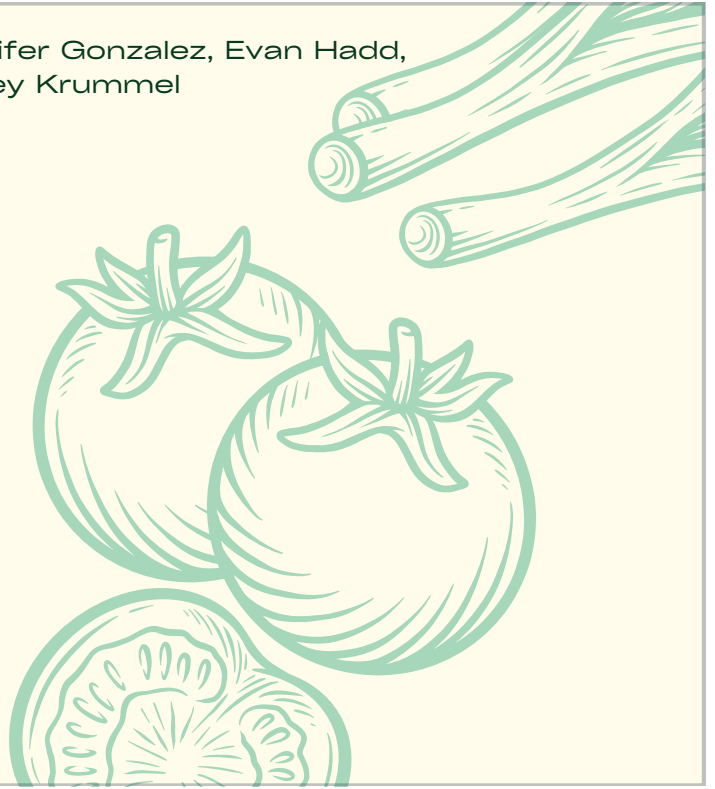


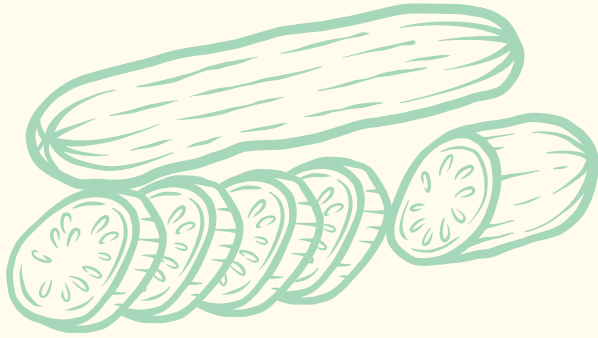
Jose Nico Currea, Jenna Ferguson, Jennifer Gonzalez, Evan Hadd,  
Muhammad Ibrahim, Ramzi Kattan, Hadley Krummel

# SnapChef

**Personalized Recipe  
Recommender**



Imagine you're an exhausted MSBA student and you've just come home from a long day of spending six hours trying to figure out your lift calculation code and you open the fridge and have no clue what you should make for dinner. Well, we've got the app for you!

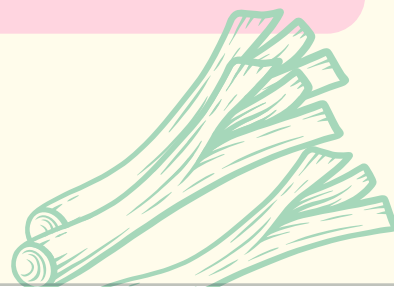
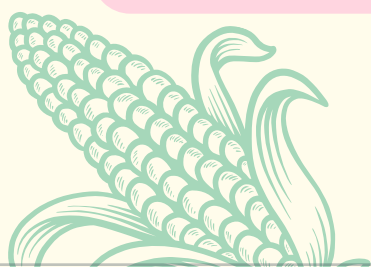


- 01** - Proof of Concept
- 02** - Backend Workflow
- 03** - Future Work
- 04** - Conclusions





# Proof of Concept





"I just got back from class and I only have two hours until my cheerleading practice. I don't have time to clean up and I don't want to be too full at practice and throw up."

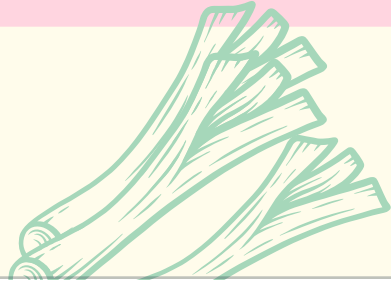
With SnapChef, user's are able to snap pictures of their fridge, pantry, and anywhere they store ingredients in the house. They then speak to personalized SnapChef chatbot to describe their current recipe mood, the meals context, and anything else!

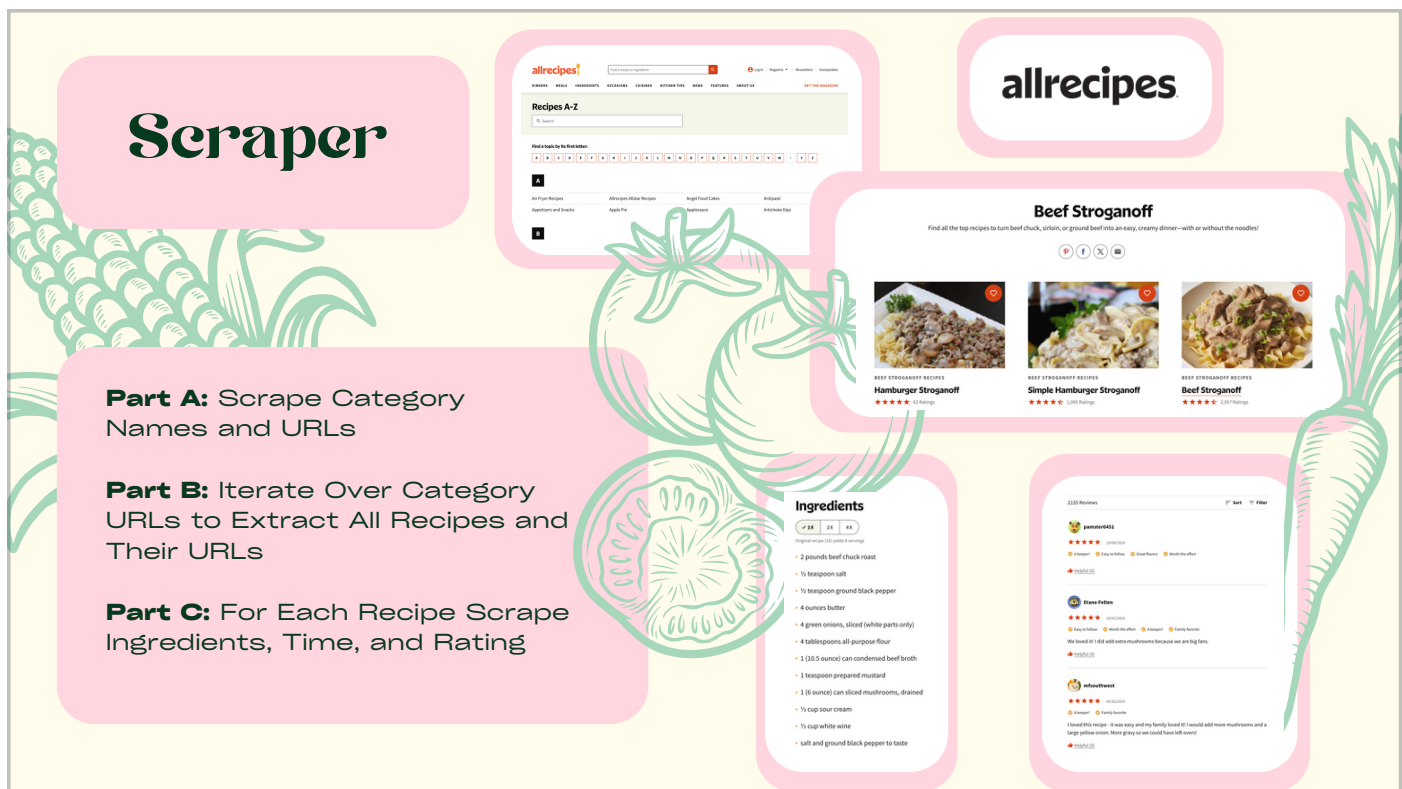
What can you think of making with ingredients from these pictures and this description.

Hang on tight and SnapChef will tell you!



# Backend Workflow





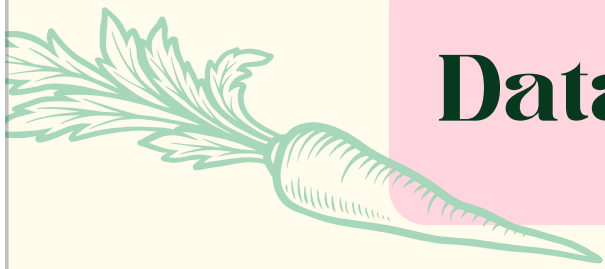
We worked on scraping AllRecipes.com, focusing on extracting recipe data such as ingredients, cooking times, ratings, and user comments. The tools we used for this process were Python, specifically the Selenium and BeautifulSoup libraries for web scraping, along with Pandas for data manipulation.

We started by scraping the list of recipe categories from AllRecipes and stored the category names, recipe names, and URLs into a DataFrame. From there, we iterated through each URL, accessed individual recipe pages, and extracted important details like total cooking time, ratings, and the list of ingredients.

Once that was done, we moved on to scraping user comments. Using Selenium, we automated the process of clicking the "Load More Reviews" button to fetch more comments. For each recipe, we scraped up to 100 user reviews, then appended each comment as a new row in the DataFrame, ensuring that each comment was stored individually.

Afterward, we cleaned the dataset by dropping the ingredients column and counted the number of ingredients for each recipe. Finally, we saved the cleaned DataFrame as a CSV file for further analysis or reporting.

This process allowed us to create a rich dataset containing recipe metadata and user feedback directly from AllRecipes, which can now be used for various kinds of analysis like sentiment or ingredient-based trends.

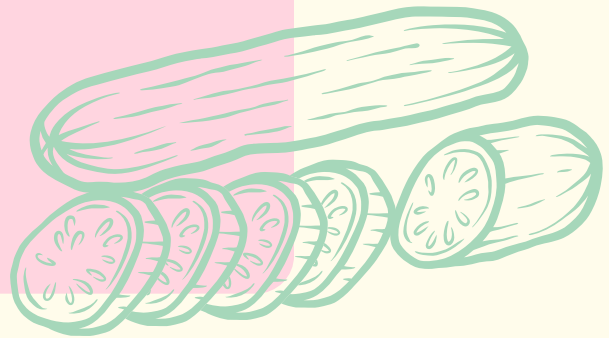


# Data Preprocessing

Take a Recipe and return key ingredients:

- 1/2 cup mashed banana
- 1 teaspoon vanilla extract
- 2 cups of flour

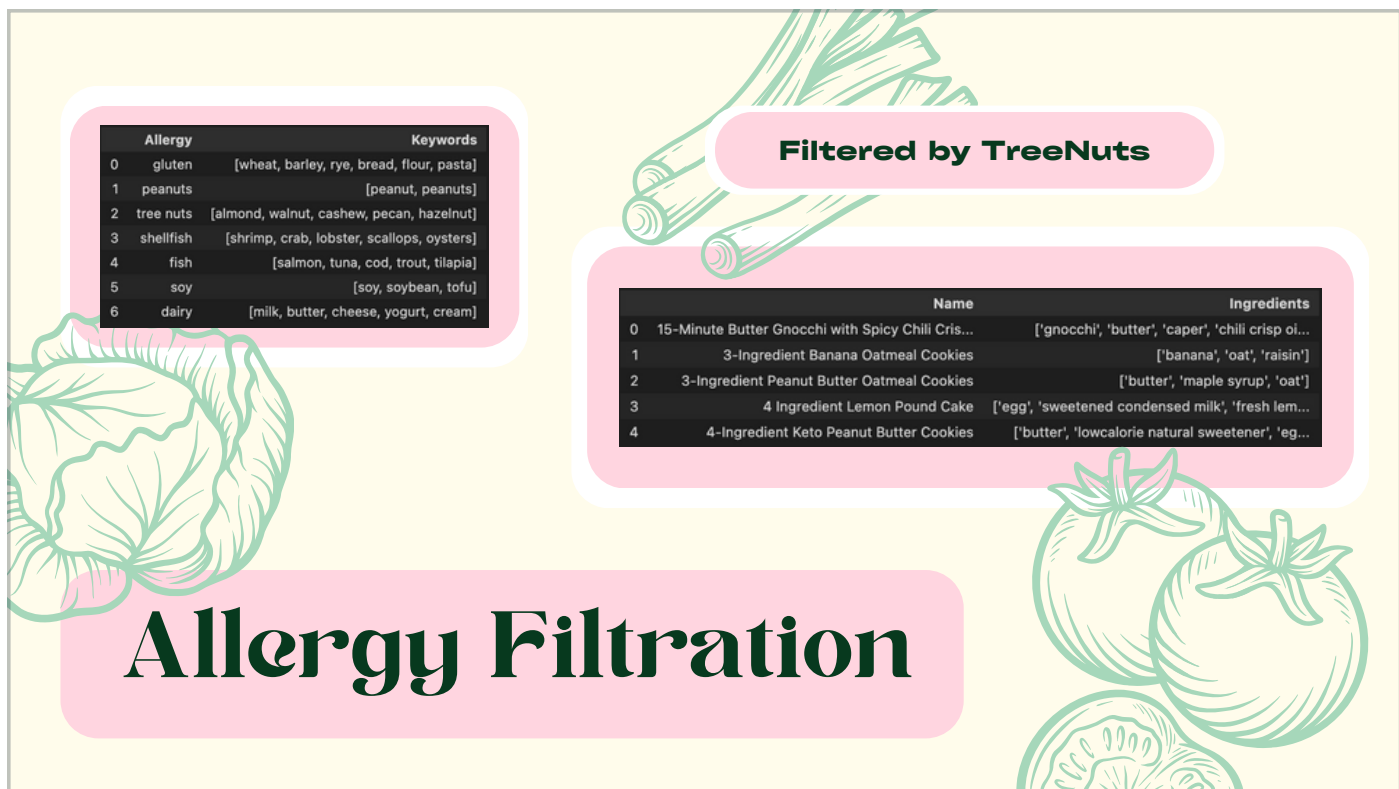
**Into: bananas, vanilla, flour**



In the data preprocessing, there were two main goals. First, take the scraped ingredients and create a file that has one line per recipe with the name in one column and the key ingredients in the next column. The second goal was to create a file where every line matched one review and then included the recipe name, category, and cook time in minutes.

For the first goal, the hardest part was taking the unstructured recipe and working to extract the key ingredients to be able to do a similarity analysis. I tried a couple of different methods like using a 5000-word list to find matches, but ran into problems with plurals. For instance, the name list had bananas, but the ingredients had bananas, which did not match. What ended up working the best was using some lemmatized functions and regular expression to get rid of measurements, descriptors, and extra words.

For the second goal, it was a lot easier. All that had to be done was removing the punctuation and stopwords from the reviews and then reorganizing the data into a way that could be best used for the recommender system.



We felt that it would be important to sort the data to include the allergy tolerances of the user, so we included the allergy filtration feature. We created groupings for the for the categories gluten, peanuts, tree nuts, shellfish, fish, soy, dairy.

Basically, when the user inputs their allergy, in this case, tree nuts, the dataset is shortened to exclude any recipes that contain anything in this category.



# Image Analytics and Attribute Extraction



The process of extracting ingredients and attributes uses large language models.

The user is prompted to input what recipes they're in the mood for. This allows for very informal responses as the OpenAI gpt-4o model is used to sift through the user's prompt and extract the most recipe-related attributes. For example, if the user inputs:

"I just got back from class and I only have two hours until my cheerleading practice. I don't have time to clean up and I don't want to be too full at practice and throw up."

The model was prompted with:

"You are a helpful assistant that analyzes a user's description of their recipe preferences. Return a list of recipe attributes based on the user's preferences. The list should be provided in comma-separated format (e.g., 'attribute1, attribute2, attribute3'). Provide only the two lists without any extra commentary."

The model extracts the attributes:

quick to prepare  
easy cleanup  
light meal  
low-calorie

The user is then able to upload as many photos of their fridge, pantry, and anywhere else they

store food.

Since OpenAI requires links to be public in order for the model to analyze it, AWS S3 Buckets were used as a public location to host images that generates a public URL. The boto3 and AWS API was used to automate the process of image upload using a secret key and the bucket key. The images automatically uploaded one by one and the image URLs are generated.

The next step is the image analytics. First, Google Vision was used to attempt to label the ingredients in the user's images, however Google Vision's classes were found to be too general. For example, for a peanut butter jar, the model returned 'container' and 'food.'

Next, a Faster R-CNN network was set up to be fine-tuned for labeling food items. The issue here was finding datasets with relevant class labels. A relevant dataset called the Grozi-3.2K dataset with bounded items was found, but the class labels were specific to swiss grocery store items. Another dataset called the ISIAFood500 with over 500,000 training data was found, but downloading the dataset was infeasible.

Finally, the OpenAI gpt-4o model was found to be multi-modal with the capability to accurately describe items, even in obscure images. The system was prompted with:


"You are a helpful assistant that processes images and extracts ingredients from the image. Return a list of ingredients used for cooking detected in the image. Do not include vague descriptions or any commentary. The list should be provided in comma-separated format (e.g., 'ingredient1, ingredient2, ingredient3')."

The model returned for the two images the following ingredients:

jars with spices  
pasta  
cereal  
legumes  
apples  
onions  
oranges  
lemons  
pickles  
jam  
mayonnaise  
milk  
hot sauce  
soda  
eggs

cheese  
ketchup  
mustard  
butter  
soda  
yogurt

# Ingredient Requirements Recommender



name	feasibility_score
Chef John's Cuban Sandwich	0.55
Copycat McDonald's Apple Pies	0.55
Marry Me Chicken Pasta Bake	0.53
Beer Cheese Corn Chowder	0.50
Easy Overnight Oats	0.50

LLM



"You are an ingredient matching evaluator. Your task is to assess how feasible it is to make each recipe based on the ingredients a user has. Return a score between 0 and 1 where 1 means the user has all the ingredients in the recipe and a lower score means some of the ingredients are missing. An exact match is not required, for example if the user has 'onion' and the recipe calls for 'white onion'. Consider the importance of an ingredient in a recipe, for example, 'beef' in a burger recipe. Do not provide any explanations or contextual information. Only return the score."

In order to find the feasibility score for every recipe, Llama 3.0 API was used with the following system prompt:

"You are an ingredient matching evaluator. Your task is to assess how feasible it is to make each recipe based on the ingredients a user has. Return a score between 0 and 1 where 1 means the user has all the ingredients in the recipe and a lower score means some of the ingredients are missing. An exact match is not required, for example if the user has 'onion' and the recipe calls for 'white onion'. Consider the importance of an ingredient in a recipe, for example, 'beef' in a burger recipe. Do not provide any explanations or contextual information. Only return the score."

Parallel processing was used to speed up the API calls by using 3 workers.

Feasibility scores were then attached to each recipe, and then merged with the reviews dataframe to combine all recipes with their reviews for the next step.

# Attribute Review Recommender

Filtered by Feasibility Score  $\geq 0.60$

spaCy

Evaluation Score = 70% x Feasibility Score + 30% x Similarity Score

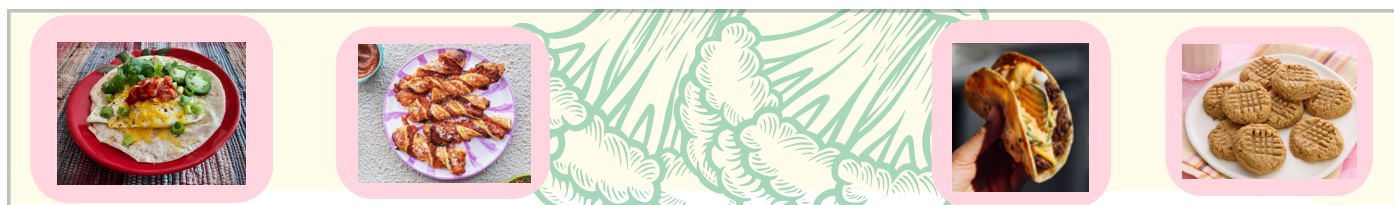
name	ingredients	cook_time	rating	category	url	Feasibility Score	Review Similarity	Evaluation Score
Classic Macaroni Salad	['uncooked elbow macaroni', 'mayonnaise', 'white sugar', 'distilled white vinegar', 'prepared yellow mustard', 'salt', 'ground black pepper', 'stalk celery', 'onion', 'green bell pepper seeded', 'carrot', 'pimento pepper']	270.0	4.4	Kosher	<a href="https://www.allrecipes.com/recipe/81108/classic-macaroni-salad/">https://www.allrecipes.com/recipe/81108/classic-macaroni-salad/</a>	0.6	0.576630	0.592989

First, the recipes were filtered by Feasibility Scores. A feasibility threshold of 60% was set to allow for recipes that would require a quick trip to the grocery store. This also allows for more recipes to be considered when finding similarity scores.

To find similarity scores between the user's attributes and the reviews of each recipe, spaCy's medium sized model was used to create word embeddings for the attributes and reviews of the feasible recipes. SpaCy was used because the user's attributes such as 'quick cook time' can be said in many different ways, and therefore we desired more flexibility when calculating similarity. We found bag of words resulted in much lower similarity scores.

An overall evaluation score was found by adding 70% of the feasibility score and 30% of the similarity score. More weight was given to the feasibility score because a recipe with high similarity score but low feasibility score is not useful because the user can't cook that recipe without the ingredients.

These evaluation scores were averaged over all reviews for each recipe. Recommendations were found by sorting recipe average evaluation scores in descending order and taking the top 5 recipes.




name	ingredients	cook_time	rating	category	url	Feasibility Score	Review Similarity	Evaluation Score
Smash Burger Taco	['mayonnaise', 'ketchup', 'mustard', 'relish', 'hot sauce', 'lb ground round', 'kosher salt', 'ground black pepper', 'flour tortilla', 'american cheese cut half lengthwise', 'iceberg lettuce', 'onion']	20.0	4.6	Tacos	<a href="https://www.allrecipes.com/smash-burger-taco-recipe-7485747">https://www.allrecipes.com/smash-burger-taco-recipe-7485747</a>	0.75	0.583860	0.700158
Apple Cheddar Twists	['apple', 'lemon juice', 'butter', 'brown sugar', 'puff pastry', 'extra sharp cheddar cheese', 'sea salt']	80.0	4.5	Christmas	<a href="https://www.allrecipes.com/apple-cheddar-twists-recipe-8709986">https://www.allrecipes.com/apple-cheddar-twists-recipe-8709986</a>	0.60	0.642528	0.612758
4-Ingredient Keto Peanut Butter Cookies	['butter', 'lowcalorie natural sweetener', 'egg', 'sugarfree vanilla extract']	25.0	4.0	Sugar-Free Recipes	<a href="https://www.allrecipes.com/recipe/261181/4-ingredient-keto-peanut-butter-cookies/">https://www.allrecipes.com/recipe/261181/4-ingredient-keto-peanut-butter-cookies/</a>	0.60	0.610753	0.603226
Fried Egg Tortilla	['butter', 'egg', 'taco seasoning taste', 'cheddar cheese', 'flour tortilla']	9.0	4.3	Cooking for One	<a href="https://www.allrecipes.com/fried-egg-tortilla-recipe-7561904">https://www.allrecipes.com/fried-egg-tortilla-recipe-7561904</a>	0.60	0.603685	0.601105
Applesauce	['apple cored', 'water', 'ground cinnamon', 'ground clove', 'white sugar']	35.0	4.7	Applesauce	<a href="https://www.allrecipes.com/recipe/54346/applesauce/">https://www.allrecipes.com/recipe/54346/applesauce/</a>	0.60	0.587820	0.596346

## Final Recommendation

Here are the 5 recipes:

Smash Burger Taco  
 Apple Cheddar Twists  
 4-Ingredient Keto Peanut Butter Cookies  
 Fried Egg Tortilla  
 Applesauce

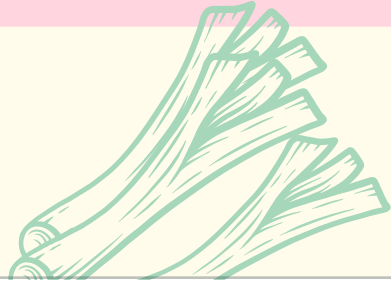
Some key takeaways are extracted from this:

1. Category: Someone heading to cheer practice after a long classes likely doesn't want applesauce! Applesauce does not make much sense in the context of the user's day. We believe we can solve this by integrating a filter for Breakfast, Lunch, Dinner, Snack, Dessert... items. Another way to automate this is to use time of day to assign a lower feasibility score to items based in unlikely category using an LLM. A simple edit of the system prompt could account for the user's prompt in the user message to account for context and time of day.
2. Importance of Ingredient: We are recommending the user make Smash Burgers... with no meat! This should be accounted for in the feasibility scores. Ingredients should be assessed by the importance of the ingredient in the recipe. Again, the system prompt can be tweaked to account for this.
3. Cook Time: Although the user specified a quick cook time, a high similarity was found for Apple Cheddar Twists which takes 80 minutes! The likely reason for this was because Apple Cheddar Twists are a baking item, and 80 minutes in baking is not that long! We believe adding in meal type would reduce the chances of being recommending meals that are inconsistent with your

cook time preferences.



# Future Work

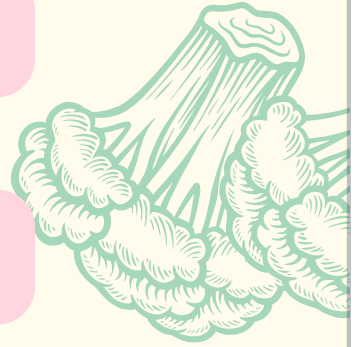




# App Design & Chatbot UI

## Memory Features

## Grocery Store Integration



1. These ideas are great but it would need to be turned into a cohesive application that incorporates the image analysis into a flawless user experience. First steps would include incorporating a chatbot, and second steps could be speaking to the AI feature to input the desired meal attributes.
2. In the future, the application will be programmed to remember the allergies and preferences of the user to make more informed decisions. Then, it would allow the user to snap a picture of their spices, and it will remember which spices they have and even how much of each they have left.
3. We thought that a lucrative addition to this application would be to work with various grocery stores. This would allow users to have the groceries they still need to be delivered right to their doorstep. We could expand this to include features that find the closest grocery store and also the cheapest options for the user.