Capstone One - Project Proposal

Fridge Raiders

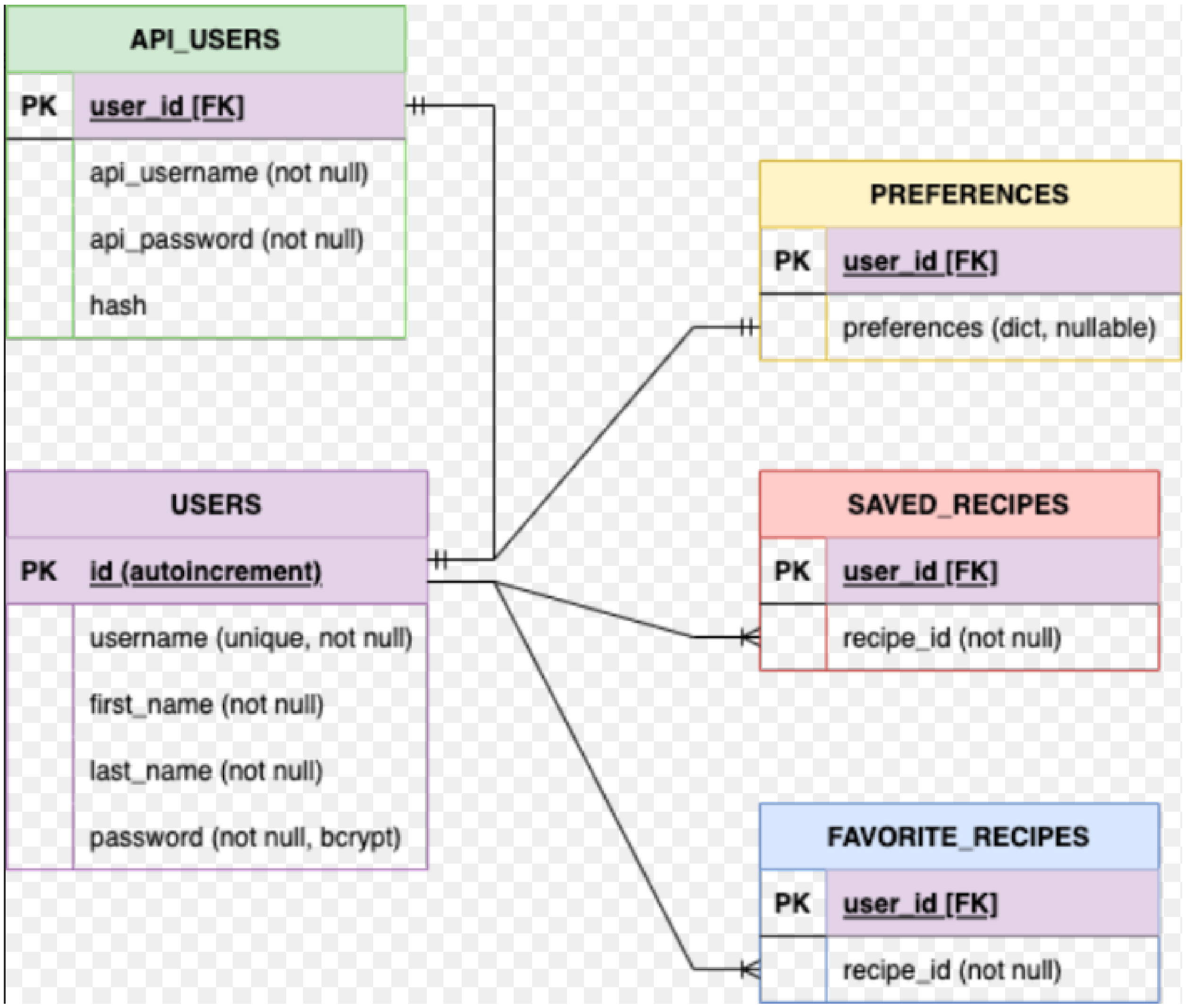
Introduction:
This website aims to tackle the goal - "What should we have for dinner?". It is a question uttered in households worldwide; whether when planning a weekly menu or more last minute, everyone wants to know what they should make for dinner each night. Fridge Raiders aims to help answer that question for users by helping to come up with recipe ideas based on the ingredients they already have in their fridge, pantry or cupboards. The demographic for this site would be 18-65, there is no small grouping of people who face this question and it would appeal to any and all who are preparing a meal for themselves, friends, or their family.

Data Requirements/API Information:
The API which will be used to build this site is the Spoonacular Food API. The desired endpoints would be:

1) POST https://api.spoonacular.com/users/connect - connect signed up users to the API so that user specific requests can be made (eg. saving/favoriting recipes and creating a curated shopping list from chosen recipes). Will need to save username and hash to database to be included in user specific API calls.
2) GET https://api.spoonacular.com/recipes/findByIngredients - which will allow users to input ingredients they already have and return recipes that use those ingredients. Options can be given for users to choose whether to maximize used ingredients or minimize missing ingredients. Users could choose a recipe from those returned and the list of missing ingredients can be captured from that API data.
3) GET https://api.spoonacular.com/recipes/{id}/information - once a specific recipe has been chosen by the user an API call can be made to get full information about the recipe by passing the id from the previous API call to this one.
4) GET https://api.spoonacular.com/food/ingredients/substitutes - give users the option to search for alternatives for ingredients by having a clickable list of ingredients for the recipe which will make an API call to find alternatives.
5) POST https://api.spoonacular.com/mealplanner/{username}/shopping-list/items - users can click a button to make a shopping list for their chosen recipe. This will use the data combined from the findByIngredients call to pick out the missed ingredients and automatically add to the shopping list. If a user searches for alternatives for an ingredient they can choose to add any missing alternatives to their shopping list by clicking a button. User information will need to be passed from the database.
6) DELETE https://api.spoonacular.com/mealplanner/{username}/shopping-list/items/{id} - users can delete an ingredient from their shopping list (eg. if they have chosen an alternative for that item or are choosing to leave it out of their version of the recipe or they've already found/bought it). User information will need to be passed from the database.
7) GET https://api.spoonacular.com/mealplanner/{username}/shopping-list - show a users shopping list. User information will need to be passed from the database.
8) GET https://api.spoonacular.com/recipes/complexSearch - give users the option to search recipes without inputting ingredients. Users can input any intolerances, preferred cuisines, ingredients to include or exclude, plus many more options that can be passed to the API (but don't want to add too many and make things more complicated).

9) GET https://api.spoonacular.com/recipes/{id}/similar - users can request similar recipes from ones that they have chosen/favorited/searched by clicking a button and a request being made to the API using the previous recipe's id.

Database Schema:
Schema Diagram



Potential Issues and Things to Consider:

- Possible issues with making correct API calls and including all relevant information in the request body/headers as lots of API endpoints are suggested to include for the site.
- Trying to combine searchByIngredient with intolerances and exclude ingredient options for complexSearch so that users can save preferences but combine the outputs for searchByIngredient to reflect those preferences.
- Trying to store user preferences.
- Passwords will need to be secured using bcrypt for users signing up and logging in.
- API usernames, passwords and hash information will need to be stored

Functionality/User Flow/Features:
(highlighted features go beyond CRUD and/or stretch goals)

1) Homepage - introduce concept, show register and login buttons in navbar, allow anyone to search recipes without signing up as only some API endpoints require a logged in user. API GET requests for complexSearch can be made (possibly include randomSearch endpoint also). Notice for users about needing to signup/login to use saving/favoriting/searching by ingredients/shopping list functionality.
2) Registration Form page - collect all USERS table information using WTForms, send API POST request to connect user to API and store returned data in API USERS table of db.
3) Login Form Page - returning users can login by providing just username and password using WTForms. API USERS table should already have their API user information stored for making user connected API requests.
4) Once a user is logged in:
    a) User homepage - update navbar to show name of user and logout button plus links to saved recipes and favorite recipes (get recipe_ids from database and make an API call to get recipe information for each recipe), show search options with WTForms: complexSearch endpoint for searching without inputting specific ingredients but getting to select specific criteria (eg. intolerances, excluded ingredients, etc), or searchByIngredient endpoint to input fridge/pantry ingredients to find recipes containing those ingredients.
    b) If the user chooses to use complexSearch endpoint route, display results for matching recipes. Make each recipe have a clickable button to choose (API call to get detailed recipe info), display detailed recipe info, an option to create a shopping list for the recipe and an option to see similar recipes by clicking a button (API call to similar endpoint) and then display similar recipes with an option to create a shopping list. Give users the option to save their preferences.
    c) If the user chooses to use the searchByIngredient endpoint route, display results for matching recipes. Possibly include a sort feature to sort by number of missed ingredients. Similar to the previous route, each recipe can have a clickable button to choose and display detailed info. Users can then choose to make a shopping list by adding ingredients that are missing. From the shopping list they can search for alternatives and then update their shopping list by deleting any unnecessary ingredients or adding alternatives.