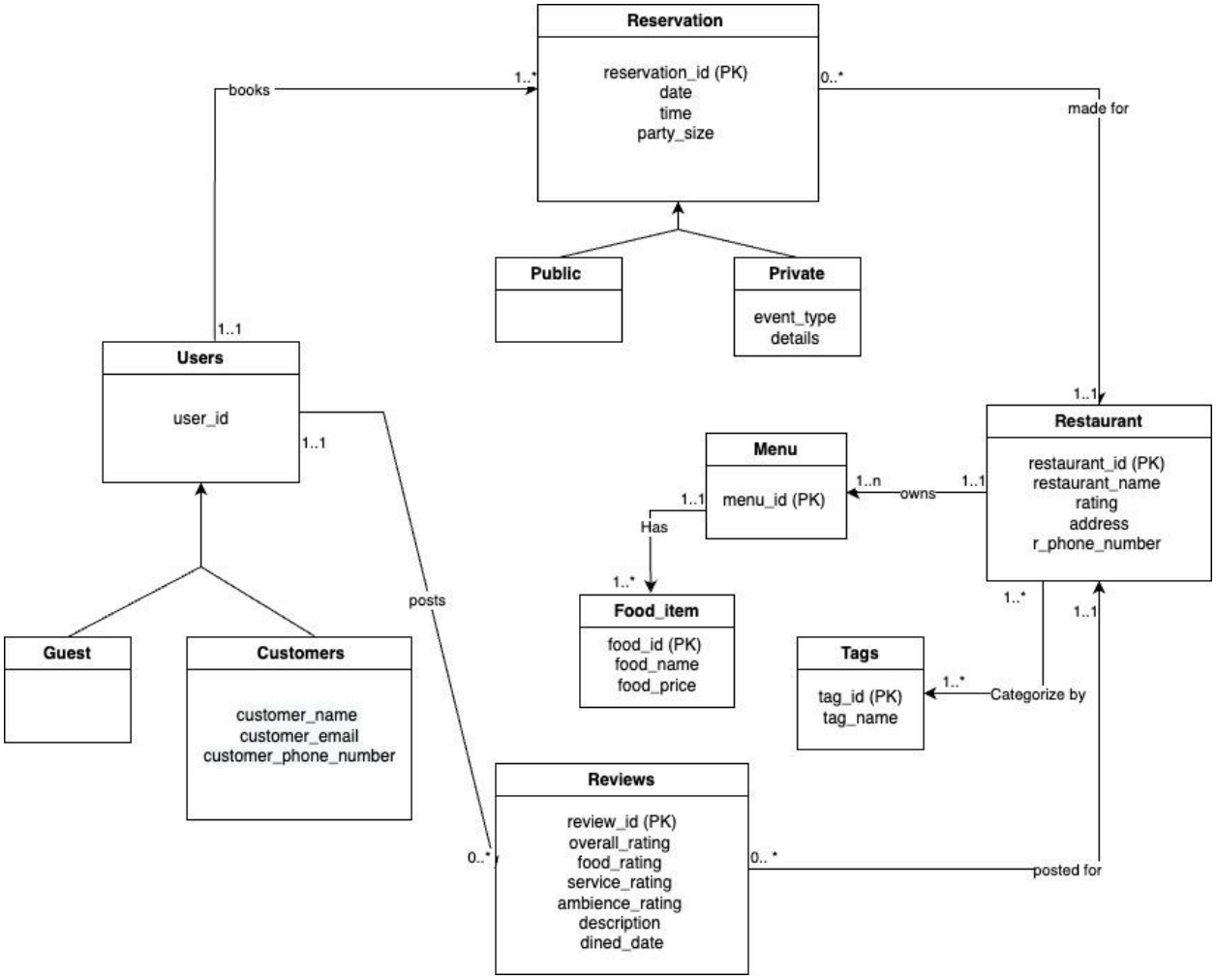
Team S3-6
Jiatong Li (jiatong4)
Anthony Jiang (akjiang)

Phase 2 Final Deliverable

Updated Conceptual Model/ User Stories



Assumptions: Customers can make more than 1 reservation. A restaurant can have 0 to multiple reservations for it. Customers don't have to leave a review, but each review can only be written by one customer. A restaurant can have 0 to many reviews. Restaurants have at least one tag, and can have multiple tags. Restaurants have at least one menu, and can have multiple menus (for drinks, desserts, main dishes, etc.). A menu must have at least one food item on it and a food item is unique to that menu. Guests and customers both count as users, but only customers have additional information (email, name, phone number) registered in the system

Users:

1. **Restaurants** - consists of managers/owners of a restaurant that aim to grow and improve the business
2. **Potential guests** - people who are exploring restaurants through OpenTable but have not booked a reservation or ordered anything
3. **Customers** - people who have interacted with a restaurant(s) in some way through OpenTable (booking a reservation, ordering takeout, posting a review, etc.)

10 User stories :

ID	Simple/ Complex/ Analytical	As a <role>	I want to <goal>	So that <reason>
US1	Complex	Potential guest	make reservations at a specific restaurant for a specific date and time	I won't have to wait in line when I get to the restaurant
US2	Complex	Potential guest	filter for restaurants by cuisine	I can find restaurants that suit my tastes
US3	Analytical	Potential guest	see how many high rated reviews and low rated reviews there are for a restaurant	I can judge if the restaurant is worth going to.
US4	Complex/ Analytical	Restaurant	compare the number of reservations at my restaurant to the number of reviews made from a reservation	I can see how many customers that booked a reservation

				also decided to give a review.
US5	Analytical	Restaurant	To see the number of times my restaurant was booked on a certain day	I can analyze which days are most popular and analyze the growth of my restaurant.
US6	Simple/Analytical	Restaurant	See the list of returning customers (people that have made a reservation at least twice at my restaurant)	I can analyze how well my restaurant is doing in attracting the same customers back and also provide special discounts to them
US7	Complex/ Analytical	Restaurant	Compare the amount of public to private reservations made at my restaurant	I can analyze why people choose to make reservations at my restaurant.
US8	Complex	Customer	I would like to see the details of my reservation (date, time, party size, restaurant name, restaurant address)	I can check when and where I have to go to make it to the reservation.
US9	Complex	Potential guest	See the items on the menu of the restaurant (food name and food price)	I can decide if I want to make a reservation or not based on if there's something I would like to try from the

				restaurant
US10 (new)	Complex	Customer	add other people into your reservation	Other people in my reservation can receive the reservation details

Relational Model (primary key in bold, foreign key in underline)

Reservation (**reservation_id**, date, time, party_size, user_id, restaurant_id)

Public_Reservation **(reservation_id**)
Private_Reservation (**reservation_id,** event_type, details)

Restaurant (**restaurant_id**, restaurant_name, rating, address, restaurant_phone_number)

Menu (**menu_id**, restaurant_id)
Food_item (**food_id**, food_name, food_price, menu_id)

Tags (**tag_id**, tag_name)

Reviews (**review_id**, overall_rating, food_rating, service_rating, ambience_rating, description, dined_date, user_id, restaurant_id)

Users (**user_id**)
Guests (**user_id**)
Customers (**user_id**, customer_name, customer_email, customer_phone_number)

Categorized_by (**r.restaurant_id, t.tag_id**) d

Functional Dependencies

Reservation:

- Reservation_id → date, time, party_size, user_id, restaurant_id (assuming that each reservation is unique)

- Date, time, restuarant_id → user_id, party_size, reservation_id (assuming that there is only one reservation slot available for a specific time on any day for each restaurant, so each reservation at the restaurant is unique)

Public_Reservation: No functional dependencies
Private_Reservation:

- Reservation_id → event_type, details (assuming, details are provided by the customers and both event_type and details are associated with that specific reservation represented by the reservation_id)

Restaurant:

- Restaurant_id → restaurant_name, rating, address, restaurant_phone_number
- Address → restaurant_name, restaurant_phone_number, rating, restaurant_id (assuming addresses are unique to restaurants)

Menu:

- Menu_id → restaurant_id (assuming menus are made specifically for the restaurant so having the menu_id would mean we know what restaurant it is)

Food_item:

- Food_id → food_name, food_price, menu_id (assuming a food item can only be unique to one menu)

Tags:

- Tag_id → tag_name (assuming each tag name has a unique tag id associated with it)

Reviews:

- Review_id → overall_rating, food_rating, service_rating, ambience_rating, description, dined_date, user_id, restaurant_id
- User_id, restaurant_id → overall_rating, food_rating, service_rating, ambience_rating, description, dined_date, review_id (assuming a customer would only write one review for a restaurant)

Users: No functional dependencies
Guests: No functional dependencies
Customers:

- User_id → customer_name, customer_email, customer_phone_number
- Customer_phone_number → customer_name, customer_email, user_id (assuming a phone number is unique to the customer)

- Customer_email → customer_name, customer_phone_number, user_id (assuming a email is unique to the customer)

Categorized_by: No functional dependencies

Normalization:

Reservation: This relation is already in BCNF because all the functional dependencies are good. The primary key, reservation_id, is a unique identifier for each reservation made through OpenTable and other attributes will naturally depend on it. Assuming that only one reservation can be made for a specific date, time and restaurant, those attributes will lead to the user_id, party_size and reservation_id.

Private Reservations: they are only identified by the reservation_id, which will lead to the type of event and other details. Type of event cannot lead to details because the details will be uniquely provided by the customers and similarly, details cannot lead to event type.

Restaurant: This relation is already in BCNF because all the functional dependencies are good. Primary key, restaurant_id, would be the unique identifier of each restaurant registered with OpenTable and therefore lead to all other attributes. Assuming the address of a restaurant is unique to that restaurant, it can also lead to all other attributes.

Menu: This relation is already in BCNF because all functional dependencies are good. Assuming that a menu is unique to the restaurant, menu_id would lead to restaurant_id.

Food_id: This relation is already in BCNF because all functional dependencies are good. The food_id is a unique identifier for a specific item on a specific menu. Food_name wouldn't be able to indicate the price of that item because items can be priced differently on different menus.

Tags: This relation is already in BCNF because all functional dependencies are good. The tag_id is a unique identifier for each tag name.

Reviews: This relation is already in BCNF because all functional dependencies are good. The review_id is a unique identifier for each review posted on OpenTable and would lead to all other attributes in the table. Assuming that a customer can only post one review for a specific restaurant, user_id and restaurant_id would also lead to all the other attributes in the table.

Customers: This relation is already in BCNF because all functional dependencies are good. The user_id is a unique identifier for any user that has an account with OpenTable and

would lead to all other attributes in the table. Assuming that email and phone number are also unique identifiers for each customer, they would also lead to the other attributes.