Mark Piccirilli
cs340 Project step 3 Final Version
7/10/18

http://people.oregonstate.edu/~piccirim/cs340_Project/

Feedback by the peer reviewers on step 3

I did not receive any feedback

Upgrades to draft version

- substantially improved the layout, readability and flow of the site.
- Added update account page
- fixed handlebars pages
- Added many data manipulation quarries
- Added servings, low_cal, and low_sodium attributes to the recipes entities
- Ingredient_quantity attribute added to the recipe_ingredients relationship table

Feedback by the peer reviewers on step 2

Review 1
I don't have any helpful commentary based on your excellent ER/Schema. Everything was superbly layed out and easily understood. Thank you for the opportunity to review.

Review 2
Recipe database Are the attributes for each entity in the ERD same as that described in the database outline? Yes. I'm sure you were getting at the same idea, but one of the attributes under the cookware entity is called price and outline is called cost. Is the participation of entities in the relationships same as that described in the outline? yes Is the cardinality of entities in the relationships same as that described in the outline?yes Is there something that could be changed/improved in the E R Diagram and/or the overall database design? I may be wrong, but I'm wondering if the Users save recipes is correctly implemented? The best peer review for a Schema would answer all of the following questions: Are the relationship tables present where required and correctly defined, when compared with the database outline?yes. I would check if recipe_contributor table is necessary, given that it is a one to many relationship. Are foreign keys present where required and correctly defined, when compared with the database outline? yes Do the entity attributes match those described in the outline?yes Is there something that could be changed/improved in the Schema and/or the overall database design? I may be wrong, but I'm wondering if the arrows in your schema is correct. For instance, I believe that the recipe_id in recipe ingredients should point(references) the id in recipes. The ideal peer review for a DDQ file would answer all of the following questions: Is the SQL file syntactically correct? This can be easily verified by importing/copy-pasting it in phpmyadmin. (Do not forget to take backup of your own database before you do this!) No. Error in SQL syntax arises upon importing. Around the DROP TABLE IF EXISTS `recipes` .Around line 10. Are the data types appropriate considering the description of the attribute in the database outline?yes Format seems to be correct within creating a table; however, I would double check if the necessity of certain tables. Great job!

Actions based on the step 2 feedback

- One reviewer pointed out that I had an attribute that was labeled cost in one place and price in another. I fixed this and then proceeded to ensure that all attributes and entities were the same throughout the outline, both diagrams and the SQL code in the DDQ.

- A reviewer pointed out that I had created a unnecessary table in my schema for a one to many relationship. I then discovered that I left out one of my many to many relationships. I removed the unnecessary table for the schema and replaced it with the necessary one.
- A reviewer suggested that my users save recipes relationship was incorrectly implemented in my database design. I reviewed the relationship and I believe that the design is correct and effective so I left it as it is.

Upgrades to the draft version from step 2

- I ensured that all attributes and entities are named the same thing throughout and made appropriate changes
- I fixed errors in my schema uncovered by the peer review. See *Actions based on the feedback* for details.
- I've decided to remove the not null constraint from the contributor attribute of the recipes entity because this allows users to contribute recipes anonymously, or delete there account and not have their name permanently attached to the site.
- I changed the attribute name password to user_password to avoid using a sql keyword
- changed the user_password data type to varchar(255) from char(20) because I learned more about how to handle passwords.
- I added units to the attribute in the ingredients table
- added serving size and serving size unit attribute
- change the data types for calories, fat, sodium, sugar, protein attributes to decimal(19,1)
- Adjusted the code in my DDQ so everything runs correctly

Feedback by the peer reviewer on step 1

The best peer review for a general Outline would answer all of the following questions. • Is the outline useful for you to understand broadly, the universe that the database is set in ? (For eg, if the project is about designing a website for BattleStar Galactica, does the outline mention what is BattleStar Galactica)? The outline was broad, but clear and straight to the point. I was able to quickly identify the database Mark wanted to setup which involve anything relating to cooking recipes. • Is there something else that you wish was mentioned in the outline to understand the topic better? One possible small change could be shortening each entity up. For example, for Ingredient entity, the Vitamin could be categorized under Vitamins. Overall no, I actually like how detailed each entity is. Good job on selecting good choices that relate to your main database topic of cooking recipes. The best peer review for a Database Outline, In Words would answer all of the following questions: • Are there sufficient entities as required? Yes there are more than number of required entities. Entities also all relate and connect well with overall database outline/topic. • Are there sufficient number and types of relationships as required? Yes; there are a variety of relationships. This is good to ensure database is unique. • Is there a good reason to have the things described as entities to be entities? Or can they be just attributes of other entities? Entities are good to stay as entities. In this case, if all the attributes are just attributes the Id and Name attributes would get a confusing or messing. • Do the various relationships make sense to you ? Yes, the relationships are very well thought. I was able to make sense of it from first read. • Is there sufficient information about each attribute to justify it's presence ? Yes, each attribute has a quick definition. I also really like that fact outline listed number type such as "string of max length 255". • Are the data types mentioned for attributes ? Yes! It need not be an actual keyword from MySQL/MariaDB like "varchar" but is it sufficiently described as a string, number, date, etc? Yes, each attribute has a data type associated. This will make it very clear for when project needs to start. Do the data types make sense

Actions based on the step 1 feedback

- The reviewer mostly just confirmed that I had completed the specifications and objectives for this assignment.
- They did suggest the possibility of simplifying some of the entities by removing some attributes but also said that they like how detailed the entities are. I like having lots of attributes to describe the entities in this database and I want to use all of that information in my site, so I am going to keep all the attributes.

Upgrades to the draft version from step 1

- In the project outline I added a line to clarify that users of the site did not have to contribute recipes. Users will be able to create and account and use the site search the database, save recipes and follow other contributors without ever contributing a recipe. I also changed the word contributor to user in a few places.
- I corrected the contributor attribute of recipes entity to contain the unique id of a particular user rather than containing the users name which does not have to be unique.
- In my draft version I have attributes for the many to many relationships listed under the entities. I removed these because these many to many relationships will be stored in a separate table. If they were stored in the entity table it would require storing a list and the database would no longer be atomic. All of these relationships are still described in the relationships section.
- I corrected a mistake I found. I had “users save recipes” described as a one to many relationship when it is actually and many to many relationship.
- I added lines to indicate the relationships that would require an extra table.

Recipe Database

Project Outline

For this project, I am going to create a database of cooking recipes to drive a website for sharing recipes. The database will allow the user to look up recipes and find the necessary ingredients, cookware and health information. The user will also be able to contribute their own recipes to the site. The website will allow users to search for healthy recipes or search by meal type, ethnic cuisine, or contributor. In this document users are sometimes referred to as contributors, but non contributing users will be able to use the site to search and save recipes as well.

Database Outline

The entities in the database are:

recipes: Recipes are the key entities for this database. The have relationships with all of the other entities. Their attributes include:
- **id**: This will be a not null, auto incremented integer to uniquely identify each recipe in the database. This will be the primary key
- **name**: This will be the name of the recipe. It will be a string with a max length of 255 characters. This cannot be null.
- **instructions**: This attribute will be a varchar(max) providing cooking instructions for the recipe. This cannot be null.
- **meal_type**: This will provide the recipes meal type. Options will include breakfast, Lunch, Dinner, Desert, Appetizer, Snack. This attribute will allow users to search by meal type.
- **ethnic_cuisine**: This will provide any specific ethnic cuisine the recipe may belong too. This will allow users to search by ethnic cuisine.
- **low_calorie**: This attribute will indicate if an item is low calorie. It will be default null and hold a 1 if the item is low calorie.
- **low_sodium**: This attribute will indicate if an item is low sodium. It will be default null and hold a 1 if the item is low sodium.
- **servings**: This attribute is a number indicating how many serving the recipe makes

- **contributor**: This attribute will provide the Id of the user that contributed the recipe.
- **date_contributed**: This attribute will provide the date contributed. Its data type will be a date. This cannot be null.

ingredients:
- **id**: This will be a not null, auto incremented integer to uniquely identify each ingredient in the database. This will be the primary key.
- **name**: This attribute will be a string of max length 255 that provides the name of the ingredient. This cannot be null.
- **serving_size**: This will be the serving size of the ingredient
- **serving_size_unit**: This will be the unit of the serving size
- **calories**: This attribute will contain the number of calories in the ingredient. It will be a decimal. This cannot be null.
- **fat_g**: This attribute will contain the amount of fat in the ingredient. It will be and decimal. This cannot be null.
- **sodium_mg**: This attribute will contain the amount of sodium in an ingredient. It will be a decimal. This cannot be null.
- **sugar_g**: This attribute will contain the amount of sugar in the ingredient. It will be a decimal. This cannot be null.
- **protein_g**: This attribute will contain the amount of protein in the ingredient. It will be a decimal. This cannot be null.
- **vitaminA_pdv**: This attribute will contain the amount of vitamin A in the ingredient. It will be an integer to represent a percent. This cannot be null.
- **vitaminC_pdv**: This attribute will contain the amount of vitamin C in the ingredient. It will be an integer to represent a percent. This cannot be null.

cookware:
- **id**: This will be a not null, auto incremented integer to uniquely identify all cookware in the database. This will be the primary key.
- **name**: This is the name of the cooking item. It will be a string of max length 255. This cannot be null.
- **cost**: This attribute will contain the cost of the cooking item. Its data type will be small money.

user:
- **id**: This will be a not null, auto incremented integer to uniquely identify each contributor in the database. This will be the primary key.
- **first_name**: This will be the user's first name. It will be a string of max length 255. This cannot be null.
- **last_name**: This will be the user's last name. It will be a string of max length 255. This cannot be null.
- **user_name**: This will be the user's user name. It will be a string of max length 255 characters. This cannot be null.
- **user_password**: This will be the user's password. It will be a string of max length 255 characters. This cannot be null.
- **email**: This will be the user's email. It will be a string of max length 255. This cannot be null.
- **cooking_experience**: This will be an optional attribute for the user to enter their cooking experience if any. This will be a varchar(max);

The relationships in the database are:

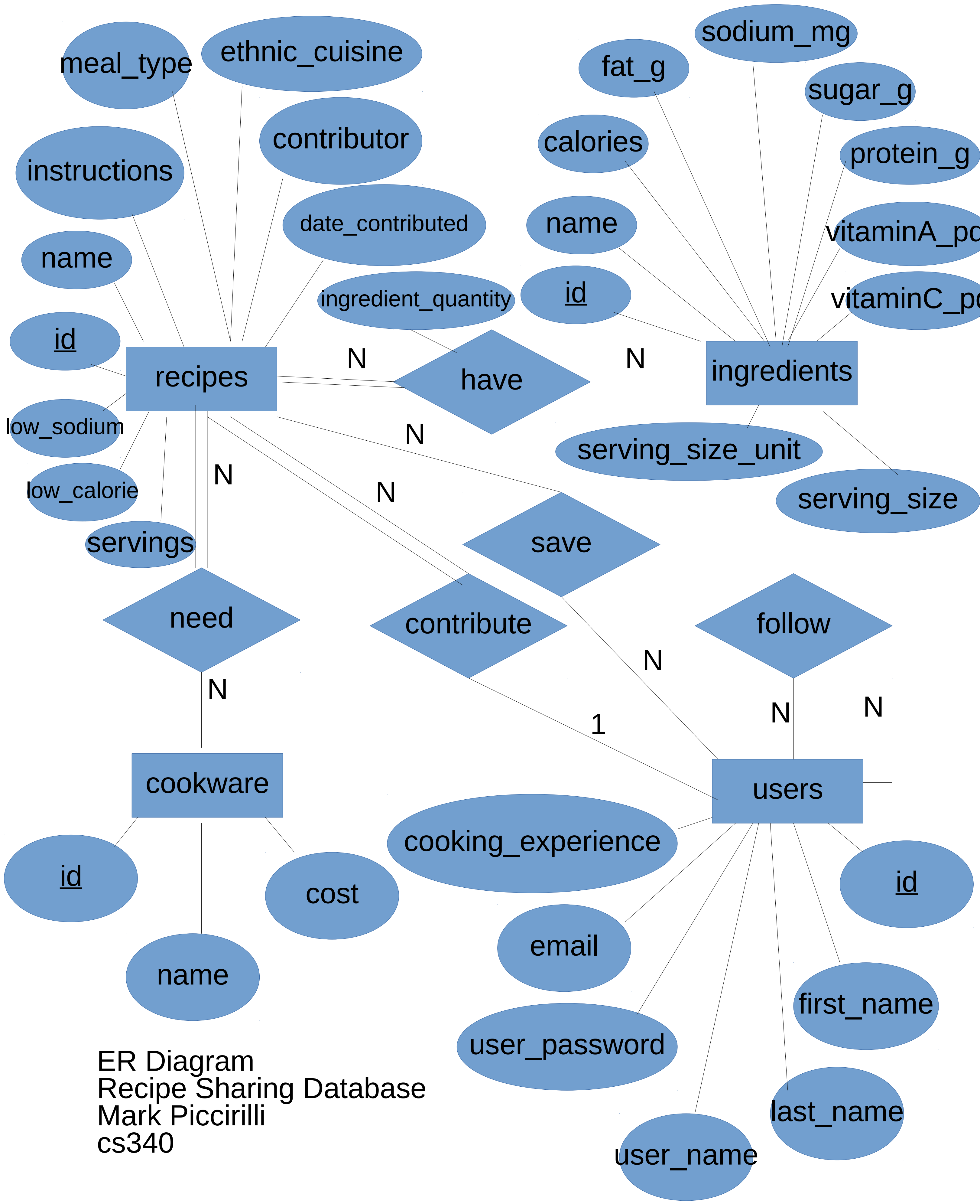
recipes have ingredients: This is a *many to many* relationship. Many recipes have many ingredients and many ingredients are used in many different recipes. This relationship will require its own table. This relationship will have an additional attribute called ingredient quantity.

recipes have cookware: This is a *many to many* relationship. Many recipes use a lot of cookware and different cookware is used in many different recipes. This relationship will require its own table.

users contribute recipes: This is a *one to many* relationship because one contributor can contribute to many recipes but each recipe only has one contributor.

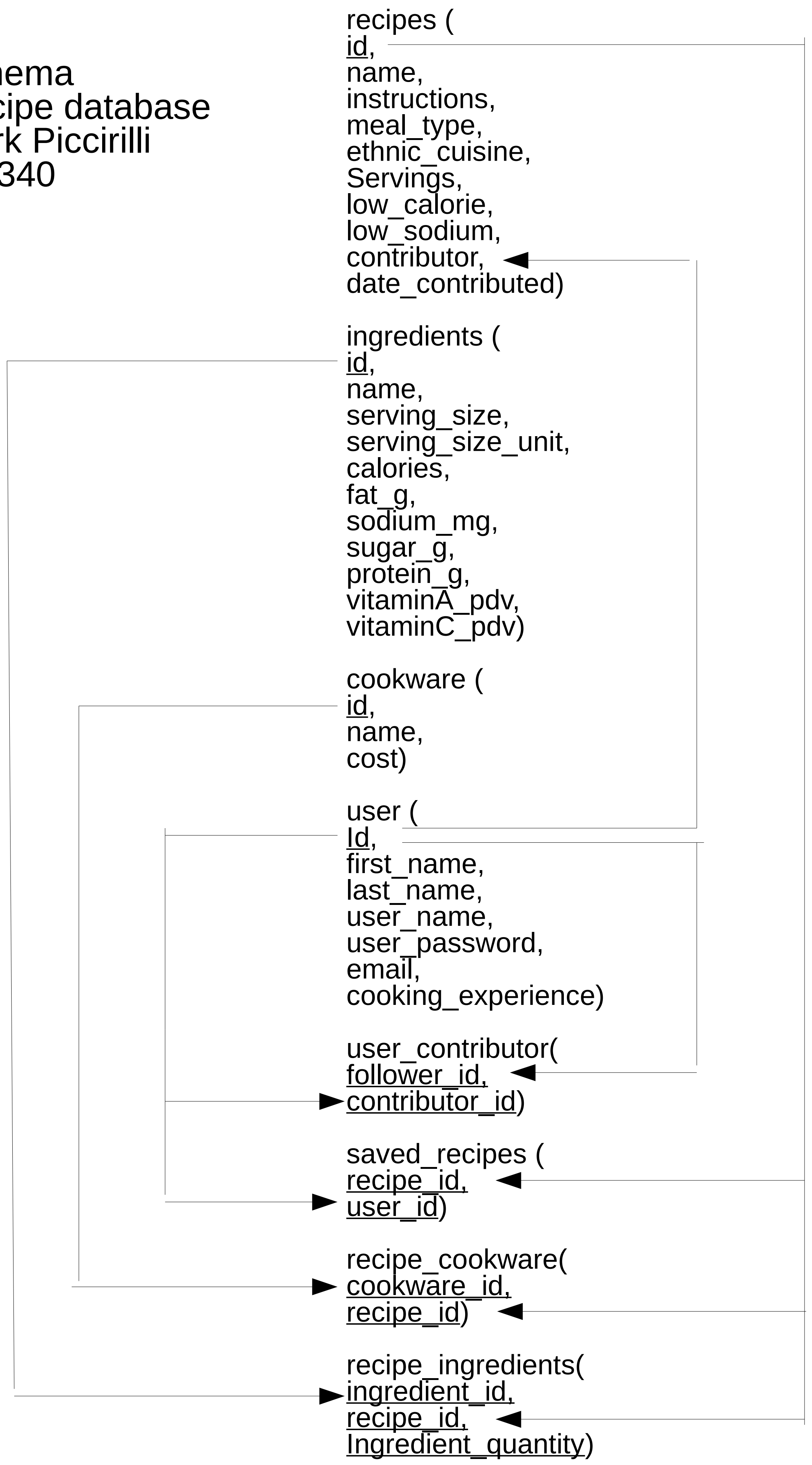
users follow other users: Like social networking a user can follow other user and receive notice when a user they follow has posted a new recipe. This is a *many to many* relationship because a user can be followed by many other users, and a user can follow many other users. This relationship will require its own table.

users save recipes: Users can save recipes so that they can go back to the later. This is a *many to many* relationship because one user can save many recipes and on recipe can be saved by many users. This relationship will require its own table.



ER Diagram
Recipe Sharing Database
Mark Piccirilli
cs340

Schema
Recipe database
Mark Piccirilli
Cs 340



recipes (
id,
name,
instructions,
meal_type,
ethnic_cuisine,
Servings,
low_calorie,
low_sodium,
contributor,
date_contributed)

ingredients (
id,
name,
serving_size,
serving_size_unit,
calories,
fat_g,
sodium_mg,
sugar_g,
protein_g,
vitaminA_pdv,
vitaminC_pdv)

cookware (
id,
name,
cost)

user (
Id,
first_name,
last_name,
user_name,
user_password,
email,
cooking_experience)

user_contributor(
follower_id,
contributor_id)

saved_recipes (
recipe_id,
user_id)

recipe_cookware(
cookware_id,
recipe_id)

recipe_ingredients(
ingredient_id,
recipe_id,
Ingredient_quantity)