Riley Libens, Alex Ely, Phillip West, Winson Li  
4/25/2022

Saifuddin Mahmud  
Task 1  


Restaurant Info and User Review Database

**Purpose**

The purpose of our project is to create a database for users to post data pertaining to a restaurant’s foods and other important information, and to post reviews and ratings for each restaurant. We want to offer a wide variety of information for providing greater functionality and convenience in the food review community. Our application will store information such as the location of a restaurant, the type of service and food, the cost of food, nutritional value, service rating, food rating, cost rating, general information on the restaurant itself, and a section where each user can input their contact information like their email or phone number. The main idea behind our application is to make it easier and more intuitive when finding restaurants, posting info about restaurants, and posting reviews, by having a database with a large quantity of details regarding various restaurants, their food, their reviews, and the reviewers.

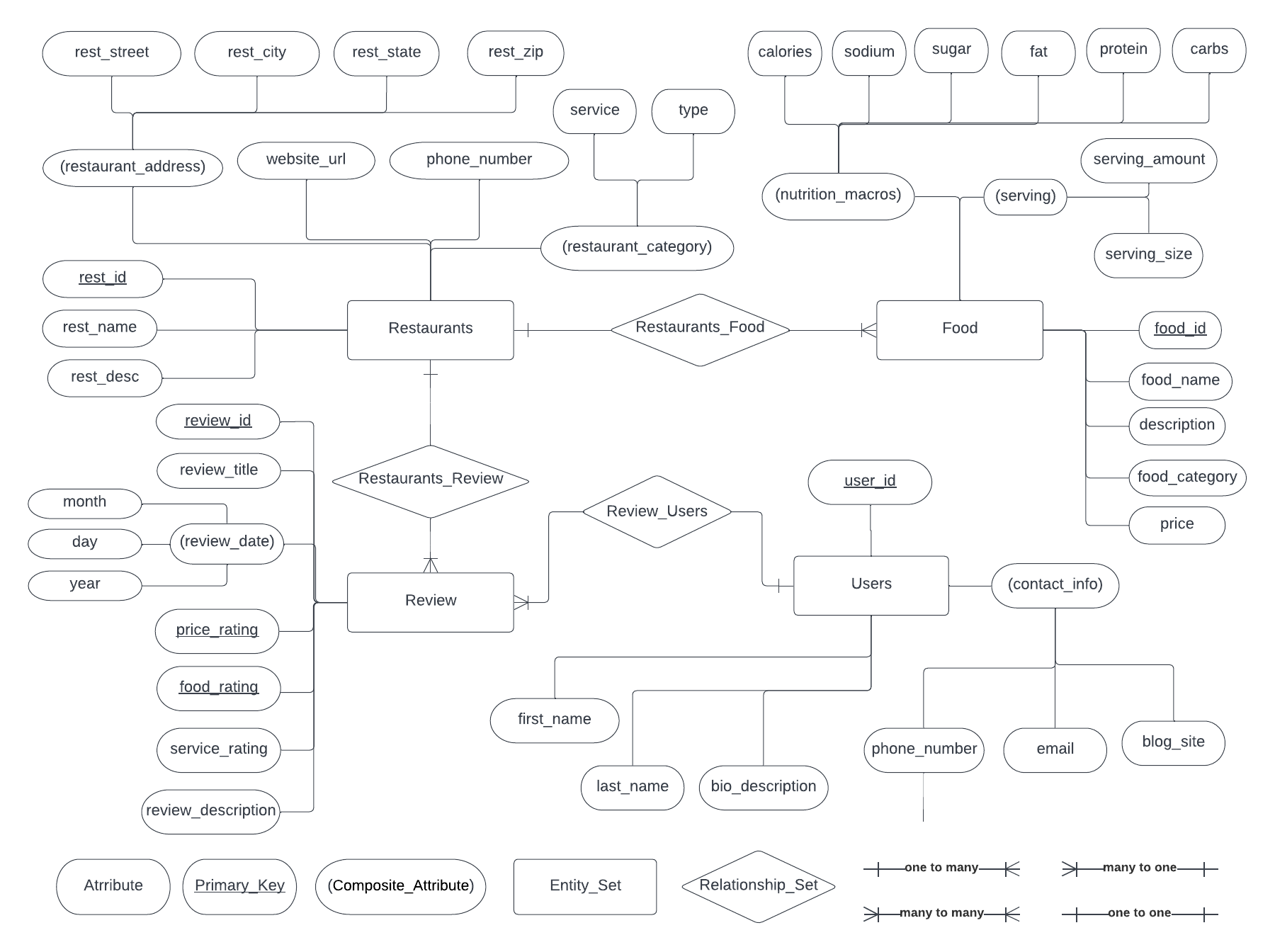
**Data Requirements**

We want our data to be informative to users about the food items the restaurants offer, contact info, the prices, food info, and to have a uniform location for all restaurant data, as well as reviews and reviewer profile info.

* Restaurants
  + Unique RestaurantID
  + Restaurant Name
  + Restaurant Description
  + Street Name
  + City Name
  + State
  + Zip
  + Service (Delivery, Dine-in, Take-out, etc.)
  + Type (American, Chinese, Italian, Mexican, etc.)
  + Phone Number
  + Website URL
* Food
  + Unique FoodID
  + Food Name
  + Food Description
  + Food Category (appetizer, side, entrée, vegan, etc.)
  + Price
  + Description
  + Calories
  + Protein
  + Carbs
  + Fat
  + Sodium
  + Salt
  + Carbs
  + Serving Size
  + Serving Amount
* Review
  + Unique ReviewID
  + Review Title
  + Month
  + Day
  + Year
  + Price Rating
  + Food Rating
  + Service Rating
  + Review Description
* Users
  + Unique UserID
  + User First Name
  + User Last Name
  + User Email
  + User Phone Number
  + User Blog Site
  + User Profile Bio Description

**Functional Requirements**The application will be hosted on a local XAMPP server using MarinaDB. The application will present a detailed view of various restaurants, each specified by a restaurant ID, and will include information on restaurant foods, location, prices, nutrition, etc. The view is broken down into various restaurants and their specific food items, as well as other details such as user review and details on the user. The final planned project is a web application mainly designed with the desktop view in mind, but will also be usable on mobile. In our web application, we will offer users the ability to update the information on restaurant and food listings, as well as the ability to update their review, rating, and user profile information. However, the application users will mainly be restaurant customers, as it is used to view restaurant details such as location and contact information, food item details, such as prices, nutritional details, etc., and to post reviews. Our Restaurant Info and Review Database will act as a valuable and useful tool for customers to find and review restaurants as well as for reviewers to possibly contact each other.

Task 2

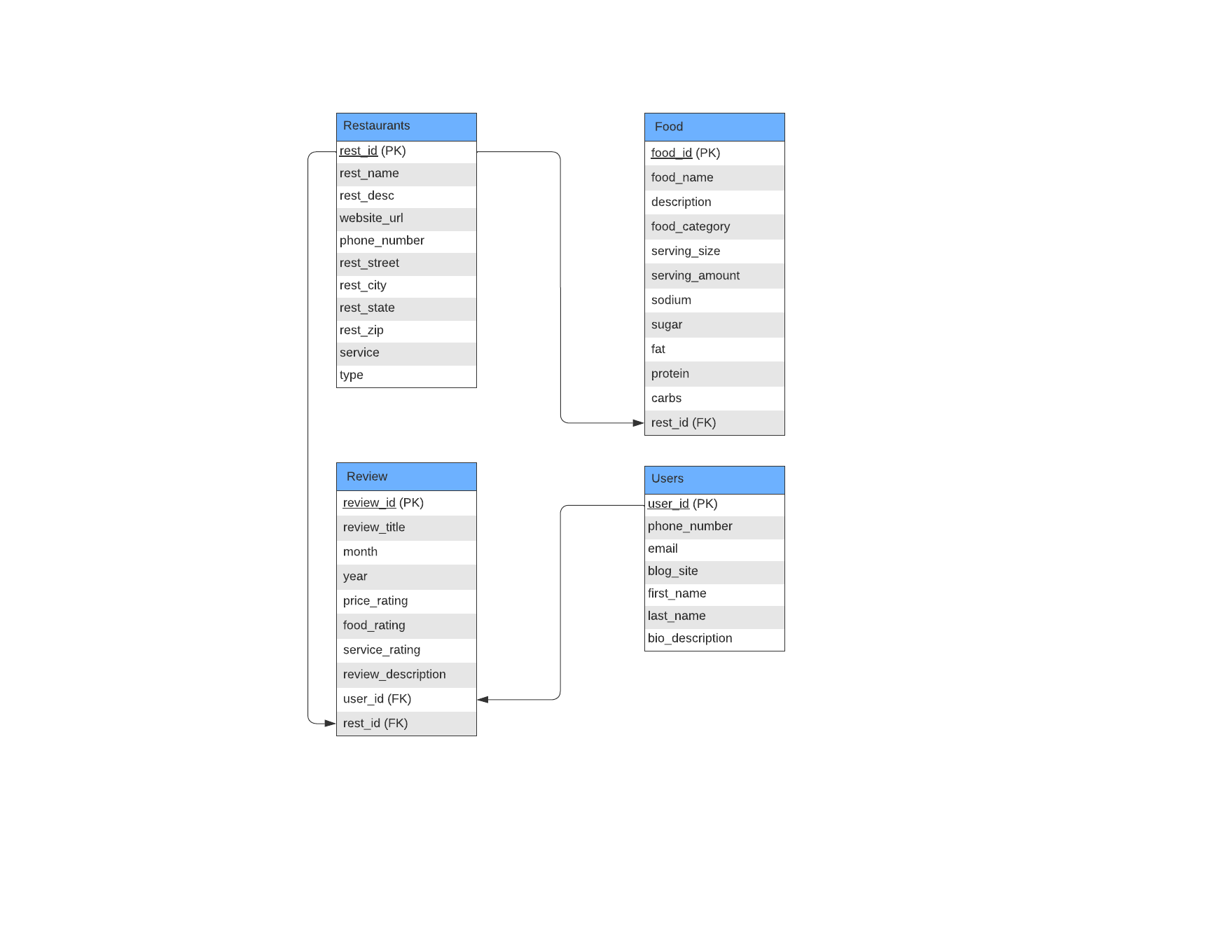
**Provide a conceptual design of your database: Identify possible entities and relationships among the entity sets then draw an EntityRelationship (ER) diagram for your database. Please make sure that you underline primary keys in the ER diagram.**  
  
  
**Explain the notations you have used in the ER diagram (You are open to use any notations, but you must describe them)**  
  
The ER diagram has a key that shows the notations at the bottom. The ER diagram has attributes as oval circles and entity sets are in rectangular boxes. Attributes that are composite have parentheses around the inside text while primary keys have the text underlined. There are no multivalued attributes for this ER diagram so it is not listed in the key. Of course, connections are shown using lines. The ER uses the line notations for a one to many relationship where on the left is one small line parallel to the connecting line and on the right has a parallel line where the connecting line forks to three lines. The ER uses the line notations for a many to one relationship where, on the left there is a parallel line and the connecting line forks to three lines, while the right is one small line parallel to the connecting line. The ER uses the line notations for a many to many relationship where the left has a parallel line and the connecting line forking to three lines and the right does also. While it is shown in the key, this ER diagram does not have a many to many relationship in it. One to one uses one small line parallel to the connecting line for both right and left. While it is shown in the key, this ER diagram does not have a one to one relationship for its entity sets.

**Describe at least two relationships among entity sets from your ER diagram.**  
The entities Restaurants and Food have a one to many relationship called Restaurants\_Food in the ER diagram with Restaurants being the one and Food being the many. A single instance of Restaurants would be linked to many instances of Food since a restaurant would have many food entries but a food entry would not have different restaurants. This is because even if two restaurants have the same food, the different name, price, other details, and most importantly the food\_id would differ. The food\_id means that each entry of Food is uniquely associated with an instance of Restaurants. In a relational schema, Food would have a rest\_id foreign key.  
  
The entities Review and Users have a many to one relationship called Review\_Users in the ER diagram with Review being the many and Users being the one. A single instance of Review would be linked to one instance of Users since a review would only have one user, or reviewer/poster, but a user could have many reviews. In a relational schema, Review would have a user\_id foreign key.  
  
The entities Restaurants and Review have a one to many relationship called Restaurants\_Review in the ER diagram with Restaurants being the one and the Review being the many. A single instance of Restaurants would be linked to many instances of Review because a restaurant would have many reviews but one review would not go to many restaurants since a review has a unique review\_id. In a relational schema, Review would have a rest\_id foreign key.

**Convert ER diagram to Relational Schemas:**  
Restaurants(rest\_id, rest\_name, rest\_desc, rest\_street, rest\_city, rest\_state, rest\_zip, website\_url, phone\_number, service, type)  
Food(food\_ID, food\_name, description, food\_category, price, serving\_size, serving\_amount, calories, sodium, sugar, fat, protein, carbs, *rest\_id*)

Review(review\_ID, review\_title, month, day, year, price\_rating, food\_rating, service\_rating, review\_description, *user\_id*, *rest\_id*)

Users(user\_id, phone\_number, email, blog\_site, first\_name, last\_name, bio\_description)

**From Relational Schemas/directly from the ER diagram, draw the Schema Diagram of your database.**

**SQL queries and features such as insert (open bank account), delete (delete account), update ( update balance after money withdrawal), joins, view, trigger etc.**Our database has implemented features to insert, delete, and update entries to restaurants, food, reviews, and users. There are views used to bring up certain data or sets for site users like inserting or updating restaurant food which projects elements where there’s details from the food table and restaurants id from restaurants table to show which restaurant is being edited. More details for the queries and features can be found in the project folder’s source files, the backend folder and .sql database file.  
  
  
  
  
  
  
  
  
**Website Documentation**

**Google Doc:** [**https://drive.google.com/drive/u/0/folders/1sRaUM0dIH-gKOUGxFYIKHDgWIUAVKqcz**](https://drive.google.com/drive/u/0/folders/1sRaUM0dIH-gKOUGxFYIKHDgWIUAVKqcz)

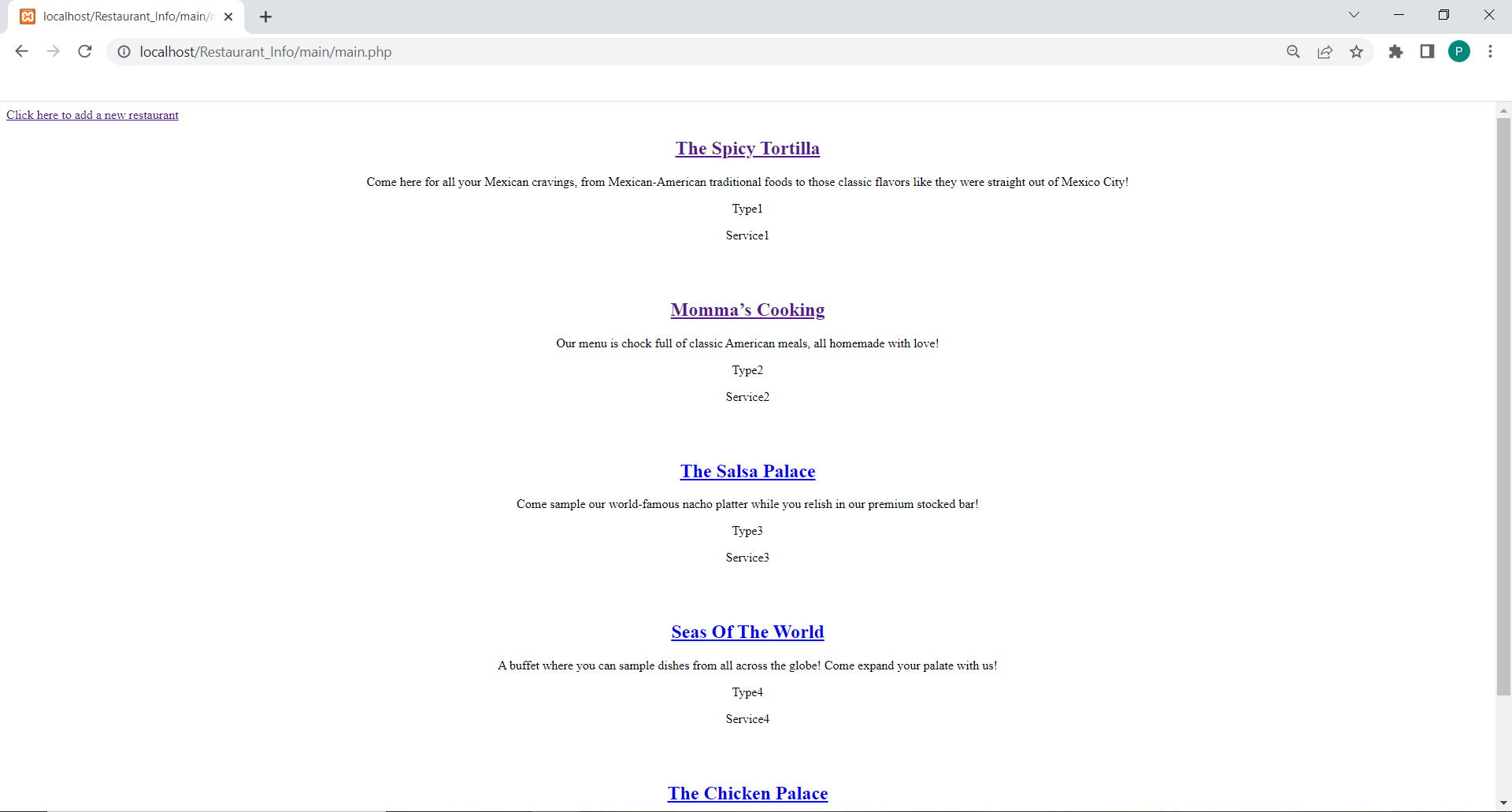
**Github:**

[**https://github.com/Alexguy891/CS33007/tree/main/Restaurant\_Info**](https://github.com/Alexguy891/CS33007/tree/main/Restaurant_Info)

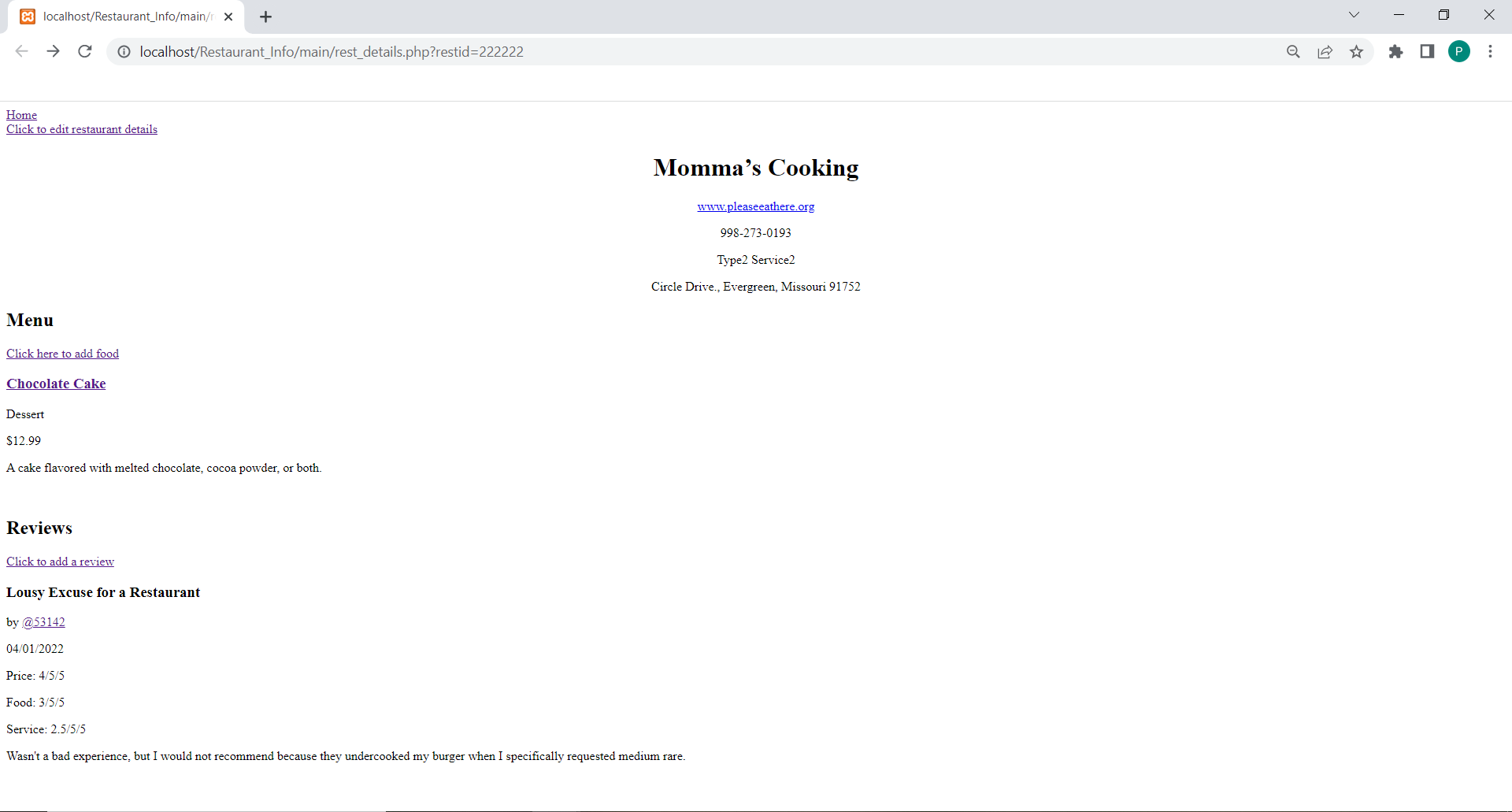
**Running this Database:**

Download the Restaurant\_Info folder from the Google Drive (unzip it if it downloads as a zipped file). Place the Restaurant\_Info folder into the htdocs folder within your xampp folder. Run the Apache and MySQL services in your XAMPP program. In localhost/phpMyAdmin in your web browser, create a database called restaurant\_info and run the SQL statements written in the file labeled “TestDB\_SQL\_Statements.rtf”. Create a user with username user, password pass, operating on host localhost, and give it permissions to the entire database. In a web browser, go to the following page: <http://localhost/Restaurant_Info/main/main.php>, and you will have access to the database.

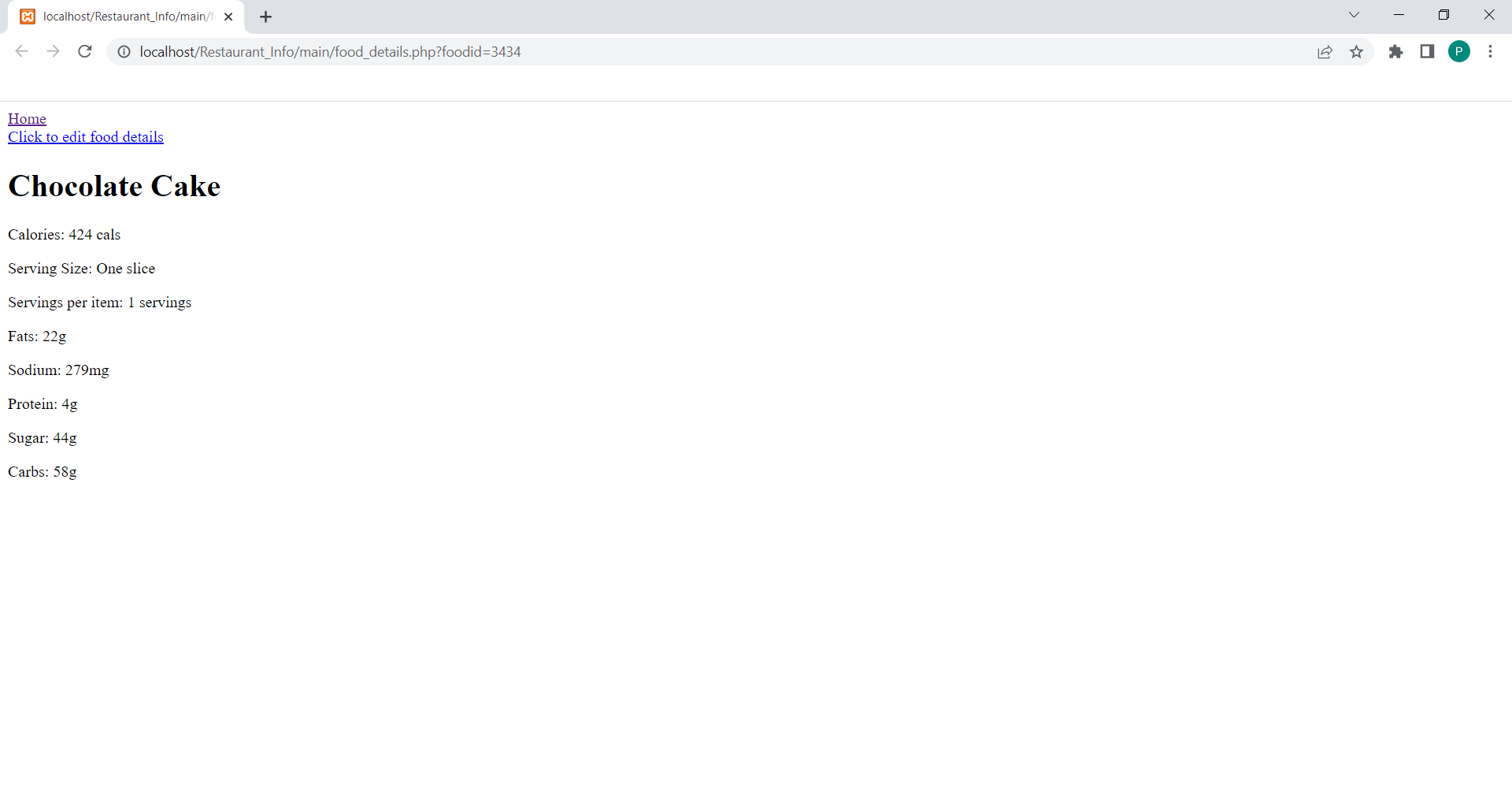
**Screenshots and Summaries:**

****

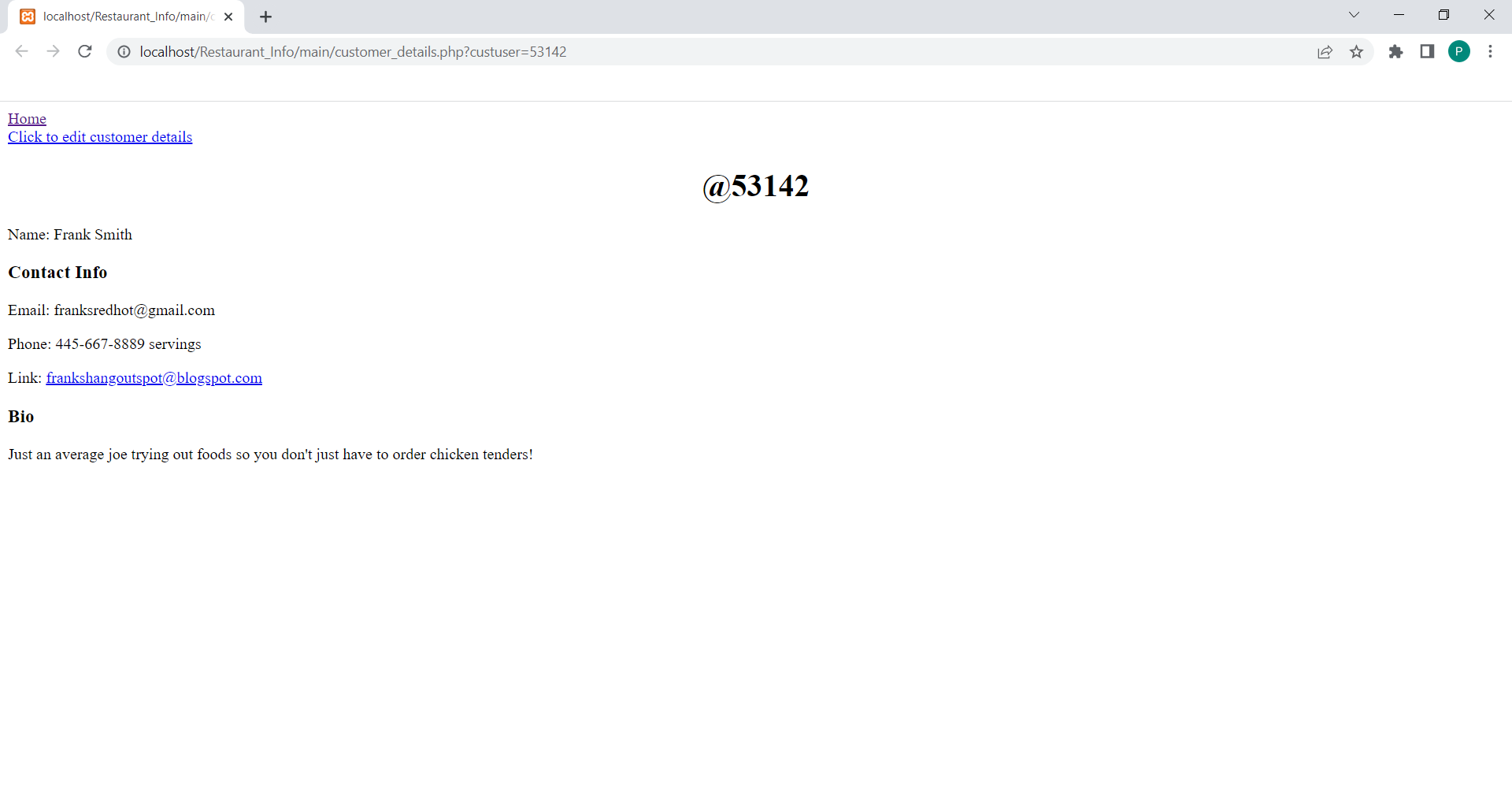
Shown: The main page of our restaurant database, displaying its name, description, type and service offered.



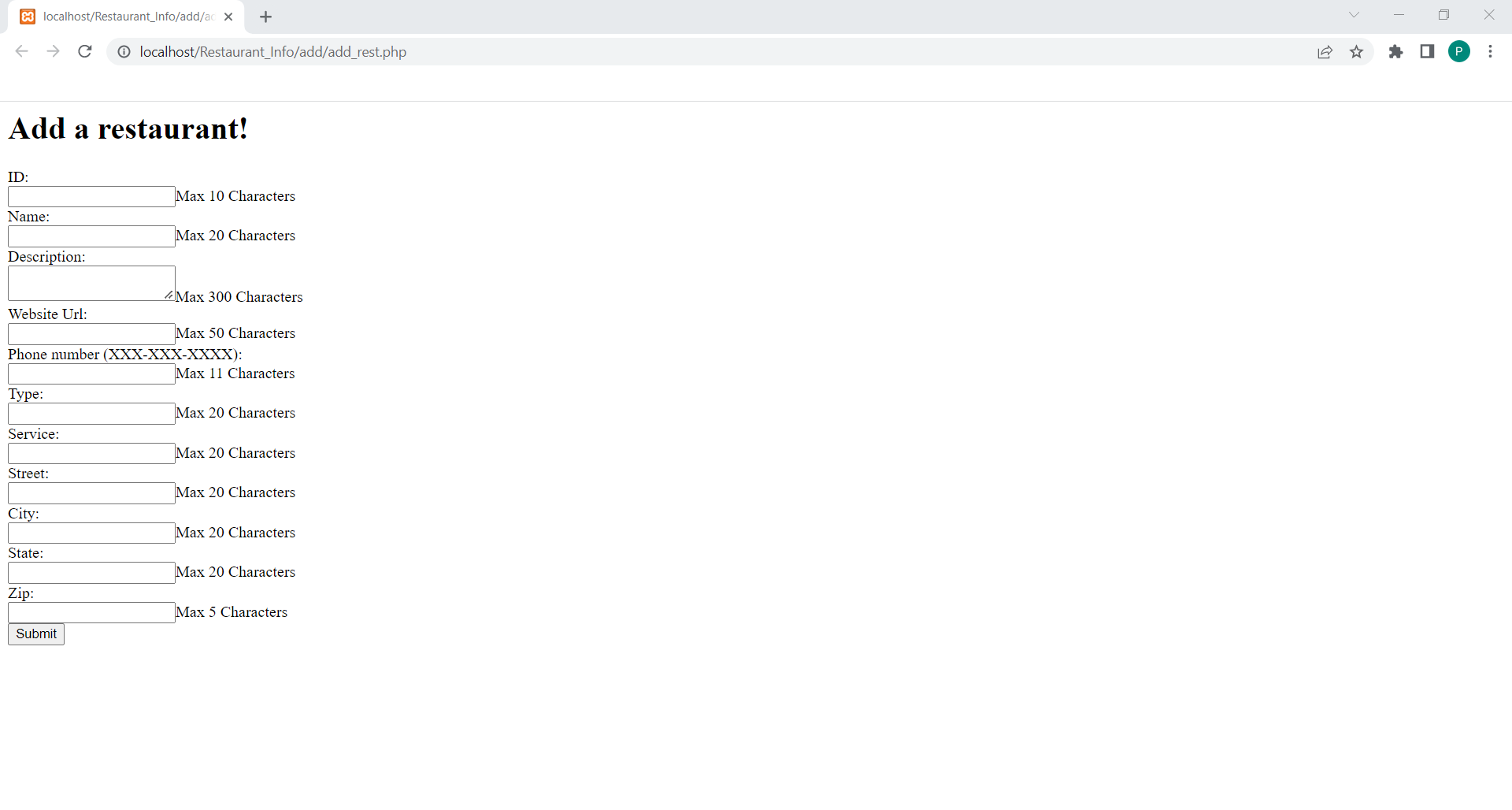
Shown: A restaurant page, displaying its location and webpage, menu, details about each food item offered, and reviews for that particular restaurant.



Shown: A web page showing nutritional information for a particular food item.



Shown: A web page showing the name, contact information and bio for a particular user.



Shown: An example edit/addition page, this one in particular for adding a new restaurant to the database, with all the fields required to create a new restaurant.

There is a page to add a new restaurant, edit details for a restaurant, add food to a particular restaurant, edit nutritional information for a particular food, add a review, and edit user details.

**Future plans:**

For future plans, we would like to make our database and web page more autonomous and user-friendly. Inclusions like bounds and error checking, better design and functionality, account login capability, and user and employee differences would create more integrity within the database and keep checks on what additions and edits are passed.