Grocery list app / Recipe Sharing

Brainstorming!!

-so this app would have users upload their own recipes and see other peoples recipes
- the recipes will have ingredients and instructions
- you can mark the recipes as public or private so a boolean
- users can add the ingredients of a recipe to their grocery list
- users can also assign recipes to specific occasions.

Users

- user_id SERIAL PRIMARY KEY,
- email VARCHAR(50), - because its a set amount of letters
- password VARCHAR(50), - because its a set amount of letters
- grocery_list TEXT, - this could be an infinite number of letters
- recipes_created TEXT, - this could be an infinite number of letters
- overall_chef_rating INTEGER, - this is a rating so it needs a number
- friends_list TEXT- this could be an infinite number of letters

Recipes

- recipe_id SERIAL PRIMARY KEY
- chef INTEGER REFERENCES users(user_id), - this is a an id number that connects to users
- occasion_type VARCHAR(30) REFERENCES occasions(occasion_id), - this is a set amount of letters (limited to only a couple occasions)
- public_or_private BOOLEAN, - this can only be true or false so its a boolean
- ratings INTEGER - this is a number that is calculated for each chef

Friends

- friend_id SERIAL PRIMARY KEY,
- user_id INTEGER REFERENCES users(user_id), - this is an id number that connects to users
- friended Boolean - this can only be true or false so its a boolean

Occasions

- occasion_id SERIAL PRIMARY KEY,
- occasion_type VARCHAR(30), this is a set amount of letters (limited to only a couple occasions)
- recipe_of_occasion TEXT - this could be an infinite number of of letters

Grocery list

- grocery_list_id SERIAL PRIMARY KEY,
- ingredients TEXT, - this could be an infinite number of characters
- store_id INTEGER REFERENCES store(store_id) - this is a store number so its an integer and references Store

STORE

- store_id SERIAL PRIMARY KEY,
- location TEXT - this is an unlimited number of letters
- ingredients TEXT - this is an unlimited number of letters

RELATIONSHIPS

USERS => RECIPES: One to Many
- users can have many recipes but a recipe can only have one author.

USERS => FRIENDS: Many to Many
- Users can have many friends, and friends can have many users

RECIPES => OCCASIONS: one to Many
- Recipes can have only one occasion but an occasion can be assigned to many reciepes

-
USER =>GROCERY_LIST: One to One
- A user can only have one grocery list and a grocery list can only have one user

ingredients=>Store : Many to Many
- A store can have multiple ingredients and an ingredient list can be fulfilled at multiple stores.
-

GROCERY_LIST => Ingredients Many to Many
- A grocery store can have many ingredients and ingredients can be on multiple grocery lists

```
CREATE TABLE users(
        user_id SERIAL PRIMARY KEY,
        email VARCHAR(50),
        password VARCHAR(50),
        grocery_list TEXT,
        recipes_created TEXT,
        overall_chef_rating INTEGER,
        friends_list TEXT
);

CREATE TABLE recipes(
        recipe_id SERIAL PRIMARY KEY,
        chef INTEGER REFERENCES users(user_id),
        occasion_type VARCHAR(30) REFERENCES occasions(occasion_id),
        public_or_private BOOLEAN,
        ratings INTEGER
);

CREATE TABLE friends(
        friend_id SERIAL PRIMARY KEY,
        user_id INTEGER REFERENCES users(user_id),
        friended BOOLEAN
);

CREATE TABLE occasions(
        occasion_id SERIAL PRIMARY KEY,
        occasion_type VARCHAR(30),
        occasion_recipe TEXT
);

CREATE TABLE grocery_list(
        grocery_list_id SERIAL PRIMARY KEY,
        ingredients TEXT,
        store_id INTEGER REFERENCES store(store_id)
);

CREATE TABLE store (
        store_id SERIAL PRIMARY KEY,
        location TEXT,
        ingredients TEXT
);
```

```sql
-- INSERT INTO users(email, password, grocery_list, recipes_created, overall_chef_rating,
friends_list)
-- Values('notreal@aol.com', '123', 'carrots, pepperoni, apples, oranges, thai tea, baking soda',
--          'My recipe for pepperoni pizza!', 7, 'JellyBean, Garrett, Joely, Scott');



SELECT * FROM users;
```