Stage2

relational schema

--Entities
Table-Location(LocationID:int[PK], Address:varchar(255), Street:varchar(255), City:varchar(255), State:varchar(30), ZipCode:int)

Table-Schedule(ScheduleID:int[PK], DayOfWeek:varchar(10), OpenTime:time, OpeningDuration:int)

Table-Restaurant(Restaurant:int[PK], Name:varchar(255), PhoneNumber:int, AveragePrice:real,Cuisine:varchar(255))

Table-User(UserID:int[PK], UserName:varchar(40), Password:varchar(30), Email:varchar(50))

Table-Reviewer(ReviewerID:int[PK], Name:varchar(255))

Table-Review(ReviewID:int[PK], Rating:int, Comment:text, ReviewDate:date)

– Weak Entity
Table-OfferFood(FoodID:int[PK], RestaurantID:int[PK][FK to Restaurant.RestaurantID], Price:real)

– Relationships

Table-Locate(RestaurantID:int[PK][FK to Restaurant.RestaurantID], LocationID:int[FK to Location.LocationID])

Table-BusinessHours[RestaurantID:int[PK][FK to Restaurant.RestaurantID], ScheduleID:int[FK to Schedule.ScheduleID]]

Table-Likes(UserID:int[PK][FK to User.UserID], RestaurantID:int[FK to Restaurant.RestaurantID] LikeTime:date, Notes:text)

Table-Writes(ReviewID:int[PK][FK to Reviewer.ReviewerID], ReviewID:int[FK to Review.ReviewID])

Table-Contain(ReviewID:int[PK][FK to Review.ReviewID], RestaurantID:int[FK to Restaurant.RestaurantID]

Assumptions of the ER/UML diagram (-0.5% for each missing description for each entity)

Each restaurant is uniquely identified by its RestaurantID, and has the attributes Name, Phone Number, averagePrice, averageRating, Cuisine. We chose RestaurantID as our primary key because we are identifying each restaurant by their RestaurantID. For example, although MacDonald's could have different locations in a city with the same name for each restaurant, each MacDonald's restaurant will have a different RestaurantID.

Each Location is uniquely identified by its LocateID and has attributes Address, Street, City, State, Zip Code. We choose LocateID to be the primary key. All of the other attributes are probably not distinct. So we choose to create a LocateID.

Each Schedule is uniquely identified by its ScheduleID and has attributes ScheduleID, DayOfWeek, OpenTime, OpeningDuration. Our primary key here is ScheduleID, other attributes may repeat. The extra assumption is that the restaurants will not take a break once they are open. For example, restaurants will not take a break between lunch and dinner.

Each Food is uniquely identified by its FoodID and the RestaurantID. It has attributes FoodName and Price. We chose the primary key as FoodID and RestaurantID because FoodName and Price may be repeated across different locations. Although restaurants could have the same food, the prices might also vary.

Each Review is uniquely identified by its RatingID and RestaurantID, and has attributes Rating and Comment. A reviewer is only allowed to review multiple restaurants but they are only allowed to review once per restaurant. This is to avoid reviewers to post similar reviews repeatedly that might skew the average rating of a restaurant.

Each Reviewer is uniquely identified by its ReviewerID, and has an attribute Name. A reviewer is those who write reviews on the restaurants. We chose ReviewerID as the primary key as different reviewers can have the same name. So we used ReviewerID to identify them.

Each User is uniquely identified by its UserID, and has attributes UserName, Password and Email. Users and reviewers are different such that reviewers are those who wrote reviews (based on data we scrape online) but users are those who log in to our website and view the restaurant information and reviews. They can 'like' the restaurant and write notes about it.

A description of each relationship and its cardinality (-1% for each missing description)

A Reviewer writes at least one Review, but a Review is written by exactly one Reviewer.

A Restaurant can contain zero to multiple Reviews, while a review used to rate exactly one restaurant

A restaurant is located in exactly one place, and one location could have exactly one restaurant.

A restaurant could offer zero to multiple food and one kind of food is offered by exactly one restaurant. Although technically, it is impossible that a restaurant has zero food, we might not be able to obtain the menu items for all restaurants. Therefore, we include the possibility of having zero food in our database.

A restaurant could be liked by any number of users and one user can choose to like any number of restaurants. Every time a user likes a restaurant, we want to store the time, specifically the attribute “LikeTime” and store the note the user added, specifically the attribute “note”

As a restaurant could be open at different times on different dates, we have a schedule entity as well. As there are only 7 days a week, each restaurant can have zero to seven days of schedules. Zero is for cases where we are unable to obtain their opening hours. There could be days where the restaurants are not open so we are putting 0 to 7 to account for that possibility. Each schedule will only belong to one restaurant.