Group 69
David Trannam/Kevin Luk

Peer Review Updates

a) Fixes based on Feedback from Step 1:

We did not get any form of feedback from TA/Instructor so we did not make any adjustments there. This list is from our peer reviews on Ed Discussion only.

Feedback by the peer reviewer

- Feedback from Eric Tolson:
    - I think the overview could benefit from a description of the entity relationships.
    - I think the facts need to be more tied to the entities and their attributes.
    - I believe there needs to be a connection entity between tags and reviews though because it is M:M. If there are multiple tags on a review, this current diagram doesn't seem to support that since there is only space for one tagID.
- Feedback from Bryan Zierk
    - There is one M:M relationship between Review and Tags, I believe there should be an intersection table for "Reviews_has_Tags" to be able to cross reference those tables.
    - The 'reviewStars' and 'restaurantPriceRange' attributes seem like they could benefit from being their own entities with defined values
- Feedback from Benni Taylor (they/them)
    - It might be worth thinking about what the difference between a "category" and a "tag" is?
    - There is a slight inconsistency with the overview using restaurantHasNutritionInfo and the ERD using restaurantHasNutritionInformation.
- Feedback from Charles Nguyen
    - I am slightly curious how Restaurants being M:1 with Categories will work out, if for no other reason than it sounds like there are both type (dessert) and region (Thai) categories available.

Actions based on the feedback

- We are not adding an intersection table, because as the assignment specified this document is a high-level overview of our project, e.g, a schema is not necessary – we are simply using an ERD. ERD's do not require intersection tables by definition.
- We will provide clarity on the following points:
    - What the attributes "reviewStars" and "resturantPriceRange" are to better explain why they aren't needed as separate entities (Zierk)
    - We have removed the tag and category system from our ERD and our project. Based on the feedback, we received a lot of confusion on how tags and categories would be used. In our project, we wanted tags to be ways to filter out reviews and categories to filter out restaurants. Instead we have decided to

remove our tags as a whole and flesh out our categories for better filtering of restaurants.

- We have updated our diagram to fix minor discrepancies such as
    - restaurantHasNutritionInfo and the ERD using restaurantHasNutritionInformation. (Taylor)

Upgrades to the Draft version

- The biggest changes we made was to scrap our tags and categories ideas. There was a lot of confusion regarding these two separate systems. We found that it would be easier and more user friendly to simply remove tags. We will now be making categories M:M instead of 1:M to allow users to better filter out restaurants they are viewing within the area based on feedback.

Group 69
David Trannam/Kevin Luk

MONCH: Project Step 1 Draft

b) Project Outline and Database Outline - Updated Version:
This is our updated version for Project 1 Feedback that adjusted some of the issues brought to us in our peer review.

Overview:

FRONT END DESCRIPTION:

Ever had a transcendental out-of-body experience at a restaurant that you just HAVE to share with the world in a quantifiable way? Picture this, Los Angeles is estimated to have around 30,000 restaurants alone making it nearly impossible to narrow down your restaurant choices. How would your average person choose what to eat? The sheer number of possibilities is mind-boggling. What if there was a way to curate a list of these restaurants to your tastes based on categories, rating, and location. This would make it much easier to find the perfect restaurant to satisfy your cravings and put you one step closer to a state of Nirvana. With the average household spending over $3000 on food a year, this database driven website will help narrow what to eat by recording reviews of restaurants.

BACK END DESCRIPTION:

There are an estimated 30,000 restaurants alone in Los Angeles. Now, imagine how many restaurants there are in the entire world! Given that, we propose an app that on the front end will allow users to submit reviews of any given restaurant at any given location. These reviews will include categories of the restaurants, the price range of the restaurant, if the restaurant includes nutritional information, and a string that will be the user's review of the restaurant. **Given that there are essentially an unlimited number of people who can write at most 1 review for EVERY single restaurant on the entire planet and given that there are many restaurants world-wide, there needs to be a way for the administrators of our app to manage/sort/edit/delete these both restaurants, reviews and users.** This is where our back-end database driven solution comes in. Administrators will be able to use both the data in our 'Reviews' table and our 'Restaurants' table to extrapolate information and manage reviews/restaurants/users. This will be critical for data reporting as well as the general administrative duties that come with managing what would be a HUGE database in the form of CRUD functionality.

Our project attempts to solve this problem by implementing some of the following features the following:

- Admins are able to CREATE/UPDATE/DELETE restaurants (and their attributes) to be reviewed by users
- Admins are able to see all reviews submitted by users and approve/reject them (READ/UPDATE/DELETE)
- Users are able to create reviews for restaurants. They can edit/delete them as well. (CREATE/UPDATE/DELETE)

- Users are able to see what others have written about particular resturants (READ)
- Users can filter their choices of restaurant based on categories, the name of the restaurants, and the city of the restaurant. (FILTERING)
- Eventually (outside the scope of this project) Users can also filter restaurants based on the proximity to their location.

Database Outline

- **Users Table**
    - Summary: This will contain a list of users that have created accounts. Users are able to generate reviews that they have visited forming a 1 to many relationship where each user can generate multiple reviews. The user table will hold generic information about the users such as email, birthday, and their location.
    - Schema:

userID	INT, AUTO-INCREMENT, PK, NOT NULL
userEmail	VARCHAR, NOT NULL
userBirthday	DATE, NOT NULL
userLocation	VARCHAR, NOT NULL

    - Relationships
        - 1 → Many with ‘Reviews’ - It is optional for a user to have a review

- **Reviews Table**
    - Summary**:** This will contain a list of reviews generated by users about restaurants they have visited. Each review will need to have a key linking it to both a user and a restaurant. The review will hold the content of the review, the date of the review, along with a star rating which is a numeric value out of 5 or 10.
    - Schema:

reviewID	INT, AUTO-INCREMENT, PK, NOT NULL
userID	INT, NOT NULL, Foreign Key
restaurantID	INT, NOT NULL, Foreign Key
reviewContent	VARCHAR, NOT NULL
reviewDate	DATE, NOT NULL, DEFAULT (Today)
reviewStars	INT, NOT NULL

    - Relationship:
        - M → 1 with ‘Restaurant’’ - Reviews are required to have a restaurant.
        - M → 1 with “User” - Reviews are required to have a user who authored the content

- **Restaurants**
    - Summary: This table will contain a list of restaurants that are in our system to be reviewed. It will form a 1 to many relationship to reviews where one restaurant can have many (or no reviews). It will also form a relationship with categories to help narrow options when searching for restaurants. This will be M:N as each

restaurant can have multiple categories to help filter out searches on the user side. Each entry will hold the name, address, and nutrition information (if available). Additionally, it will hold a price range as an integer where an integer corresponds to a range that we will provide. (Such as 1 for $0-$10, 2 for $11-$20). There will be no additional tables for price ranges as we can view the integers placed in the restaurant as a rating of the price rather than something that holds meaning. For example, when comparing something that is 1 ($) vs. 4 ($$$$), we can simply conclude that one restaurant is maybe a fast food or cheap to eat vs. a gourmet, dine in restaurant.

- Schema:

restaurantID	INT, AUTO-INCREMENT, PK, NOT NULL
restaurantName	VARCHAR, NOT NULL
restaurantAddress	VARCHAR, NOT NULL
restaurantAddress	VARCHAR, NOT NULL
restuarantPriceRange	INT, NOT NULL
restaurantHasNutritionInfo	BOOL, NOT NULL

- Relationship:
  - 1 → M with ‘Reviews’ - It is optional for restaurants to have reviews
  - M → M with ‘Category’ - It is optional that a restaurant contains a category. We will use an interception table to handle this (thus no foriegn key in this table)

- **Categories**
  - Summary: This table will hold categories that a restaurant may have. For example, if we are interested in a Korean-Mexican fusion restaurant, we can have the category of: Korean, Mexican, Fusion to better represent the restaurant
  - Schema:

categoryID	INT, AUTO-INCREMENT, PK, NOT NULL
categoryName	VARCHAR, NOT NULL

  - Relationships:
    - M → M with ‘Restaurants’ - it is optional for categories to have a restaurants

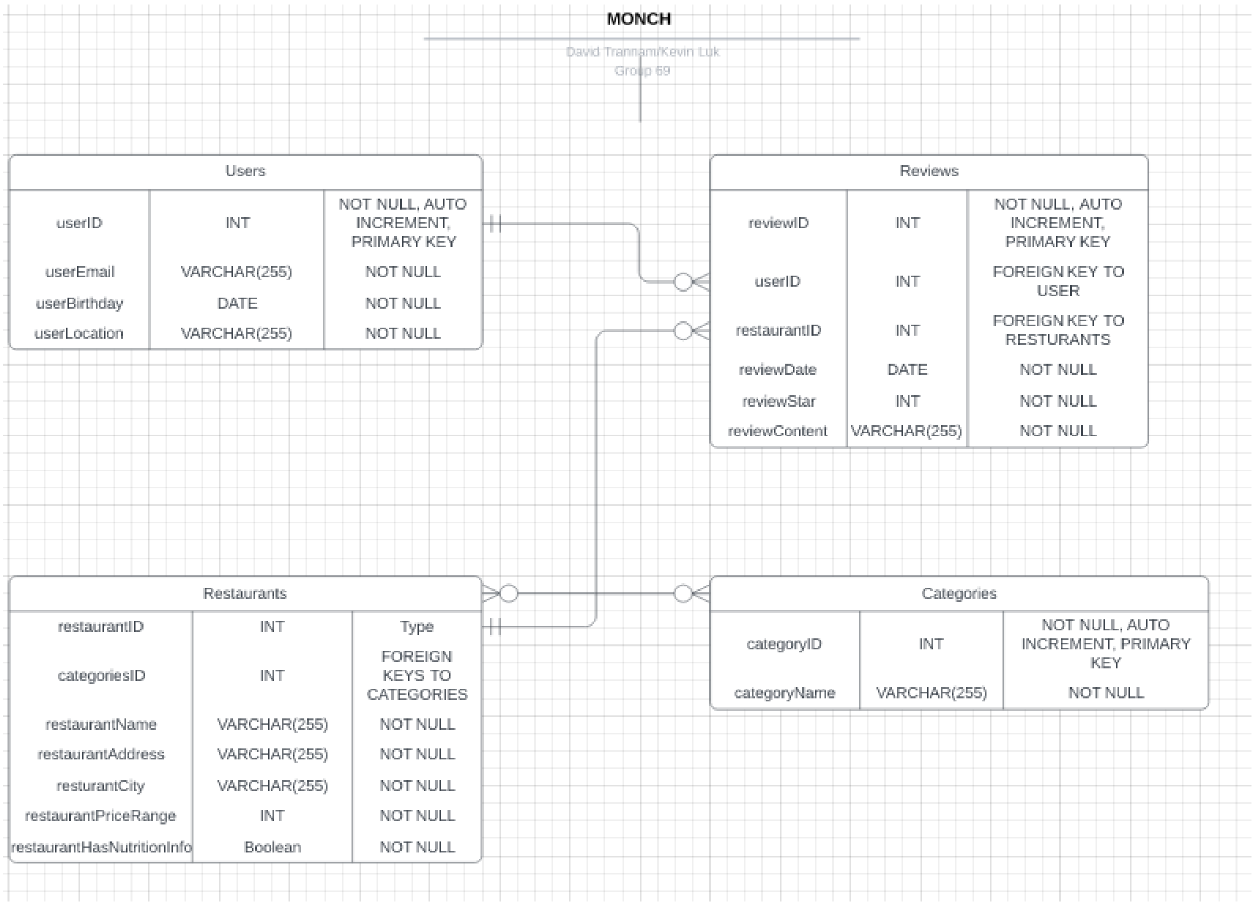
Work Assignments

- Based on the structure of this assignment, the work can be broken down as followed:
  - Kevin: Handles the restaurants and categories
  - David: Handles the users and reviews.
- At the end of the assignment, we should be able to link the store to the reviews (that’s linked to the users)

Group 69
David Trannam/Kevin Luk

ER Diagram

c) Entity-Relationship Diagram:
This contains our updated ER diagram as a schema. We provided an updated ER diagram that matches what we have stated from step B.



Users

userID	INT	NOT NULL, AUTO INCREMENT, PRIMARY KEY
userEmail	VARCHAR(255)	NOT NULL
userBirthday	DATE	NOT NULL
userLocation	VARCHAR(255)	NOT NULL

Reviews

reviewID	INT	NOT NULL, AUTO INCREMENT, PRIMARY KEY
userID	INT	FOREIGN KEY TO USER
restaurantID	INT	FOREIGN KEY TO RESTURANTS
reviewDate	DATE	NOT NULL
reviewStar	INT	NOT NULL
reviewContent	VARCHAR(255)	NOT NULL

Restaurants

restaurantID	INT	Type
categoriesID	INT	FOREIGN KEYS TO CATEGORIES
restaurantName	VARCHAR(255)	NOT NULL
restaurantAddress	VARCHAR(255)	NOT NULL
resturantCity	VARCHAR(255)	NOT NULL
restaurantPriceRange	INT	NOT NULL
restaurantHasNutritionInfo	Boolean	NOT NULL

Categories

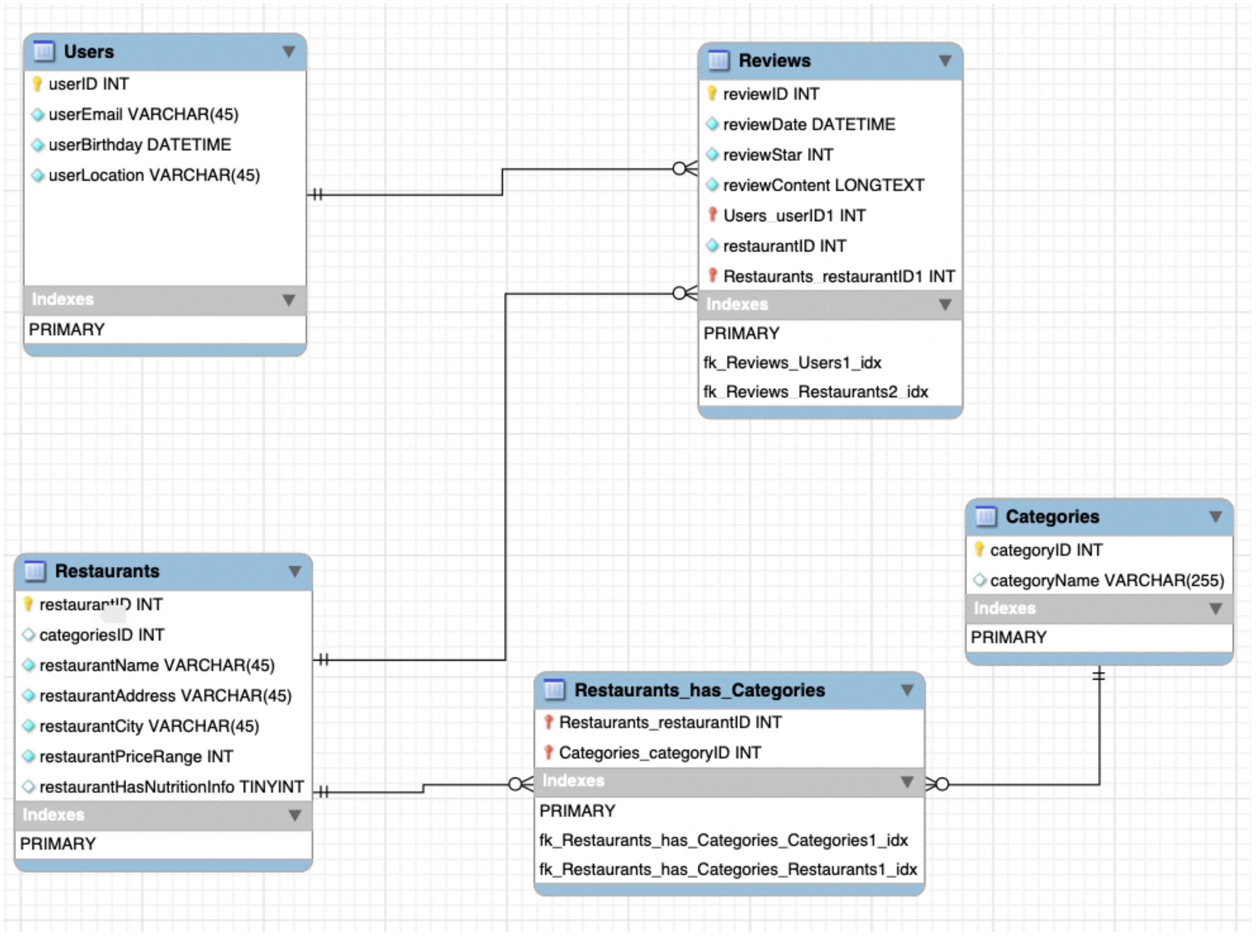
categoryID	INT	NOT NULL, AUTO INCREMENT, PRIMARY KEY
categoryName	VARCHAR(255)	NOT NULL

Group 69
David Trannam/Kevin Luk

ER Diagram

d) Schema:

This contains our Scheme as defined in our DDQ.SQL file after our normalization process.



Group 69
David Trannam/Kevin Luk

Sample Data

e) All the example data from each table in your schema should be pasted into your report.

User Table

userID	userEmail	userBirthday	userLocation
1	richie@lam.com	10/10/1996	San Francisco
2	nathan@perkins.com	6/5/1800	San Francisco
3	kevin@luk.com	9/4/94	Las Vegas

Review Table

reviewID	userID	reviewDate	reviewContent	restaurantID	reviewStar
1	1	1/1/2012	"Wow, really good"	1	5
2	2	5/4/2022	"Great!"	2	4
3	2	1/3/2004	"Tastes like my dry elbow"	2	1

Restaurant Table

restaurantID	restaurantName	restaurantAdress	restaurantCity	restaurantPriceRange	restaurantHasNutritionInfo
1	"Hugo's Cellar"	"202 Fremont Street Experience"	Las Vegas	3	FALSE
2	"Cheesecake Factory"	"251 Geary St, San Francisco"	San Francisco	2	TRUE
3	"Jjanga AYCE Sushi"	"6125 S Fort Apache Rd"	Las Vegas	2	FALSE

Tags Table

categoryID	categoryName
1	American
2	Japanese
3	Romantic
4	Desserts

Group 69
David Trannam/Kevin Luk

Restaurant has Categories Table

categoryID	restaurantID
1	1
3	1
1	2
3	3