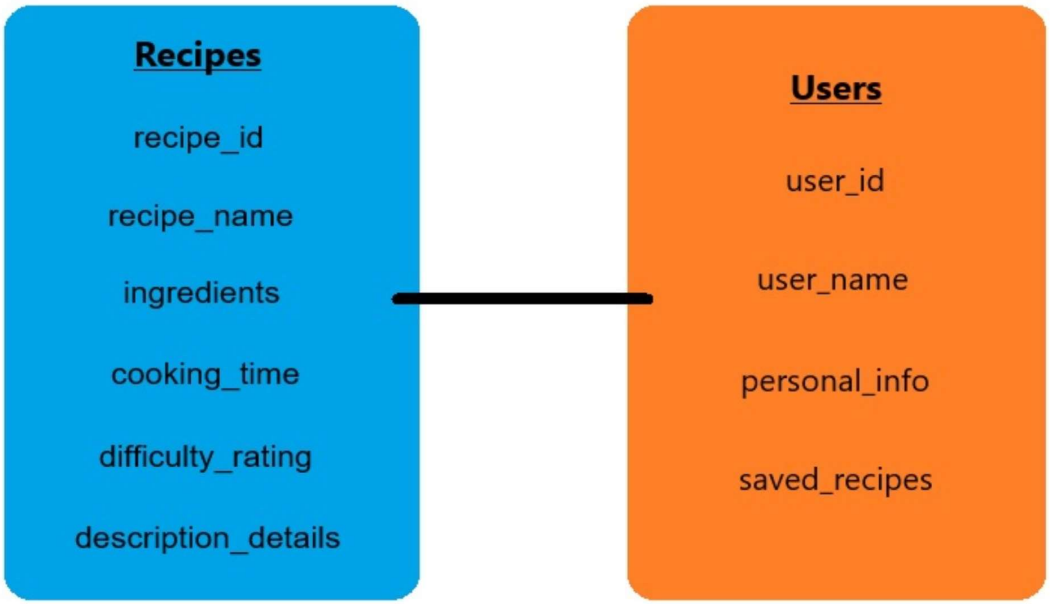
This is my initial model (schema) before I knew the things that I know now, and plans have changed :



Explanation :

These models were arranged during the work in Task 2.3. I didn't altogether know what I would need, what would be allowed, and/or what would even work properly. I also didn't define the data types or arguments therein, when designing the very simple image. It was just a good starting point to give myself something to reference before actually trying to design the function in 'models.py'; as the assignment instructed.

New Models :

With what I've learned since Task 2.3, I can list these models with some basic level of articulation that I couldn't before. As well, there are some necessary changes for optimization as well as achieving basic functionality. Many of these changes were discovered through crashing into my attempts to build the models on my own, running into various problems and errors from the 'tests.py' reports returned in console and browser, and the resulting research that these inspired. I will go through each attribute, giving explanations (where needed) as well as adding their respective data types and arguments.

Key:
Black = Remains: defining data type
Red = Removed
Blue = Changed (or expecting to *be* changed)

src**recipes**\models.py :

- **'Attribute'** - data type = 'DataType'. Explanation.

- **‘recipe_id’ -** This was removed due to a built-in MySQL process that assigns each data entry with a unique ID number. To also have this attribute in the function would be both redundant, as well as problematic in confusing users, and developers alike. This also align with the DRY (**D**on’t **R**epeat **Y**ourself) principle that helps in code cleanliness, optimization, and avoidance of errors and coding mistakes.

- **‘recipe_name’** : data type = ‘CharField’. I put a ‘max_length=255’ argument here just for the practice in using it. A character limit could be useful in stopping a user from accidentally adding something like ingredients or descriptions to this field. As well, strangely long recipe names could create a lot of visual clutter and make the users’ otherwise nice looking list of recipes look confusing or hard to read.

- **‘ingredients’** : data type = ‘TextField’. A text field works here because I don’t see any reason to inhibit the user from what they want to enter. It is also for my own exercising of using different data types successfully. However, I expect I will be changing this as I learn a better way of inputting recipes. Eventually, I am going to want the user to be able to search for recipes based on ingredients, and I expect the best way to do this is for each ingredient to be added as individual objects (prompting the user to add them in one at a time). I can imagine this looking like separate small input fields on the browser admin interface.

- **‘cooking_time’** : data type = ‘IntegerField’. I did not like this as a text or character field due to the messiness of varying human input. I like prompting the way it was done in Achievement 1 where the user is notified that they are entering only a number that represents “minutes”. I may end up arriving at new reasoning for this decision and changing accordingly.

- **‘difficulty-rating’** : data type = ‘CharField’. A Character field is what makes sense when thinking about the end result: Easy, Moderate, Hard, Very Hard. I am looking forward to see how I change this when it comes time to building the function that calculates the difficulty. When the result is populated by the result of a separate function, the data type may need to change to facilitate that result.

- ‘description_details’ : data type = ‘Text Field’. A text field should be ideal here, because this field can be populated virtually any imaginable way. This field needs to cooperate with the endless possibilities of a human input. The user/developer who is inputting the recipe should be free to add information here in any way they think is appropriate with little or no guidance from programmatic logic.

src**users**\models.py :

- **‘Attribute’ -** data type = ‘DataType’. Explanation.

- **‘user_id’** : This is probably going to be removed. For the same reason mentioned before in my explanation for removing ‘recipe_id’ from the ‘recipes\models.py’. However, I am leaving it in here, for now, as a way of experimenting and learning how my Python functions will interact with the database. In this case, I am intending on specifying that the key name will be “user_id” as opposed to the default “id” that would

otherwise be assigned to it by MySQL. If my understanding is correct, the default "id" will not be generated due to my assigning the 'primary_key=True' argument to this attribute.

- **'user_name'** : data type = 'CharField'. This attribute is pretty straight forward. A character field should fulfill the simple requirements of receiving the users' chosen names for themselves.

- **'personal_info'** : This is being replaced with the next listed attribute, 'email_address'. When I first listed this attribute, I wasn't sure what the task(s) were going to ask of me, or what information I would want to be added to the users' profiles. For now, I am thinking that email address should be enough information for the purposes of this application. An email address can make sure each user is unique.

- **'email_address'** : data type = 'EmailField'. This attribute replaces the 'personal_info' attribute that I had as a vague placeholder, while I hadn't yet known what all I would want or need for this part of the application. The built-in 'EmailField' data type is convenient in making sure the input is structured correctly as an email address (and not inadvertently populated with some other inappropriate data.

- **'saved_recipes'** : I removed this for several reasons. I initially added this thinking I could just have it as a placeholder for what I thought would be a feature we would add in another task. I expect we will. But, for now, implementing even placeholder logic here is causing more problems than I am currently equipped to deal with. I was receiving all kinds of errors that I am still not yet capable of handling without a huge research dive where I would be learning a plethora of new concepts. What I would need to learn may very well be an entire task (or portion of one) that I haven't got to, and require more time and research than the rest of that task (2.3) combined. This would not have been a good use of my time, as I expect that if this is to be implemented, it will be taught. After all, that is why I employed this class to begin with: To receive focused guidance on the learning and implementation of these concepts in a way that is more efficient and effective than if I spend the many hours I would otherwise spend teaching it to myself.