Django Delights

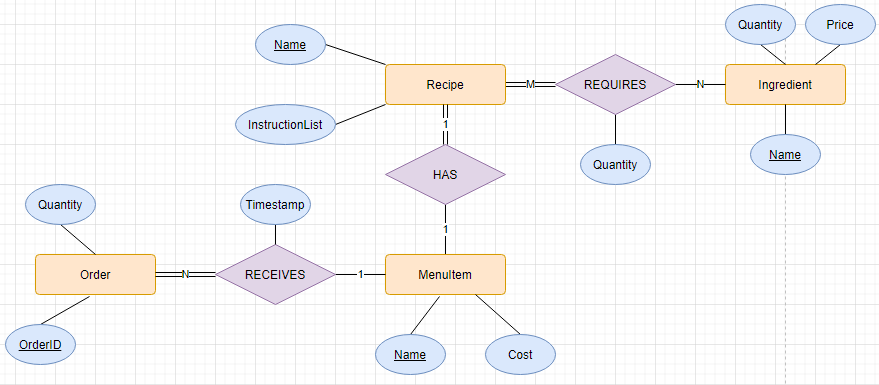
**Problem Description**

* You’ve been asked by a restaurant owner to build an application that will help keep track of how much food they have throughout the day. The owner starts the day with:
  + An inventory of different Ingredients, their available quantity, and their prices per unit
  + A list of the restaurant’s MenuItems, and the price set for each entry
  + A list of the ingredients that each menu item requires (RecipeRequirements)
  + A log of all Purchases made at the restaurant
* Knowing that information, the restaurant, Django Delights’ owner has asked for the following features:
  + They should be able to enter in new MenuItems along with their recipe requirements, and how much that menu item costs.
  + They should also be able to add to the inventory a name of an ingredient, its price per unit, and how much of that item is available.
  + They should be able to enter in a customer purchase of a menu item. When a customer purchases an item off the menu, the inventory should be modified to accommodate what happened, as well as recording the time that the purchase was made.
* Here are some helpful tips to get you started thinking about the project:
  + Your ingredients, recipes, and purchase data should be stored in a database, and should be rendered back to the Django views.
  + Your Django backend should supply endpoints to create new recipes via a form submission, submit customer purchases from a different form, and get information about the total cost of inventory, the total revenue for the day, the different purchases that were made, and how much inventory is required to restock (as an initial example) to render them into a Django view.

**Project Requirements**

* Build an inventory and sales application using Django
* Version control your application with Git and host the repository on GitHub
* Use the command line to manage your application locally and test out queries
* Users can log in, log out, and must be logged in to see the views
* Users can create items for the menu
* Users can add ingredients to the restaurant’s inventory and update their quantities
* Users can add the different recipe requirements for each menu item
* Users can record purchases of menu items (only the ones that are able to be created with what’s in the inventory!)
* Users can view the current inventory, menu items, their ingredients, and a log of all purchases made

**Models**

* 
  + A MenuItem RECEIVES an Order
    - A MenuItem can be related to 0 to N Orders (an item on the menu can be ordered multiple times a day).
    - An Order must be related to exactly one MenuItem (each order of a MenuItem is unique). An Order cannot exist without being related to a MenuItem.
  + A Recipe REQUIRES Ingredients
    - A Recipe must use at least 1 and up to M ingredients.
    - An Ingredient can be used in 0 to N Recipes.
  + A MenuItem HAS a Recipe
    - A MenuItem must can have no recipe or up to one recipe.
    - A Recipe must be related to exactly one MenuItem.
* Relations
  + Ingredient[name, price, quantity]
  + MenuItem[name, cost, description]
  + Order[OrderID, item, quantity, timestamp]
    - Order.item references MenuItem.name
  + Recipe[name, item]
    - Recipe.item references MenuItem.name
  + Requires[item, ingredient]
    - Requires.item references MenuItem.name
    - Requires.ingredient references Ingredient.name
* Notes
  + Stored REQUIRES M:N relationship as separate entity
    - This will be a table consisting of instances relating one Recipe with one ingredient, along with the quantity of that item.
    - A Recipe for a MenuItem can be obtained by querying Requires for Ingredients by Recipe. This will return all ingredients for a particular recipe.
  + Could have displayed a recipe without making Recipe an entity; MenuItem could be in the M:N relationship with Ingredients, and the process would be the same as above for displaying and creating a recipe. However, the instructions indicate that Recipe should be an entity.
  + Should Recipe maybe be a weak entity? It doesn’t make sense to have a Recipe without a MenuItem.
    - It would make sense though, because maybe you would want to enter a Recipe for a MenuItem you will include at a later date.

**Views.py**

* Home
* Ingredient Inventory
  + List
    - Display all ingredients, their remaining quantity, and cost per unit
    - Display total cost of inventory?
    - Display how much of ingredient is need to restock for next day.
  + Create
    - Form that allows a User to enter a new ingredient
  + Update
    - Form that allows a User to update an ingredient’s cost or quantity
      * Do not cascade to previous order records, or to menu.
  + Delete
    - Form that allows a User to remove an ingredient from inventory.
      * Do not cascade deletion; set to null.
* Restaurant Menu
  + List
    - Display all available MenuItems and their cost
  + Create
    - Form that allows a User to enter a new MenuItem, along with a Recipe.
  + Update
    - Change cost of a MenuItem
      * Do not cascade.
  + Delete
    - Remove a MenuItem from
      * Do not cascade.
* Recipes
  + List
    - Display all Recipes
  + Create
    - Form that allows User to enter new Recipe—ingredients and instructions.
  + Update
    - Form that allows User to modify a Recipe—add or remove ingredients and instructions
  + Delete
    - Form that allows User to delete a Recipe
      * Cascade delete to null 🡪 MenuItem should refer to null if it’s recipe is deleted.
* Purchases
  + List
    - Display record of all transactions
    - Display total revenue for the day
  + Create
    - Allow a User to enter in a new Order of a MenuItem.
      * Cascade: when a new Order is entered, the inventory of ingredients shoud be updated to reflect the used up stock.
      * Check first if there are enough ingredients for that MenuItem.
  + Update
    - No need for an Update form. Users should not be able to modify past transactions. If an Order needs to be modified to another MenuItem, just delete and enter new Order.
  + Delete
    - Allow a User to delete an Order
      * This raises a problem: how to update ingredient list if it was an order entered by mistake?
        + Could return a page asking if deletion was for accidental entry, and then it would update the inventory.
        + Could subclass Orders into active and past Orders.
      * It would be ideal to keep a record even of deleted Orders. Perhaps provide a dropdown Menu asking for deletion reason, and then add to list of RemovedOrders.