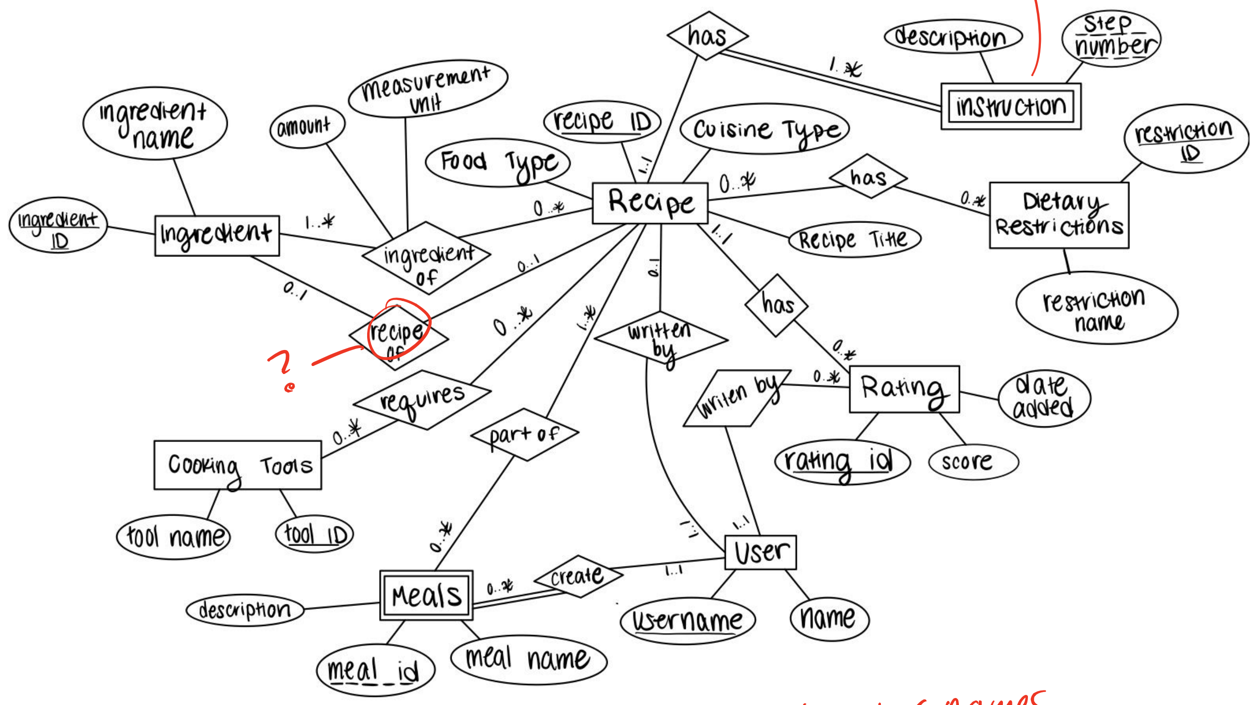
10/10 great start

Anna Smith
Jalen Tacsiat
CPSC 321 - Databases
Project 3

Possibly not needed, but might also be related to Tool, then don't need "requires"



★ Better to use singular names, eg, Meal not Meals

Explanation

Our application is centered around recipes, which is why there are many relationships between recipes and other entities. The attributes of the recipe are the recipe ID for uniqueness, food type, cuisine type and recipe title. The first relationship is between recipes and ingredients. A recipe must have at least one ingredient, but potentially many. If the recipe containing an ingredient is deleted, the ingredient will remain in the database, which is why we said that an ingredient can be a part of 0 or many recipes. For each ingredient in a recipe, it has an amount as well as a unit of measurement. Ingredients themselves can also be recipes. For example, caesar salad may call for caesar dressing as an ingredient, and caesar dressing has its own recipe. An ingredient can either have no corresponding recipe, or one. And a recipe may likewise correspond to either 0 or 1 ingredients only.

Recipes can also have instructions, where instructions are an entity. Recipes have at least one instruction, but can also have many. Instructions don't exist if the corresponding recipe is deleted, so it is a weak entity.

Recipes also have dietary restrictions such as vegan, vegetarian, etc. A recipe can have 0 dietary restrictions, but can also have multiple. Similarly, dietary restrictions may not correspond to any recipes, but they can also correspond with many.

Recipes also require certain cooking tools. A recipe may require no cooking tools/unknown cooking tools, so specified a 0 or many cardinality constraint. Cooking tools can also be used in 0 or many recipes.

Another key entity in our diagram is the user. Users can create meals, write recipes, and rate recipes. Users have a unique username as well as their real name.

For the relationship between user and recipe, we specified that each recipe can only be written by one user, but one user can write many recipes. Likewise, each rating can only be written by one user, but one user can write many ratings. A ratings' attributes are its rating ID, score (out of 5), and the date added. A rating can only belong to one recipe, but a recipe may have 0 or many ratings.

Meals are created by users and contain various recipes. A meal's attributes are its description, meal_id, and meal_name. A meal can contain 0 recipes, but on the other hand, a recipe can be part of 1 or more meals. Additionally, meals cannot exist without a user, therefore if a user is deleted the meals created by that user will also be deleted.

A recipe can also have ratings. A ratings' attributes are its rating ID, score (out of 5), and the date added. A rating can only belong to one recipe, but a recipe may have 0 or many ratings.

While designing our ER diagram, one glaring issue we had was since each entity has a relationship with Recipe, we were unsure of how to treat each entity if a Recipe was deleted. Therefore, we decided that each entity would be handled on their own. The only entity that is going to be deleted if a recipe is deleted is instruction, since it is a weak entity. All others are going to remain in the database.

Another thing we ended up changing during the design process was how we handled ingredients. We initially only had one relationship between recipe and ingredient: recipe calls for an ingredient. Then we realized that it should be possible for an ingredient to also have its own recipe. So we added another constraint labeled "recipe of" to connect ingredients to recipes of their own. In addition to that we also realized that ingredients may have measurements. For example, 4 oz of water. Therefore, we decided amount and measurement units as attributes.

3. It was a bit difficult to model the various attributes of a recipe. For example, each recipe has dietary restrictions (zero or many). However, we knew that if we had a foreign key to a table for dietary restrictions, that table wouldn't need to contain anything other than the name of the dietary restriction. So it seemed a bit unnecessary to be making tables for things such as dietary restrictions or cooking tools. We ended up adding an ID attribute to these tables; these are what other tables would use as foreign keys, but we're not sure if this is the best way to model this.