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

Brainstorming

Sign in process

- Email
- Password
- User ID
- First Name
- Last Name
- Password hint

Users can create recipes

- Ingredients text
- Instructions text
- User ID
- Recipe ID
- isPublic ?

Grocery list

- User ID
- Ingredients list
- Category ID

Occasions

- Occasion ID
- Recipe ID
- Occasion Name
- User ID

Users can follow each other

- User ID 1
- User ID 2
- isBlocked?

Table ideas

User Table

This table will hold information about the users, each row will be an individual user.

- User ID => one to one
- UserName
- Email
- Password
- First Name
- Last Name
- Password hint

Recipe Table

This table will hold information about the recipes, each row will be a recipe with a unique Recipe ID.

- User ID => User Table => one to many
- Recipe ID => one to one
- Recipes name
- isPublic?
- Instructions

Ingredients Table

This table will hold information about the ingredients pertaining to the recipes.

- Ingredients ID => one to one
- Recipe ID => Recipe Table => one to many
- Ingredients Name

Category Table

This table will hold consistent information about the categories pertaining to the ingredients.

- Category ID => one to one
- Ingredients ID => one to one
- Category Name

Occasions Table

This table will hold information about the Occasions in relation to the Recipes. Each row will be a unique Occasion.

- Recipe ID => Recipe Table => one to many
- Occasion ID => one to one
- Occasion Name
- User ID (Creator) => User table => one to one

Follow table

- FollowID => one to one
- User ID 1 => many to many
- User ID 2 => many to many
- isBlocked?

Relationships

One-to-one

User Table - User table will have User ID which is a one to one relation since there is only one user for each User ID.

Recipe Table - Recipe Table will have Recipe ID which is a one to one relationship since there is only one Recipe for each Recipe ID.

Ingredients Table - Ingredients Table will have Ingredients ID which is a one to one relationship since there is only one Recipe for each Recipe ID

Category Table - Category table will have Category Id which is a one to one relationship since there is only one Category for each Category ID.
It also has Ingredients ID which is a one to one relationship since the Ingredient will always have the same category.

Occasions Table - Occasions Table will have Occasions ID which is a one to one relationship since there is only one occasion for each occasion ID.

One-to-many

Recipe Table-

-has User ID which is one to many since the User can create many different recipes but there is only one user.

Ingredients Table - In the ingredients table, the unique recipe ID defines many ingredients.

Occasions Table - In the Occasions table, the unique recipe ID defines many Occasions.

Many-to-many

Follow table-

Has UserID 1 and UserID2 which are both many to many since many users can follow many users.

Columns

User Table

- User ID => serial - this is the primary key and we want it to automatically increment.
    - UserID is stored so that we know how to track the unique values of the User table.
- UserName => varchar -This will be characters that we want to limit to save space.
    - Username is stored so that users do not have to display their email to the public.
- Email => varchar-This will be characters that we want to limit to save space.
    - Used is stored so that we have a unique identifier for when users sign in.
- Password => varchar-This will be characters that we want to limit to save space.

        - Stored so that we can determine if criteria is met upon signing in.
    - First Name => varchar-This will be characters that we want to limit to save space.
        - Stored for a more personalized user experience.
    - Last Name => varchar-This will be characters that we want to limit to save space.
        - Stored for a more personalized user experience.
    - Password hint => varchar-This will be characters that we want to limit to save space.
        - Stored to help some users who choose to add a hint to help them remember their password.

Recipe Table

- User ID => integer -this is a foreign key which is previously defined as a primary key which is always an integer.
    - Stored so that we can link the Recipes to the user who created them.
- Recipe ID => serial -this is the primary key and we want it to automatically increment.
    - Stored so that we know how to track the unique values of the Recipe table.
- Recipe-name => varchar-This will be characters that we want to limit to save space.
    - Stored so that we have a title for each recipe.
- Recipe-description => varchar-This will be characters that we want to limit to save space.
    - Stored so that we have a description for each recipe.
- isPublic? => boolean- because we need to know whether it's a true or false statement for the public. If false then Private.
    - Stored so that we know whether it’s public or Private.

- Instructions => varchar-This will be characters that we want to limit to save space.
    - For users to see how to complete the recipe.

Ingredients Table

- Ingredients ID => serial-this is the primary key and we want it to automatically increment.
    - Stored so that we know how to track the unique values of the ingredients table.
- Recipe ID=> integer -this is a foreign key which is previously defined as a primary key which is always an integer.
    - Stored so that we can link the ingredients to their corresponding recipes.
- Ingredients Name => varchar-This will be characters that we want to limit to save space.
    - Stored so we know the specific name for the ingredient

Category Table

- Category ID => serial-this is the primary key and we want it to automatically increment.
    - Stored so Stored so that we know how to track the unique values of the category table
- Ingredients ID=> integer -this is a foreign key which is previously defined as a primary key which is always an integer.
    - Stored so that we can link the categories to their corresponding ingredients.
- Category Name => varchar-This will be characters that we want to limit to save space.
    - So that we can identify the name of each category.

Occasions Table

- Recipe ID=> integer -this is a foreign key which is previously defined as a primary key which is always an integer.
    - Stored so that we can link the occasions to the recipes.
- Occasion ID => serial-this is the primary key and we want it to automatically increment.
    - Stored so that we know how to track the unique values of the Occasions table.
- Occasion Name => varchar-This will be characters that we want to limit to save space.
    - Stored so we know the names of the occasion.

- User ID (Creator)=> integer -this is a foreign key which is previously defined as a primary key which is always an integer.
    - Stored so that we can link the occasions to the Users that created them.

Follow table

- FollowID => serial-this is the primary key and we want it to automatically increment.
    - Stored so that we know how to track the unique values of the follow table.
- User ID 1=> integer -this is a foreign key which is previously defined as a primary key which is always an integer.
    - -this is a foreign key which is previously defined as a primary key which is always an integer.
- User ID 2=> integer -this is a foreign key which is previously defined as a primary key which is always an integer.
    - -this is a foreign key which is previously defined as a primary key which is always an integer.
- isBlocked?=> boolean- because we need to know whether it's a true or false statement for the public. If false then Private.
    - Stored so that we know whether a user is blocked or not.

```
CREATE TABLE userTable(
        userID serial PRIMARY KEY,
        user_name varchar(30) unique NOT NULL,
        email varchar(50) unique NOT NULL,
        password varchar(20) NOT NULL,
        first_name varchar(30),
        last_name varchar(30),
        password_hint text
 );

 CREATE TABLE recipeTable(
   recipeID serial PRIMARY KEY,
        userID integer NOT NULL REFERENCES
userTable(userID),
```

```
        recipe_name varchar(200) NOT NULL,
        recipe_description varchar(500) NOT NULL,
        isPublic boolean NOT NULL,
        instructions varchar(500) NOT NULL
    );

  CREATE TABLE ingredientsTable(
    ingredientsID serial PRIMARY KEY,
        ingredients_name varchar(20) NOT NULL,
        recipeID integer REFERENCES
recipeTable(recipeID)

    );

 CREATE TABLE OccasionsTable (
  occasionID SERIAL PRIMARY KEY,
  occasion_name VARCHAR(30) NOT NULL,
  userID INTEGER NOT NULL REFERENCES
UserTable(UserID)
);


 CREATE TABLE categoryTable(
    categoryID serial PRIMARY KEY,
        ingredientsID integer NOT NULL REFERENCES
ingredientsTable(ingredientsID),
        category_name varchar(30)NOT NULL

      );
  CREATE TABLE followTable(
    followID serial PRIMARY KEY,
        UserID1 integer NOT NULL REFERENCES
userTable(UserID),
        UserID2 integer NOT NULL REFERENCES
userTable(UserID),
        isBlocked boolean
       );
```

```
INSERT INTO userTable(user_name, email, password,
first_name, last_name, password_hint)
VALUES ('katrynaY', 'katrynay@gmail.com',
'goodPassword', 'Katryna', 'Yaworski', 'its a good one'),
('IsirO', 'IsirO@gmail.com', 'badPassword', 'Isir',
'Osman', 'its a bad one'),
('MohamedN', 'Mohamedn@gmail.com',
'decentPassword', 'Mohamed', 'Nourin', 'its a decent
one');


select * From userTable
```

```
UserTable {

    UserID serial pk increments

    user-name varchar(30) unique

    email varchar(50) unique

    password varchar(20)

    first-name varchar(30)

    last-name varchar(30)

    password-hint text

}
```

RecipeTable {

recipeID serial pk increments

userID integer >* UserTable.UserID

recipe-name varchar(200)

recipe-description varchar(500)

isPublic boolean

instructions text(500)

}

IngredientsTable {

ingredientsID serial pk increments

ingredients-name varchar(20)

recipeID integer >* RecipeTable.recipeID

}

CategoryTable {

categoryID serial pk increments

```
    ingredientsID integer >* IngredientsTable.ingredientsID

    category-name varchar(30)

}


OccassionsTable {

    occasionID serial pk increments

    occasion-name varchar(30)

    userID integer > UserTable.UserID

}


FollowTable {

    followID serial pk increments

    UserID1 integer *>* UserTable.UserID

    UserID2 integer *>* UserTable.UserID

    isBlocked boolean

}
```