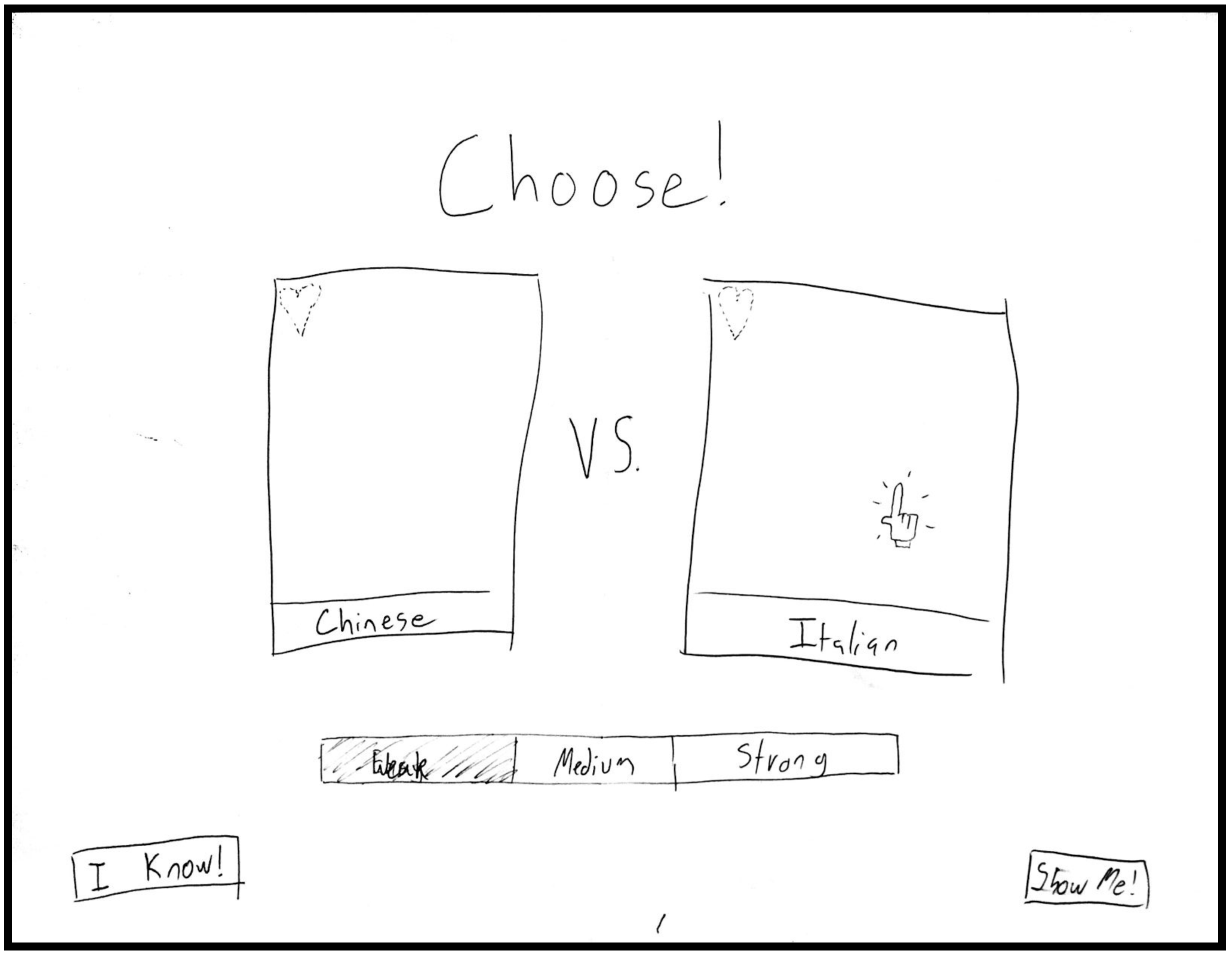
# It’s Like 20 Questions But For Your Food (Working Title)

## *Product Description*

The product is a restaurant recommendation engine that uses a cuisine chooser and a ratings filter to generate user specific output. When the user first opens the web application, he is greeted with a small series of questions that will be used to filter the restaurant set. The user can toggle settings include max distance and price to ensure all recommended restaurants are feasible. Upon completion, the user is moved into the decision-making tool, which seeks to help the user understand what type of food he is craving at that time. The user will be presented with two cuisine options (i.e. Italian and Japanese) and asked to choose the option he would rather have. Upon making a choice, the user would be prompted with another two cuisines and could continue to make choices for as long as he’d like. After the user has finished making comparisons, they would be prompted a final time to rank the importance of the three components of any restaurant: food, service, and ambiance. From the user’s choices in the cuisine matchups, the application would then generate a list of restaurants. The list would then be ranked by an aggregation of each restaurant's ratings across the 3 major restaurant review sites (Yelp, Google, TripAdvisor) but reviews would be weighted depending on how the user valued the three aforementioned restaurant components. If the user double-clicked into one of the restaurant options, it would deep link him out to the corresponding Yelp page so that he could utilize the functionality that Yelp offers (call the restaurant, see its hours, make a reservation, etc.)

The functionality previously described can also be utilized to help a group come to a decision about where to go and eat. When the user first arrives at the web app, they could choose to create a group URL and then send it to any number of friends who would be eating with them. This unique URL will bring each person to the decision-making tool where they could also select their cuisine preferences for as long as they would like. Then, the algorithm would factor in all of the group’s preferences when generating a ranked list of restaurants that best fit the group.

After a user’s first use of the product, they would be asked to create a profile that would save his restaurant preferences. Then, if he used the product in the future, the engine would use the user’s past information to better inform its recommendations. For example, if after a few uses, it became clear that the user strongly preferred Japanese food, the decision-making tool could present Japanese cuisine earlier on in comparisons to determine if the user was, once again, wanting sushi. Additionally, it could employ the user’s restaurant information to send them recommendations offline. If a significant traffic was going to a Japanese place that had recently opened, and therefore had a low number of reviews, the engine could send a note to all of the users in that area who had previously scored Japanese cuisine highly.



***Product Need***

The engine is built on the premise that two major parts of the restaurant choosing process are not customized for the user. First, it has been shown it is extremely difficult for humans to evaluate more than three choices at one time. The mind is significantly more successful at comparing two to three choices and choosing which is best and, in fact, making a choice from a set larger than three is really a series of these choices among the possible pairs in the set. But, in this series of pair comparisons, the quantity and speed of data is overwhelming and humans often do not make the optimal choice. The product seeks to improve that decision-making by only presenting the user with two choices. By recording the results of this smaller decision, theoretically, less mistakes would be made and the user would be more satisfied with his eventual choice.

On top of improving decision-making, we believe that the product can be valuable because it solves the second flaw of restaurant evaluation: ratings systems. These systems are the average opinion of the masses, not based on people similar to the user. A 3-star restaurant on Yelp maybe have a lower rating because the service is notoriously bad. But if you're someone who only prioritizes the quality of the food, this may not matter to you and you would give it a higher rating. The ratings compiler component of the product allows users to get this level of customization, providing a ranking that is more relevant to them than the current “Best Match” algorithms on ratings websites. Overall, by empowering users, and groups of users, to both better understand their restaurant desires and see restaurants that best fit their desires, we believe the product will streamline a time-consuming process and lead to higher satisfaction scores in the restaurant experience.

### ***Audience***

In theory, the product could be useful for any person, or group of people, who has struggled when deciding where to eat. That being said, only a subset of that group would actually spend the time on an application designed to help them with that decision as opposed to choosing at random. In 2016, Rasmussen reported that 58% of American adults (140.4M people total) ate at a restaurant once a week. Thus, if we conservatively estimate that 50% of these users have, at some point, had trouble deciding where to eat, the total possible market would be around 70.2 M people.

While the 70 million number was a conservative estimate, perhaps a more accurate estimate could come from the people who have actually demonstrated that they are willing to put in effort when choosing a restaurant. A good proxy for that population could come from Yelp’s user base. In 2016, Yelp reported 96.7M American unique users a month. But, considering that only 19% of listings on Yelp are for restaurants, we can conservatively assume that only 19% of these users went to Yelp for its restaurants (18.4M users). To be clear, this number for the total market is still conservative because even though only 19% of listings are restaurants, more than 19% of users use Yelp to look at restaurants. Additionally, it doesn’t include users who use other websites (e.g. Google, TripAdvisor) to look at reviews.

If we use this 18.4M users as our potential audience, we can assume the baseline technical sophistication of Yelp users: comfort using web and mobile applications. Therefore, the product would need to account for this level of ability by making the comparison choice intuitive and the flow between gathering input and generating output seamless.

### ***Competing Products***

In 1979, Zagat emerged as the first restaurant ratings compiler and, ever since then, the space has grown considerably. Yelp officially launched in 2004, kicking off the wave of online, crowd-sourced rating sites that still dominates the landscape. This space is extremely crowded, with Google, TripAdvisor, and OpenTable also offering products. The restaurant recommendation space, however, is less crowded. Most recently, a company called Ness launched a recommendation service that’s user input was how they rated restaurants they had been to along with their social media posts to recommend new restaurants. Ness was acquired in 2014 by OpenTable but there has yet to be a release of OpenTable’s software that incorporates the Ness algorithm.

More recently, a research paper from Cornell used machine learning to recommend restaurants. It used Yelp reviewers’ previous reviews to guess which other reviewers they shared taste with and then picked restaurants that those similar reviewers valued highly. So, while there are diverse examples restaurant recommendation, as of right now, there is no product that attempts to determine cuisine preference in real time and then pair that with the user's valuation of food, service, and ambiance to make a customized recommendation.

***Technical Design***

To retrieve information about each restaurant, we will utilize the Search functionality of the Yelp API to retrieve cuisine types and restaurants near the user for each type of cuisine.

The web app itself will be made using the web2py framework, leveraging Materialize for CSS styling. The site will be hosted on an Amazon EC2 instance. We plan on allowing users to create accounts using either the Facebook API, the OAuth 2.0 Google API, or a form on our site - depending on the user's preference. User data will be stored in a MySQL database hosted by the AWS Relational Database Service.

***Resource Requirements***

In order to build our project, we would not need anything out of the ordinary. We aim to create a web site which would need to be hosted in some capacity as well as linked to a database service in order to track user ratings and comparisons between users. We will use our git repository as a means of transferring code and all have the necessary hardware (a laptop of comparable speed) as well as the appropriate permissions to the repository.

While we do not have any unusual requirements, one major challenge will be finding the algorithm for recommending restaurants and comparing food types. The majority of our service relies on the idea that people need assistance while making decisions; our tool aims to solve this issue but the presentation is extremely important. Users these days could find a multitude of reviews on any given restaurant, so we must ensure that our product accomplishes its original goal of assisting the decision making process without over-burdening users.

***Potential Approaches***

For other approaches, we considered different means of procuring user restaurant preferences as well as alternate device types. For user preferences, we debated whether or not users should be quizzed every time, how permanent their cuisine preferences should be, how to compare cuisines between users, and more. Our approach aims to create some sort of culinary permanence; a user can prefer Korean to Chinese food in general, but the quiz will take recent preferences into account, allowing users to travel both paths of permanent and temporary cravings. We also considered the idea of allowing people to choose between different restaurants instead of cuisines at first. We decided this would be information overload; comparing two restaurants might allow you to process them quicker than a list but the number of restaurants fitting a users' beginning criteria could make for an extremely long process.

In terms of technology, we decided to use a web app over a mobile app for a number of reasons, the most important of which being our lack of mobile experience. We acknowledged the convenience of making group choices in a mobile app but also anticipated the problems that would come with our lack of knowledge. Ideally, we would be able to make our app into a mobile application going forward, but for now we wanted to master the recommendation concept using a web application. Furthermore, this web app will allow for easier database integration based on our previous experience, allowing us to focus on functionality and desirability of our project over minor technical issues.

***Risk Assessment***

Relating back to the issue stated in Resource Requirements, we face a major issue in specifying the usefulness of our product. When people need to find a restaurant, they have set routines for browsing, most of which involve search bars, long lists, and ambivalent ratings. Our product has the potential to re-design this experience, but not without it's own risks. Since users have set steps for restaurant searches, we must make our experience appealing enough to intrigue users to use it over their current system. How well does our comparison framework find their initial cuisine preferences? How often to cuisine preferences change? How does cuisine selection translate for individual users to a group (i.e. is the average favorite cuisine the most desirable for the group?) With UI iterations and different fidelity prototypes, we can develop a well-designed layout.

***Next Steps***

To build our product, we need to focus on the base product of allowing users to choose between multiple culinary choices and creating the ELO system. This involves making the backend for storing a user's food preferences in a database and figuring out how to compare two users' preferences. The base UI will be quite simple, showing two options and signaling that the user should click an option to continue. Once those are functioning, we can work on tweaking our UI with tutorials or an onboarding process as well as additional functionalities for group searches or restaurant filters. Furthermore, we need to collect restaurant data in order to edit our criteria for restaurant recommendations. We will do this by scraping information through existing API's like Yelp or Google Reviews.

Possible Features to extend MVP

- Group recommendation
    - Users can either use their past preferences or continue to indicate their preferences more by selecting between genres, or if a genre has been selected, restaurants.
    - Once individual preferences have been established results are aggregated across a group to minimize a cost function (where higher ranked genres and restaurants have lower costs), to provide a restaurant to the group

    - Also can integrate with calendar, when2meet or doodle to match restaurant and group availability
- Progress Bar
    - Indicates how strong the Elo rankings are based on number of rankings they’ve chosen
- Localized Cuisine Chooser
    - Initially just give two random cuisines
    - Later can rank “best” of the cuisines in the surrounding area by number of restaurants and yelp ranking
- Filter by service
    - Use natural language processing to filter positive and negative reviews by quality of service, food, atmosphere and allow users to select the importance of those relative categories.