CPSC 304 Project Cover Page

Milestone #: 4

Date: August 5

Group Number: 34

Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
Jack Yao	29843927	s9n0s	jackyao2004@gmail.com
Joshua Calalang	66829102	u6x4h	joshuacalalang@hotmail.com
Rahul Kamath	25650656	t9i7n	rahul.kamath2512@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

Project Description:

Our database application is designed for the food and online delivery industry, focusing on the seamless management of interactions between customers, delivery drivers, and restaurants. This application enables customers to create and log into their accounts, specify dietary preferences, order food from restaurants that match their preferences, and review their past orders.

For reasons in maintaining simplicity to account for the short summer course time, we have decided to keep our app local to Vancouver. Behold, Vancouver Eats!

Schema Changes:

In our final schema, we changed the data type of the OrderID attribute to an INT since this allows us to compare IDs easier. Similarly, we also changed CustomerPhoneNum to be VARCHAR rather than CHAR, as comparisons with CHAR enforces the length, which unnecessarily complicates the code in the server.

Project Schemas:

Ingredient(IngredientName, IngredientType)
- IngredientName: varchar, PK
- IngredientType: varchar

Customer(CustomerID, CustomerName, CustomerAddress, CustomerPhoneNum)
- CustomerID: varchar, PK
- CustomerName: varchar
- CustomerAddress: varchar, not null
- CustomerPhoneNum: varchar

HasDietaryPreference(**CustomerID**, **IngredientName**, PreferenceType)
- CustomerID: varchar, PK FK
- IngredientName: varchar, PK FK
- PreferenceType: varchar

Orders(OrderID, OrderDate, **CustomerID**, **LicenseNum**, **RestaurantAddress**)
- OrderID: int, PK
- OrderDate: date
- CustomerID: varchar, FK
- LicenseNum: varchar, FK
- RestaurantAddress: varchar, FK

DeliveryPerson(LicenseNum, DeliveryPersonName, DeliveryPersonRating, PhoneNum)
- LicenseNum: char[8], PK
- DeliveryPersonName: varchar
- DeliveryPersonRating: float
- PhoneNum: char[10], CK unique

Vehicle(LicensePlateNum, InsuranceExpiryDate, Brand, VehicleModel, VehicleColour)
- LicensePlateNum: varchar, PK
- InsuranceExpiryDate: date
- Brand: varchar
- VehicleModel: varchar
- VehicleColour: varchar

OwnsVehicle(**LicenseNum**, **LicensePlateNum**)
- LicenseNum: char[8], PK FK
- LicensePlateNum: varchar, PK FK

Restaurant(RestaurantAddress, **RestaurantName**, RestaurantRating, RestaurantPhoneNum)
- RestaurantAddress: varchar, PK
- RestaurantName: varchar, FK
- RestaurantRating: float
- RestaurantPhoneNum: char[10], CK unique

RestaurantCuisine(RestaurantName, RestaurantCuisineType)
- RestaurantName: varchar, PK
- RestaurantCuisineType: varchar

Menu(MenuName, **RestaurantAddress**, MealType)
- MenuName: varchar, PK
- RestaurantAddress: varchar, PK FK
- MealType: varchar

MenuItem(MenuItemName, Calories)
- MenuItemName: varchar, PK
- Calories: int

MenuFeaturesItem(**MenuItemName**, **MenuName**, ItemPrice)
- MenuItemName: varchar, PK FK
- MenuName: varchar, PK FK
- ItemPrice: float, not null

ItemMadeWith(**MenuItemName**, **IngredientName**)
- MenuItemName: varchar, PK FK
- IngredientName: varchar, PK FK

OrderContains(**OrderID**, **MenuItemName**)
- OrderID: int, PK FK
- MenuItemName: varchar, PK FK

Drink(**MenuItemName**, Volume, AlcoholContent)
- MenuItemName: varchar, PK FK
- Volume: int
- AlcoholContent: float

RotationalItem(**MenuItemName**, StartDate, EndDate)
- MenuItemName: varchar, PK FK
- StartDate: date, not null
- EndDate: date, not null

RotationalDrink(**MenuItemName**, Volume, AlcoholContent, StartDate, EndDate)

MenuItemName: varchar, PK FK
StartDate: date, not null
EndDate: date, not null
Volume: int
AlcoholContent: float

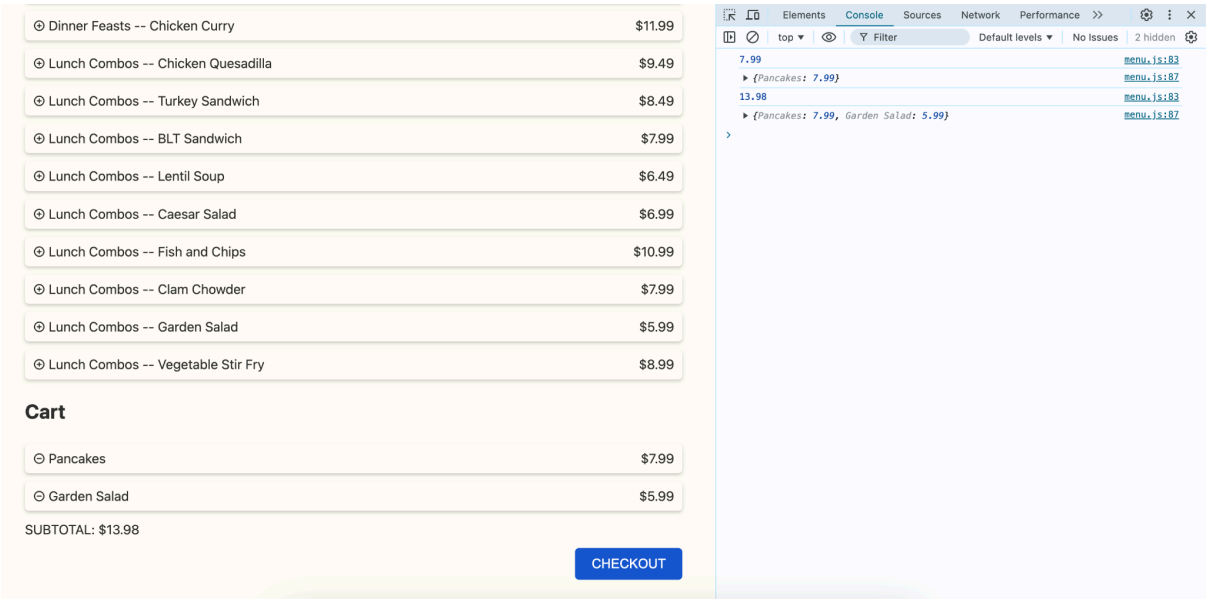
Tables:

CS304 Project Tables

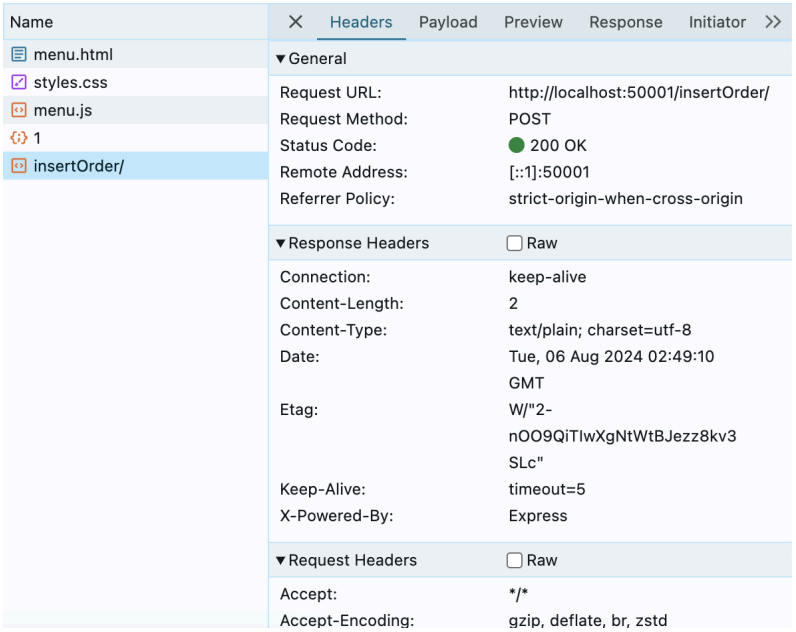
Query Result Screenshots:

INSERT: Creating an Order, which would be inserted into the Orders table in the SQL database. [appService.js, line 360]

(items in cart, ready to make request to insert)

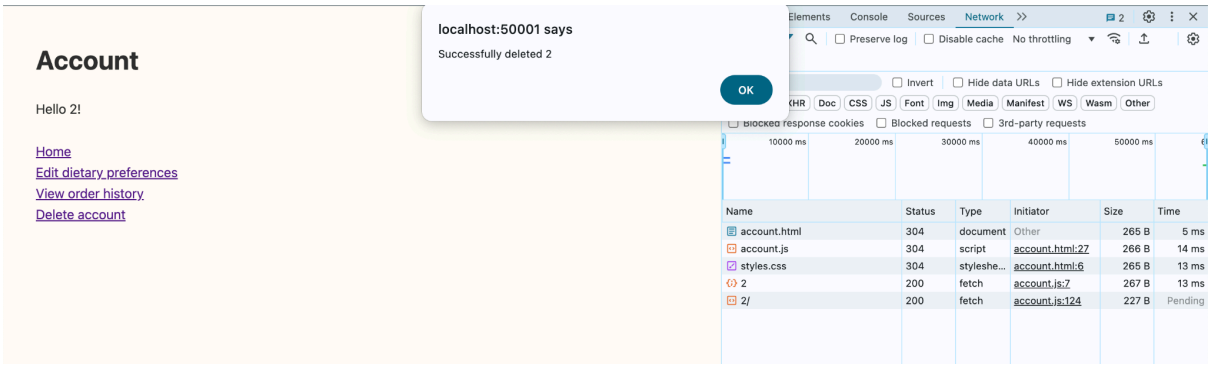


(status code 200 returned from server as confirmation that it has been inserted)



DELETE: Deleting one's account, which would subsequently delete the user's ingredients preferences by cascading [appService.js, line 109]

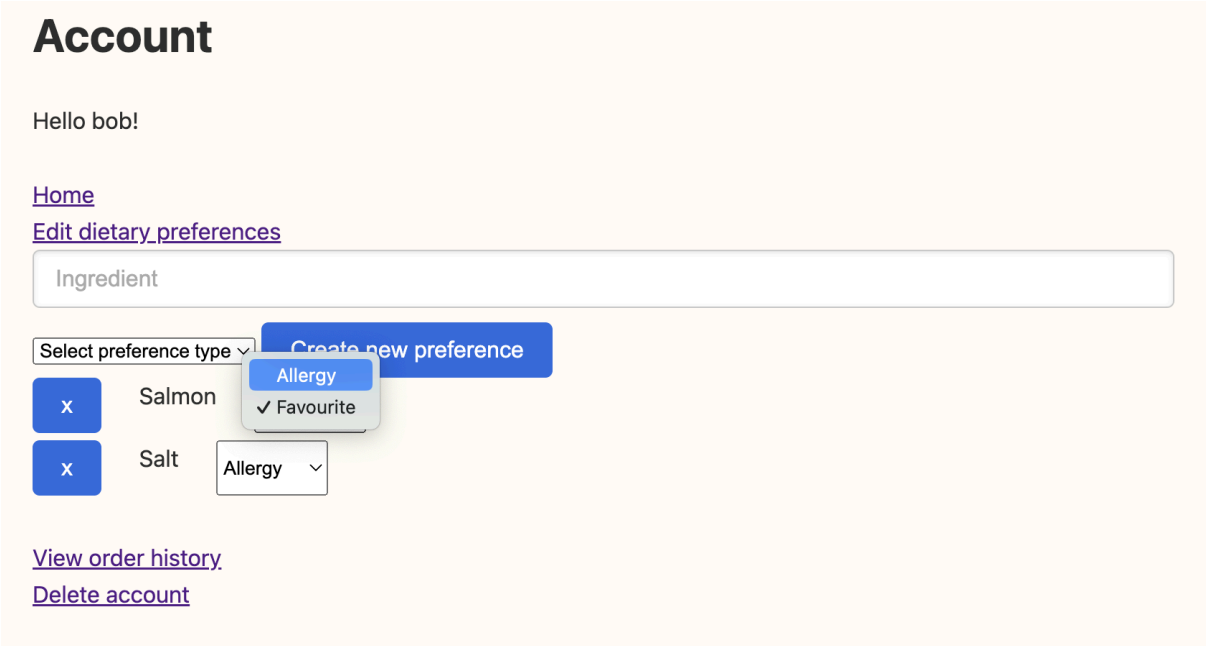
(deleting account from UI and receiving code 200)

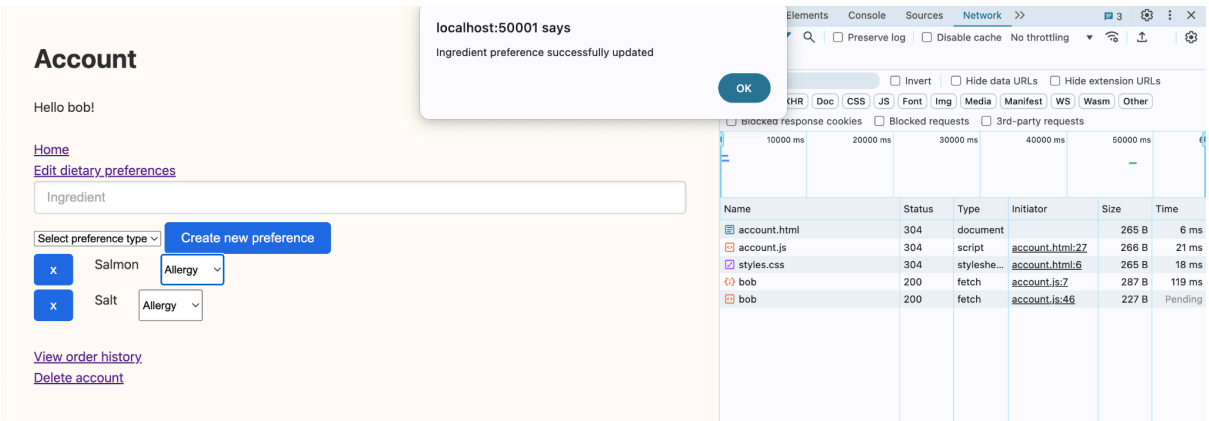


(database before and after account deletion, showing that user 2 had allergies to peanuts, and no longer existing in database afterwards)

```
SQL>select * from hasdietarypreference where customerid = 2;

CUSTOMERID INGREDIENTNAME
---------- --------------------------------------------------
PREFERENCETYPE
--------------------
2          Peanuts
Allergy


SQL> select * from hasdietarypreference where customerid = 2;

no rows selected

SQL>
```

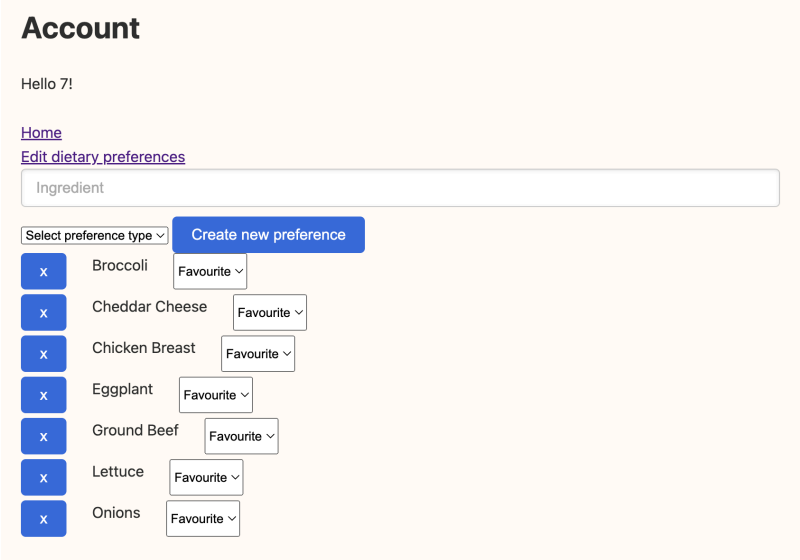
UPDATE: Updating ingredient preferences of the user. [appService.js, line 152]



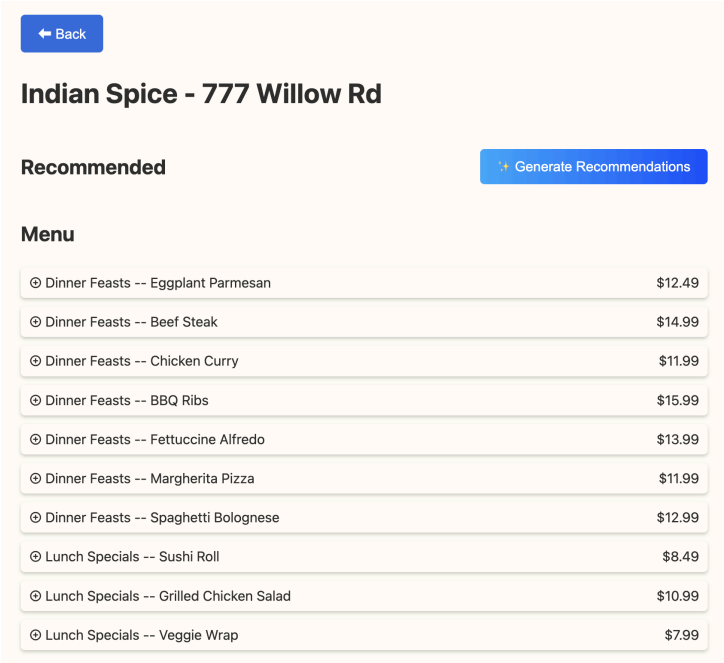


SELECTION: Select MenuItems in a menu associated with a specific Restaurant. Filter out menu items that contain an ingredient that a user is allergic to. [appService.js, line 190]

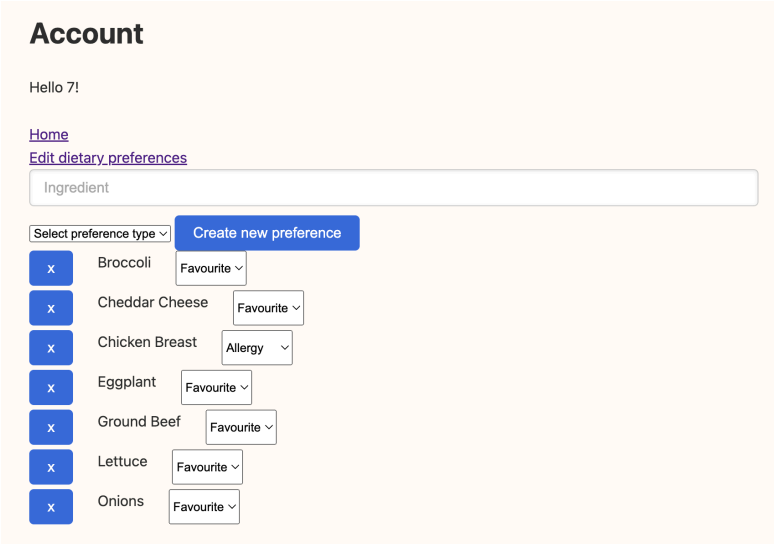
(All dietary preferences are set to “Favourite”)



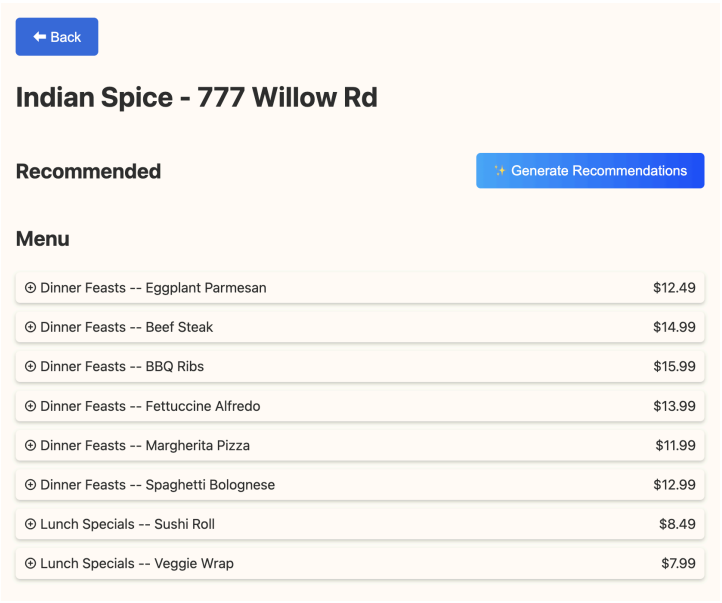
(Full Menu Shown (no filtering based on “Allergy” dietary preference))



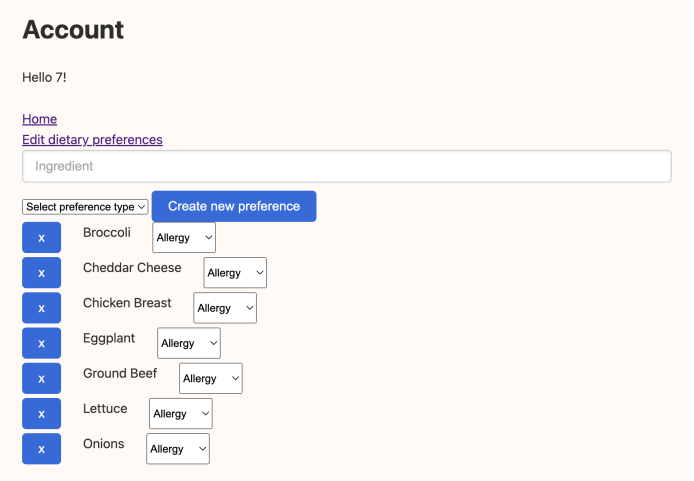
(Change “Chicken Breast” to “Allergy”)



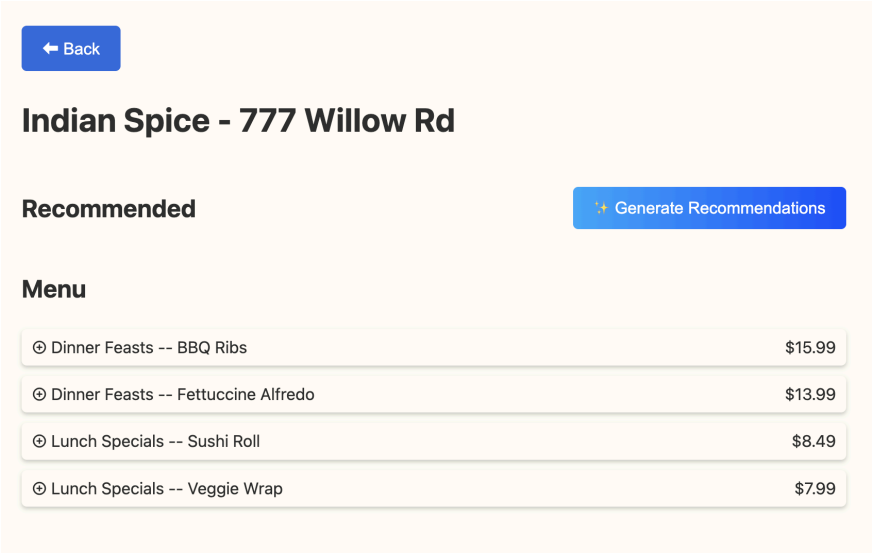
(Items containing “Chicken Breast” ingredient automatically filtered out when user views menu again)



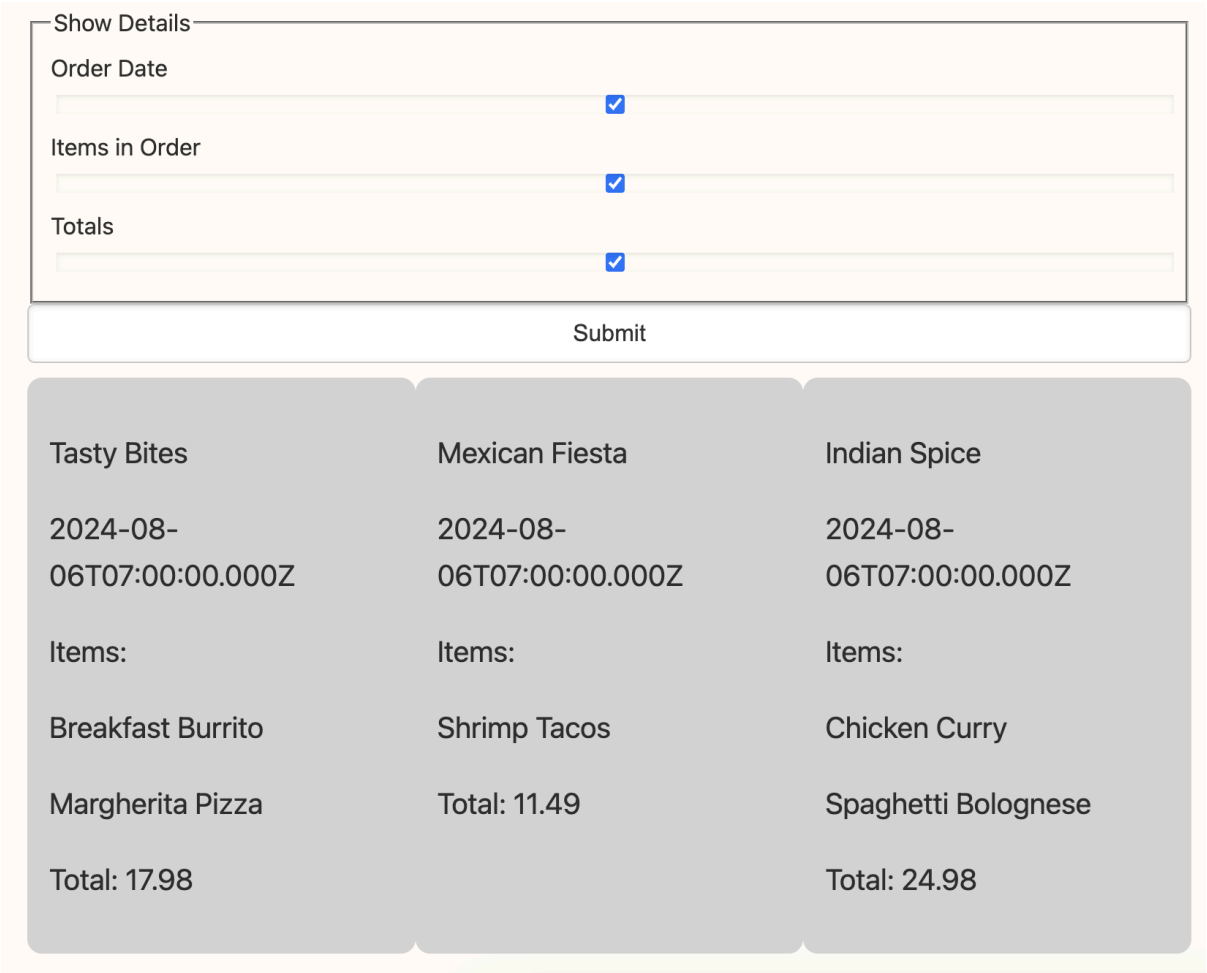
(Arbitrarily set all dietary preferences to “Allergy”)



(Menu further filtered)



PROJECTION: Filtering out what attributes to view in the order history. [appService.js, line 244]



Show Details

Order Date

Items in Order

Totals

Submit

Tasty Bites	Mexican Fiesta	Indian Spice
2024-08-06T07:00:00.000Z	2024-08-06T07:00:00.000Z	2024-08-06T07:00:00.000Z
Total: 17.98	Total: 11.49	Total: 24.98

Show Details

Order Date

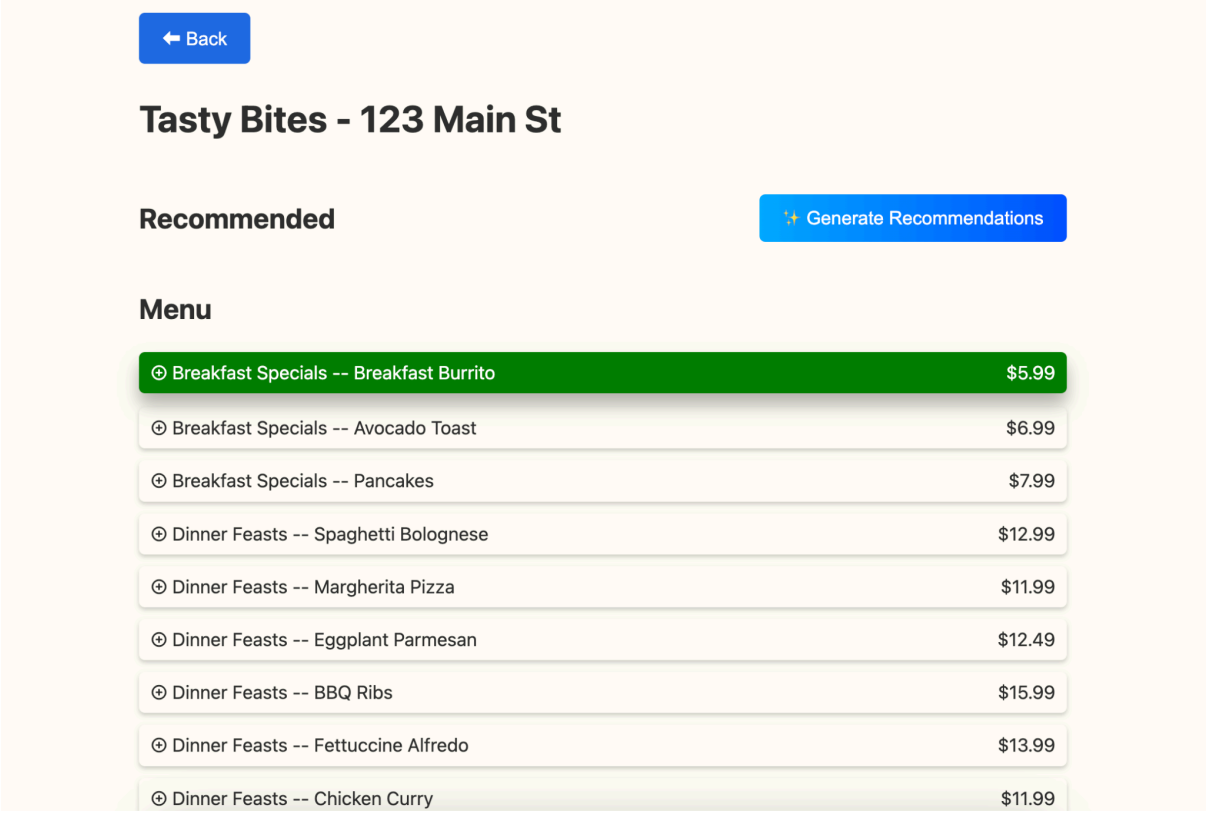
Items in Order

Totals

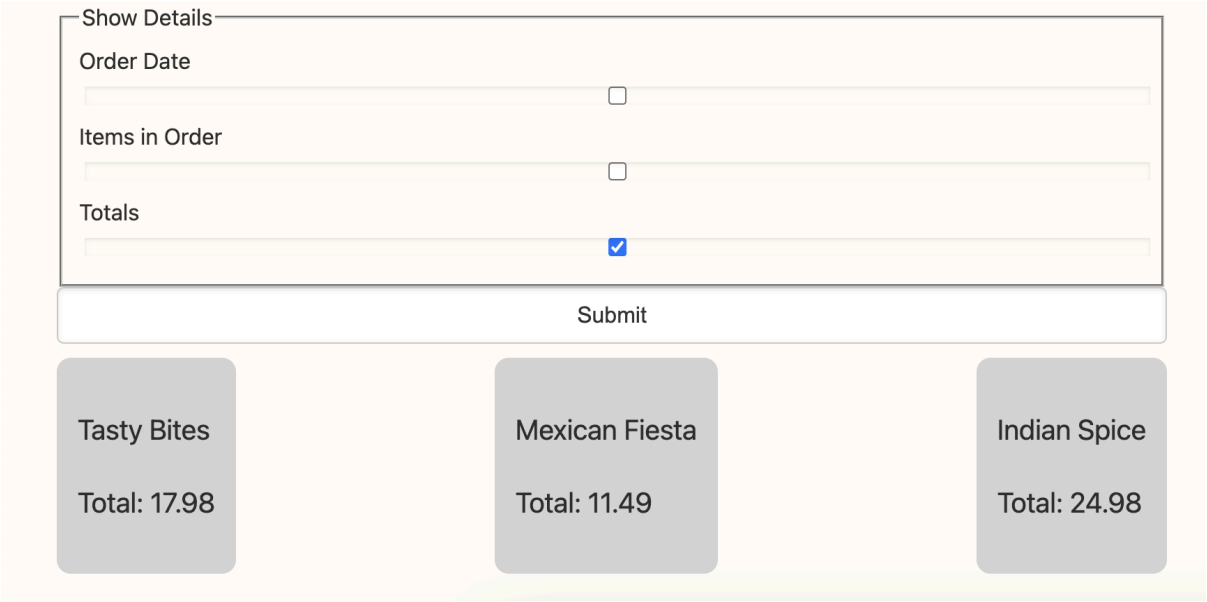
Submit

Tasty Bites	Mexican Fiesta	Indian Spice
2024-08-06T07:00:00.000Z	2024-08-06T07:00:00.000Z	2024-08-06T07:00:00.000Z

JOIN: Retrieving menu items and selecting the items. [appService.js, line 190]

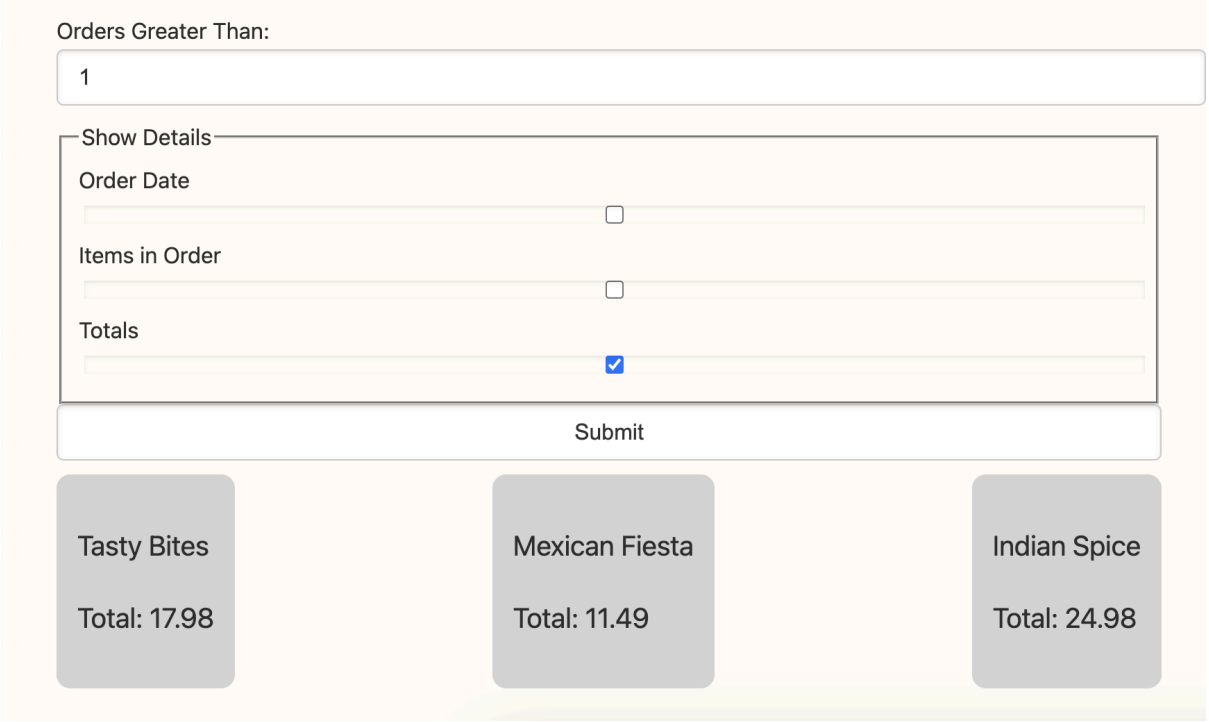


AGGREGATION WITH GROUP BY: Finding the totals of each order in history. [appService.js, line 269]

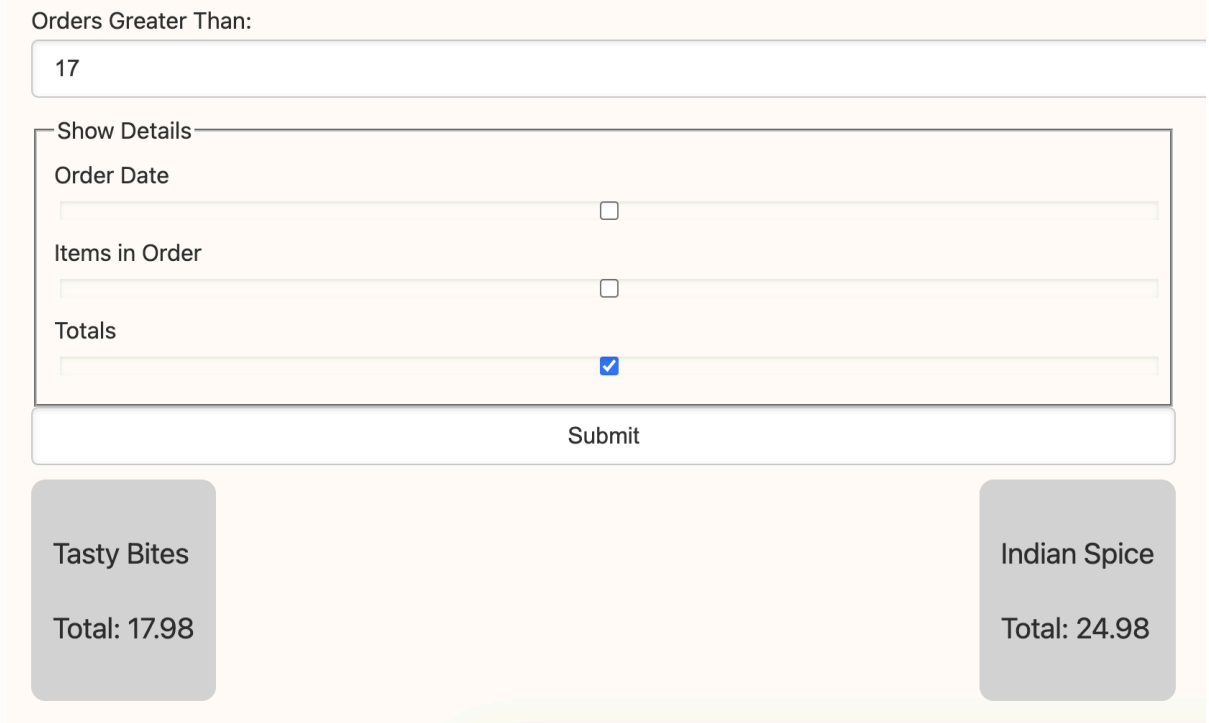


AGGREGATION WITH HAVING: Finding the totals of orders that are greater than a threshold. [appService.js, line 315]

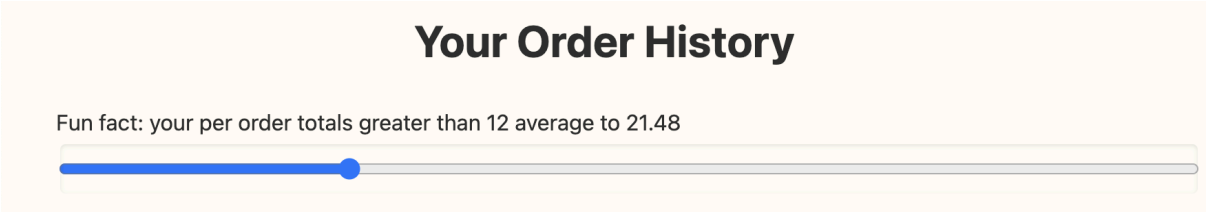
(order history when the threshold is set at 1)



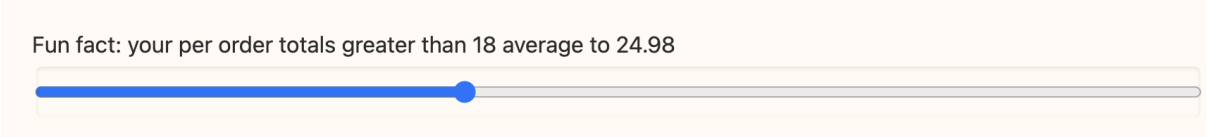
(order history when the threshold is set at 17)



NESTED AGGREGATION WITH GROUP BY: Calculates the average of orders given a threshold. [appService.js, line 289]

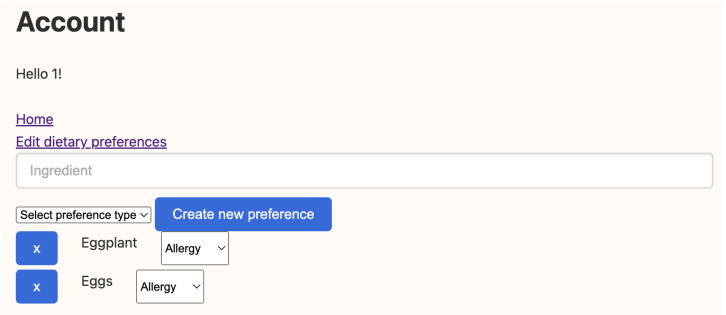


(after raising it to 18)

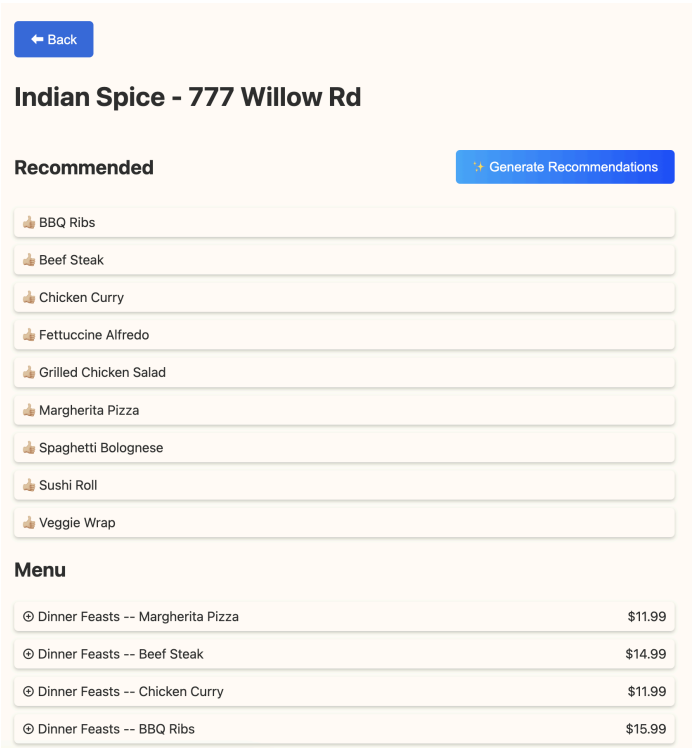


DIVISION: Select the MenuItemName of a MenuItem at a restaurant that contains ALL of a user's favourite ingredients. [appService.js, line 212]

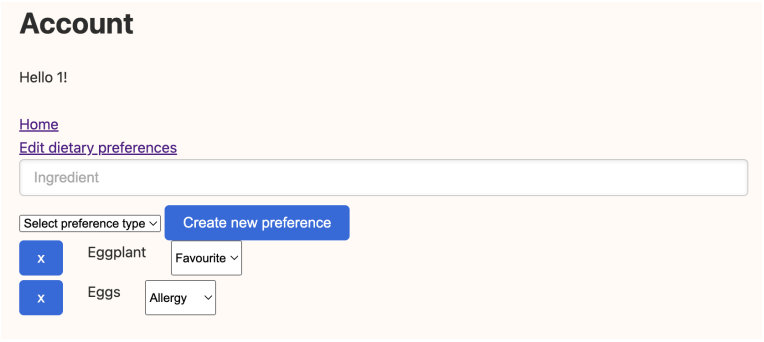
(No "Favourites" under dietary preferences)



(Every item is recommended because user has no "Favourite" preference)



(Change eggplant preference to “Favourite”)



(Only eggplant parmesan is recommended)