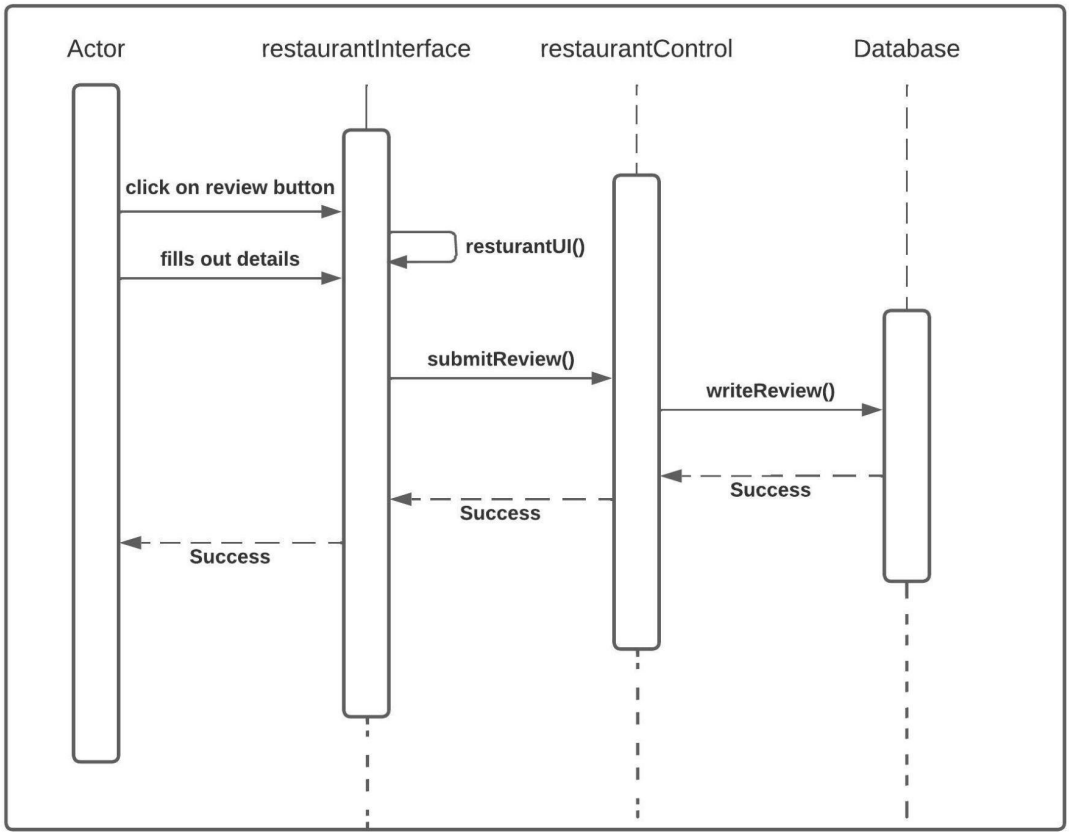
Methods to add:

Use Case 5: Randomize Restaurant



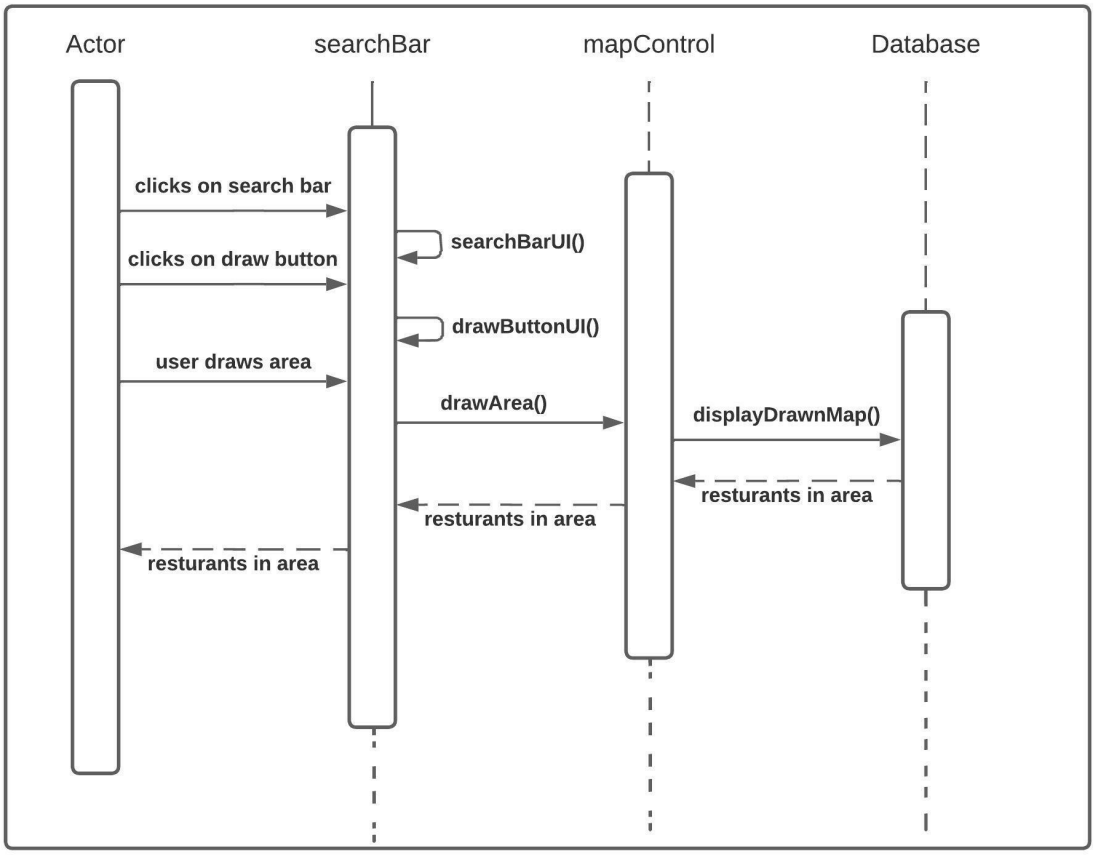
Methods to add:

Use Case 6: Review of Restaurant



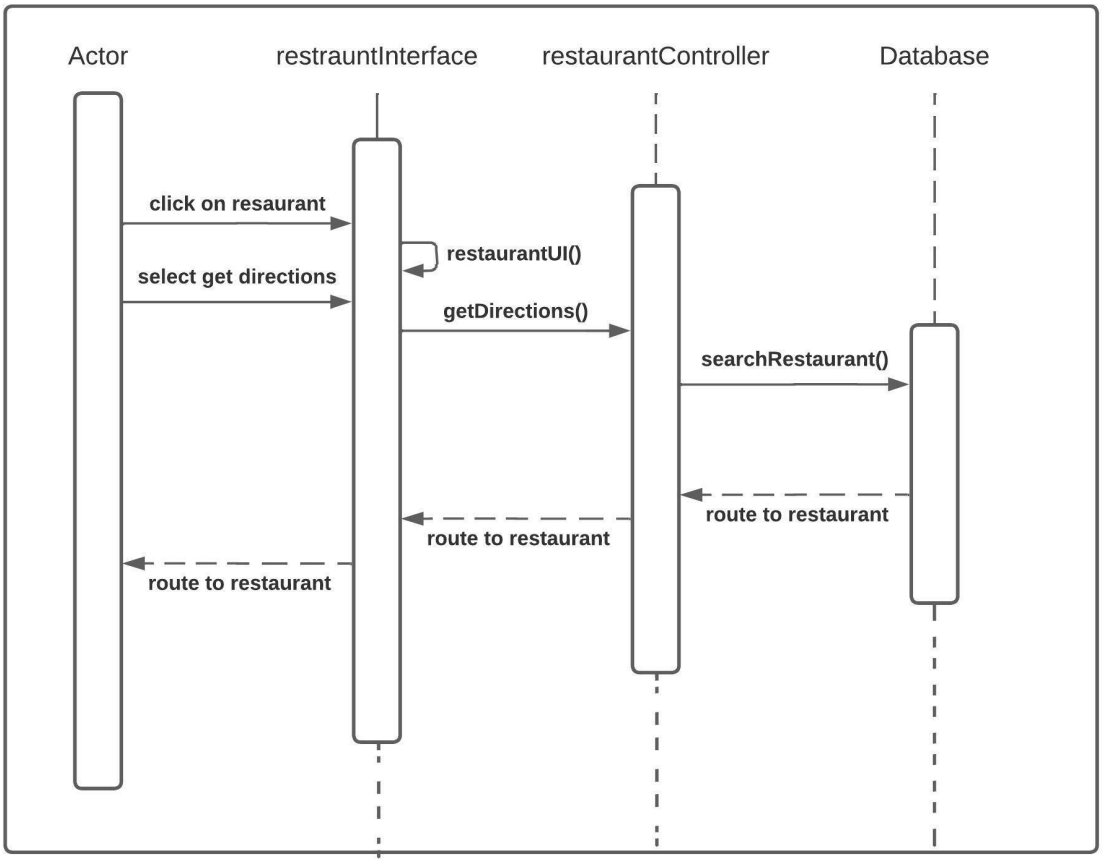
Methods to add:

Use Case 8: Draw Search Area



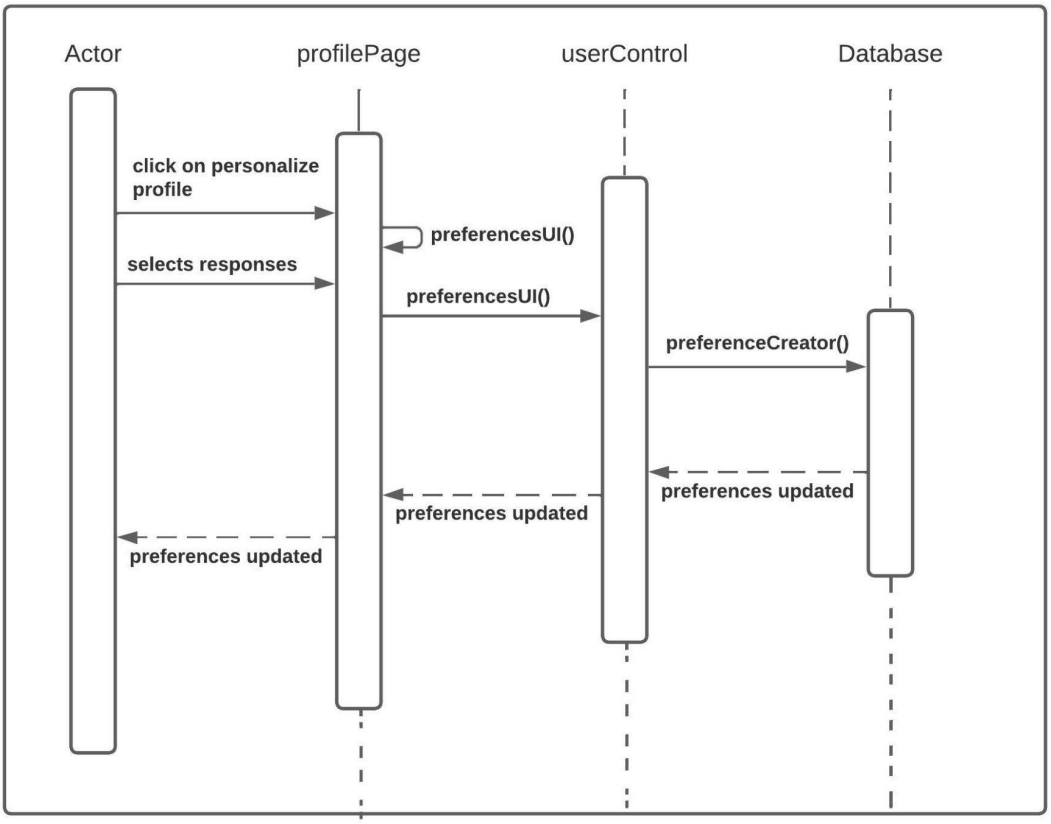
Methods to add:

Use Case 9: Find route



Methods to add:

Use Case 10: Profile Preferences



Methods to add: