DB Planning

Recipe App

Brainstorm:

- users-email/username and password
- Recipes-ingredients and instructions,
- Public/private recipes
- Grocery list-add ingredients, per user
- Occasions-users and recipes

Tables Ideas:

- Users-hold the user's profile info such as username/name, email,
    - Each row is a user
    - User_id, full_name, email
- Auth-hold the user's password and username
    - Each row is a user's info
    - auth_id, user_id, username, password
- Recipes-holds the recipe name and link to ingredients,instructions, and privacy
    - Each row is recipe name, ingredients link, instructions link, and privacy boolean
    - Recipe_id,user_id, recipe_name, ingredients_id, instructions_id,ingredient_amount, privacy
- Grocery-Holds what user created/own list, and the required ingredients
    - Each row is user, and ingredients
    - Grocery_id, grocery_name, user_id, ingredients_id
- Occasions-holds the user/owner and recipes
    - Each row is a user and recipe and occasion name
    - Occasion_id, occasion_name, user_id, recipe_id
- Instructions-holds the body instruction for a recipe
    - Each row will hold a set of instructions
    - Instructions_id, body
- Ingredients- hold the ingredient details
    - Ingredient_id, ingredient_name,recipe_id

Relationships:

One to One:

- User to Auth(only need the user and auth communicating)
- Instructions to recipes(only one instruction can match a recipe)
- Ingredient to recipes(only one ingredient can go per recipe due to amount size)

One to Many:

- User to grocery(a grocery list is tied to a specific user but they can have multiple grocery list)

Many to many

- User to Recipes(Associations)(Users can have multiple recipes and recipes can be owned by many users)
- User to Occasions(Middle)(Occasions will link users and recipes together by grouping recipes)
- Ingredient to grocery(many ingredients can go into many grocery lists)

Columns:

All tables contains a primary key(integer) to differentiate each row

User:

- Full_name(varchar): limit the user to having a long name and to customize the user's profile to themselves.
- email(varchar): connect the user's profile to their email for login and support purposes

Auth:

- user_id(int):connection between the user table and auth to confirm if credentials are correct.
- username(varchar): allows the user to login with either username or email
- password(varchar): stored as hash

Grocery:

- user_id(int): connects to show what the specific user's grocery list is
- grocery_name(varchar): the name of the list so the user can create multiple lists
- ingredient_id(int): connects to which ingredient belongs with what grocery_id

Recipe:

- user_id(int):links to who's the owner of the recipe
- recipe_name(varchar):the name of the recipe
- ingredient_id(int): links what ingredients needed for the recipe
- instruction_id(int): links what set of instructions needed for the recipe
- privacy(boolean): if this recipe should be private or public

Ingredient:

- ingredent_amount(int): the amount or size of each ingredient needed
- ingredient_name(varchar): hold what the ingredient is

Instruction:

- body(text):hold the full instructions of each recipe

Occasion:

- occasion_name(varchar):the title of the occasion
- user_id(int): who's the owner/creator of the occasion
- recipe_name(int): what recipe is on each occasion. Since each recipe_id is different, we grab by the name since it'll have multiple rows due to ingredients

SQL

User:

```
CREATE TABLE users(
 user_id SERIAL PRIMARY KEY,
 full_name VARCHAR(255) NOT NULL,
 email VARCHAR(255) NOT NULL);
```

Auth:

```
CREATE TABLE auth(
   auth_id SERIAL PRIMARY KEY,
        user_id INTEGER REFERENCES users(user_id),
        username VARCHAR(255),
        password VARCHAR(255)
);
```

Grocery:

```
CREATE TABLE grocery(
   grocery_id SERIAL PRIMARY KEY,
        user_id INTEGER REFERENCES users(user_id),
        grocery_name VARCHAR(255),
        ingredient_id INTEGER REFERENCES
ingredient(ingredient_id)
);
```

Recipe:

```
CREATE TABLE recipe(
   recipe_id SERIAL PRIMARY KEY,
   user_id INTEGER REFERENCES users(user_id),
        recipe_name VARCHAR(255),
        ingredient_id INTEGER REFERENCES
ingredient(ingredient_id),
        instruction_id INTEGER REFERENCES
instruction(instruction_id),
        privacy BOOLEAN
);
```

Ingredient:

```
CREATE TABLE ingredient(
   ingredient_id SERIAL PRIMARY KEY,
   ingredient_name VARCHAR(255),
   ingredient_amount INTEGER
);
```

Instruction:

```
CREATE TABLE instruction(
   instruction_id SERIAL PRIMARY KEY,
        body TEXT
);
```

Occasion:

```
CREATE TABLE occasion(
   occasion_id SERIAL PRIMARY KEY,
        occasion_name VARCHAR(255),
        user_id INTEGER REFERENCES users(user_id),
        recipe_name VARCHAR(255) REFERENCES
recipe(recipe_name)
);
```

Inserts

Users:

```
INSERT INTO users (full_name, email)
VALUES ('Mitchell Nguyen', ‘moneymitch@example.com'),
       ('Vector', ‘moonstealer@example.com'),
       ('Kevin Durant', ‘easymoneysniper@example.com');
```

Auth:

```
INSERT INTO auth (user_id, username, password)
VALUES (1, 'moneymitch', 'password123'),
       (2, 'vexxy', '123password'),
       (3, 'kd35', 'imissrussel');
```

Grocery:

```
INSERT INTO grocery (user_id, grocery_name, ingredient_id)
VALUES (1, 'Weekly Shopping List', 1),
       (1, 'Weekly Shopping List', 2),
       (1, 'Weekly Shopping List', 3);
```

Recipe:

```
INSERT INTO recipe (user_id, recipe_name, ingredient_id, instruction_id,
privacy)
VALUES (1, 'Spaghetti’, 1, 1, false),
       (1, 'Spaghetti’, 1, 2, false),
       (1, 'Spaghetti’, 1, 3, false);
```

Ingredient:

```
INSERT INTO ingredient (ingredient_name, ingredient_amount)
VALUES ('Noodles', 2),
       (‘Tomato Sauce’, 2),
       (‘Ground Beef’, 3);
```

Instruction:

INSERT INTO instruction (body)
VALUES (

‘Steps the the perfect spaghetti:

1. In a large pot of salted, boiling water, cook spaghetti according to package directions until al dente.
2. Heat oil over medium-high heat. Add the ground beef and cook until browned and cooked through.
3. Add the canned tomatoes and bring to a simmer. Reduce heat to low and simmer for 30-40 minutes or until the sauce has thickened and reduced.
5. Season the sauce with salt and pepper to taste. Serve the spaghetti with the sauce.');

Occasion:

INSERT INTO occasion (occasion_name, user_id, recipe_name)
VALUES ('Family Dinner', 1, ‘Spaghetti’);