Course: CS 340 - Fall 2021
Assignment: Project Step 7 (Portfolio Assignment)
Group: 49
Team members: Chi Hang Leung, Tobi Fanibi
Team's name: DROP TABLE mic;
Project Title: Restaurant Table Manager

Project Website:

http://137.117.84.69:7000

Project Outline:

In the US, customers experienced a wait in nearly 93% of restaurants (FSR magazine, 2013). The average restaurant wait time is about 23 minutes per party and nearly a third of them have wait time of more than 30 minutes (FSR magazine, 2013). To provide an efficient solution to ease the wait time problem, the webapp we are creating will be a management system designed for restaurant managers or hostesses to manage a restaurant's tables, waiters, and customers. The database will store pertinent information about the tables with their features, waiters, customers, and waiting lists. This website will allow managers or hostesses to manage a waiting list of customers and find the tables that fit customers' needs the most. It also keeps track of each visit by recording details such as the information about the table/waiter/customer, time spent, and check/tips amount.

Citation:

FSR magazine. (2013, October 8). *Study released on Average Restaurant Wait Times*. Retrieved October 7, 2021, from https://www.fsrmagazine.com/content/study-released-average-restaurant-wait-times.

Database Outline

dining_tables: Representing actual tables in a restaurant.

table_id: int, auto_increment, unique, not null, primary key (Table's ID)
num_seat: int, not null (Number of customers that the table can be seated)
feature_id: int, default null, foreign key (represents the feature a dining table has)

Relationships:

- 1:M relationship with *visits* using table_id as the foreign key in *visits*. A dining table does not have to be used in a visit but can be used repeatedly in many visits (at different times).
- M:1 relationship with *table_features* using feature_id as the foreign in dining_tables. A feature of the table does not have to be in any dining tables but can appear in many of them.
- M:M relationship with waiters, using *visits* as composite entities. A dining table can have different waiters while servicing customers at different times. A waiter can also serve customers at multiple dining tables at the same or different times.
- M:M relationship with customers, using *visits* as composite entities. A customer can be served by the same waiter at different times and a waiter can serve many different customers at the same or different times.

waiters: Represents working waiters in a restaurant.

waiter_id: int, auto_increment, unique, not null, primary key (Waiter's ID)
waiter_name: varchar(255), not null (Full name of the waiter)

Relationships:

- 1:M relationship with *visits* using waiter_id as the foreign key in *visits*. A waiter does not have to serve in every visit but can participate in many visits.
- M:M relationship with dining_tables, using *visits* as composite entities. A waiter can serve at many dining tables and a dining table can have many waiters while visited by customers at different times.
- M:M relationship with customers, using visits as composite entities. A waiter may serve the same customer many times during different visits and a customer can also be served by many waiters during different visits.

customers: Representing the customer (per table) who makes a reservation or requests a table.

customer_id: int, not null, primary key (Customer's ID)
customer_name: varchar(255), not null (Full name or preferred name of the customer)
customer_phone: varchar(255) (Phone number of the customer, if they don't mind leaving a number. Can be null)

Relationships:

- 1:M relationship with *waiting_lists* using customer_id as the foreign key in *waiting_lists*. A customer can make zero to many reservations at different times. On the other hand, each reservation corresponds to at most one customer. Setting Null to customer_id in the waiting_lists will remove the relationship and indicate that the reservation has been cancelled.
- 1:M relationship with *visits* using customer_id as the foreign key in *visits*. A customer can visit the restaurant many times and have many visits but they can also be new customers with reservations and have not been served yet.
- M:M relationship with waiters, using *visits* as composite entities. A customer can be served many times by the same waiter and the same waiter can serve many different customers.
- M:M relationship with dining_tables, using *visits* as composite entities. A customer can be seated at the same table at different times and a table can be seated by many different customers at different times.

waiting_lists: A waiting list for the tables. Reservations and walk-in requests are treated the same.

queue_id: int, not null, primary key (ID for the queue)
customer_id: int, foreign key (Customer's ID)
num_seat: int, not null (Number of seats that the reservation is for)
reserved_time: datetime, not null (Date and time the customer want to dine)
requested_feature_id: int, default null, foreign key (represent what kind of table the customer want)
is_seated: tinyint, not null (Whether (1) or not (0) the customer has been seated)

Relationships:

- M:1 relationship with *table_features* as many customers on the waitlist may request the same kind of table but only one feature of a table can be requested for each reservation.
- M:1 relationship with *customers* using customer_id as the foreign key in *waiting_lists*. Each item on the waiting_lists corresponds to at most one customer. If an item does not correspond to any customer with Null as the customer_id, then it means the reservation has been cancelled. However, a customer can make many reservations at different times.

table_features: A list of possible features that a dining table can have. Customers can request a specific type of table while making reservations.

feature_id: int, not nul, primary key (ID for the feature of a dining table)
feature_description: varchar(255) not null (the description of the special feature of a dining table)

Relationships:

- 1:M relationship with dining_tables as many different tables may have the same special feature but each dining table may only have one feature.
- 1:M relationship with waiting_lists as many different reservations may request for the same type of table but each reservation can only have one request at the most.

visits: Represent each visit by the customer with his/her guests at a single table.

visit_id: int, auto_increment, unique, not null, primary key (ID for each visit)
table_id: int, not null, foreign key (Table's ID)
customer_id: int, not null, foreign key (Customer's ID)
waiter_id: int, not null, foreign key (Waiter's ID)
num_guest: int, not null (Number of guests within a single visit)
time_start: datetime, not null (Date and Time of seating at the table)
time_stop: datetime, not null (Date and Time of leaving the table)
check_amount: decimal(5, 2), not null (Dollar amount of the check not including tips)
tips_amount: decimal(5, 2), not null (Dollar amount of the tips for the wait staff)

Relationships:

- M:1 relationship with *customers* using customer_id as the foreign key in *visits*. Each visit is attributed to exactly one customer only but the same customer can have multiple visits at different times.
- M:1 relationship with *dining_tables* using table_id as the foreign key in *visits*. Each visit is attributed to exactly one dining table only but the same dining table can be used in multiple visits at different times.
- M:1 relationship with *waiters* using waiter_id as the foreign key in *visits*. Each visit is attributed to exactly one waiter only but a waiter can participate in multiple visits.

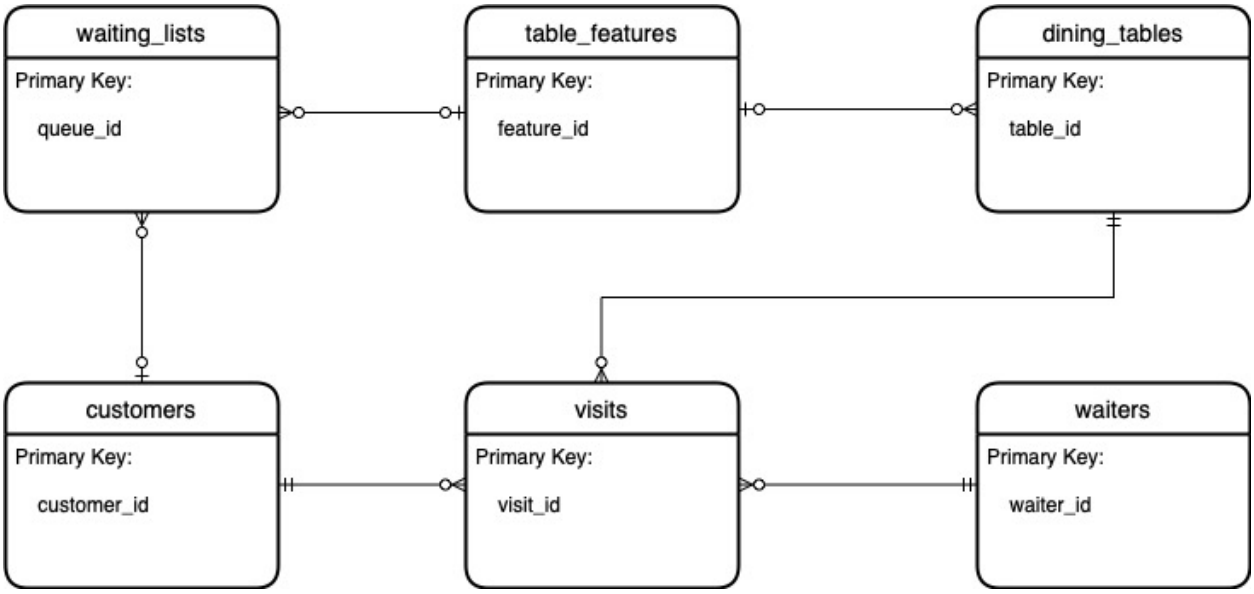
Relationships Summary

1:M customers : waiting_lists
customers : visits
dining_tables : visits
waiters : visits
table_features: waiting_lists
table_features: dining_tables

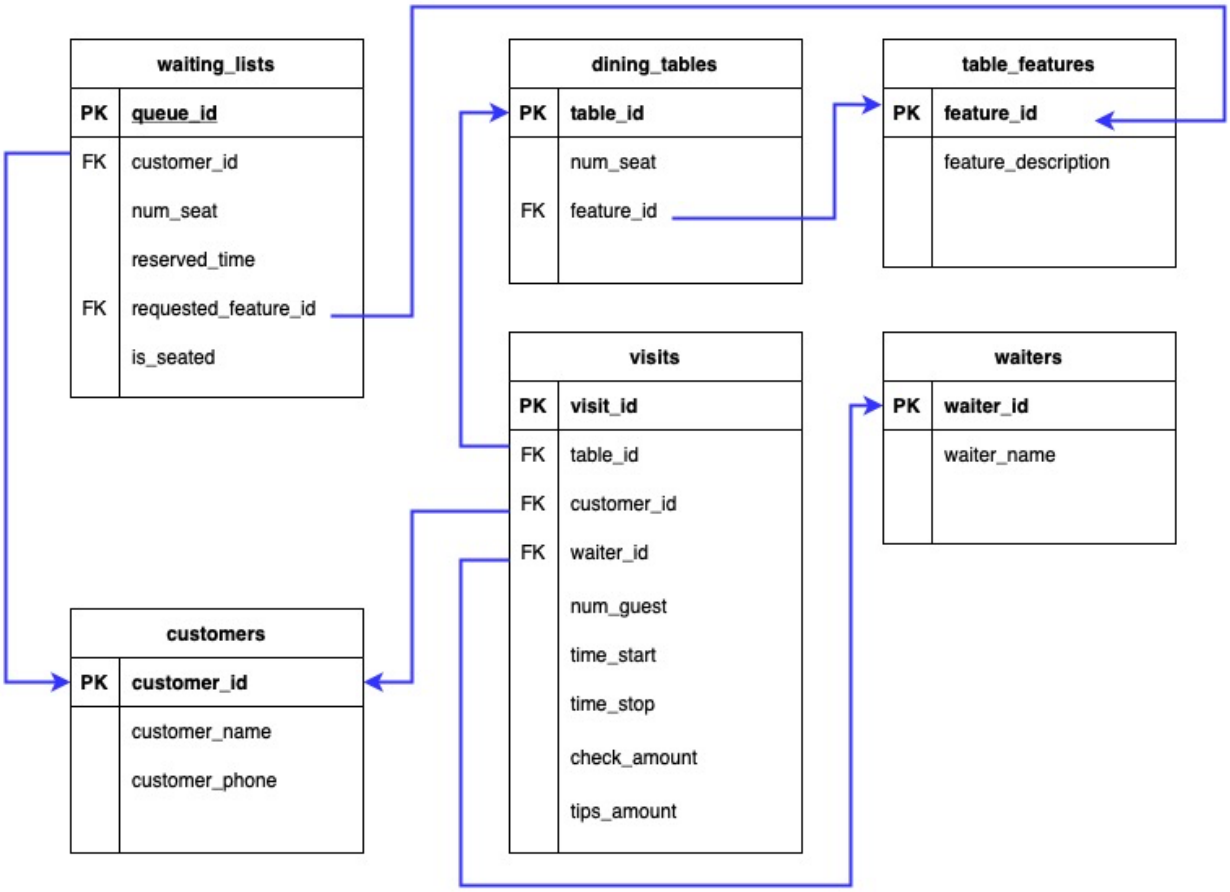
M:M customers : waiters
customers : dining_tables
waiters : dining_tables

Note: All M:M relationships are using *visits* as the composite entity.

Entity Relationship Diagram

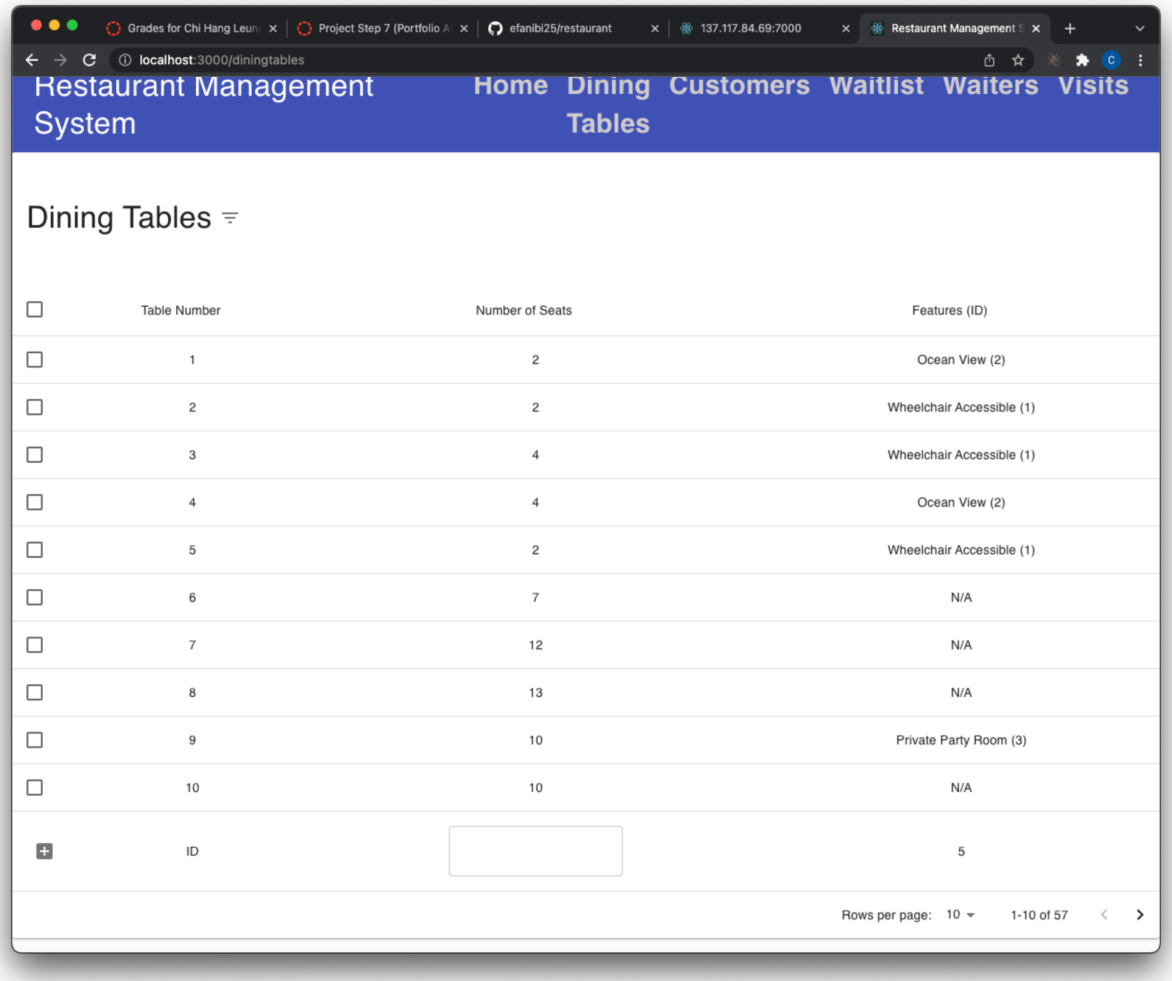


Schema:

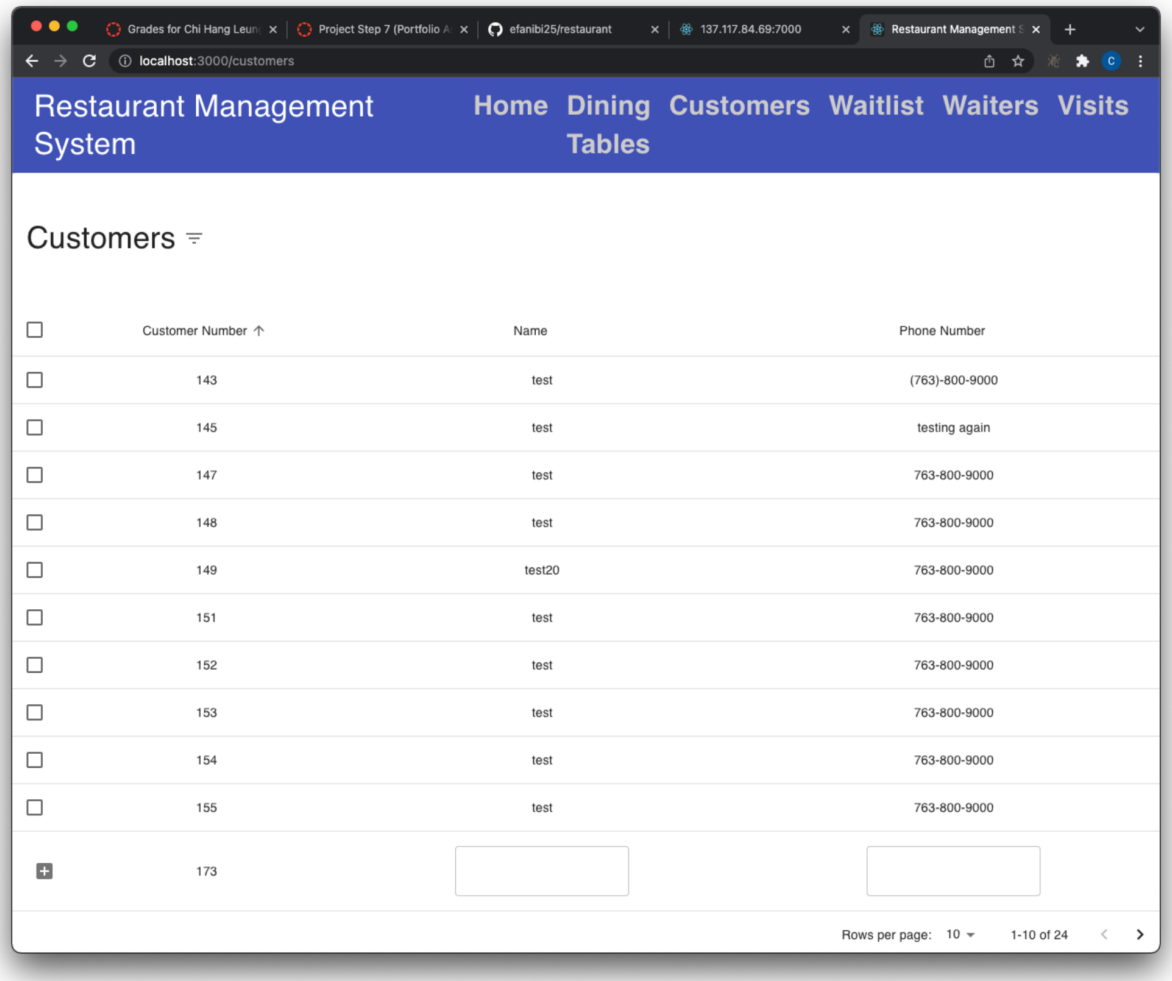


Screen Captures

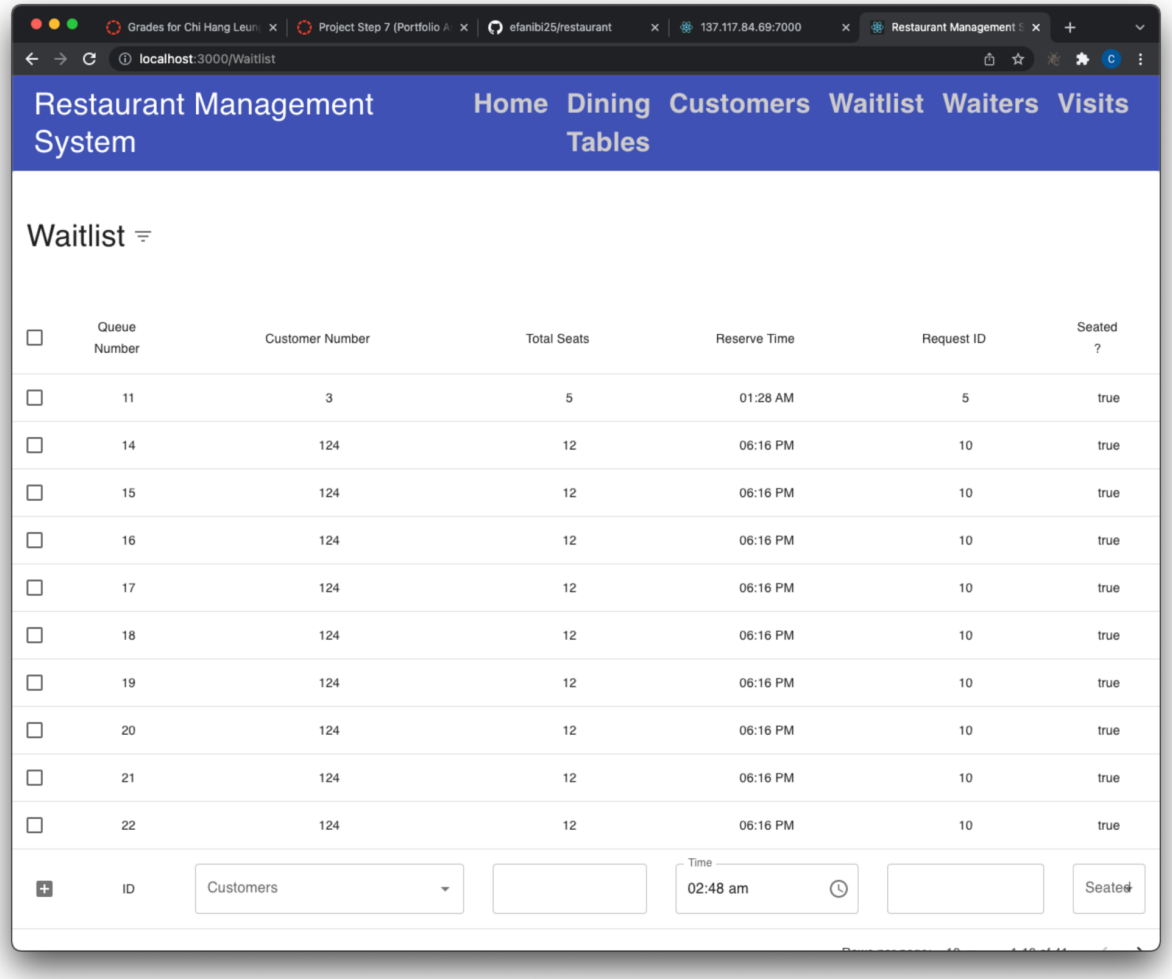
CREATE / READ / UPDATE / DELETE capable Dining Tables Page



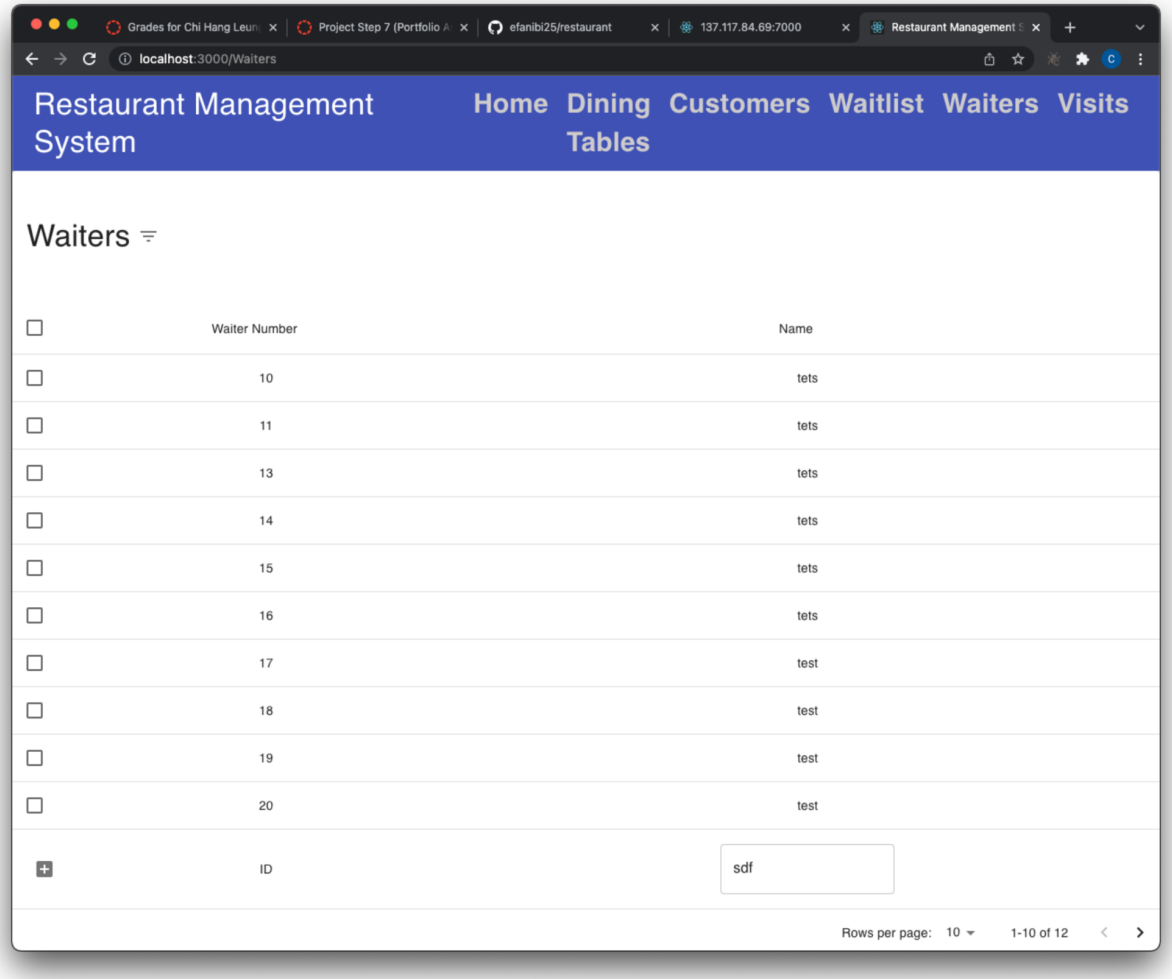
CREATE / READ / UPDATE / DELETE capable Customers Page



CREATE / READ / DELETE capable Waitlist Page



CREATE / READ / UPDATE / DELETE capable Waiter Page



CREATE / READ / DELETE capable Visits Page