Part 1, step 1a: Brainstorming

- **Sign in with email and password / OAuth**
- **Create recipes**
    - Can be public or private
    - Can view other peoples recipes
    - Can give a rating to the recipe with a review
- **Can create shopping lists**
    - Add ingredients from recipes to shopping list
- **Users can create profile**
    - Add profile image and other shareable details
- **Can follow or friend other users**
    - Can share recipes
- **Search**
    - Can find recipes based on difficulty or cuisine type
    - Can search for other users
- **Social Feed for friends to view recipes tried**
    - Displays difficulty level to make
    - Displays a picture and a review of the recipe shared
- **Can leave reviews on recipes**
- **Can share recipes with social media e.g. Instagram, Twitter, Facebook**

Step 1b: User flows / user stories

1. The user logins and enters a new recipe for a chocolate cake. He/she/they enters the ingredients which includes chocolate sprinkles, flour, and eggs. The user then shares the recipe with his/her/their friends.
2. As a user I want to be able to view my collection of recipes under my profile so that I can quickly access them to use/share. Login>view profile>recipes>use/share
3. The user logs in to find a recipe to make to impress his boss. He sees that a friend of his recently tried to make lobster thermidor and gave it a good rating. The user then adds the ingredients to his cart and is able to find them at the grocery store. He then tries to make the recipe and fails. He leaves a bad review and unfriends his friend.
4. When a user tries a recipe they want to be able to post a picture, rating, and review of the recipe tried. User logs in>clicks + (for post)> Adds image, rating, review> posts to feed.
5. User logs in and posts a recipe for tacos. He then clicks the share button to share it to social media so his friend can see it.

Step 2: Table ideas

1. **Recipes:** holds info about the recipe, ingredients, reviews, etc
2. **Reviews:** needs info about user as well as the recipe they are reviewing
3. **Friends:** info about the user being friended and the person friending
4. **Share:** info about user, the review being shared, and the destination its being shared to
5. **Sign-In/Register:** columns for email, password plus user name, and other optional profile data if registering
6. **Shopping list:** columns include food_type, quantity to purchase, foreign_key to link tables together
7. **User table / profile:** password, email, description/summary, profile_image.
8. Multiple Recipes sorted from A-Z, Z-A, by Star Review
9. Feed, social feed for recipes shared by other users

Step 3: Relationships

One to one

1. One user has a Shopping List
2. User to profile. Each user can only have one profile

One to many

1. Recipes to reviews because each recipe can have many reviews, but a review is only for one specific product.
2. User to friends. Each user can have many friends
3. Share. One user can share with multiple/all users
4. Reviews. One user has many reviews

Many to many

1. Multiple posts can be shared to multiple people
2. Users can have many recipes, recipes can have many users

Part 2 step 2: Columns

User

- User_id (serial), password (varchar), email (varchar), profile_img (img), bio (varchar), recipe_id (serial), friend_id (serial)
- The data stored is pertinent to the user so they can login/have a personal profile/biopage and be able to view recipes and other users/friends.

Friends

- Friend_id (serial), recipe_id (serial)
- This data is stored so that the user has access to their friends profile and recipes
-

Shopping List

- ShoppingList_id (serial), ingredients (varchar)
- The data allows the user to keep track of the ingredients they will need to make their planned recipes

Recipe

- Recipe_id (serial), ingredients(varchar), review_id(serial)
- We need the recipe data to find out what ingredients we need to make the food dish.

Feed

- feed_id(serial), friend_id(serial), recipe_id(serial), review_id(serial)
- Feed is stored so that users can view each others posts, comment, and save on recipes.
- We chose the data types because each post needs a unique id for, user (user/friend), and recipe.

Share

- Share_id (serial), share_destination (varchar), recipe_id (from recipe)
- The data is useful to the user so they post a share link to their friends and with social media.

Reviews

- Review_id (serial), review_content(varchar)
- The review data is required in order to save the reviews that users write in this application.

<u>Part 3: Create table statement for each table</u>

Build order: 1 reviews -> 2 recipes -> 3 share -> 4 friends -> 5 feed -> 6 shopping list

User 7)

```
CREATE TABLE user_table(
  user_id SERIAL PRIMARY KEY,
  password VARCHAR(500),
  email VARCHAR(500),
  profile_img BYTEA,
  bio VARCHAR(10000),
  recipe_id SERIAL REFERENCES recipes(recipe_id),
  friend_id SERIAL REFERENCES friends(friend_id)
```

```
);
```

Friends 4)

```
CREATE TABLE friends (
  friend_id SERIAL PRIMARY KEY,
  recipe_id SERIAL REFERENCES recipes(recipe_id));
```

Shopping List 6)

```
CREATE TABLE shoppingList (
  shoppingList_id SERIAL PRIMARY KEY,
  ingredients VARCHAR(5000) REFERENCES recipes(ingredients)
)
```

Recipes 2)

```
CREATE TABLE recipes(
  recipe_id SERIAL PRIMARY KEY,
  ingredients VARCHAR(5000) UNIQUE,
  review_id SERIAL REFERENCES reviews(review_id)
)
```

Feed 5)

```
CREATE TABLE feed(
  feed_id SERIAL PRIMARY KEY,
  friend_id SERIAL REFERENCES friends(friend_id),
  recipe_id SERIAL REFERENCES recipes(recipe_id),
  review_id SERIAL REFERENCES reviews(review_id)
)
```

Share 3)

```
CREATE TABLE share(
  share_id SERIAL PRIMARY KEY,
  share_destination VARCHAR(100),
  recipe_id SERIAL REFERENCES recipes(recipe_id)
```

```
)
```

Reviews 1)

```
CREATE TABLE reviews(
  review_id SERIAL PRIMARY KEY,
  review_content VARCHAR(1000)
)
```

Dbdesigner.net - Schema