Dine Dash Final Report

Changes from Stage 1 Proposal

For the most part our project matches the direction of our stage 1 proposal. The main difference is that in our final outcome we limited users to only see the menu items of one restaurant at a time. This allows for simplicity purposes to be able to focus on one restaurant at a time since the amount of restaurants we had would've been overwhelming to populate all of their menu items at once.

Another difference is that we didn't focus as much on allergen response due to the lack of clarity of ingredients in each food item. Without a clear list of ingredients for each food item, it is impossible to be sure of what allergens would potentially be in that item.

The other main change we had from our stage 1 proposal is the UI design difference. Although there are some similarities such as in the color scheme, the overall structure of the page changed as we made our website.

Application Usefulness

Since our website follows the same purposes as the proposal, our usefulness is still relatively the same. The application achieved the ability for users to filter based on eating preferences such as vegetarian or halal food items, cuisine category, menu item name, and a price range minimum and/or maximum. These are vital filters that allow users to pick meals that are within their dietary restrictions and budget constraints as well as their preferences in general.

Another important feature of our website is the ability to estimate order costs. Users can add their menu items from a restaurant into a cart so that the website can estimate the overall cost of that order. In order to meet requirements we added a drop down menu to display 8 different 'default queries' that the user can see. Some include: Most popular Menu Item, Least Popular Menu Items, Restaurants above average Rating in Area. The feature allows users to apply a narrower search on the database using criteria they do not have information about such as other user's orders and the items.

Schema Changes from Data source

We kept the dataset that we originally selected, but we did make several changes to it due to various issues when trying to put it onto MySQL Workbench. We also created some data to allow creation for other tables (such as users). The original data itself needed some cleaning up in order to be used.

Changes from ER diagram & Table Implementation

- Major Changes
    - We changed the primary key of menuitem to MenuID because Name and associated RestaurantID were not unique. We didn't realize this until we were creating the database for stage 3.
    - We made OrderItems a weak entity and made the primary key a composite key based on OrderID and MenuID
        - The original design did not require adding UserID, RestaurantID or Name as the primary keys allowed us to obtain the information from Orders and MenuItems joined with the table
        - Since we change the primary key of MenuItems we no longer needed RestaurantID and Name within the table
        - ItemID was not required as using MenuID and OrderID create a unique key for each Order
- Changes to relationships:
    - We clarified that restaurants and orders have a one to many relationship (a restaurant can have many orders, but an order belongs to one restaurant). We forgot to include this relationship when we originally created the ERD for stage 2, but realized the mistake when making stage 3.
- Changes to data types:
    - The database was originally designed on Workbench which did not allow for booleans so Vegetarian and Halal are tinyints
    - The descriptions, name, category were larger than 225 characters and Workbench made them text

Functionality Changes

We maintained the functionalities from stage 1, with the exception of a filter for allergens. As we have already established, creating a filter for allergens would've resulted in uncertainty on the accuracy of it because the menu items don't include an ingredient list. Without an ingredient list it is very difficult to speculate the various allergens for every food item.

Advanced database programs on Application

- In our project, we have used stored procedures that included advanced queries throughout the application. Users can filter for top restaurants and popular menu items, which involve several complex criteria for selection and quite useful filters for users. Managing orders also required the use of stored procedures.

    - Order submission has been defined as a Transaction to ensure that orders and order items remain in sync. This is required for the basic functioning of the application.
    - ·Some advanced queries have been used for intermediate steps, such as calculating the total price of the order items selected by the user.

Technical Challenges

- Ashley
    - Editing data on an excel spreadsheet is not ideal as the format is changed and data needs to be edited manually and extensively in order to fix errors in the CSV converted file
    - Our original design for the table was based on the information provided by the source we used , however we implemented users, orders, and orderItem tables. One issue we had between the tables is inconsistencies in attribute punctuation. In some tables the same variable would be upper or lowercase for certain letters, making it more difficult to run queries as changes had to be consistent.The attributes were inconsistent due to Workbench's settings forcing all names to be lowercase but did not standardize the punctuation throughout the database.
    - Workbench SSH does not work well, would be best to create the database in GCP at the start or workbench and create a SQL file of the database to upload to GCP to allow collaboration
    - While connecting to GCP on VS code, we had to add the @app.route(url, methods=[]) in order to allow access to the Database for queries.
- Lindsey
    - The dataset required a lot of data cleaning in order to be functional. When I first tried importing the data into Workbench it wouldn't import everything due to the several formatting issues with our data file (that weren't obvious until further examination). Future groups would benefit from choosing a much smaller dataset so it would be easier to manually check rows in it (ie choose dataset with closer to 1,000 rows instead of 50,000 rows)
- Manas
    - Setting up the environment for the application front-end was challenging due to ongoing code changes and the required environment setup. It took a significant amount of time to establish the web environment, as my primary focus was on the backend. Each new codebase required a fresh environment setup, and debugging issues was highly time-consuming. Due to time constraints, we could not complete the setup for Google Cloud Platform (GCP). Incorporating a functional environment as part of Stage 2 or Stage 3 would facilitate smoother progress in Stage 4.

- Deepika
    - We first had to establish a connection with our database that is on the google cloud platform. This took us a while to figure out as we needed to have the correct authentication credentials to establish the connection. This connection was established in python.
    - While writing sql queries to obtain the data we wanted, we noticed some of our data in the database had some special characters, carriage returns and some other issues. We had to work around them when creating our queries. Future groups would benefit from cleaning their data prior to uploading it to gcp.
    - We experienced issues connecting our backend with our frontend. This included not sending the correct data to the backend api endpoint, not posting to the api endpoint in a correct manner.

Application Changes from Stage 1 Proposal

The largest visual change from our proposal to our current application is the UI itself. We expanded on our original wireframe to allow for a more intuitive website so users would be able to easily navigate it. Rather than have a single page render every time the user applies a change we optimize to use a navigation bar users can access restaurants and orders. In order to limit our queries and inhibit some features we only allowed users to either view restaurants or Menu Items at the selected restaurant to reduce query latency and reduce the overall factors we needed to consider. Additionally a user can modify their orders on the orders tab and on the menu items tab to either 'add to order' or 'create a new order'. We removed the Order by capabilities of the filtering for users due to time constraints on creating our UI and implementing backend and the keyword search is based on Restaurant Name as opposed to any item within the database.

Another change we implemented was the "add to cart" functionality, a user can add one item to cart and the order is visible on the page but no longer available to modify as a signal that they placed the order.

Application Improvements

There are many features that could be added to improve the application. A very simple addition would be to allow users to order directly from our application. This would involve routing orders to the restaurant's website, but would improve user experience overall.

An important feature would be the ability for users to directly contact the restaurants (ie via a website or a phone number), but this was a limitation for us since our original dataset did not include similar information.

Another feature that could be implemented is to allow users to do group orders together. This functionality would be useful for catered events so that users are able to select their own items from the restaurant.

Additionally users could have the ability to view more criteria regarding the Restaurant or Menu Item but due to constraints with our data the feature was not plausible.

We can modify the schema to include an OrderDate column in the orders table. This change would help in identifying orders on a given day, which could be used to determine daily sales and add one more criterion for selecting popular restaurants and menu items.

We can introduce a feature to identify Star Customers (those who place many orders within a period, e.g., a week or a month). These customers could receive special discounts upon login. To support this, we would need to add a starTier column in the users table to track whether a customer is in the Silver, Gold, or Platinum tier, and apply discounts accordingly. Additionally, we can create another table to maintain the discount levels for each tier.

We could implement a daily email batch job in the application to send promotions and discounts to users.

Labor Division & Teamwork

- Ashley
    - Successfully connected to db (and resolved data issues), finalized db creation, performed explain analysis on queries, frontend UI design
    - Worked on creating the UI design and modified it to fit the needs of our queries and show basic CRUD operations
    - Worked on creating basic API endpoints for simple queries like fetching user data and storing it on localStorage for caching and latency reduction
- Lindsey
    - Created ER diagram and relationship assumptions / reasoning
    - Tried to connect to db, drafted db creation
    - Wrote advanced queries, coordinated writeup
- Manas
    - Wrote advanced queries, stored procedures and dynamic filtering for restaurants and menu items.
- Deepika
    - Integrating frontend and backend
    - Focused on creating the GCP connection with our SQL Database to allow us to work on VS Code
    - Added Triggers, transactions, stored procedures to our database through GCP to test if they worked before implementing it to front end
- Overall, we worked well together as a group. It was very helpful that we all met and discussed the project thoroughly for stage 1. This allowed us to get to know each other better, know each other's strengths and weaknesses, and have a good understanding of our project. We maintained teamwork throughout the project by having a group chat for active communication and updates regarding project work.