BRAINSTORMING

1. Sing-In
    a. Username
    b. Password
    c. User_id
2. Create Recipies
    a. Ingredients
    b. Descriptions
    c. Quantities
    d. Instructions
    e. Duration
    f. Share
    g. Save
3. Share Recipies
    a. Recipe
    b. Private
    c. Public
    d. Share options
4. Buy Ingredients
    a. Recipe
    b. Ingredient
    c. Quantity
    d. Price
    e. Nearest location
    f. Total
    g. Similar items
5. View
    a. Recipe Name
    b. Recipe Description
    c. Meal Type
    d. Ingredients
    e. Duration
    f. Buy Ingredients
    g. Save
    h. Search
6. Occassions
    a. Recipies
    b. People
    c. Buy Ingredients
    d. Meal Types
    e. Total Prep
    f. Total cost

TABLES

1. User: Will store data about the user that will later be used at sign-in, to create recipes, make occasions, view recipes, and add items to the grocery list.
    a. user_id SERIAL **PRIMARY KEY**,
    b. First Name **VARCHAR(50),**
    c. Last Name **VARCHAR(50),**
    d. Date of Birth INTEGER **NOT NULL**
    e. Zip Code INTEGER **NOT NULL**
    f. Email Address **VARCHAR(50),**
    g. Username **VARCHAR(50),**
    h. Password **VARCHAR(50),**
    i. Dietary Preference INTEGER **NOT NULL REFERENCES** dietary_pref(dietary_id),
    j. Status BOOLEAN
2. Saved Recipes
    a. Username VARCHAR(50) **NOT NULL REFERENCES** user(username)
    b. Saved recipes VARCHAR(800) **NOT NULL REFERENCES** recipies(save_status),
    c. ingredients **VARCHAR(200) NOT NULL REFERENCES** ingridents(ingredients_id)
3. Dietary Preferances: This table will store data regarding the users dietary preferences, but it is also used as a tag for searching recipes.
    a. dietary_id SERIAL **PRIMARY KEY**,
    b. Dietary_pref_name **VARCHAR(50),**
        i. Low-Carb
        ii. High-Protain
        iii. Low/No-Sodium
        iv. Gluten-Free
        v. Lactose-Free
        vi. Vegetarian
        vii. Paleo
        viii. Vegan
        ix. Nut-Free
        x. Pescetarian
    c. Recipes **VARCHAR(200) NOT NULL REFERENCES** recipes(recipe_id)
    d. ingredients **VARCHAR(200) NOT NULL REFERENCES** ingridents(ingredients_id)
4. Recipies: This table will store information relevant to recipes. This information will be accessible and used when the user creates, views and shares a recipe, as well as when they purchase items off of the list.
    a. recipe_id SERIAL **PRIMARY KEY**,
    b. Recipe title **VARCHAR(50),**
    c. Description **VARCHAR(200),**
    d. Ingredients**VARCHAR(50),**

e. Quantity **VARCHAR(50),**
f. Instructions **VARCHAR(1000),**
g. Prep Time INTEGER **NOT NULL**
h. Cook Time INTEGER **NOT NULL**
i. Total Time INTEGER **NOT NULL**
j. Servings INTEGER **NOT NULL**
k. Share_status BOOLEAN **NOT NULL REFERENCES** account(status),
l. Print_status BOOLEAN **NOT NULL**
m. Save_status BOOLEAN **NOT NULL**
n. Status **REFERENCES** account(status),
o. course INTEGER **NOT NULL REFERENCES** courses(course_id),
p. Meal_type INTEGER **NOT NULL REFERENCES** mealType(meal_id),
q. Tag INTEGER **NOT NULL REFERENCES** dietary_preferences(dietary_id),
r. **cost INTEGAR NOT NULL REFERENCE** grocery_list(price)

5. Grocery List: This table will store items pulled from recipes for purchase. It will also allow for the user to identify similar items they could swap ingredients with. Lastly, it will pull data from relevant stores to identify the price and where the items can be purchased near the user.
   a. list_idSERIAL **PRIMARY KEY,**
   b. ingredient_id **INTEGAR NOT NULL REFERENCE** ingredients(ingredient_id)
   c. Ingredient_name **VARCHAR(50) NOT NULL REFERENCE** ingredients(name)
   d. Description **VARCHAR(50) NOT NULL REFERENCE** ingredients(description)
   e. Size INTEGER **NOT NULL**
   f. Price INTEGER **NOT NULL**
   g. Swap BOOLEAN **NOT NULL**
   h. nearestLocation_id INTEGER **NOT NULL REFERENCES** store_locations (nearestLocation_id),
   i. suggested_ingredients_to_SWAP INTEGER **NOT NULL REFERENCES** ingredients(ingredient_id),
   j. total
6. Occasions: This table will be used to store data when the user creates different occasions. This table must be able to access recipes, grocery lists, ingredients, courses, and meal types.
   a. Occasion_id SERIAL **PRIMARY KEY**,
   b. Name **VARCHAR(50),**
   c. Description **VARCHAR(200),**
   d. Number of People INTEGER **NOT NULL**
   e. Meal_id INTEGER **NOT NULL REFERENCES** meal_type(meal_id),
   f. Recipe INTEGER **NOT NULL REFERENCES** recipies(recipe_id),
   g. Ingredients INTEGER **NOT NULL REFERENCES** ingredients(ingredient_id),
   h. course INTEGER **NOT NULL REFERENCES** courses(course_id),
   i. Total prep INTEGER **NOT NULL REFERENCES** recepies(cook_time),
   j. Total cost INTEGER **NOT NULL REFERENCES** grocery_list(total),
7. StoreLocations: This table is used to store information regarding locations near the user. It feeds data to the shopping cart function.

   a. nearestLocation_id SERIAL **PRIMARY KEY**,
   b. Zip Code INTEGER **NOT NULL REFERENCES** account(zipcode),
   c. Street Address **VARCHAR(200),**
   d. City **VARCHAR(50),**
   e. State **VARCHAR(2),**
   f. Price_comp **INTEGER NOT NULL REFERENCE** grocery_list(price)
   g. Ingredients_availabile **VARCHAR(200) REFERENCE** ingredients(ingridient_id)

8. MealTypes: This table is used to search and label recipes. It is accessible for the occasions feature.
   a. Meal_id
   b. meal_name
      i. Breakfast **BOOLEAN**
      ii. Lunch **BOOLEAN**
      iii. Dinner **BOOLEAN**
      iv. Brunch **BOOLEAN**
      v. Coffee **BOOLEAN**
9. Courses: This table is used to identify the type of dish. It facilitates the searching, creating, and sharing of recipes.
   a. Course_id SERIAL **PRIMARY KEY**,
   b. course_name
      i. Appetizer **BOOLEAN**
      ii. Main dsh **BOOLEAN**
      iii. Dessert **BOOLEAN**
      iv. Drink **BOOLEAN**
10. Ingredients: This table is accessible to the create, view, groceries and sawp functions. It's also utilized to feed dietary preferences and creating occasions.
    a. Ingredient_id SERIAL **PRIMARY KEY**,
    b. Name **VARCHAR(50),**
    c. Description **VARCHAR(50),**
    d. dietary_id INTEGER **NOT NULL REFERENCES** dietary_preference(dietary_id),
    e. Similar items **VARCHAR(50),**
    f. **cost INTEGAR NOT NULL REFERENCE** grocery_list(price)

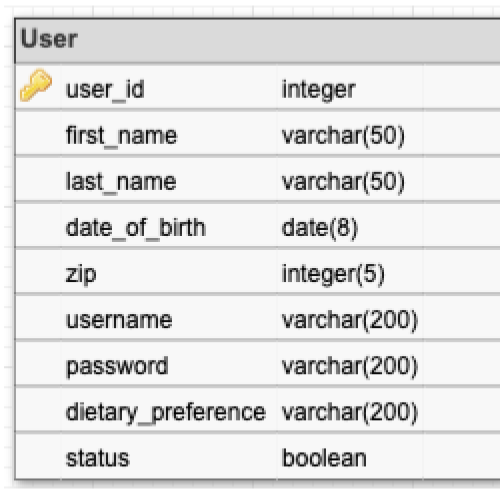
<u>RELATIONSHIPS</u>

One-to-One	One-to-Many	Many-to-Many
User → Status User → ZipCode	User → Occasions Meal → Course Occassion → Meal Meal → recipies Occassions → prep time Occassions → cook time	User ↔ Dietary Preferences User ↔ Recipies User ↔Grocery List User ↔ Store Locations User ↔ Ingredients User ↔ Saved

		Recipe ↔ Ingredients Recipe ↔ Dietary Preferences Meal ↔ Recipies StoreLocation ↔ Ingredients Grocery List ↔ Ingredients Grocery List ↔ Courses Grocery List ↔ Occasions Prep Time ↔ Occasions Prep Time ↔ recipies Cost ↔ Occasions Cost ↔ Ingredients Cost ↔ recipies Cost ↔ Store Locations

dbdesigner

Columns

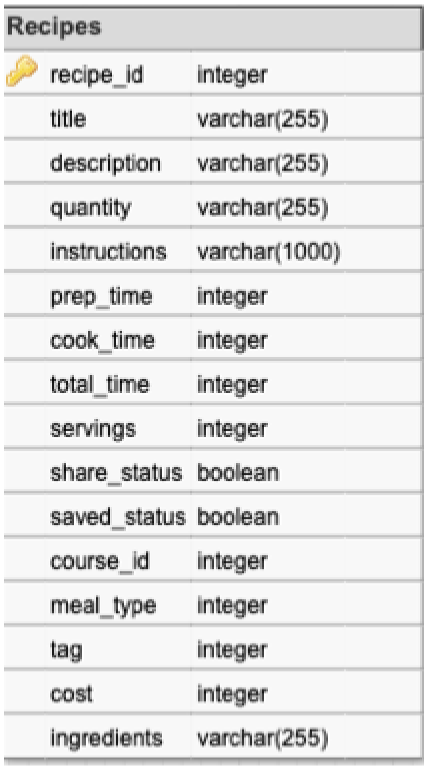


User	
user_id	integer
first_name	varchar(50)
last_name	varchar(50)
date_of_birth	date(8)
zip	integer(5)
username	varchar(200)
password	varchar(200)
dietary_preference	varchar(200)
status	boolean

→ stored for creating and sharing purposes

→ stored for legal purposes, to check for the user's age
→ stored to find nearest store locations
→ stored for log-in purposes

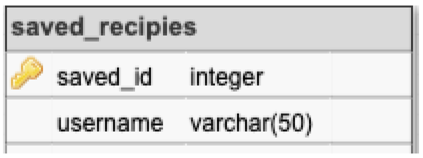
→ stored for creation and searching purposes



Recipes	
recipe_id	integer
title	varchar(255)
description	varchar(255)
quantity	varchar(255)
instructions	varchar(1000)
prep_time	integer
cook_time	integer
total_time	integer
servings	integer
share_status	boolean
saved_status	boolean
course_id	integer
meal_type	integer
tag	integer
cost	integer
ingredients	varchar(255)

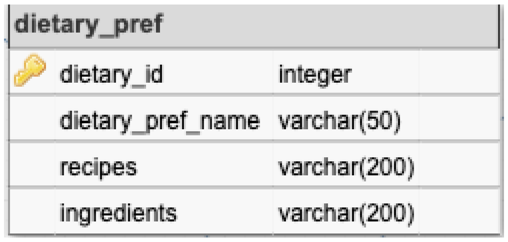
→ stored for ease of searching
→ stored to entice user to save, or make recipe.
→ stored to see how many ingredients are needed.
→ stored so users understand how to make each recipe
→ stored for users to determine if they want to use the recipe and as part of the instructions

→ stored for yield purposes
→ stored to see if the system should share the recipe
→ stored to see if the system should save the recipe
→ stored for the user to identify if it's an apetizer, main course, etc.
→ stored to see if it's a recipe for breakfast, lunch, dinner, etc.



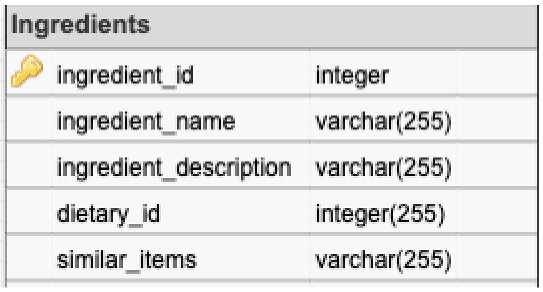
saved_recipies	
🔑 saved_id	integer
username	varchar(50)

→ stored to create a link between the user and saved recipes



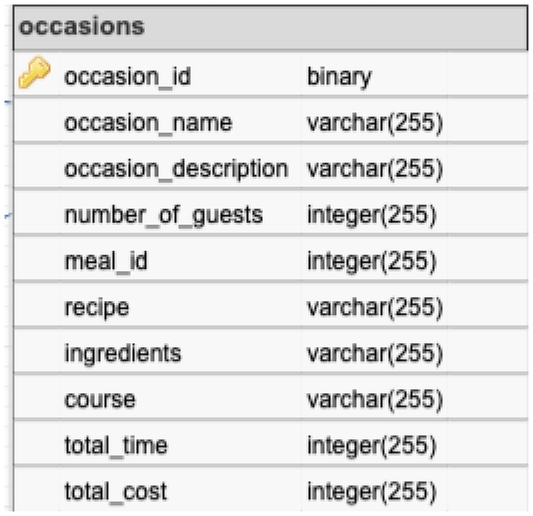
dietary_pref	
🔑 dietary_id	integer
dietary_pref_name	varchar(50)
recipes	varchar(200)
ingredients	varchar(200)

→ stored so that the user can specify their dietary preferences
→ stored so the user can search/view pased on their preferences
→ stored so the system knows which ingredients relate to which preferences



Ingredients	
🔑 ingredient_id	integer
ingredient_name	varchar(255)
ingredient_description	varchar(255)
dietary_id	integer(255)
similar_items	varchar(255)

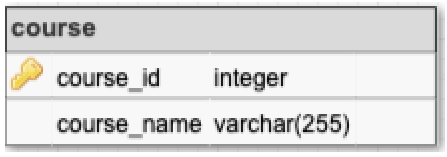
→ stored for recipes, preferences, meals, etc.
→ stored for recipes, preferences, meals, etc.
→ stored so the system knows which ingredients relate to which preferences
→ stored for swap suggestion



occasions	
🔑 occasion_id	binary
occasion_name	varchar(255)
occasion_description	varchar(255)
number_of_guests	integer(255)
meal_id	integer(255)
recipe	varchar(255)
ingredients	varchar(255)
course	varchar(255)
total_time	integer(255)
total_cost	integer(255)

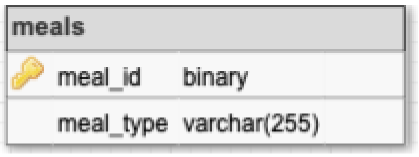
→ stored for visibility purposes
→ stored for planning purposes
→ stored for yield purposes
→ stored to identify which mealtype it is: breakfast, lunch, etc.
→ stored to see which recipes are associated with the event.
→ stored to see which ingredients are required.

→ stored to see if it's a entree, main course, etc.
→ stored for planning purposes on the user side.
→ stored for planning purposes on the user side.



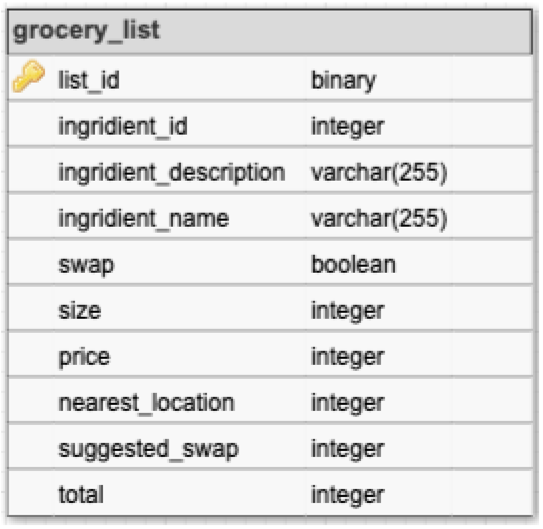
course	
🔑 course_id	integer
course_name	varchar(255)

→ stored to help the user identify if it's an entree, main course, etc.



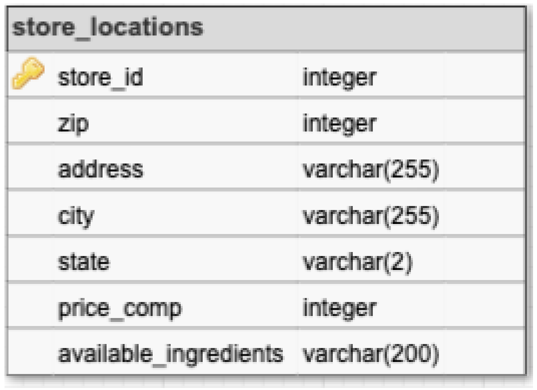
meals	
🔑 meal_id	binary
meal_type	varchar(255)

→ stored to identify which mealtype it is: breakfast, lunch, etc.



grocery_list	
list_id	binary
ingridient_id	integer
ingridient_description	varchar(255)
ingridient_name	varchar(255)
swap	boolean
size	integer
price	integer
nearest_location	integer
suggested_swap	integer
total	integer

→ stored to see the name of the ingredient required
→ if user selects it, the system must populate options
→ helps user identify how much they need vs how much it contains
→ helps calculate the cost and price comp for each item
→ helps user identify where to get the item
→ stored to find similar items that could replace the ingredient if needed
→ helps user identify total cost



store_locations	
store_id	integer
zip	integer
address	varchar(255)
city	varchar(255)
state	varchar(2)
price_comp	integer
available_ingredients	varchar(200)

→ helps the system identify stores near the user
→ provides the user with a physical address to pick up the groceries

→ shows the price difference between one store and another
→ shows ingredients available for purchase at that location

Postgress Sandbox

```
CREATE TABLE profile(
  profile_id SERIAL PRIMARY KEY,
  first_name VARCHAR(50),
  last_name VARCHAR(50),
 date_of_birth INTEGER NOT NULL,
 zip INTEGER NOT NULL,
 username VARCHAR(50),
 password VARCHAR(50),
 status BOOLEAN
);
```

NOTE: ALTER TABLE to add “dietary_preference INTEGER NOT NULL REFERENCES dietary_pref(dietary_id),” once the table has been created.

```
  CREATE TABLE dietary_pref(
    dietary_id SERIAL PRIMARY KEY,
    dietary_pref_name VARCHAR(50)
);
```

NOTE: ALTER TABLE to add:

```
    associated_recipies VARCHAR(200) NOT NULL REFERENCES recipes(recipe_id),
    associated_ingredients VARCHAR(200) NOT NULL REFERENCES ingredients(ingredient_id)
```

```
  CREATE TABLE saved_recipes(
    saved_id SERIAL PRIMARY KEY,
    username_associated VARCHAR(50) NOT NULL REFERENCES profile(username)
);
```