We want to create a recipe creating/sharing and grocery list app. You'll be planning out what tables we'll need, what information they'll store, and how the data will relate to each other.

Features

- users can sign into the app with their email and password
- users can create recipes with ingredients and instructions
- recipes can be marked as public or private
- users can view other people's recipes
- ingredients from recipes can be added to user's grocery lists
- users can create their own occasions and assign recipes to occasions

Step 1

Brainstorming

- **A user table** with user id# (serial primary key) first name, last name, username, email, password (maybe in a different table)
    - One to many relationships between user table and recipes table

- **An ingredients table** with ingredient id# (serial primary key), name of ingredient, type of ingredient (vegetable, fruit, spice, etc.)

- **A recipes table** with recipe id# (serial primary key), recipe name, public/private, and instructions
    - Many-to-many relationship between ingredients and recipe table (association table linking them)

- **A grocery list table** with grocery list id# (serial primary key), and grocery list name
    - One-to-many relationship between user table and grocery list table
    - Many-to-many relationship between grocery list table and ingredients table (association table linking them)

- **An occasions table** with occasions list id# (serial primary key), and occasions list name
    - One-to-many relationship between user table and occasions list table
    - Many-to-many relationship between occasions list table and recipes table (association table linking them)

Step 2

Table Ideas

1. **A user table**: this table will hold user information, with each row dedicated to a different user. It will contain important information, like name, username, email, and password.

2. **A password table:** this is needed in order for the password to be kept safe in a different file.

3. **An ingredients table:** this table will hold information about the ingredients available to add to grocery lists, each row will be an individual ingredient/food item

4. **A recipes table**: this table will hold information about user-generated recipes, each row will be a different recipe. It will include a public/private column and an instructions column

5. **A grocery list table:** this table will hold information about user-generated grocery lists.

6. **An occasions table:** this table will hold information about user-generated occasions lists.

Step 3

Relationships

One-to-one

1. **User table to password table:** *Each user only has one password associated with it and each password only has one user associated with it.*

One-to-many

1. **User table to recipes table:** *each user can have multiple recipes, but a recipe can only have one user who created it. Since there is no feature for other users to comment on or like a recipe, this is one-to-many. Many users can view a recipe, but we do not need to store the viewing data at present. There could potentially be reasons to store it, but it doesn't seem necessary based on the instructions.*

2. **User table to grocery list table:** *each user can have multiple grocery lists, but a grocery list can only have one user who created it (because there is no feature for other users to comment on/like/view grocery lists).*

3. **User table to occasions table:** *each user can have multiple occasions, but an occasion can only have one user who created it (because there is no feature for other users to comment on/like/view occasions).*

Many-to-many:

1. Ingredients to Recipes: *Ingredients may appear in multiple recipes and inversely, many recipes may contain the same/many ingredients*
2. Grocery list to Ingredients: *Ingredients may appear on different grocery lists and inversely, many grocery lists may contain the same/many ingredients*

3. Occasions to Recipes: *A recipe can be assigned to multiple occasions (nye, Halloween, etc), as well as an occasion can store multiple Recipes (how to make bread, lasagna).*

Columns

User table
User_id: We are storing this to keep track of the users. We are using integers because this needs to be an incremented whole number.

First_name: We are storing this to keep track of the users' names. We are using VARCHAR because names are strings and we want their length to be finite.

Last_name: We are storing this to keep track of the users' names. We are using VARCHAR because names are strings and we want their length to be finite.

Username: We are storing this to keep track of the users' usernames. We are using VARCHAR because usernames are strings and we want their length to be finite. We are requiring this to be unique so we do not have repeating usernames.

Email: We are storing this to keep track of the users' emails. We are using VARCHAR because emails are strings and we want their length to be finite. We are requiring this to be unique so we do not have repeating emails.

Password table
Password_id: We are storing this to keep track of the password. We are using integers because this needs to be an incremented whole number.

Password: We are storing this to keep track of the users' passwords. We are using VARCHAR because passwords are strings and we want their length to be finite.

User_id: We are creating a one-to-one relationship here with the users table. We are using integers because the user_id column in the users table is an integer and it needs to match. We have selected "unique" because I believe that creates a one-to-one relationship(?).

Ingredients table
Ingredients_id: We are storing this to keep track of the ingredients. We are using integers because this needs to be an incremented whole number.

Name: We are storing this to keep track of the name of the ingredient. We are using VARCHAR because it will be a string and we want its length to be finite. We have selected "unique" to avoid repeating ingredients.

Type: We are storing this to keep track of the type of the ingredient. We are using VARCHAR because it will be a string and we want its length to be finite.

Recipes table

Recipes_id: We are storing this to keep track of the recipes. We are using integers because this needs to be an incremented whole number.

Name: We are storing this to keep track of the name of the recipe. We are using VARCHAR because it will be a string and we want its length to be finite.

is_private: We are storing this to keep track of the privacy setting of the recipe. We are using a boolean because this would be a true/false response.

Recipe_instructions: We are storing this to keep track of the instructions for the recipe. We are using VARCHAR because it will be a string and we want its length to be finite.

User_id: We are creating a one-to-many relationship here with the users table. We are using integers because the user_id column in the users table is an integer and it needs to match. We have not selected "unique" because I believe that creates a one-to-many relationship(?).

Part 3:
Go to https://postgres.devmountain.com/

- Write a create table statement for each of your tables
- Copy and paste each of the statements into your doc so you can keep track of them and turn them in

```
CREATE TABLE users (
user_id SERIAL PRIMARY KEY,
first_name VARCHAR(40),
last_name VARCHAR(40),
username VARCHAR(40),
email VARCHAR(40)
);
```

```
INSERT INTO users(first_name, last_name, username, email)
VALUES('Paul', 'Tonn', 'paultonn' ,'paultonn@gmail.com'),
('Nancy', 'Botwin', 'nancybotwin', 'nancybotwin@gmail.com'),
('Travis', 'Barker', 'travisbarker' , 'travisbarker@gmail.com'),
( 'Kylie', 'Rogers', 'kylierogers', 'kylierogers@gmail.com'),
('Stormy', 'Celebrity', 'stormycelebrity', 'stormycelebrity@gmail.com')
```

```
CREATE TABLE password (
password_id SERIAL PRIMARY KEY,
user_id INTEGER REFERENCES users(user_id),
password VARCHAR(50)
);
```

```
INSERT INTO password(user_id, password)
VALUES (1, ‘qwerty12345’),
(2, ‘password123’),
(3, ‘mypassword’),
(4, ‘ilovecats888’),
(5, ‘plantsarecool99’)


CREATE TABLE ingredients (
ingredients_id SERIAL PRIMARY KEY,
Name VARCHAR(50),
Type VARCHAR(50)
);

CREATE TABLE recipes (
recipes_id SERIAL PRIMARY KEY,
Name VARCHAR(50),
Is_private BOOLEAN,
Instructions VARCHAR(10000),
User_id INTEGER REFERENCES users(user_id)
);

CREATE TABLE grocery_lists (
grocerylists_id SERIAL PRIMARY KEY,
Name VARCHAR(50),
User_id INTEGER REFERENCES users(user_id)
);

CREATE TABLE occasions (6
occasions_id SERIAL PRIMARY KEY,
Name VARCHAR(50),
User_id INTEGER REFERENCES users(user_id)
);


CREATE TABLE grocery_lists_ingredients (
grocerylists_ingredients_id SERIAL PRIMARY KEY,
grocerylists_id INTEGER REFERENCES grocery_lists(grocerylists_id),
ingredients_id INTEGER REFERENCES ingredients(ingredients_id)
);

CREATE TABLE ingredients_recipes (
ingredients_recipes_id SERIAL PRIMARY KEY,
ingredients_id INTEGER REFERENCES ingredients(ingredients_id),
recipes_id INTEGER REFERENCES recipes(recipes_id)
);

CREATE TABLE occasions_recipes (
occasions_recipes_id SERIAL PRIMARY KEY,
occasions_id INTEGER REFERENCES occasions(occasions_id),
recipes_id INTEGER REFERENCES recipes(recipes_id)
);
```