Harihar Subramanyam
Tawanda Zimuto
Ryan Chipman
Danielle Man

11 November, 2014
6.170 Final Project
Design Document

menyou

the menu for you

Overview

Motivation

Introduction

Menyou was created out of an interest in improving the current dining experience for consumers. There are many different platforms out there that help a user decide where to eat, facilitate the discovery of new venues, and aggregate reviews and ratings for comparison purposes. The problem is that sometimes users don't actually know what to eat. Popular apps for this task today, namely Yelp and Foursquare, simply provide a long list of restaurants and leave the user to decide which one to go to - they pay no attention to the user's taste in food or to the menus at each restaurant. Menyou is different from these apps in that it recommends a meal for you, not a restaurant. Menyou analyzes a user's taste profile while simultaneously reading all of the menus that are close to that user, thus providing the user with the best possible meal to suit his or her own preferences.

Purpose

At its core, the purpose of menyou is:

1) Recommend meals the user will enjoy
    a) That is, we want to suggest dishes at restaurants which the user will enjoy eating based on their unique taste.

However, let us add some refinements to this purpose in order to flesh it out and make our concept development easier. These refinements are:

2) Do not violate user's dietary restrictions
    a) We aim to avoid recommending foods that the user does not like to eat or cannot eat. For instance, if the user is a vegetarian, we do not want to recommend a hamburger. If the user is allergic to dairy, we do not want to recommend them a milkshake. If the user hates eggplant, then we won't recommend the eggplant parmesan. Et cetera.
3) Only recommend meals near the user
    a) We will search for restaurants near (within 10s of miles) the user, because these are likely to be the ones that the user will actually go to.

With that, let us consider the concepts.

Context Diagram

Since interaction with many external systems is complicated and fragile, we keep menyou's context simple. The context diagram is shown below:

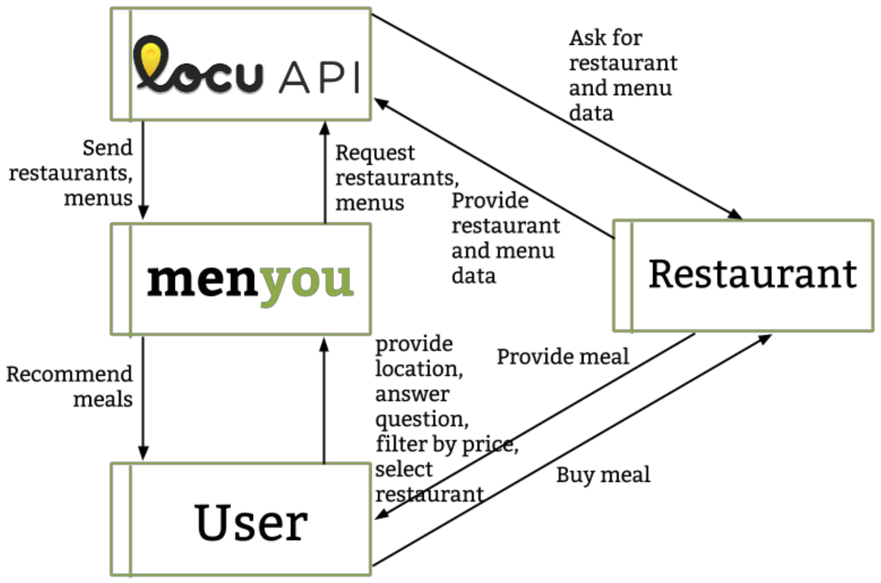

Locu API
Ask for restaurant and menu data
Send restaurants, menus
Request restaurants, menus
Provide restaurant and menu data
menyou
Restaurant
Recommend meals
provide location, answer question, filter by price, select restaurant
Provide meal
Buy meal
User

Figure 1: Context diagram

In order to keep the diagram focused and clear, we have omitted interactions and systems that are inconsequential (ex. the user and restaurant may process payments with the help of a bank, but including the bank would add complexity to the diagram without providing us with useful insights to build menyou).

There four systems of interest here. Ultimately, we (menyou) care about the interaction between the user and the restaurant, because the user wants a meal and the restaurant provides meals. However, we do not have enough time to meet and interact with the restaurants directly. Therefore, we acquire restaurant and menu information from the Locu API. Thus, menyou asks for restaurants/meal data and Locu provides it. The user provides their taste to menyou (by answering questions which menyou uses to build a taste profile). Then, menyou recommends meals to the user.

Note that we do not control the interaction between the user and restaurant, nor do we control the interaction between the restaurant and Locu. This is not ideal. For instance, if some restaurants are inadequately represented (or not represented at all) on Locu, we will not know. In addition, we cannot observe the user at the restaurant and gauge their enjoyment of the meal. However, given our time limit and resources, trusting Locu and using questions to determine a user's taste profile is a reasonable decision.

Design

Concepts

We shall take a top down approach in discussing the concepts - we begin with the most important concepts first.

meal recommendation - This concept is ultimately what achieves menyou's purpose. It consists of a meal and the restaurant that serves the meal (ex. "Try vegetarian pad thai at Pepper Sky's").

meal - A dish served at a restaurant. It consists of a name, a description (ex. "Char-grilled chicken served shish-kebob style"), and a price. Since the purpose of the app is to recommend meals, we need this concept to define what meal actually is.

menu - A list of meals. The Locu API provides us with menus, from which we extract meals in order to recommend them. Our purpose is to recommend meals, and menus are the containers for meals.

restaurant - A venue that serves food. It consists of a name, an address, and a geographic location. Our purpose requires us to recommend meals **near the user**, so we introduce this concept to ensure that recommended meals are at nearby restaurants.

taste profile - Our purpose is recommend meals that the user will **enjoy** while respecting their **dietary restrictions**. Therefore, we introduce the concept of a taste profile, which captures the following aspects of a user's taste:

1) likes - A list of strings reflecting aspects of meals that a user likes (ex. potato, eggplant, spicy)
2) dislikes - A list of strings reflecting aspects of meals that a user dislikes.
3) forbidden - A list of strings reflecting aspects of meals that a user will NOT tolerate (ex. if the user is vegetarian, then "beef" would be forbidden)

The taste profile captures a user's taste. Thus, it can help us identify meals that the user will enjoy, which means helps us meet our purpose of recommending meals.

question - An interrogative statement posed to the user to help build their taste profile. For instance, "Do you like spicy food?" will determine whether to add "spicy" to the the list of likes or to the list of dislikes on a user's taste profile. We utilize this concept to improve the taste profile as the user answers questions. This way, we can better accomplish our purpose of recommending meals that the user will **enjoy**.

Recommendation

In order to understand the data model for menyou, we must first understand the recommendation algorithm. While recommendation is typically done using machine learning techniques such as collaborative filtering, we will not be adopting that approach because setting up and tuning such an algorithm would be too time consuming (and we don't have a great deal of time). Therefore, we shall adopt a simpler approach that involves filtering out forbidden foods, and assigning (or subtracting) points to liked and disliked foods.

Input

The input to our algorithm will be a **menu** (a list of meals) and a **taste profile** (the list of a user's tastes).

Denote the menu by M. Let M[i] be the ith meal on the menu. Let M[i].name, M[i].description, M[i].price, M[i].restaurant, M[i].address, and M[i].location be the name, description, price, restaurant name, restaurant address, and restaurant latitude/longitude of the ith meal on the menu.

Denote the taste profile by T. Let T.likes, T.dislikes, and T.forbidden be the lists of strings. For instance, if a user likes eggplant, then “eggplant” would be an element in T.likes. If a user absolutely will not eat dairy, then “milk”, “yogurt”, and “cheese” would be elements of T.forbidden.

As a demonstrative example, consider the following values for M and T.

```
M = [{
        “price”: 10,
        “name”: “Eggplant parmesan”,
        “restaurant”: “Antico Forno”,
        “address”: “123 Something Way, Somewhere, AK, 12345”
        “location”: [42.12312, -71.2342],
        “description”: “Breaded fried eggplant with marinara sauce and parmesan cheese, sprinkled with
        basil”
},
{
        “price”: 5,
        “name”: “Spring rolls”,
        “restaurant”: “Thai 123”,
        “address”: “125 Something Way, Somewhere, AK, 12345”
        “location”: [42.312, -71.22342],
        “description”: “Fried pastry rolls filled with cabbage and carrot. Served with
        hoisin dipping sauce”
}]

T = {
        “likes”: [“eggplant”, “cabbage”, “carrot”, “pastry”],
        “dislikes”: [“hoisin”],
        “forbidden”: [“cheese”, “milk”, “yogurt”]
}
```

Output

The output of the recommendation algorithm will be a list of recommended dishes, call it R. The list R will be a subset of M, but each meal will be augmented with an attribute “points”, where the higher the points are, the more highly recommended the meal is.

Algorithm

The pseudocode for the algorithm is described below:

```
1) Let R be a new list
2) For each meal, m, in M
    a) Let points = 0
    b) For each forbidden, f, in T.forbidden
        i) If (m.description + m.name) contains f (case insensitive)
            1) skip to the next iteration of the for loop on line 2
    c) For each like, e, in T.likes
        i) If (m.description + m.name) contains e (case insensitive)
            1) points = points + 1
        d) For each dislike, d, in T.dislikes
            i) If (m.description + m.name) contains d (case insensitive)
                1) points = points - 1
        e) m.points = points
        f) Add m to R
    3) Return R
```

The algorithm is quite straightforward. It simply searches in m’s name and description for each of the strings in T.likes, T.dislikes, and T.forbidden. If it finds a forbidden keyword, then it skips m and does not add m to the list R. For each keyword in T.likes that is found in m’s name or description, points is incremented. Similarly, for each keyword in T.dislikes that is found in m’s name or description, points is decremented. Finally, m is augmented with the “points” attribute and added to R. After all the meals in M are processed this way, the list R is returned.

Running this algorithm on the example input above, we would get:

```
R = [{
        “price”: 5,
        “name”: “Spring rolls”,
        “restaurant”: “Thai 123”,
        “address”: “125 Something Way, Somewhere, AK, 12345”
        “location”: [42.312, -71.22342],
        “description”: “Fried pastry rolls filled with cabbage and carrot. Served with
        hoisin dipping sauce”
        “points”: 2
}]
```

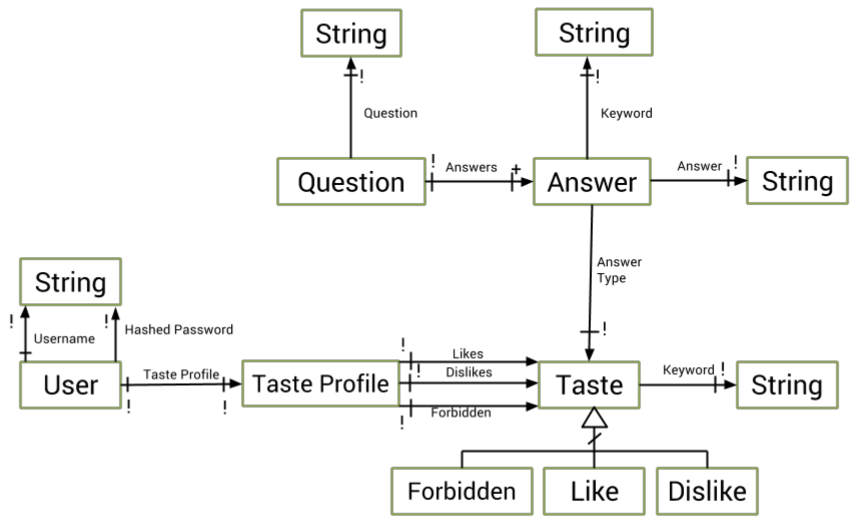
Notice that eggplant parmesan is not include in R because the taste profile indicates that “cheese” is forbidden. The spring rolls get 3 points for “cabbage”, “carrot”, and “pastry”, but they lose 1 point for having “hoisin” - the net total is 2 points.

Notice that the price is not used in this algorithm, because filtering by price is handled on the client side (i.e. the user selects a price range and client hides the recommendations which fall outside the price range). Similarly, the address, restaurant, and location are not used in the algorithm, but are passed to the client for displaying the restaurant on a map.

Now that we understand the algorithm, let us turn our attention to the data model.

Data Model

The data model is displayed below:



A Taste Profile's "Dislikes" attribute must be a set of "Dislike" entities.
A Taste Profile's "Likes" attribute must be a set of "Like" entities.
A Taste Profile's "Dislikes" attribute must be a set of "Dislike" entities.
All usernames are distinct.

Figure 2: Data Model

The data model can be overwhelming, so we will discuss each part.

First, we have the User entity set. Each user must have exactly one username (a string), and that username should not change (since it uniquely identifies the user). A User has exactly one password (a string), but the user may change that password. Each user also is associated with exactly one Taste Profile, which records the user's tastes. Since it does not make sense for a user to be assigned a new Taste Profile, the Taste Profile is immutable (however the attributes within a Taste Profile are mutable).

A Taste Profile entity has three attributes - a set of likes (which must contain only Like entities), a set of dislikes (which must contain only Dislike entities), and a set of forbiddens (which must contain only Forbidden entities). These sets need not exist (ex. suppose a user has no dietary restrictions, so there are no forbidden foods), and they can change as the user's taste chances.

Like, Dislike, and Forbidden are each subsets of an entity set called Taste. Each Taste is associated with a single keyword indicating the taste (ex. eggplant, spicy). Recall that our recommendation algorithm searches for keywords in the description of foods. The keyword cannot change. Each Taste must be associated with exactly one user profile, which cannot change. Note that if two two users both like eggplant, they will have separate taste entities. Note that if a user no longer has a particular taste, the taste entity is deleted.

Next, we have the Question entity set. A Question has a Question attribute (i.e. the text content of the question, such as "Do you like spicy foods?"). The Question also has an immutable set of one or more Answer entities (and each Answer is immutably linked to exactly one Question).

Finally, we have the Answer entity set. An Answer represents a possible response to a Question. An Answer entity has an Answer attribute (i.e. the text content of the answer, such as “Yes, I love spicy foods!”, “No, I can’t stand spicy food at all!”, or “I don’t particularly like spicy food”). An Answer entity also has an Answer Type and Keyword, which indicate the part of the taste profile to append to and the keyword to append - this is useful for building a user’s taste profile based on the answer to a question. For instance, if Answer Type is “Likes” and Keyword is “spicy”, then if the answer is selected, the keyword “spicy” should be appended to the “likes” list of the user’s taste profile.

Behavior

Security Concerns

Security Requirements

- An authenticated user can only edit his/her own profile details & taste profile.
- Users cannot see any details about other users, including names, emails, and taste profiles.

Potential Risks

- Hackers who break into the system to extract personal information
- Spam advertisers gaining access and inserting incorrect/irrelevant information

Threat Model

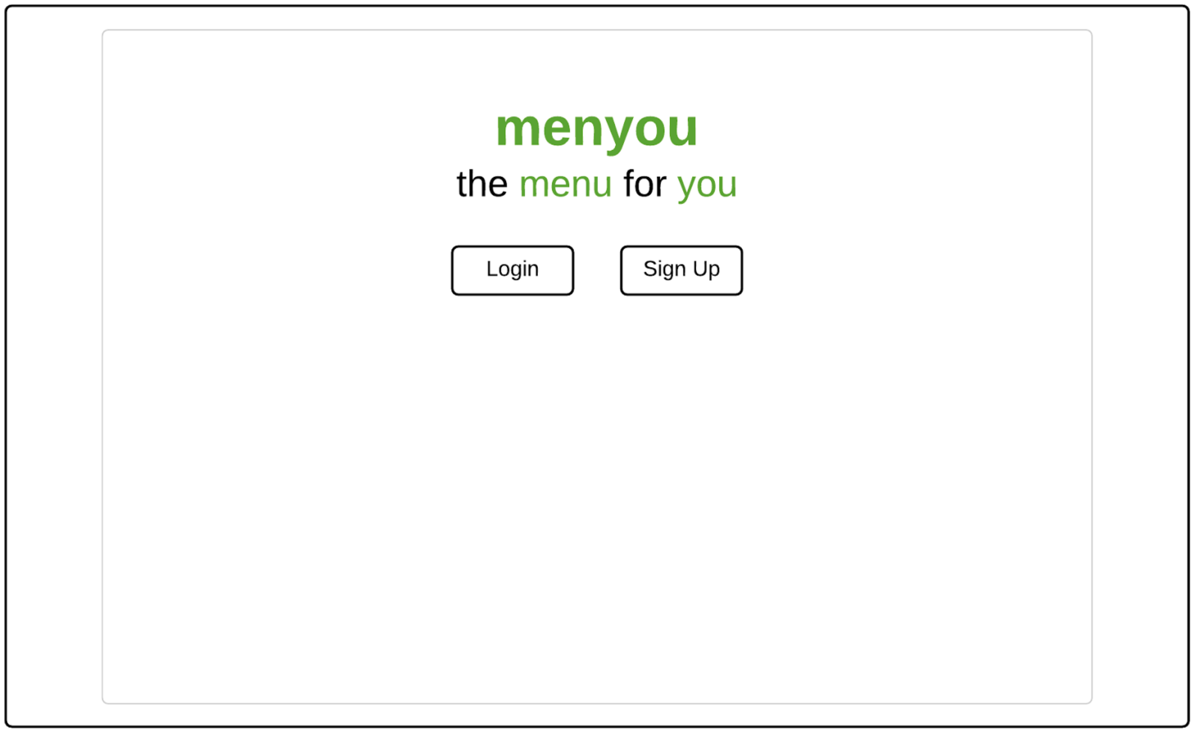
- We can assume that there is minimal interest for hackers retrieving profile information, because there are much easier ways to aggregate name/email information than hacking into menyou.
- Menyou involves no interaction between users, so there is no risk of manipulation/fraud between users within the system.
- There is no payment component to Menyou, so we don’t need to worry about that getting hacked.
- Spammers are the most likely of our threats.

Mitigations

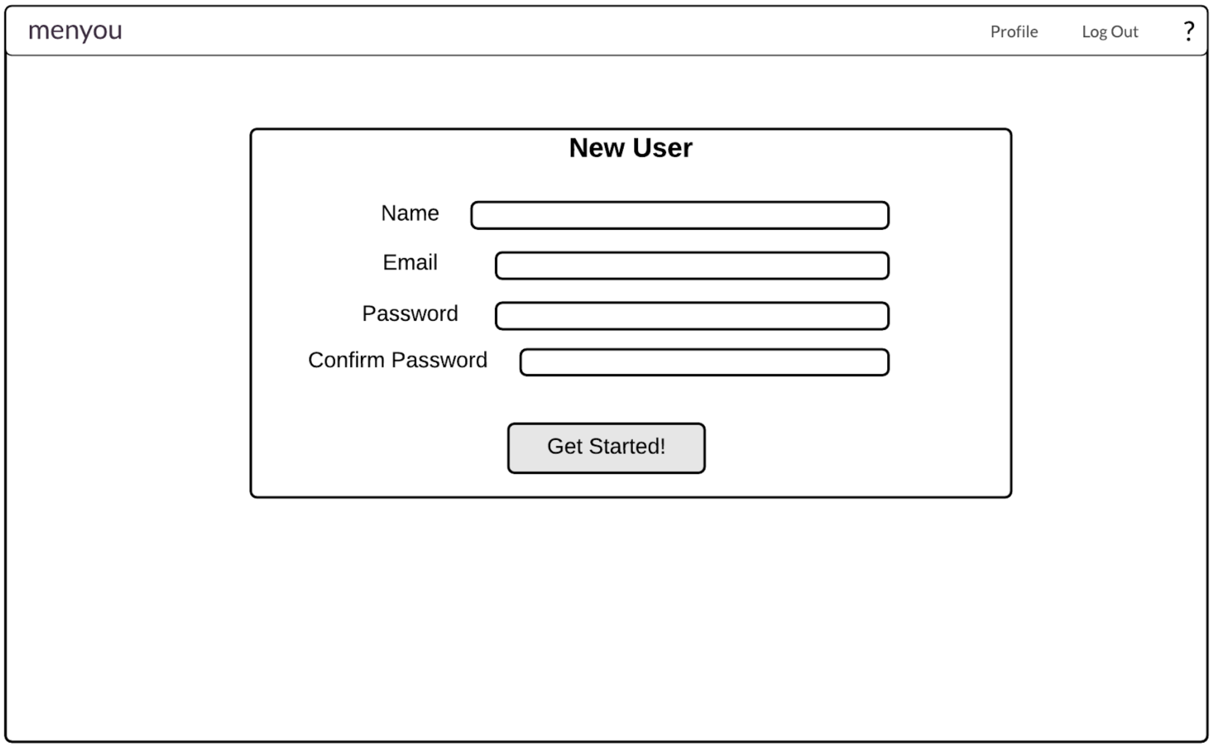
- Profile Information
    - Properly implement access control
    - Require access token with every request. Don’t use cookies to prevent session hijacking.
- Spammers
    - Trust Locu to have accurately maintained data
    - Allow users to ‘report’ inaccurate/spam entries
    - Standard countermeasures against injection, XSRF, etc.

User Interface

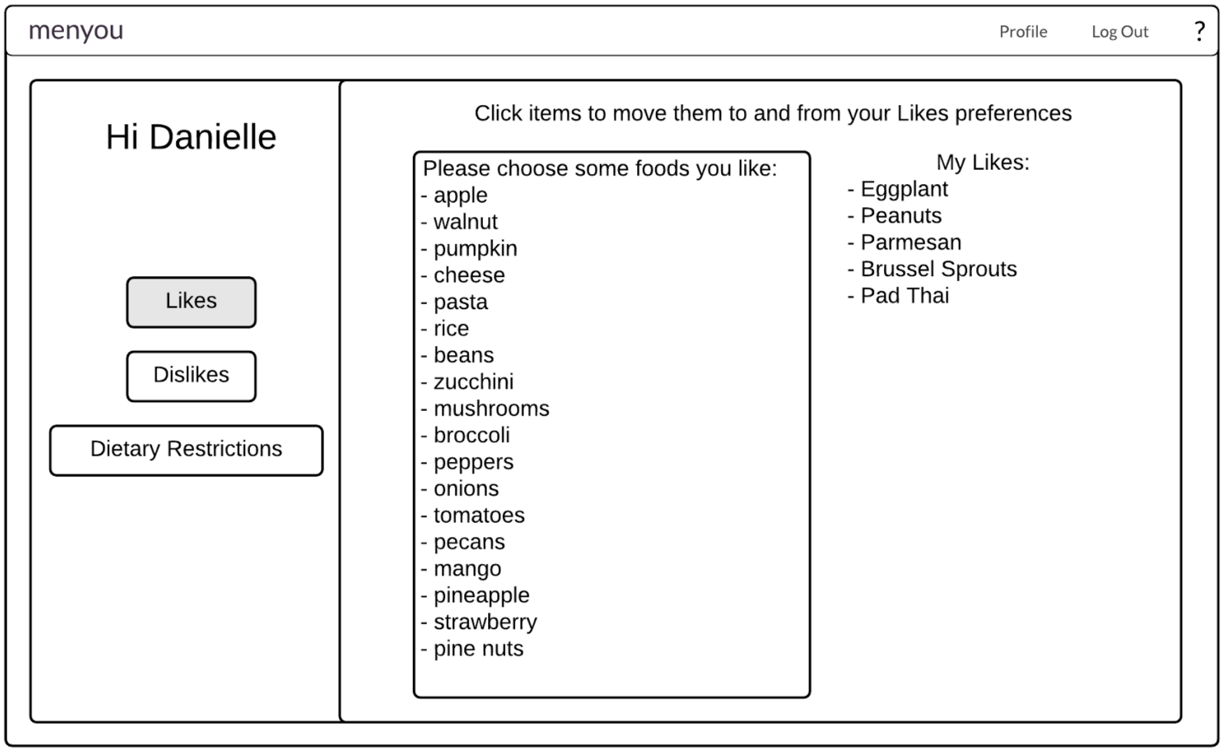
User not logged in:



User Sign Up page:



User Profile page:



menyou | Profile | Log Out | ?

Hi Danielle

Likes

Dislikes

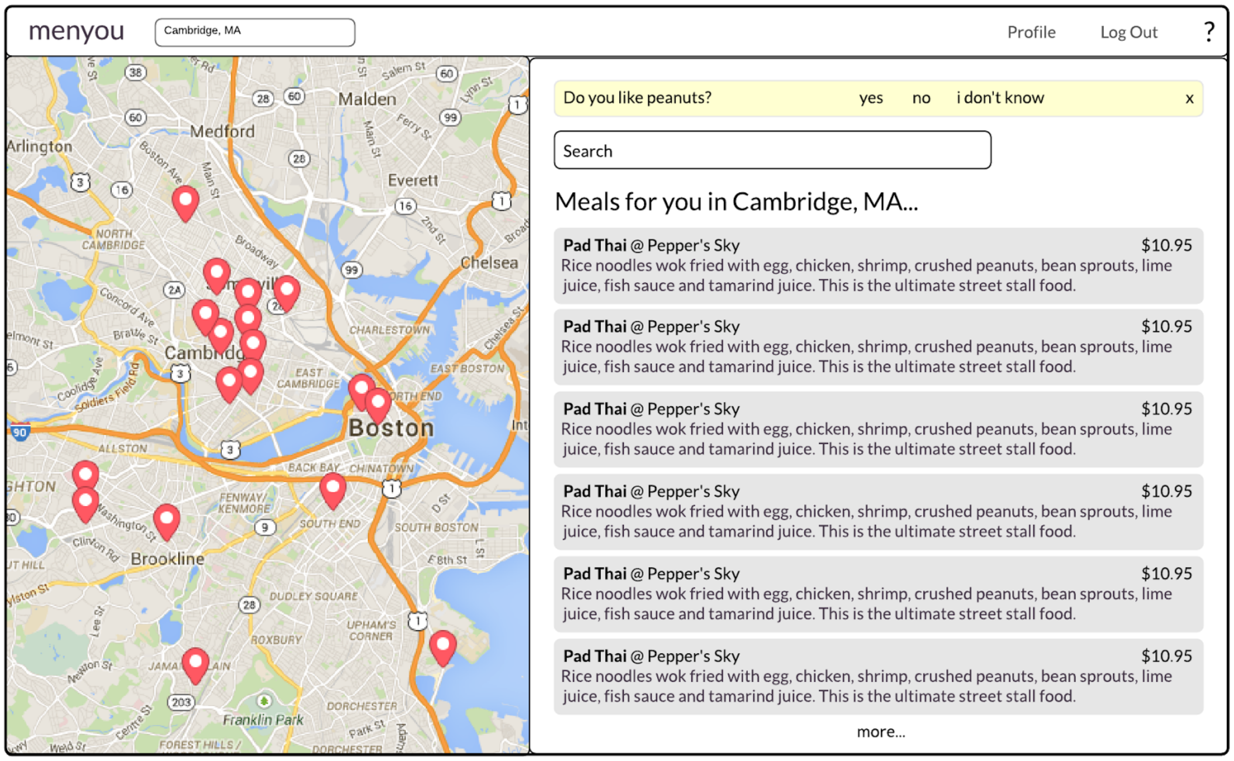
Dietary Restrictions

Click items to move them to and from your Likes preferences

Please choose some foods you like:
- apple
- walnut
- pumpkin
- cheese
- pasta
- rice
- beans
- zucchini
- mushrooms
- broccoli
- peppers
- onions
- tomatoes
- pecans
- mango
- pineapple
- strawberry
- pine nuts

My Likes:
- Eggplant
- Peanuts
- Parmesan
- Brussel Sprouts
- Pad Thai

User Home page:



menyou | Cambridge, MA | Profile | Log Out | ?

Do you like peanuts? yes no i don't know x

Search

Meals for you in Cambridge, MA...

Pad Thai @ Pepper's Sky $10.95
Rice noodles wok fried with egg, chicken, shrimp, crushed peanuts, bean sprouts, lime juice, fish sauce and tamarind juice. This is the ultimate street stall food.

Pad Thai @ Pepper's Sky $10.95
Rice noodles wok fried with egg, chicken, shrimp, crushed peanuts, bean sprouts, lime juice, fish sauce and tamarind juice. This is the ultimate street stall food.

Pad Thai @ Pepper's Sky $10.95
Rice noodles wok fried with egg, chicken, shrimp, crushed peanuts, bean sprouts, lime juice, fish sauce and tamarind juice. This is the ultimate street stall food.

Pad Thai @ Pepper's Sky $10.95
Rice noodles wok fried with egg, chicken, shrimp, crushed peanuts, bean sprouts, lime juice, fish sauce and tamarind juice. This is the ultimate street stall food.

Pad Thai @ Pepper's Sky $10.95
Rice noodles wok fried with egg, chicken, shrimp, crushed peanuts, bean sprouts, lime juice, fish sauce and tamarind juice. This is the ultimate street stall food.

Pad Thai @ Pepper's Sky $10.95
Rice noodles wok fried with egg, chicken, shrimp, crushed peanuts, bean sprouts, lime juice, fish sauce and tamarind juice. This is the ultimate street stall food.

more...

Challenges

Design Challenges

Design & Concepts

Taste Profile Refinement

Problem Description

Our taste profile will by no means be perfect after the user's initial specification of their likes/dislikes. In order to provide consistently satisfying recommendations, menyou needs to incorporate a feedback loop for improving the quality of the recommendations by augmenting the taste profile.

Potential Solutions

- Dedicated Profile Editor
- Infer Preferences
- Questions

Our Decision

We decided to include both the first and third options in our design. The fine-grained control provided by a dedicated profile editor is useful and worth including, but having an incremental way to add to the taste profile (ex. by asking the user a simple question visible to them while they use the app) we are able to get data to improve a user's taste profile more often.

API & Backend Implementation

Authentication

Problem Description

Our authentication scheme needs to satisfy two key criteria. First, since we are providing menyou as an API backend, we need the authentication scheme to be flexible and adaptable to many kinds of clients. Second, it needs to provide adequate protection against the security risks outlined earlier in the document.

Potential Solutions

- OAuth
- API Tokens
- Username/Password + Cookies

Our Decision

We decided to implement authentication using the second of the three schemes. Except for the new-user route (which requires no authentication) and the login route (which takes username/passwd in the request body), every API call expects a bearer-token to be provided with the Authentication request header. OAuth was overkill for our particular use-case, but a token-based implementation provides major conveniences in the form of cross-platform compatibility and easy session invalidation (invalidating a user's sessions requires only generating a new API token). Cookies and other such conveniences will be implemented client-side; the backend will use the same implementation regardless of client platform.

Scoring Meals

Problem Description

In order to provide menyou's users with recommendations, we need to have some method of ranking meals. Given a user and a meal, we need to produce a nonnegative integer that represents how strongly we recommend that meal for that given user. Distilling such subjective factors as food preferences into a ingle integer requires some careful consideration about what makes a good recommendation.

Potential Solutions

- Machine Learning (ex. collaborative filtering)
- Recommendations from other users
- Point/filter based recommendation algorithm

Our Decision

We decided to go with the third option. Machine learning algorithms would be complicated to set up and integrate in the app given our limited time, so we chose to avoid them. Getting recommendations from other users would also require a good deal of code to be implemented to handle building a relationship between users and a way for users to recommend food to each other. However, recommendations from other users may not be very useful, because people have very different tastes, and a recommendation from somebody else may not fit your tastes just right.

UX & Frontend Implementation

Questions

Problem Description

As mentioned in the 'Design & Concepts' section above, we refine a user's taste profile by asking them small, simple questions. In order for this method to be successful, people need to actually answer the questions. To make this happen, it is essential that our front-end implementation of these questions strikes a balance between prominence (so that a user actually sees the questions we want them to answer) and unobtrusiveness (so that we don't annoy the user and cause them to perceive our questions as 'spammy').

Potential Solutions

Modals/Pop-Ups

In list of menu items

Our Decision

We shall display the small, simple questions at the bottom of the menu list. This way, we do not interrupt the user as we would with a modal/pop-up, and we ensure that the question is always onscreen so that the user can respond to it.

Focusing on Meals

Problem Description

One of menyou's key goals is to change a user's dining decision process so that the focus falls on choosing a good meal (rather than choosing a restaurant). That being said, information about the restaurant is still important and needs to be provided to the user, even though our *primary* focus may be meals. Our challenge is to find a way to make all the information easily accessible to the user without allowing the secondary/extraneous information to overshadow the individual meals.

Potential Solutions

- Full-Page Map
- Full-Page Meals List
- Meals by Restaurant
- Meal List + Restaurant Map

Our Decision

We will be implementing the third of the three options discussed above. It is the only of the three that actually allows the user to see his best meal options without obscuring the secondary (but still important) information like location. Furthermore, it provides that additional information in a simple, easily digestible visual format.