University of British Columbia, Vancouver
Department of Computer Science

CPSC 304 Project Cover Page

Milestone #: 4

Date: 29 - 11 - 2024

Group Number: 60

Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
Christopher Lew	87799201	a8t6b	christopherlew4@gmail.com
Jennifer Tanojo	40026460	b7a5y	jennifer.olive.tan@gmail.com
Jennifer Wang	72344476	f9p0l	jennifer26wang@gmail.com

By typing our names and student numbers in the above table, we certify that the work in the attached assignment was performed solely by those whose names and student IDs are included above. (In the case of Project Milestone 0, the main purpose of this page is for you to let us know your e-mail address, and then let us assign you to a TA for your project supervisor.)

In addition, we indicate that we are fully aware of the rules and consequences of plagiarism, as set forth by the Department of Computer Science and the University of British Columbia

Milestone 4 - Project Implementation

A Brief Project Description:

a) Domain of the application

The domain of the application is food journaling and restaurant interaction, combining elements of personal food documentation, restaurant review and waitlist platforms, and online food services.

b) Aspects of the domain modeled by the database

The application's goal is to serve as a comprehensive food experience platform, where users can not only document their food experiences but also interact with other users and restaurants in various ways. This project is best understood as a personal food journaling and interaction platform, where users can not only document and share their dining experiences but also have restaurant-related activities in ways that enhance their journaling: leaving reviews, creating a food bucket list, ordering food online, and joining a waitlist for dine-in.

c) Differences between the final schema and the schema we had turned in:

Our schema is the same as our schema in Milestone 2 since we did not find any issues with our existing one.

Repository Link:

https://github.students.cs.ubc.ca/CPSC304-2024W-T1/project_a8t6b_b7a5y_f9p0l

Queries

2.1.1 INSERT Operation

The insert query allows a user to rate a restaurant (using its name and location) based on its food, service and affordability. The rating's comments come from its reviewID.

```
INSERT
INTO RATES (foodRating, serviceRating, affordabilityRating, reviewID,
restaurantName, restaurantLocation)
VALUES (:foodRating, :serviceRating, :affordabilityRating, :reviewID,
:restaurantName, :restaurantLocation)
```

(appService.js - line 88)

2.1.2 UPDATE Operation

Depending on the non-primary key chosen by the user, the following queries allows a user to update a tuple of choice by providing the column name shown in the interface, journalID (Primary Key), old value and new value.

```
UPDATE REVIEW2 SET tags = :newValue WHERE journalID = :jid
```
(appService.js - line 261)
```
UPDATE REVIEW2 SET accountID = :newValue WHERE journalID = :jid
```
(appService.js - line 277)

2.1.3 DELETE Operation

The delete query would allow a user to delete their journal of choice by inputting its specific title and description (case sensitive).

```
DELETE
FROM JOURNAL2
WHERE title = :title AND description = :description
```
(appService.js line - 112)

2.1.4 SELECTION Operation

The selection operation will allow the user to select–or search–a restaurant based on its name, location, and/or waitlist ID. We implemented this operation with dynamically generated dropdown of AND/OR options and user inputs (the user enters a string specifying the conditions).

```
SELECT * FROM Restaurant2 ${conditions ? `WHERE ${conditions}` : ""}
```
(appService.js – line 203)

2.1.5 PROJECTION Operation

The projection operation operates on the Restaurant1 table and allows users to see what types of cuisine are available and also their menus. The SQL query gets the whole table and the projection happens afterwards, only showing the columns that the user has selected.

```
SELECT * FROM Restaurant1
```
(appService.js - line 136)

2.1.6 JOIN Operation

The join operation allows the user to view the staffs' job title and position given the restaurant name and location.

```
SELECT Restaurant_Staff1.staffID, Restaurant_Staff1.position
FROM Restaurant_Staff1, Restaurant2
WHERE Restaurant_Staff1.restaurantName = Restaurant2.name AND
Restaurant_Staff1.restaurantLocation = Restaurant2.location AND Restaurant2.name
= :name AND Restaurant2.location = :location
```
(appService.js - line 293)

2.1.7 Aggregation with GROUP BY

The aggregation with `group by` query allows the user to compute the count of pick-up orders made by each account (in other words,we are displaying how many dine-in orders have been made by a single account).

```
SELECT accountID, Count(*) AS orderCount FROM PICKUPORDER GROUP BY accountID
```

(appService.js - line 225)

2.1.8 Aggregation with HAVING

This query allows the user to average a restaurant's ratings over restaurants that have a certain number of reviews written.

```
SELECT CAST(AVG(foodRating) AS DECIMAL(3, 2)) AS avgFoodRating,
CAST(AVG(serviceRating) AS DECIMAL(3, 2)) AS avgServiceRating,
CAST(AVG(affordabilityRating) AS DECIMAL(3, 2)) AS avgAffordabilityRating,
restaurantName, restaurantLocation
FROM Rates GROUP BY restaurantName, restaurantLocation
HAVING Count(*) >= 2
```

(appService.js - line 214)

2.1.9 Nested Aggregation with GROUP BY

The nested aggregation with group by query will allow the user to see the account IDs that have average total price of dine-in orders that is above 100 dollars.

```
SELECT *
FROM (SELECT accountID, CAST(AVG(totalPrice) AS DECIMAL(6, 2)) AS
      avgTotalPrice
      FROM DineInOrder
      GROUP BY accountID)
WHERE avgTotalPrice >= 100
```

(appService.js - line 236)

2.1.10 DIVISION Operation

The division operation allows us to get the accounts that has reviewed all restaurants

```
SELECT Account2.accountID
FROM Account2
WHERE NOT EXISTS
(SELECT accountID
 FROM ((SELECT DISTINCT journal1.accountID, Review1.restaurantName
        FROM Review1, Journal1)
        MINUS (SELECT Review2.accountID, Review1.restaurantName
               FROM Review1, Review2
               WHERE Review2.JournalID = Review1.JournalID)) TempResult
 WHERE TempResult.accountID = Account2.accountID)
```

(appService.js - line 308)