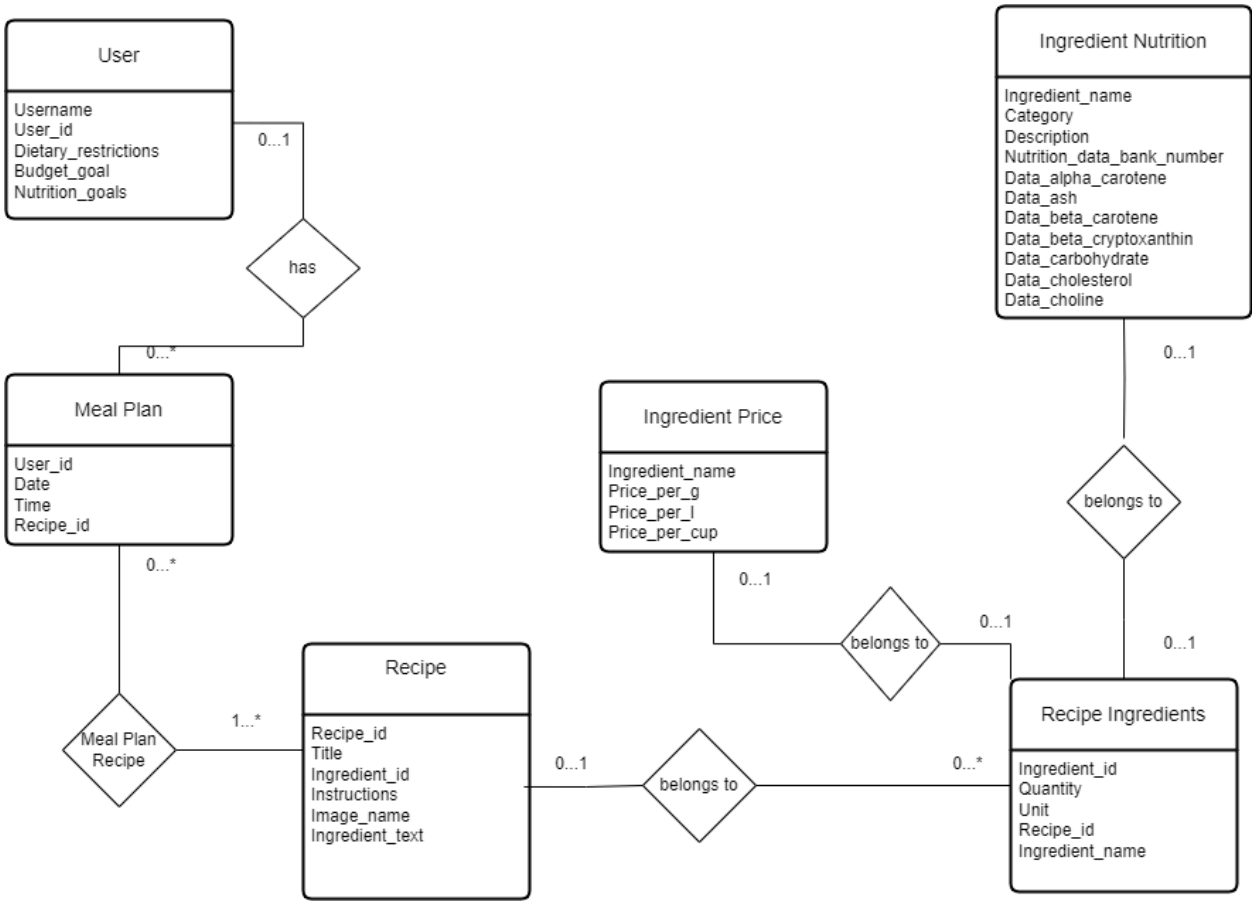
1. ER Diagram



2. Assumptions

There are 4 main sources of truth for our application. Ingredient price, recipe, recipe ingredients and ingredient nutrition. Ingredient price and ingredient nutrition are both primary key'd by ingredient name so can be in the same table. However, the nutritions come from a different source from price (most of which is mocked). Given such context, we felt that it would be more contextually appropriate to split the two tables.

When preprocessing the source recipe dataset, there were a lot of challenges due to the irregularity in format. Recipe ingredients had to be separated from the recipe table because each component of a recipe was effectively a collection of information (ingredient, quantity, unit) requiring a separate table. We assume that with the rigorous regex handling implemented in preprocessing, we were able to capture most of the original data from the recipe. We assume that there were 0 or many recipe ingredient records associated with a single recipe_id. We assume that if the unit was not parsed from the recipe data we can default to the standard 100 gram unit (which is the unit for the nutrition table). We assume that any recipe's ingredients will have at least 1 as the quantity even if the data was not correctly parsed.

As for the meal plan and user table, they are populated by user activity and we assume that each meal plan has an associated recipe and the user can plan out multiple meals.

Each relationship can be explained pretty easily. A user will be able to create multiple meal plans so it can range from 0 to *, or any. A meal plan will only be connected to one user however so it's from 0 to 1. A meal plan will contain a list of recipes so we range from 1 to however many but a recipe doesn't not necessarily have to be connected to a meal plan. From this recipe we have ingredients that belong to it and recipes should contain a certain amount of ingredients. These ingredients each have their own price and nutrition data, but they should only have 1 instance of this data since it's a unique ingredient.

3. Normalized database

Given every attribute in consideration, the following functional dependencies were found:

- Username → User_id, Dietary_restrictions, Budget_goal, Nutrition_goals
- User_id → Username, Dietary_restrictions, Budget_goal, Nutrition_goals
- User_id, Date, Time → Recipe_id
- Recipe_id → Title, Ingredient_id, Instructions, Image_name, Ingredient_text
- Ingredient_name → Price_per_g, Price_per_l, Price_per_cup
- Ingredient_id, Ingredient_name → Quantity, Unit, Recipe_id
- Ingredient_name, Description, Nutrition_data_bank_number → Category, Data_alpha_carotene, Data_ash, Data_beta_carotene, Data_beta_cryptoxanthin, Data_carbohydrate, Data_cholesterol, Data_choline

Given the keys/superkeys and the attributes they implied, the decomposition into entities were done as follows.

<u>User Schema</u>

Functional Dependencies

- Username → User_id, Dietary_restrictions, Budget_goal, Nutrition_goals
- User_id → Username, Dietary_restrictions, Budget_goal, Nutrition_goals

3NF Form: (Username, User_id, Dietary_restrictions, Budget_goal, Nutrition_goals)

<u>Meal Plan</u>

Functional Dependencies

- User_id, Date, Time → Recipe_id

3NF Form: (User_id, Date, Time, Recipe_id)

<u>Recipe</u>

Functional Dependencies

- Recipe_id → Title, Ingredient_id, Instructions, Image_name, Ingredient_text

3NF Form: (Recipe_id, Title, Ingredient_id, Instructions, Image_name, Ingredient_text)

<u>Ingredient Price</u>

Functional Dependencies

- Ingredient_name → Price_per_g, Price_per_l, Price_per_cup

3NF Form: (Ingredient_name → Price_per_g, Price_per_l, Price_per_cup)

<u>Recipe Ingredients</u>

Functional Dependencies

- Ingredient_id, Ingredient_name → Quantity, Unit, Recipe_id

3NF Form: (Ingredient_id → Quantity, Unit, Recipe_id, Ingredient_name)

Ingredient Nutrition

Functional Dependencies

- Ingredient_name, Description, Nutrition_data_bank_number → Category, Data_alpha_carotene, Data_ash, Data_beta_carotene, Data_beta_cryptoxanthin, Data_carbohydrate, Data_cholesterol, Data_choline

3NF Form: (Ingredient_name, Description, Nutrition_data_bank_number, Category, Data_alpha_carotene, Data_ash, Data_beta_carotene, Data_beta_cryptoxanthin, Data_carbohydrate, Data_cholesterol, Data_choline)

The above functional dependencies cover every single attribute and cannot be further reduced or combined.

4. Relational Schema:

a. User(Username:VARCHAR(100), User_id:INT [PK],Dietary_restrictions:VARCHAR(MAX), Budget_goal:INT, Nutrition_goals:VARCHAR(MAX))
b. Meal Plan(User_id:INT [FK to User.User_id], Date:DATE, Time:TIME, Recipe_id:VARCHAR(100))
c. Recipe(Recipe_id:INT [PK], Title:VARCHAR(100), Ingredient_id:INT [FK to Recipe Ingredients.Ingredient_id], Instructions:VARCHAR(MAX), Image_name:VARCHAR(500), Ingredient_text:VARCHAR(MAX))
d. Ingredient Price(Ingredient_name:VARCHAR(100) [PK], Price_per_g:DECIMAL, Price_per_l:DECIMAL, Price_per_cup:DECIMAL)
e. Recipe Ingredients(Ingredient_id:INT [PK], Quantity:INT, Unit:VARCHAR(50), Recipe_id:INT [FK to Recipe.Recipe_id], Ingredient_name:VARCHAR(100))
f. Ingredient Nutrition(Ingredient_name:VARCHAR(100) [PK], Category:VARCHAR(50), Description:VARCHAR(MAX), Nutrition_data_bank_number:INT, Data_alpha_carotene:INT, Data_ash:INT, Data_beta_carotene:INT, Data_beta_cryptoxanthin:INT, Data_carbohydrate:INT, Data_cholesterol:INT, Data_choline:INT)