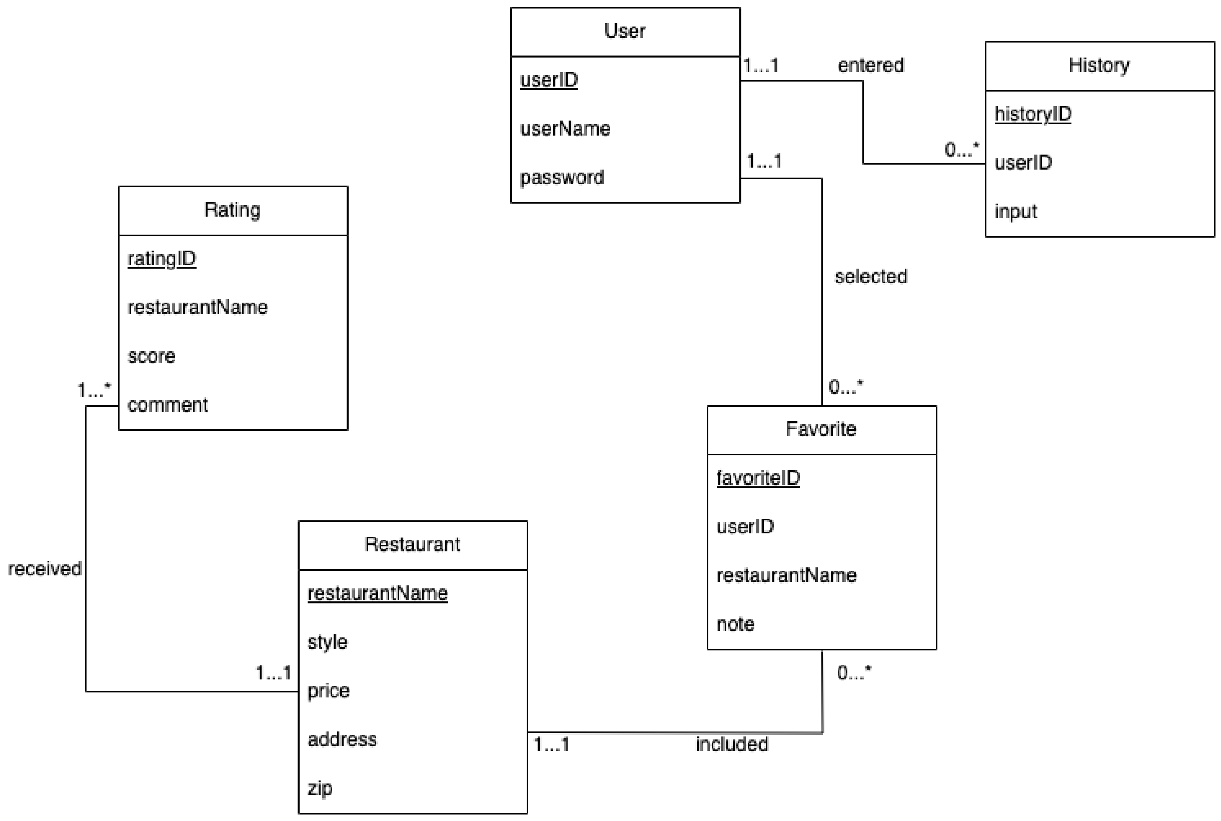
UML Diagram:



Assumptions/Description for Each Entity:

1. Users:
   a. Attributes: userID, userName, password
   b. Primary Key: userID
   c. Description: Holds information related to each user. Each user could log into his/her account and do searching operations.
2. Restaurants:
   a. Attributes: restaurantName, style, price, address, zip
   b. Primary Key: restaurantName
   c. Description: Holds information related to each restaurant. All the information is retrieved from the "top 240 restaurants recommended in Los Angeles 2.csv".
3. Rating:
   a. Attributes: ratingID, restaurantName, score, comment
   b. Primary Key: ratingID

   c. Foreign Keys: restaurantName references Restaurants
   d. Description: Holds the rating and comments given by users to restaurants.
4. History:
   a. Attributes: historyID, userID, input
   b. Primary Key: historyID
   c. Foreign Keys: user_id references Users, restaurant_id references Restaurants
   d. Description: Keeps a log of user visits to restaurants, it would store the input which is used to search by the user in the entity.
5. Favorite:
   a. Attributes: favoriteID, userID, restaurantName, note
   b. Primary Key: favoriteID
   c. Foreign Keys: userID references Users, restaurantName references Restaurants
   d. Description: Holds information related to the user's favorite restaurants.

Cardinality/Description for Each Relationship:

User entered History: 1…1 - 0…* (1 - many)
Users can enter as many Histories as they want. A user might have many histories or no history at all.

User selected Favorite: 1…1 - 0…* (1 - many)
Users can select as many Favorites as they want. A user might have many favorites or no favorites at all.

Favorite included Restaurant: 0…* - 1…1 (many - 1)
Users might have as many favorites as they want. Each favorite will match with exactly one restaurant. However, there could be a restaurant that is not selected as a favorite for any user.

Restaurant received Rating: 1…1 - 1…* (1 - many)
A restaurant might receive many ratings. Different ratings could be given to the same restaurant.

Relational Schema:

Entities:

Users (
    userID: LONG [PK],
    userName: VARCHAR(255),
    password: LONG
);

Restaurants(
    restaurantName: VARCHAR(255) [PK],
    style: VARCHAR(255),
    price: INT,
    address: VARCHAR(255),
    zip: INT
);

Rating(
    ratingID: LONG [PK],
    restaurantName: VARCHAR(255) [FK to Restaurants.restaurantName],
    score: INT,
    comment: VARCHAR(255)
);

History(
    historyID: LONG [PK],
    userID: LONG [FK to Users.userID],
    input: VARCHAR(255)
);

Favorite(
    favoriteID: LONG [PK],
    userID: LONG [FK to Users.userID],
    restaurantName: VARCHAR(255) [FK to Restaurants.restaurantName],
    note: VARCHAR(255)
);

Relationships:

Entered(
    userID: LONG [FK to Users.userID]
);

Selected(
    userID: LONG [FK to Users.userID]
);

Included(

```
        restaurantName: VARCHAR(255) [FK to Restaurants.restaurantName]
);
```

Received(

```
        restaurantName: VARCHAR(255) [FK to Restaurants.restaurantName]
);
```

Normalization to BCNF:

1. Users:
    a. Users (userID, userName, password)
    b. FD = {userID -> userName, userID -> password}
    c. Analysis: Since userID is primary key as well as super key so the relation, Users, is in BCNF.
2. Restaurants:
    a. Restaurants (restautantName, style, price, address, zip)
    b. FD = {restautantName -> style, restautantName -> price, restautantName -> address, restautantName -> zip}
    c. Analysis: Since restautantName is primary key as well as super key so the relation, Restaurants, is in BCNF.
3. Rating:
    a. Rating (ratingID, restaurantName, score, comment)
    b. FD = {ratingID -> restaurantName, ratingID -> score, ratingID -> comment}
    c. Analysis: Since ratingID is the primary key as well as super key so the relation, Rating, is in BCNF.

4. History:
    a. History (historyID, userID, input)
    b. FD = {historyID -> userID, historyID -> input}
    c. Analysis: Since historyID is primary key as well as super key so the relation, History, is in BCNF.

5. Favorite
    a. Favorite (favoriteID, userID, restaurantName, note)
    b. FD = {favoriteID -> userID, favoriteID -> restaurantName, favoriteID -> note}
    c. Analysis: Since favoriteID is primary key as well as super key so the relation, Favorite, is in BCNF.

Reason:

In our case, the choice to use BCNF over 3NF is largely attributed to its ability to maintain a higher level of data integrity and to minimize redundancy effectively. For database designs where accuracy and consistency of data are paramount, like in applications involving recommendations, adhering to BCNF could be crucial.

Fixes to Stage 1:

In stage 1, we submit the link to our Google document instead of submitting the GitHub URL with an appropriate tag like stage 0. We are going to fix that by doing exactly the same submitting step as in Stage 0.