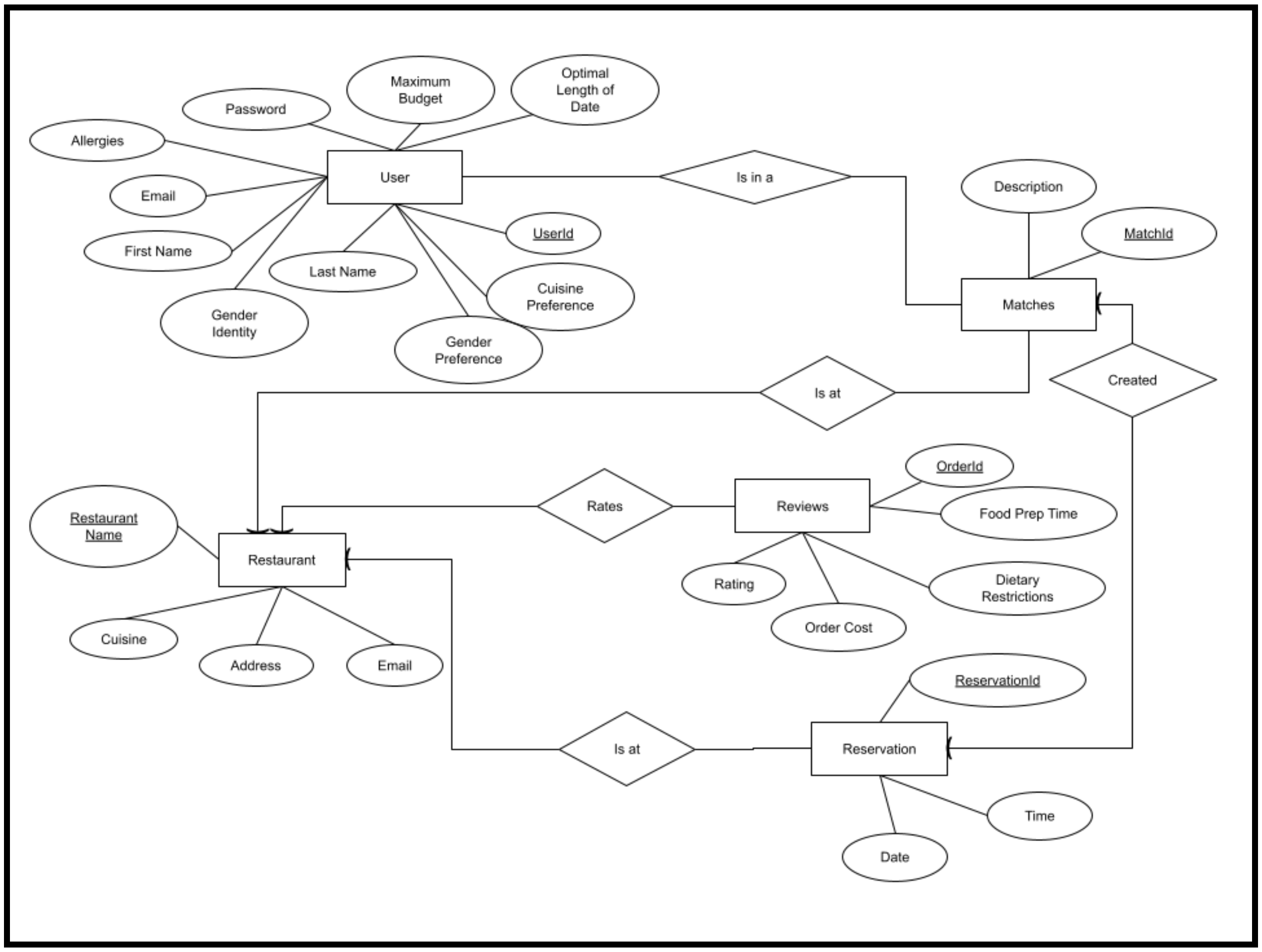
Table for Two Stage 2
KEY: **Bold** denotes Primary Keys, Underlined denotes Foreign Keys
Entities:

1. User
    - Attributes: **UserId**, Email, Password, First Name, Last Name, Gender Identity, Gender Preference, Cuisine Preference, Maximum Budget ($, $$, $$$), Optimal Length of Date, Allergies
    - Description: Holds information about the profiles within the application. The attributes reflect the user's personal information and preferences. The primary key is the UserId, which is unique and distinguishes each user from one another.
    - Assumptions: Each UserId is associated with a unique email. All users in the application are stored in this table. Maximum Budget is reflected as $, $$, or $$$ to represent the price (a custom range of prices is set to determine which category the budget falls into).
2. Restaurant
    - Attributes: **RestaurantName,** Cuisine, AverageRating, Address, RestaurantEmail
    - Description: Holds relevant information about the Restaurants in New York City. The primary key is the Restaurant Name, which is unique. The Average Rating is a foreign key to the Reviews.Rating.
    - Assumptions: The Restaurant Names are unique, and only one Address and one Restaurant Email are listed for each restaurant name. All restaurants with unique names in New York City should be stored in this table.
3. Matches
    - Attributes: **MatchId,** UserId A, UserId B, Description
    - Description: Stores the matches between two Users. The primary key is the MatchId, as each pairing of UserId's is unique. The Match information is meant for the users to view their pairing. The UserId A and UserId B attributes are foreign keys to User.UserId.
    - Assumptions: UserId A and UserId B are different, unique UserIds. All matches should be stored in this table.
4. Reservation
    - Attributes: **ReservationId,** RestaurantName, MatchId, Time, Date
    - Description: Holds information about the reservation. Intended to inform the restaurants for tracking purposes. The ReservationId is the primary key since it is unique for each reservation placed. The RestaurantName is a foreign key to Restaurant.RestaurantName and MatchId is a foreign key to Matches.MatchId.
    - Assumptions: A reservation will be placed to an existing restaurant, and the MatchId will correspond to an existing match. Can be deleted; the corresponding match will be deleted as well. All reservations should be stored in this table.
5. Reviews

- Attributes: **OrderId,** <u>RestaurantName</u>, Order Cost, Rating, Food Prep Time, Dietary Restrictions
- Description: Holds information about the experience at each restaurant. Allows users to input feedback on the restaurants they dine at.
- Assumptions: A review will be made at an existing Restaurant, and the rating is on a scale from 1 to 5. The RestaurantName is a foreign key to Restaurant.RestaurantName. All reviews should be stored in this table.

Relationships (i.e., 1-1, 1-many, and many-many):

1. Rates (Restaurant → Reviews)
    - One to Many
    - Description: A review rates a particular restaurant
    - Assumption: There are multiple (or 0) reviews for each restaurant.
2. Is in a (User → Matches)
    - Many to Many
    - Description: A user is paired up with another user and a match is formed.
    - Assumption: A single user can be in multiple (more than 0) matches.
3. Created (Matches → Reservations)
    - One to One
    - Description: When a match is made, the match also creates a reservation at the restaurant of choice.
    - Assumption: Every match creates a single reservation, and if they delete the reservation, they are no longer matched with each other.
4. Is At (Reservation → Restaurant)
    - Many to One (many reservations for each restaurant)
    - Description: A reservation is made at a chosen restaurant
    - Assumption: Each reservation is made at one restaurant and a single restaurant can have many reservations.



Normal Form:
Functional Dependencies:

User
UserId → Email, Password, FirstName, LastName, GenderIdentity, GenderPreference, CuisinePreference, MaximumBudget, OptimalLengthOfDate, Allergies
Email → UserId

Restaurant
RestaurantName → RestaurantEmail, Address, Cuisine, AverageRating
Address → RestaurantName
RestaurantEmail → RestaurantName

Reservation
ReservationId → RestaurantName, MatchId, Time, Date
MatchId → ReservationId

<u>Matches</u>
MatchId → UserId A, UserId B, Description
UserId A, UserId B → MatchId

<u>Reviews</u>
OrderId → FoodPrepTime, Rating, OrderCost, DietaryRestrictions

Our tables are already in 3NF because for every functional dependency, the left hand side is a primary key or the right hand side is a primary key. We chose to do 3NF because we wanted to preserve all the functional dependencies, which BCNF does not do. We plan on mitigating redundancies when querying from our datasets.

<u>Relational Schema:</u>
User(UserId:INT [PK], Email:VARCHAR(255), Password:VARCHAR(255), FirstName:VARCHAR(255), LastName:VARCHAR(255), GenderIdentity:VARCHAR(255), GenderPreference:VARCHAR(255), CuisinePreference:VARCHAR(255), MaximumBudget:VARCHAR(255), OptimalLengthOfDate:INT, Allergies:VARCHAR(500))

Restaurant(RestaurantName:VARCHAR(255) [PK], Cuisine:VARCHAR(255), AverageRating:REAL [FK to Reviews.Rating], Address:VARCHAR(500), Email:VARCHAR(255))

Matches(MatchId:INT [PK], UserIdA:INT [FK to User.UserId], UserIdB: INT [FK to User.UserId], Description: VARCHAR(255))

Reservation(ReservationId:INT [PK], RestaurantName: VARCHAR(255) [FK to Restaurant.RestaurantName], MatchId:INT [FK to Matches.MatchId], Time:VARCHAR(255), Date:VARCHAR(255))

Reviews(OrderId:INT [PK], RestaurantName:VARCHAR(255) [FK to Matches.MatchId], OrderCost:REAL, Rating:INT, FoodPrepTime:INT, DietaryRestrictions:VARCHAR(255))

<u>Stage 1 Fixes:</u>
We needed to explain the functionalities of our database, so here are the following operations that are allowed on our database:

- Users
    - Create a user account
    - Read data from user account

    - Update data from user account
    - Cannot delete users
        - Make it null instead of deleting
        - Delete any reservations and matches that the user was a part of
- Restaurants
    - Create a restaurant entry
    - Read data from restaurant entry
    - Update data from restaurant entry
    - Cannot delete restaurant entry
- Reservation
    - Create reservations
    - Update reservations
    - Read reservations
    - Delete reservations
        - When a reservation is deleted, the corresponding match is deleted
- Reviews
    - Create reviews
    - Read reviews
    - Cannot update reviews
    - Cannot delete reviews
- Matches
    - Able to create match
    - Cannot update a match
    - Able to read a match
    - Delete a match
        - When a match is deleted, the corresponding reservation is also deleted