We have developed a recommender system, specifically for recommending restaurants based on many food characteristics, in Prolog. The certain approach we took towards the task and the programming language we use to write our conceptual model has had varying effects on our end result including: how well it works, limitations, and usefulness.

The model we have constructed is very quick and accurate at parsing out the information given by the user, assuming it's in a standardized form. The way we have developed the parsing section of our code is by making it resemble human speech, the ideal input would be something someone would actually respond with when asked the question what they would like for dinner; “I would like burgers” as an example response and proper formatting of an input. This information will then be broken down into different parts, with the most important part, nouns being checked against our set of ‘known foods’.

Another thing the system we set up does well is given an objectively best option based on the query we have. After we parse the input we then find the closest match based on their input. We refine and narrow in on a certain restaurant after the first iteration based on more information the user provides; normally this includes getting the desired price or method of getting the food to help better our current belief. A great thing about the way our code is structured is that it is easy to add restaurants, food items, and even food categories into the preexisting knowledge base. This is an important detail that helps mitigate a minor flaw with our current system.

This issue is that we have a small array of restaurants, at least compared to the multitude of restaurants that could be included; however, the small list is varied enough to result in one restaurant for each possible outcome. This isn't a tricky problem to fix, we had this in mind and made sure it is incredibly easy to add or remove restaurants to better fit an individual's location and subsequent options. A smaller thing is each restaurant is only related with one ‘type’ of food, for instance, McDonald’s has ‘burgers’ as its specific type of food even though it has other types of food that are in our knowledge base like ‘chicken’.

Probably the largest drawback as of now is that we only keep a memory of 2 terms and this is erased upon each rerun. An example of a memory of two terms being an issue is that someone states they want ‘burgers and chicken’ then they state they also want ‘rice’, the ‘burger’ will be deleted and the user will be switched in the Chinese food category, “rice and chicken”. We determined this isn't a problem immediately with the next belief revision as the user probably would want to leave the American category. However, for all intents and purposes, the user never was in the American category according to our knowledge. A thing we wish we could change but since we are using Prolog it is very difficult to implement is remembering a user's past preferences; this would help us make better beliefs initially without a user having to repeatedly type that they prefer cheap food, for instance, if they always prefer cheaper dining options.

A limitation occurs in our output to the user, we list the number one option to the user. We have thought about and tested the idea of listing the top 3 restaurant options to the user. We decided against it because based on the number of restaurants we have in the system some weird, wildly inaccurate recommendations could sneak into third place or even second place if not enough information was provided. Contrarily, if two restaurants share the exact same foods,

dining options, and price only one restaurant will be displayed. This currently isn't a problem since these perfect overlaps are uncommon and none are currently present in our set.

We have developed a conceptual model that aims to solve a problem many people run into daily. Hopefully, if the restaurants are accurate enough for the user, our model will fit all criteria of the definition of ‘Conceptual Model’: describing how people's thoughts and perceptions influence the way they feel and behave. We take the user’s preferences and then find the right options for them, which is perfect for situations when users want to try something new, can't decide between a few options, or maybe just forgot about a restaurant. Our goal is to influence people and help them with their decision-making, and we feel that in the right situation we have accomplished this and made deciding where to eat just a little bit easier.