Table of Contents

Discovery

Proposal

Create an application to allow users to search for recipes online given an ingredient set that they can choose based off of their compatibility with each other.

The application can be divided into two distinct pieces which will be able to be leveraged independently if desired:

1. Ingredient selection process
2. Recipe search based off of an ingredient set

User Stories:

1. Main use case:
   1 User selects a first ingredient from a list from the entire data set (ingredients will be displayed alphabetically initially, later iterations of the application will include better ingredient bucketing by cuisine and other types of groupings like acidity)
   2 User selects a second ingredient from the list of “paired” ingredients associated with the parent. Each ingredient will have a number/percentage (tbd) attached to it to convey the strength of the pairing of the two ingredients
   3 User selects a third (and beyond) ingredient from a list of the “paired” ingredients associated with both the first and second ingredients
   4 Selected ingredients are passed to a recipe API which will return a list of recipes that match user’s ingredient set as closely as possible
2. Partial use case 1:
   - User performs steps 1-3 of main use case and uses the ingredient set produced without leveraging the recipe API
3. Partial use case 2:
   - User performs only step 4 by inputting their own ingredient set independent of application’s suggested pairings

Resources:

Back end

1. Data set - https://www.karenandandrew.com/books/the-flavor-bible/
2. Online epub to html converter
3. DB – PostgreSQL with SQLite as a backup (local machine currently has issues with the PostgreSQL install)
4. Language(s) – Python
5. Recipe API – https://market.mashape.com/spoonacular/recipe-food-nutrition#search-recipes-by-ingredients

Front end

1. Python, HTML, CSS, JavaScript, Bootstrap

Challenges/Limitations

Proposed source is from a book rather than a clean tabular data set

- Python epub libraries are questionable at best and should be avoided
- Book is in .epub format, will have to be converted to HTML for usability and online epub to html converters have trouble creating “correct” associations (everything is stacked rather than nested, lacks meaningful content-based tag names)
- Data set was written as a book rather than as something to be leveraged for an application so significant data cleansing/mapping efforts are required to be able to effectively recognize connections and have standard search inputs (see schema for DB table “child_ingredients”)
- In the raw dataset, 111 of the 708 'parent' ingredients are not populated with child ingredients. They seemingly fall into one of the following categories:
  1. They have a reference to another parent ingredient's child set (ex: yams has "see sweet potatoes")
  2. They are a parent wrapper (ex: wine has "see individual varietals")
  3. They have no child ingredient set, just a small quote (ex: chardonnay vinegar)
  4. They refer to a concept (ex: acidity)
- Proposed solutions for the above would be as follows:
  1. Select the referenced set
  2. Remove as an option? List all associated children? Will need to further consider how to deal with this in the child sets
  3. Refer to parent set or consider removing
  4. Remove but will need to look for these entries in child_ingredients as well
- It will be necessary to reverse engineer some of the parent ingredients from children where there are several child entries for something without a parent. Examples include brandy, butter, and cider

Design

DB Schema

Table parent_ingredients

Key	Name	Type	Description
PK	pk	INTEGER	Unique parent ingredient identifier
	name	VARCHAR	Name of the ingredient
	season	VARCHAR	The ingredient's seasonal peak(s)
	taste	VARCHAR	The ingredient's primary taste(s), e.g., bitter, salty, sour, sweet
	function	VARCHAR	The ingredient's intrinsic property, e.g., cooling vs. warming
	botanical_relatives	VARCHAR	The ingredient's closest botanical relatives
	weight	VARCHAR	The ingredient's relative density, e.g., from light to heavy
	volume	VARCHAR	The ingredient's relative flavor "loudness," e.g., from quiet to loud
	techniques	VARCHAR	The most commonly used techniques to prepare the ingredient
	tips	VARCHAR	Suggestions for using the ingredient
	search_term	VARCHAR	String to be used when first ingredient is passed to the recipe API

Notes:

- Fields season, taste, function, botanical_relatives, weight, volume, techniques, and tips will be purely cosmetic for the foreseeable future. The information was present in the book for each listed parent ingredient so it will be included as value add information when selecting an ingredient.
- Field search_term on the parent table will only be used for the first selected ingredient since all subsequent ingredients for the API will come from the child table's search_term field.

Table child_ingredients

Key	Name	Type	Description
PK	pk	INTEGER	Unique child ingredient identifier
	name	VARCHAR	Name of the ingredient
	pairing_strength	INTEGER	Strength of the pairing of parent and child ingredient
	search_term	VARCHAR	String to be used when ingredients are passed to the recipe API
FK	pairing_pk	INTEGER	PK value from the parent table used to indicate all listed pairings for a unique ingredient on the parent_ingredients table
FK	self_pk	INTEGER	PK value from the parent table to indicate which parent_ingredients record should be used when selecting an ingredient pairing list for the next selection by the user

Note: search_term fields will be included for both parent_ingredients and child_ingredients tables in order to standardize the API call parameters and return more useful search results without stripping information out of of the entries. For example: child ingredient entry “almonds (peak: october)” would be passed to the recipe API as “almonds” but the user would still see the original text when selecting ingredients.

UI

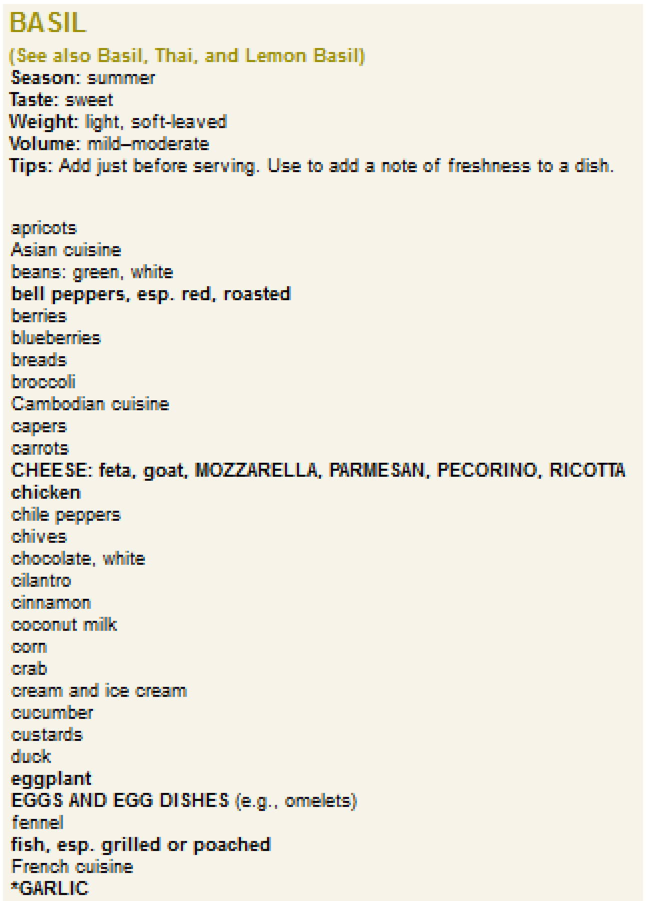
1. Start page
    1. Options (links) based off of user stories
        1. User generates only ingredient set based on pairing info and weighting
        2. Only pass user defined ingredients to recipe API
        3. User generates ingredients based on pairing info which is passed to recipe API
        4. See information on all ingredients page
2. Only generate ingredients page
    1. Start point will give a full list of possible first ingredients (entries on parent_ingredients table) and ask user to select one. Look to add something at the top to redirect by first letter and/or to search since there are ~700 parent ingredients
    2. When first ingredient is selected, second ingredient page will look like the first ingredient page but it will be only ingredients that have pairings with the first selected ingredient with their pairing strength with respect to the first ingredient
    3. Third ingredient (and beyond) will have a list based off of all pairing information from all previously selected ingredients with pairing strength calculated per methodology section of this document
3. Only leverage recipe API page

    1. Start point will have a series of inputs where users can put some predefined number of ingredients which will be passed directly to the recipe API
    2. Default is to return 5 results with id, title, image link, count of used ingredients, count of missed ingredients, and number of likes.
        1. Within the application, display should be the image, title, and number of used ingredients. The title should have a href to the recipe
        2. Recipe address can be derived by taking the image link and editing to change “recipeImages” to “recipes” and removing the “.jpg”
4. Page for both
    1. Go through full process for “only generate ingredients” page
    2. Show results page for “only leverage recipe API” page based on selected ingredients without prompting for any manual input
5. Ingredients information page
    1. List of all parent ingredients with their paired ingredients and “cosmetic” fields from the parent table like season, taste, etc.
    2. Use ‘#’ character in the address to be able to search by parent ingredient so that the pages for generating ingredients lists can have an “Info” link on each ingredient which would direct to that individual ingredient’s information

Methodologies

Pairing strengths:

- The strength of the pairing in the source material is denoted by the formatting of the text for each child ingredient. The best pairing category is marked by bold + uppercase + a leading asterisk, the second best is bold + uppercase, the third best bold, and the fourth best is unformatted. For example, the following is the entry for “Basil”:



BASIL
(See also Basil, Thai, and Lemon Basil)
Season: summer
Taste: sweet
Weight: light, soft-leaved
Volume: mild–moderate
Tips: Add just before serving. Use to add a note of freshness to a dish.

apricots
Asian cuisine
beans: green, white
bell peppers, esp. red, roasted
berries
blueberries
breads
broccoli
Cambodian cuisine
capers
carrots
CHEESE: feta, goat, MOZZARELLA, PARMESAN, PECORINO, RICOTTA
chicken
chile peppers
chives
chocolate, white
cilantro
cinnamon
coconut milk
corn
crab
cream and ice cream
cucumber
custards
duck
eggplant
EGGS AND EGG DISHES (e.g., omelets)
fennel
fish, esp. grilled or poached
French cuisine
__*GARLIC__

Pairing scores:

- Numerical scorings will need fairly extensive user testing to be properly tweaked.
- Tentative values to attribute for a numerical match in pairings would be 100 for bold + caps + asterisk, 80 for bold + caps, 60 for bold, and 40 for no additional formatting. Using the example above, if basil was chosen as the user's first ingredient, garlic would have a score of 100, cheeses would have a score of 80, chicken would have a score of 60, and apricots would have a score of 40
- The number/percentage attached to the third ingredient and beyond will have additional weighting performed to consider the strength of the pairing with all of the other selected ingredients. This could tentatively be done as an average of all of the pairing strengths with each other
- Ingredients that are listed for one ingredient but not for another could be defaulted to having a score of 0 when considering it for the unlisted value and then averaging the other score and the 0 value

Later Iterations

(Enhancements to be added after the first iteration is production-ready, **not** ordered by priority)

1. Give users the ability to randomize a number of ingredients (ex: only randomize the first ingredient, randomize two ingredients and choose one, randomize all ingredients)
2. Add tracking in the ingredient selection process to make sure it's transparent as to which parent(s) the child ingredients came from
3. Find a way to display which ingredients were matched/missed in the results of the API, may be as simple as changing a request parameter to "fillIngredients=true"
4. Add options to display additional information like cook time, difficulty, score, etc. with the possibility of adding additional filters on those attributes. This is heavily dependent on what is available through a combination of the API itself and curling the resulting recipe link
5. Add prohibited ingredients for allergies/intense dislikes. Same dependencies as point 4
6. Machine learning to tweak ingredient weightings
7. Scrape user data from recipe site and store user's preferences to suggest based off of ratings of users with similar tastes. Requires user login functionality to be implemented as well as a recipe site that has distinct user ratings/reviews
8. The flavor bible has information for what pairings to avoid as well as suggested flavor groups which is not included in current data set (due to variable structure making it very difficult to scrape)
   - Avoid would be implemented where if an "avoid" ingredient appears in the children of a subsequent parent, it is removed from the list of possible choices
   - Suggested flavor groups could be another partial use case where a user could select an ingredient and choose one of the pre-set lists. Suggested flavor groups could also be used to add more weight to certain ingredients when choosing subsequent ingredients