Team members

Martin Hill, Connor Baldes, Group 30

Project Title

Dope Eats

Website

https://web.engr.oregonstate.edu/~hillmart/CS340/

Peer Review Feedback and Actions from Step 3

Does the UI utilize a SELECT for every table in the schema? Data from each table in the schema should be displayed on the UI (Note: it is rarely acceptable for a single query to join all tables and display them).

Yes, data from every table is displayed in the UI.

Does at least one SELECT utilize a search/filter with a dynamically populated list of properties?

There's not currently a search option to look through any tables' entities.

Does the UI implement an INSERT for every table in the schema? There should be UI input fields that correspond to each table and attribute in that table.

Yes, there is an insert for every table.

Does each INSERT also add the corresponding FK attributes, including at least one M:N relationship? For example, if there is an M:N relationship between Orders and Products, INSERTing a new order, should also INSERT row(s) into the intersection table OrderDetails. Otherwise, the new Order won't be associated with any products, and an order containing no products likely doesn't make much sense.

Yes, the inserts on recipes and users both have dietary restriction as a FK added. It uses a drop down menu to select from existing restrictions, which is great!

Is there at least one DELETE and does at least one DELETE remove things from an M:N relationship? For example, if an order is deleted from the Orders table, it should also delete the corresponding rows from the OrderDetails table, BUT it should not delete any Products or Customers.

There is a delete button but does not currently functionally remove things from tables.

Is there at least one UPDATE for any one entity? For example, in the case of Products, can the attributes of an existing row/record be updated?

There is an update button for every table.

Is at least one 1:M relationship NULLable? There should be at least one 1:M relationship with partial participation. For example, perhaps users can have a row in Accounts without actually ordering anything (i.e., having no relationship with any record in Orders). Thus it should be feasible to edit an order and remove its relationship with an account (i.e., change the order's foreign key to NULL).

Yes, dietary restriction is a NULLable foreign key for users and recipes, but currently, in the HTML UI, it doesn't have a NULL option for insert or update.

Do you have any other suggestions for the team to help with their HTML UI? For example, maybe they should use AS aliases to replace obscure column names such as fname with "First Name".

The UI looks good, I would recommend having the add/update/delete options hidden unless the user clicks on their respective button in the table.

Does the UI utilize a SELECT for every table in the schema? Data from each table in the schema should be displayed on the UI (Note: it is rarely acceptable for a single query to join all tables and display them).

- Yes, there is a SELECT for every table.

Does at least one SELECT utilize a search/filter with a dynamically populated list of properties?

- No

Does the UI implement an INSERT for every table in the schema? There should be UI input fields that correspond to each table and attribute in that table.

- Yes, there is INSERT for the main tables

Does each INSERT also add the corresponding FK attributes, including at least one M:N relationship? For example, if there is an M:N relationship between *Orders* and *Products*, INSERTing a new order, should also INSERT row(s) into the intersection table *OrderDetails*. Otherwise, the new Order won't be associated with any products, and an order containing no products likely doesn't make much sense.

- Yes, userRecipes has userID and recipeID as foreign keys and recipeIngredients has recipeID and ingredientsID as foreign keys

Is there at least one DELETE and does at least one DELETE remove things from an M:N relationship? For example, if an order is deleted from the Orders table, it should also delete the corresponding rows from the OrderDetails table, BUT it should not delete any Products or Customers.

- No, deleting one parent value would essentially remove both parent value from the composite entity

Is there at least one UPDATE for any one entity? For example, in the case of Products, can the attributes of an existing row/record be updated?

- Yes, it has update for users, recipes and recipeIngredients.

Is at least one 1:M relationship NULLable? There should be at least one 1:M relationship with partial participation. For example, perhaps users can have a row in Accounts without actually ordering anything (i.e., having no relationship with any record in Orders). Thus it should be

feasible to edit an order and remove its relationship with an account (i.e., change the order's foreign key to NULL).
    - Yes, the RestrictionID for Recipes can be null

Do you have any other suggestions for the team to help with their HTML UI? For example, maybe they should use AS aliases to replace obscure column names such as fname with "First Name".
    - The html ui looks good. Although I do find the last column units Recipe Ingredients can be confusing to grasp when it's pieced with the other columns

Does the UI utilize a SELECT for every table in the schema? **Data from each table in the schema should be displayed on the UI (Note: it is rarely acceptable for a single query to join all tables and display them).**

- Yes, data from each table in the schema is displayed in the UI.

Does at least one SELECT utilize a search/filter with a dynamically populated list of properties?

- There does not seem to be a SELECT that allows for search capabilities with a dynamic list of properties. Users would be one place they could add a search feature to search for users by name.

Does the UI implement an INSERT for every table in the schema? **There should be UI input fields that correspond to each table and attribute in that table.**

- Yes, there is INSERT functionality implemented for each table.

Does each INSERT also add the corresponding FK attributes, including at least one M:N relationship? **For example, if there is an M:N relationship between *Orders* and *Products*, INSERTing a new order, should also INSERT row(s) into the intersection table *OrderDetails*. Otherwise, the new Order won't be associated with any products, and an order containing no products likely doesn't make much sense.**

- Yes. When inserting a Recipe entity, there is a FK to add a row to the RecipieIngredient table

Is there at least one DELETE and does at least one DELETE remove things from an M:N relationship? **For example, if an order is deleted from the Orders table, it should also delete the corresponding rows from the OrderDetails table, BUT it should not delete any Products or Customers.**

- No. When deleting a Recipe entity, there is no cascade effect to deleting the corresponding row in the RecipeIngredient Intersection table. This should be included otherwise there will be deletion anomalies.

Is there at least one UPDATE for any one entity? **For example, in the case of Products, can the attributes of an existing row/record be updated?**

- Yes, in there is an update implemented for the User entity

Is at least one 1:M relationship NULLable? **There should be at least one 1:M relationship with partial participation. For example, perhaps users can have a row in Accounts without actually ordering anything (i.e., having no relationship with any record in Orders). Thus it should be feasible to edit an order and remove its relationship with an account (i.e., change the order's foreign key to NULL).**

- Yes, Recipe is in partial participation with DietaryRestriction and can be updated to have a NULL id for the dietary restriction

Do you have any other suggestions for the team to help with their HTML UI? **For example, maybe they should use AS aliases to replace obscure column names such as fname with "First Name".**

- I would recommend spacing the links out on the top. They tend to blur together so it's hard for people to tell what the distinct table names are. Also, the Recipies page states "Browse Users" instead of "Browse Recipes". Finally, in UserRecipes I would make the userID and recipeID fields for the INSERT a dropdown menu rather than a text entry.

Main Review Takeaways and Addressing actions:
1. Search/Filter Feature: Currently, the user interface doesn't provide a search or filter capability with a dynamically populated list of properties. The reviewers suggest adding a search feature, particularly in the Users section to allow users to search by name.

Added search bar with dynamically populated list of properties to search for recipes friendly to certain dietary restrictions. To do this we added code to the recipe.html file to add the search bar. We also created javascript files to serve the dynamic search.

2. DELETE Functionality: When deleting a Recipe entity, the UI does not have a cascading effect to also delete the corresponding row in the RecipeIngredient Intersection table.

Implemented ON DELETE CASCADE where necessary in sql code.

3. UI Design: The reviewers recommended a few design changes to improve the user interface:
a. Spacing out the links at the top for better clarity and readability, as they currently tend to blur together.
Our team believes that the current spacing is good and chose not to change the spacing.

b. Correcting the title on the Recipes page, which currently reads "Browse Users" instead of "Browse Recipes".

Recipes Page title “Browse Users” was corrected to “Browse Recipes”

Peer Review Feedback and Actions Taken from Step 2

“There is no schema present”
Added Schema

“I would however recommend removing spaces for the entity names like User Recipes to UserRecipes and perhaps renaming “Ingredients in Recipe” to something like RecipeIngredients” Changed name to recipeIngredients

“create a userRecipeID as the PK and then have userID and recipeID as the FK” We made the primary key the combination of them both so we don;t have 2 PKs, made this more clear on ERD and Outline

“Inconsistent use of quotation marks in the INSERT INTO statements:
- ○ The values for integer columns (e.g., `servings`, `price`, `quantity`) should not be enclosed in quotation marks.
- ○ The values for date columns (e.g., `dateAdded`) should be enclosed in single quotation marks.

- Inconsistent use of quotation marks and NULL value in the INSERT INTO statements for the `recipes` table:
  - ○ The `restrictionID` column in the `recipes` table is nullable, so the value should be specified as NULL instead of `(SELECT id from dietaryRestrictions where name = 'Gluten-free')`. “

Updated use of quotation marks, NULL was already inserted correctly.

“There seems to be no normalization issues. However, to make it more clear, perhaps have repeated users in the "User Recipes" table, unless users can only have one recipe.”
Updated sample data

“Instead of using NULL to say no dietary restrictions consider using a number”
We decided to leave this as NULL just because we thought having an absence of a dietary

restriction made sense vs making users pick something (even if it was just a number that meant no restriction)

“Some of the entities’ ID do not have a specific name such as recipeID, ingredientID”
Fixed entity names
“One thing that I noticed was that the entity names in the ERD and the outline are spaced and do not use camelCase as seen in the schema, which can be fixed quickly. “
Updated entity names in ERD

Peer Review Feedback and actions from Step 1

“Yes. However I would recommend you choose either snake case or camel case, as of right now you have (Date_added) and IngredientID.”
“Everything is capitalized except for things that end in ID, which are not capitalized. It is consistent this way, and if it works it works, but it is unclear why the things ending in ID are the only ones not capitalized.”
“The names are all consistent so far but there still seems to be lots of possible error in the future so stay consistent with your naming!”
“There is some inconsistency between attribute names using camelCase versus snake_case particularly in the outline.”
Updated all attribute names to camel case.

“While none of the entities have a summary, their titles make them fairly easy to tell what they are aiming to store. However, the distinction between what is different about the ingredients in recipes database and the ingredients database is unclear.”
“Adding more descriptions of the purpose of each database would help define why you need them and what they do.”
“The entity names are fairly descriptive though it would be nice to describe the purpose in words. This can also help you pinpoint other possible attributes like unit measurements for quantity.”
Added table descriptions.

“There could be more details about how the dietary restrictions could be apart of a database.”
“Yes though I am missing some data that contextualizes the user context, such as how many users visit the site and how many recipes are there or their length.”
Updated overview to incorporate dietary restrictions and specific sizes on data stored in the database.

“The recipe entity is not complete and does have any connection to ingredients that is shown yet. This may be fixed in the future of the class though and i'm not sure how you'd store all the ingredients besides just a large string listing them all.”
“There are currently many to many relationships but I think there is potential for them here if you

merge the ingredients with the ingredients in recipes and say that ingredients can have many recipes and recipes can have many ingredients."
Ingredients are stored in the Ingredients in Recipes table.

Upgrades to Draft for Proj 1
Added FKs to Ingredients in Recipes, Recipes, User Recipes, and Users.
Added additional attributes that will give the tables more context.
Removed "not NULL" designation from restrictionID in users and

recipes. **Overview**

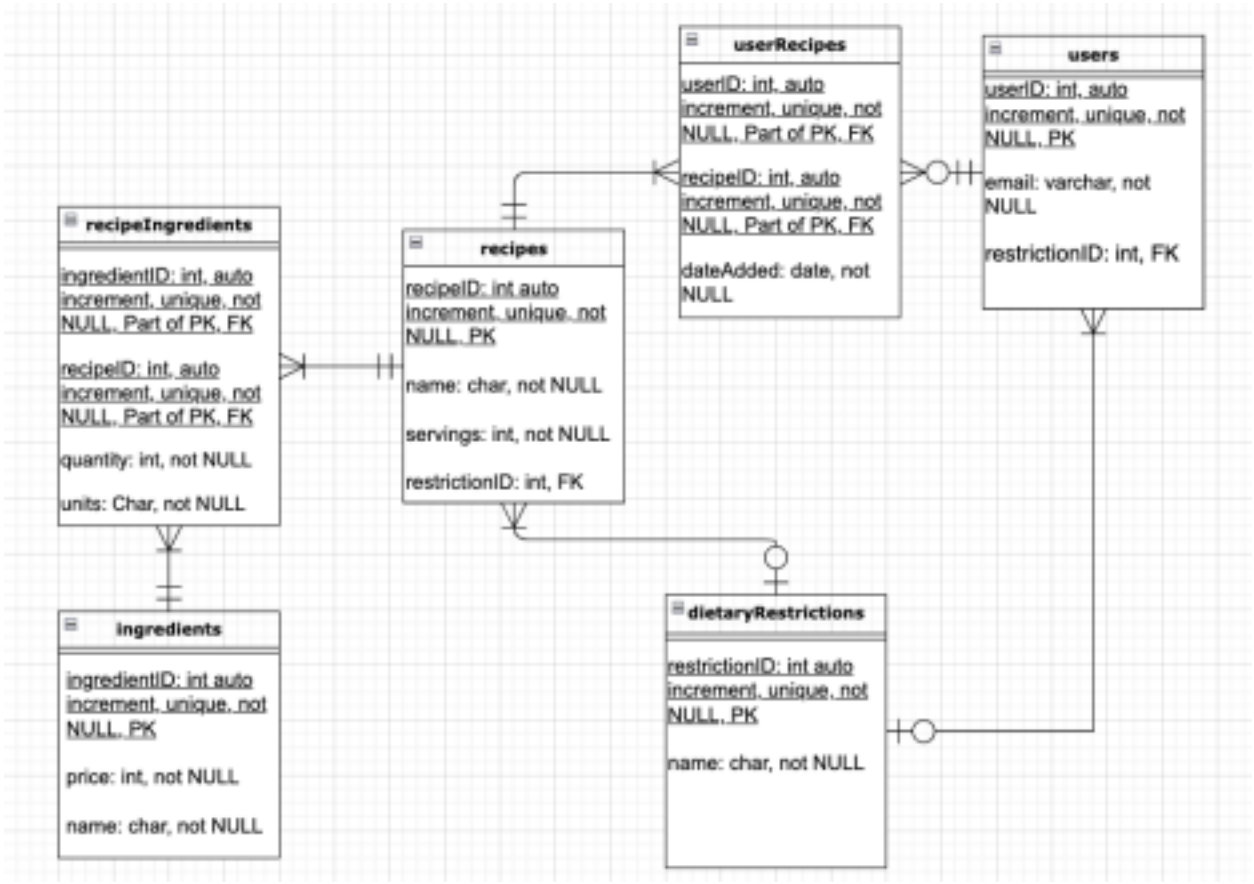
Our client is a company named "Dope Eats" which provides an online platform that connects thousands of food enthusiasts with a database of over 10,000 diverse recipes designed to accommodate a range of over 50 dietary restrictions. For this platform Dope Eats needs a relational database driven website which is effective at managing user profiles, and contains a library of recipes, ingredients, and dietary restrictions. Specifically, the platform should allow users to browse different recipes, as well as make specific searches for recipes containing desired ingredients or catering to their dietary restriction. Users should also be able to save recipes they like in a way that they will be easily accessible in the future. Dope Eats has a stable and steadily growing user base, and their desire is that the platform puts an emphasis on simplicity and usability while also requiring minimal system maintenance.

Outline

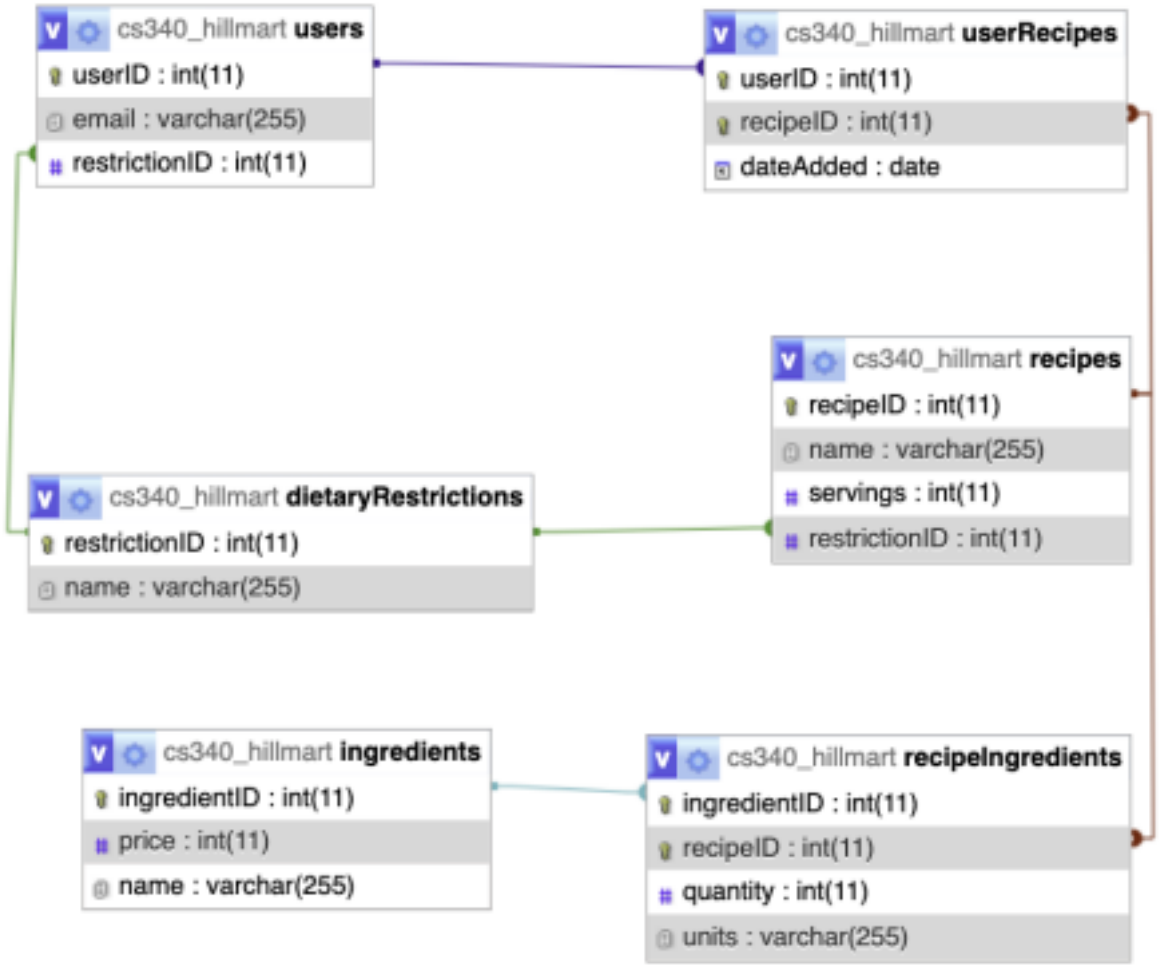
- users - database of users, storing their email, ID connects to recipes
    - userID: int, auto increment, unique, not NULL, PK
    - email: varchar, not NULL
    - restrictionID: int, FK
    - M:N relationship with Recipes
    - M:N relationship with Dietary Restrictions
- recipes - database of recipes, storing name and serving, ID connects to ingredients, users, and dietary restrictions
    - recipeID: int auto increment, unique, not NULL, PK
    - name: char, not NULL
    - servings: int, not NULL
    - restrictionID: int, FK
    - M:N relationship with Users
    - 1:N relationship with Dietary Restrictions
    - M:N relationship with Ingredients
- ingredients - database of ingredients, storing their price and name, ID connects to recipes
    - ingredientID: int auto increment, unique, not NULL, PK

    - price: int, not NULL
    - name: char, not NULL
    - M:N relationship with Recipes
- dietaryRestrictions - database of dietary restrictions storing name, ID connects to users and recipes
    - restrictionID: int auto increment, unique, not NULL, PK
    - name: char, not NULL
    - M:1 relationship with Users
    - M:1 relationship with Recipes
- userRecipes - database to join Users and Recipes. Stores both IDs and date added
    - userID: int auto increment, unique, not NULL, PK, FK
    - recipeID: int auto increment, unique, not NULL, PK, FK
    - dateAdded: date, not NULL
    - 1:N relationship with Users
    - 1:N relationship with Recipes
- recipeIngredients - database to connect recipes and ingredients, stores ID and quantity
    - recipeID: int auto increment, unique, not NULL, PK, FK
    - ingredientID: int auto increment, unique, not NULL, PK, FK
    - quantity: int, not NULL
    - units: char, not NULL
    - 1:N relationship with ingredients
    - 1:N relationship with Recipes

ERD


userRecipes
userID: int, auto increment, unique, not NULL, Part of PK, FK
recipeID: int, auto increment, unique, not NULL, Part of PK, FK
dateAdded: date, not NULL
users
userID: int, auto increment, unique, not NULL, PK
email: varchar, not NULL
restrictionID: int, FK
recipeIngredients
ingredientID: int, auto increment, unique, not NULL, Part of PK, FK
recipeID: int, auto increment, unique, not NULL, Part of PK, FK
quantity: int, not NULL
units: Char, not NULL
recipes
recipeID: int auto increment, unique, not NULL, PK
name: char, not NULL
servings: int, not NULL
restrictionID: int, FK
ingredients
ingredientID: int auto increment, unique, not NULL, PK
price: int, not NULL
name: char, not NULL
dietaryRestrictions
restrictionID: int auto increment, unique, not NULL, PK
name: char, not NULL

Schema



Normalization Spreadsheet with Example Data

3NF

users

userID	email	restrictionID
1	john.doe@example.com	1
2	jane.smith@example.com	2
3	mark.jones@example.com	3
4	emma.taylor@example.com	1
5	will.brown@example.com	2

userRecipes

userID	recipeID	dateAdded
1	1	2023-04-01
2	3	2023-04-05
2	2	2023-04-05
3	3	2023-04-10
4	4	2023-04-15
5	5	2023-04-20

dietaryRestrictions

restrictionID	name
1	Vegetarian
2	Gluten-free
3	Vegan

recipes

recipeID	name	servings	restrictionID
1	Tomato Soup	4	1
2	Grilled Chicken	2	NULL
3	Vegan Tacos	4	3
4	Gluten-free Pasta	4	2
5	Garden Salad	2	1

ingredients

ingredientID	price	name
1	1	Tomatoes
2	2	Chicken
3	1	Lettuce
4	3	Vegan Cheese
5	1	Gluten-free Noodles

recipeIngredients

recipeID	ingredientID	quantity	units
1	1	4	cups
1	3	2	cups
2	2	1	pounds
3	4	1	packages
4	5	1	packages
5	3	4	cups
5	1	2	cups