FOOD DELIVERY APP

Approach

Break Down the problem into classes which are required for the food delivery app-

Class List-
1)User Class
2)Rider Class
3)Location Class
4)Restaurant Class
5)Menu Clas
6)Order Clas

Bonuses Classes-

1)Coupon Class
2)Rating Class

Assumptions taken

a) For ease of calculating distance assuming the **user , restaurant and rider** are in a 2D-Plane **(x,y coordinate planes)**

b) We assume a **default speed** for all the **Riders** for ease of calculations.

c) For suggesting restaurants always assuming a delivery rider is nearby.

d) In order to suggest a restaurant based on cuisine and time given we sort them based on **user and restaurant distance + preparation time taken by the restaurant.**

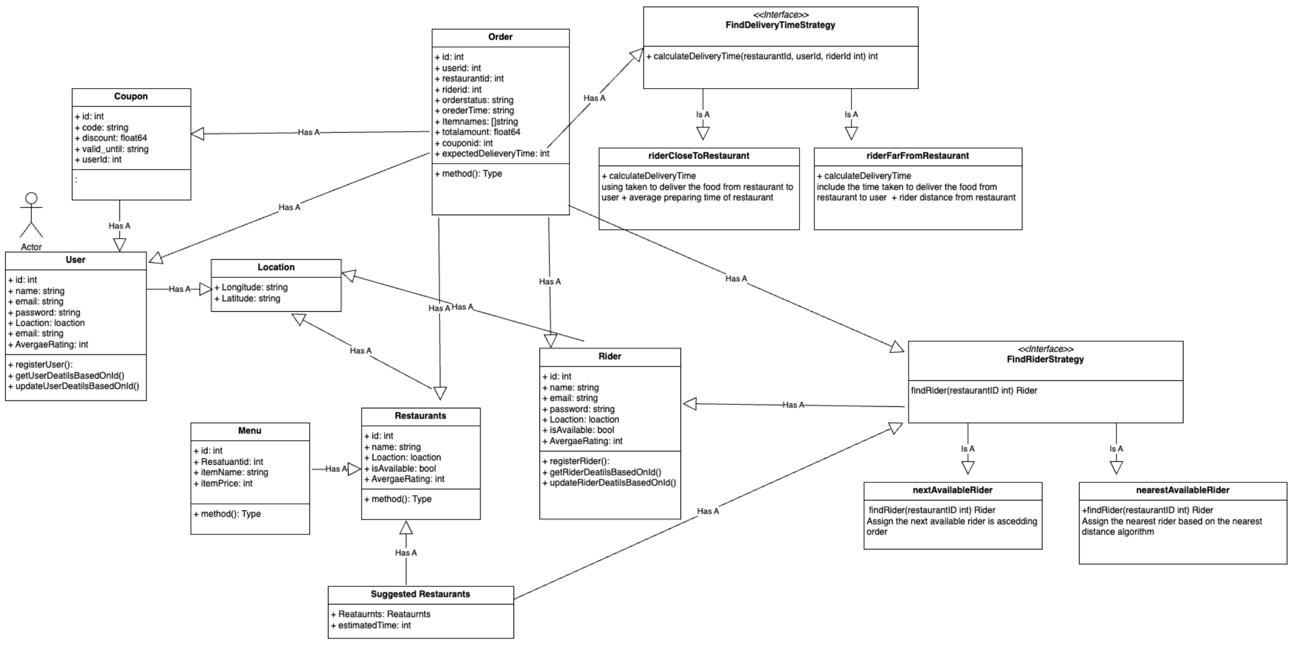
e) When order is placed for finding the rider who will deliver we can either use

-NextAvailabe Algorithm : Assign the next available rider

-NearestAvailabe Algorithm : Assign the next rider based on the nearest distance algorithm

f) **IF** coupon code is available for the user then subtract **discount%** amount from the total amount of the order

Class Diagram



API Documentation

servers: - url: http://localhost:8080

description: Local development server

1. Register a user

```
curl -X POST http://localhost:8080/register/user \
-H "Content-Type: application/json" \
-d '{
  "name": "John Doe",
  "email": "john.doe@example.com",
  "password": "securepassword",
  "location": {
    "lat": 12,
    "lon": 56
  },
  "avg_rating": 0.0
}'
```

```
curl -X POST http://localhost:8080/register/user \
-H "Content-Type: application/json" \
-d '{
  "name": "Ravi Ben",
  "email": "ravi.ben@example.com",
  "password": "securepassword",
  "location": {
    "lat": 20,
    "lon": 36
  },
  "avg_rating": 0.0
}'
```

2. Register a rider

```
curl -X POST http://localhost:8080/register/rider \
-H "Content-Type: application/json" \
-d '{
  "name": "Gpeer",
  "email": "gpeer.123@example.com",
  "password": "securepassword",
  "location": {
    "lat": 20,
    "lon": 34
  },
  "avg_rating": 0.0
}'
```

```
curl -X POST http://localhost:8080/register/rider \
-H "Content-Type: application/json" \
-d '{
  "name": "Prayag",
  "email": "pgrrid.123@example.com",
  "password": "securepassword",
  "location": {
    "lat": 34,
    "lon": 14
  },
  "avg_rating": 0.0
}'
```

3. Register a restaurant

```
curl -X POST http://localhost:8080/register/restaurant \
-H "Content-Type: application/json" \
-d '{
  "name": "Gods Food Corner",
  "location": {
    "lat": 20,
    "lon": 34
  },
  "avg_preparing_time": 15
}'
```

```
curl -X POST http://localhost:8080/register/restaurant \
-H "Content-Type: application/json" \
-d '{
  "name": "Chopsticks",
  "location": {
    "lat": 30,
    "lon": 20
  },
  "avg_preparing_time": 20
}'
```

4. Register menus to restaurants

```
curl -X POST http://localhost:8080/register/restaurant/menu \
-H "Content-Type: application/json" \
-d '{
 "restaurant_id": 1,
 "item_name": "Panner",
 "item_price": 15
}'
```

```
curl -X POST http://localhost:8080/register/restaurant/menu \
-H "Content-Type: application/json" \
-d '{
 "restaurant_id": 1,
 "item_name": "Pizza",
 "item_price": 45
}'

curl -X POST http://localhost:8080/register/restaurant/menu \
-H "Content-Type: application/json" \
-d '{
 "restaurant_id": 2,
 "item_name": "Panner",
 "item_price": 35
}'

curl -X POST http://localhost:8080/register/restaurant/menu \
-H "Content-Type: application/json" \
-d '{
 "restaurant_id": 2,
 "item_name": "Pizza",
 "item_price": 25
}'
```

5. Suggest restaurants to a user

```
curl -X GET
"http://localhost:8080/restaurants/suggest?itemName=Panner&maxTimeEx
pected=300&userId=1"
```

6. Get the menu for the restaurant

```
curl -X GET "http://localhost:8080/restaurant/menu?restaurantId=1"
```

7. Place an order

```
curl -X POST http://localhost:8080/order \
-H "Content-Type: application/json" \
-d '{
    "user_id": 1,
    "restaurant_id": 1,
    "order_time": "2022-01-01T12:00:00Z",
    "total_amount": 20.99,
     "item_names":["Pizza", "Burger", "Fries"]
}'


curl -X POST "http://localhost:8080/order?coupon_code=d50" \
-H "Content-Type: application/json" \
-d '{
    "user_id": 1,
    "restaurant_id": 1,
    "order_time": "2022-01-01T12:00:00Z",
    "total_amount": 20.99,
     "item_names":["Pizza", "Burger", "Fries"]
}'
```

8. Update rider location

```
curl -X PATCH http://localhost:8080/rider/location -H "Content-Type:
application/json" -d '{"id":1,"location": {
    "lat": 20,
    "lon": 34
  }
}'
```

9. Show history of orders placed by the user

```
curl -X GET "http://localhost:8080/user/orders?id=1"
```

10. Show history of orders completed by the rider

```
curl -X GET "http://localhost:8080/rider/orders?id=1"
```

Bonus Section

11. Generate Coupon for a User

```
curl -X POST http://localhost:8080/user/coupon \
-H "Content-Type: application/json" \
-d '{
     "code": "d10",
     "discount": 10.0,
     "valid_until": "2022-12-31T23:59:59Z",
     "user_id": 1
}'

curl -X POST http://localhost:8080/user/coupon \
-H "Content-Type: application/json" \
-d '{
     "code": "d50",
     "discount": 50.0,
     "valid_until": "2022-12-31T23:59:59Z",
     "user_id": 1
}'
```

12. Submit Ratings for a User and Rider

```
curl -X POST http://localhost:8080/submit-rating \
-H "Content-Type: application/json" \
-d '{
    "user_id": 1,
    "rider_id": 1,
    "rating": 4.5,
    "order_id": 1
}'
```

13. Get Ratings for a User

```
curl -X GET "http://localhost:8080/user/ratings?id=1"
```

14. Get Ratings for a Rider

```
curl -X GET "http://localhost:8080/rider/ratings?id=1"
```

Table Structure

```
CREATE TABLE Location (
   lat INT,
   lon INT,
   PRIMARY KEY (lat, lon)
);

CREATE TABLE Users (
   user_id INT PRIMARY KEY AUTO_INCREMENT,
   name VARCHAR(100) NOT NULL,
   email VARCHAR(100) UNIQUE NOT NULL,
   password VARCHAR(100) NOT NULL,
   lat INT,
   lon INT,
   avg_rating FLOAT,
   FOREIGN KEY (lat, lon) REFERENCES Location(lat, lon)
);

CREATE TABLE Riders (
   rider_id INT PRIMARY KEY AUTO_INCREMENT,
   name VARCHAR(100) NOT NULL,
   email VARCHAR(100) UNIQUE NOT NULL,
   password VARCHAR(100) NOT NULL,
   lat INT,
   lon INT,
   is_available BOOLEAN,
   avg_rating FLOAT,
   FOREIGN KEY (lat, lon) REFERENCES Location(lat, lon)
);

CREATE TABLE Restaurants (
   restaurant_id INT PRIMARY KEY AUTO_INCREMENT,
   name VARCHAR(100) NOT NULL,
   lat INT,
   lon INT,
   avg_preparing_time INT,
   FOREIGN KEY (lat, lon) REFERENCES Location(lat, lon)
);

CREATE TABLE Menus (
   menu_id INT PRIMARY KEY AUTO_INCREMENT,
   restaurant_id INT NOT NULL,
   item_name VARCHAR(100) NOT NULL,
   item_price DECIMAL(10,2) NOT NULL,
   FOREIGN KEY (restaurant_id) REFERENCES Restaurants(restaurant_id)
);

CREATE TABLE Coupons (
   coupon_id INT PRIMARY KEY AUTO_INCREMENT,
   code VARCHAR(50) UNIQUE NOT NULL,
   discount DECIMAL(5,2) NOT NULL,
   valid_until DATE NOT NULL,
   user_id INT NOT NULL,
   FOREIGN KEY (user_id) REFERENCES Users(user_id)
);

CREATE TABLE Orders (
   order_id INT PRIMARY KEY AUTO_INCREMENT,
   user_id INT NOT NULL,
   restaurant_id INT NOT NULL,
   rider_id INT NOT NULL,
   order_status VARCHAR(20) NOT NULL,
   order_time TIMESTAMP NOT NULL,
   total_amount DECIMAL(10,2) NOT NULL,
   item_names TEXT NOT NULL,
   expected_delivery_time INT NOT NULL,
   coupon_id INT,
   FOREIGN KEY (user_id) REFERENCES Users(user_id),
   FOREIGN KEY (restaurant_id) REFERENCES Restaurants(restaurant_id),
   FOREIGN KEY (rider_id) REFERENCES Riders(rider_id),
   FOREIGN KEY (coupon_id) REFERENCES Coupons(coupon_id)
);

CREATE TABLE Ratings (
   rating_id INT PRIMARY KEY AUTO_INCREMENT,
   user_id INT,
   rider_id INT,
   rating FLOAT NOT NULL,
   order_id INT NOT NULL,
   FOREIGN KEY (user_id) REFERENCES Users(user_id),
   FOREIGN KEY (rider_id) REFERENCES Riders(rider_id),
   FOREIGN KEY (order_id) REFERENCES Orders(order_id)
);

CREATE TABLE SuggestedRestaurants (
   restaurant_id INT NOT NULL,
   estimated_time_min INT NOT NULL,
   FOREIGN KEY (restaurant_id) REFERENCES Restaurants(restaurant_id)
);
```