Oye Rickshaw Rating System

Task

1. The passenger should be able to rate a given ride.
2. The driver should be able to see aggregated rating of his all rides
3. The driver should be able to rate the passenger after ride
4. The passenger should be able to see his aggregated rating based on all the rides he has taken.

Tech Stack

I have used NodeJs, ExpressJs and MongoDb for the given task

DataBase Details

I have used the online hosting of mongoDB database provided by mlab. I have three models:

1. **Driver** : This the Driver Object representing a Driver. It contains the following fields.
    1. **name** : Name of the Driver
    2. **age** : Age of the Driver
    3. **contact** : Contact of the Driver
    4. **licenseNo** : License Number of the Driver
    5. **avgRating** : Aggregated rating of the Driver based on all the rides
    6. **rides[ ]** : Rides Array representing all the rides made by driver

2. **Passenger** : This the Passenger Object representing a Passenger. It contains the following fields.
    1. **name** : Name of the Passenger
    2. **age** : Age of the Passenger
    3. **contact** : Contact of the Passenger
    4. **avgRating** : Aggregated rating of the Passenger based on all the rides
    5. **rides[ ]** : Rides Array representing all the rides made by Passenger

3. **Ride :** This the Ride Object representing a Ride. It contains the following fields.
    1. **startLocation :** Start Location of the Ride
    2. **endLocation:** End Location of the Ride
    3. **passengerId:** Passenger Id (Mongo Object Id) who booked the Ride

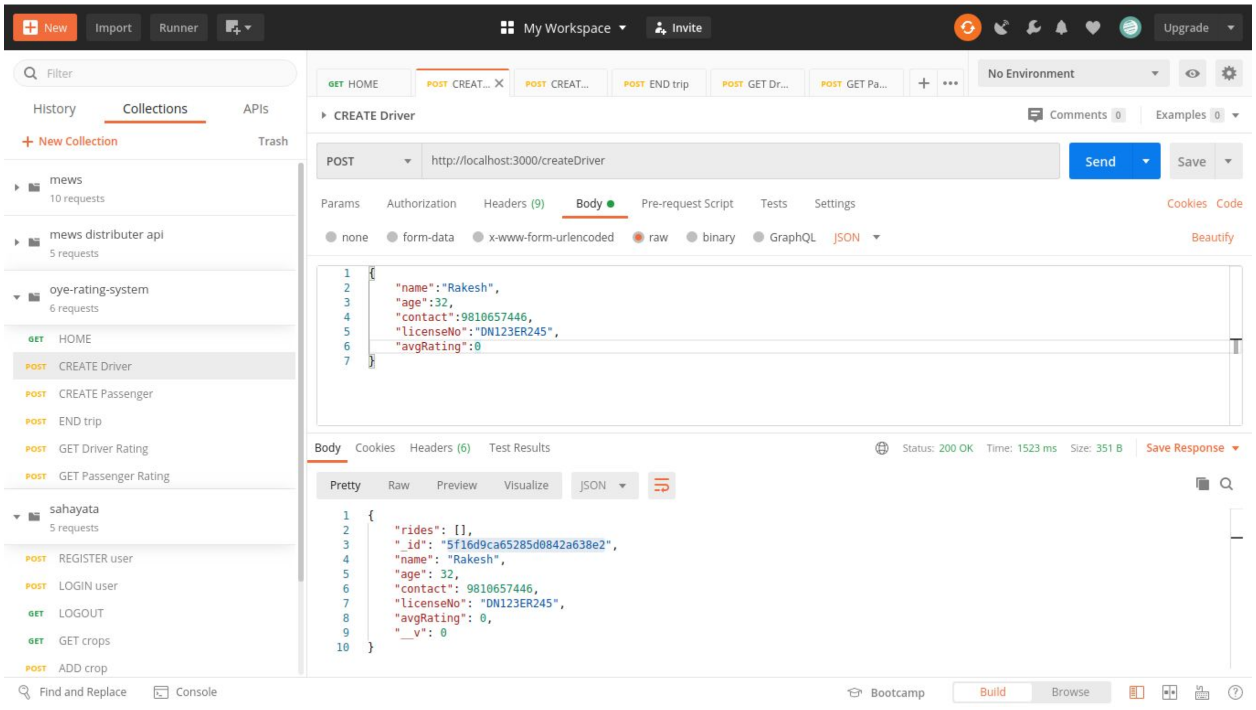
4. **driverId:** Driver Id (Mongo Object Id) who accepted the Ride
5. **driverRating:** Rating given by the Driver to the Passenger
6. **passengerRating:** Rating given by the Passenger to the Driver

Assumptions to my solution

1. We have a database of drivers
2. We have a database of passengers

Approach to my solution

1. We can create a new driver using the “/createDriver” endpoint.



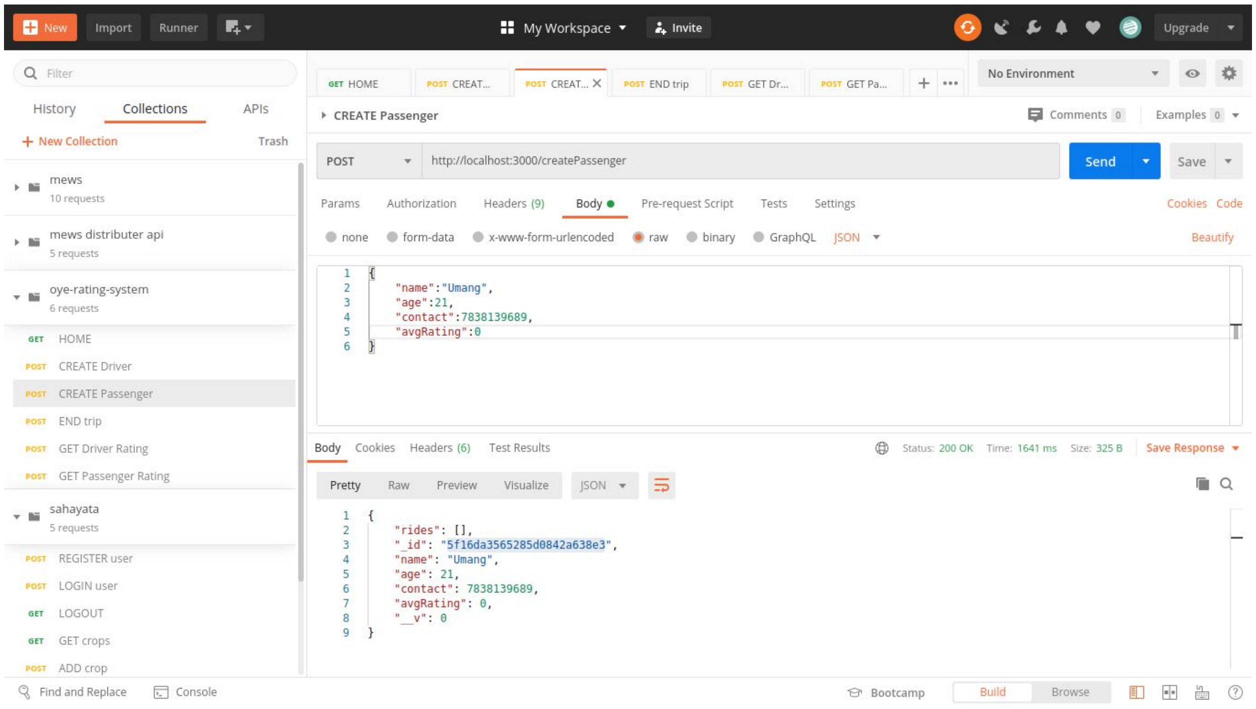
POST http://localhost:3000/createDriver

```
{
    "name":"Rakesh",
    "age":32,
    "contact":9810657446,
    "licenseNo":"DN123ER245",
    "avgRating":0
}
```

Status: 200 OK Time: 1523 ms Size: 351 B

```
{
    "rides": [],
    "_id": "5f16d9ca65285d0842a638e2",
    "name": "Rakesh",
    "age": 32,
    "contact": 9810657446,
    "licenseNo": "DN123ER245",
    "avgRating": 0,
    "__v": 0
}
```

2. We can create a new passenger using the “/createPassenger” endpoint.

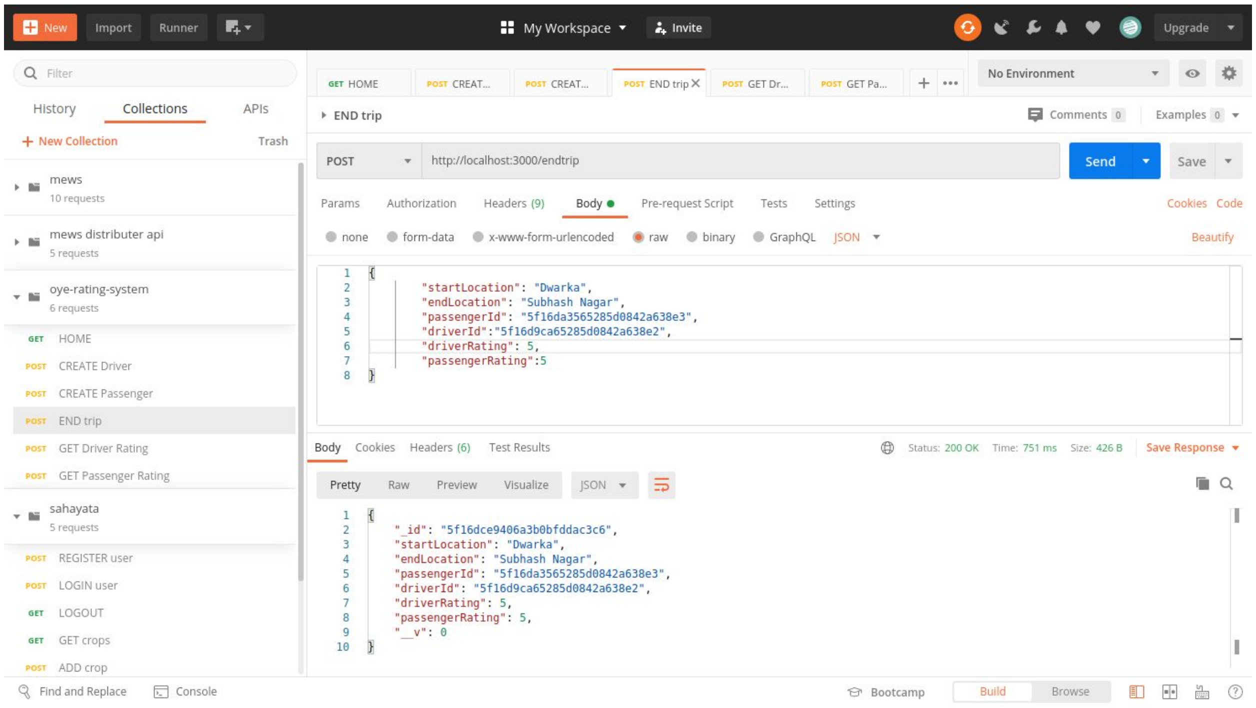


```
POST http://localhost:3000/createPassenger
```

```
{
    "name":"Umang",
    "age":21,
    "contact":7838139689,
    "avgRating":0
}
```

```
{
    "rides": [],
    "_id": "5f16da3565285d0842a638e3",
    "name": "Umang",
    "age": 21,
    "contact": 7838139689,
    "avgRating": 0,
    "__v": 0
}
```

3. Now initially the **avgRating** of both driver and passenger is 0 and rides array is empty i.e. no rides till now.
4. Suppose when the trip ends we were given information like the **startLocation, endLocation, passengerId, driverId, driverRating, passengerRating.**

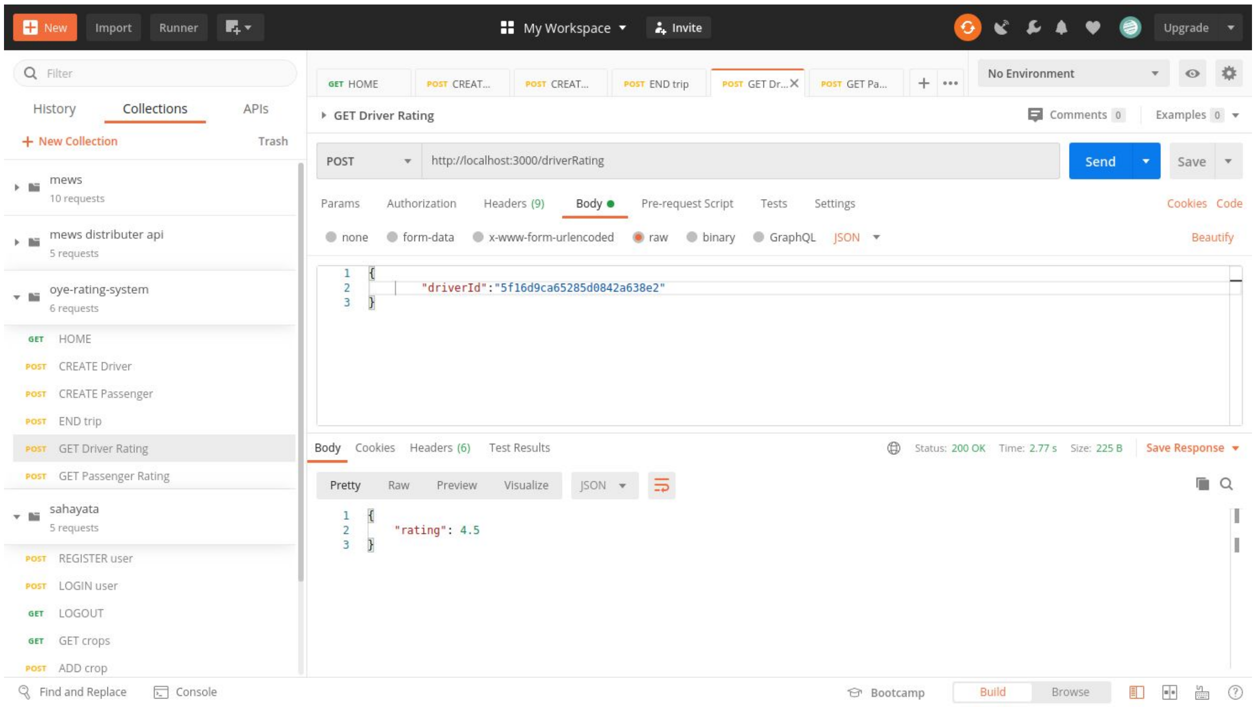


```
POST http://localhost:3000/endtrip
```

```
{
        "startLocation": "Dwarka",
        "endLocation": "Subhash Nagar",
        "passengerId": "5f16da3565285d0842a638e3",
        "driverId":"5f16d9ca65285d0842a638e2",
        "driverRating": 5,
        "passengerRating":5
}
```

```
{
    "_id": "5f16dce9406a3b0bfddac3c6",
    "startLocation": "Dwarka",
    "endLocation": "Subhash Nagar",
    "passengerId": "5f16da3565285d0842a638e3",
    "driverId": "5f16d9ca65285d0842a638e2",
    "driverRating": 5,
    "passengerRating": 5,
    "__v": 0
}
```

5. Now using the information provided in the body object of “/endtrip” endpoint we first create a **Ride** object with all the information of the ride and then we add the **Ride** details to the associated **Passenger** and **Driver** using mongoDb findById method
6. Also while adding the ride object to **Passenger** and **Driver** we update their **avgRating** field by calculating the rating given for the current ride and previous **avgRating**
7. By having a property of **avgRating** in **Passenger** and **Driver** objects we are saving the query time. If the properties are not present then we have to traverse the full **rides** array of **Passenger** and **Driver** which will take O(n) time
8. Finally we save the **Ride** object to our Database
9. We also have two endpoints “/driverRating” and “/passengerRating” to get the Aggregated rating of corresponding **Driver** and **Passenger**



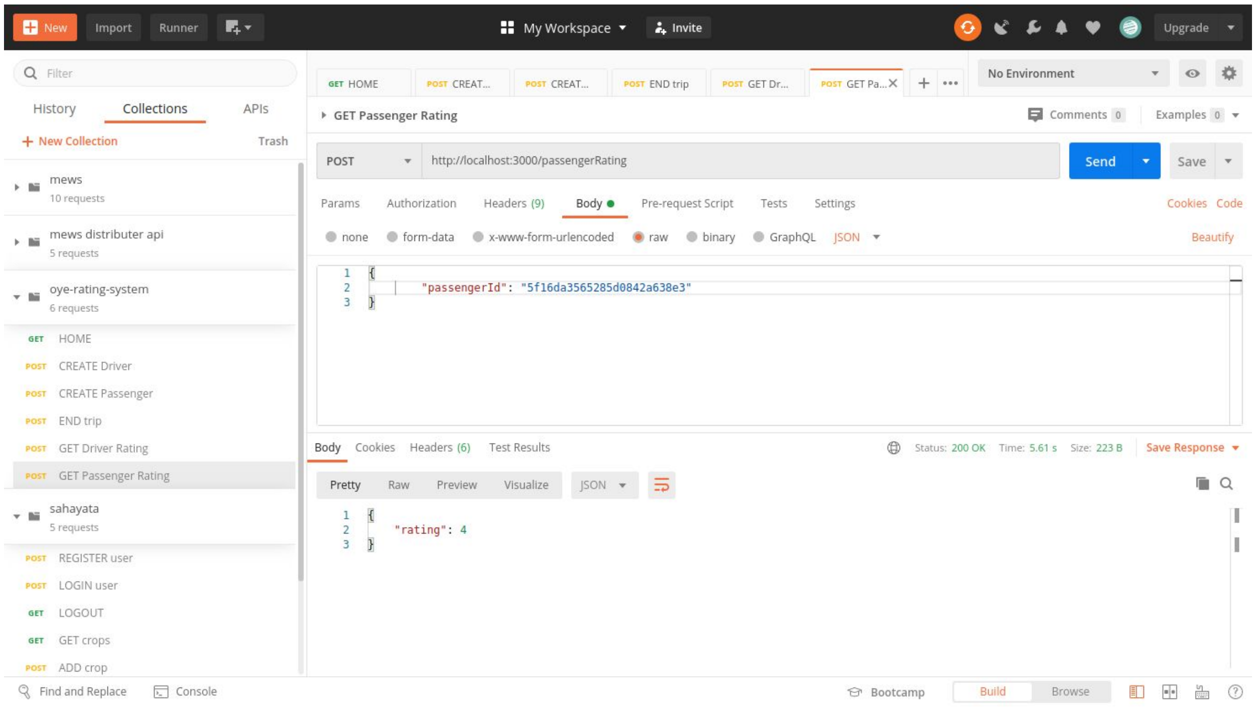
POST http://localhost:3000/driverRating

```
{
    "driverId":"5f16d9ca65285d0842a638e2"
}
```

Status: 200 OK Time: 2.77 s Size: 225 B

```
{
    "rating": 4.5
}
```

10.



GET Passenger Rating

POST http://localhost:3000/passengerRating

```
{
    "passengerId": "5f16da3565285d0842a638e3"
}
```

Status: 200 OK Time: 5.61 s Size: 223 B

```
{
    "rating": 4
}
```

Steps to run your application

1. You need to have node js installed in your system
2. Extract the .zip file
3. While inside the oye-rating-system folder do *npm install* to install all the dependencies from package.json
4. Then do *node app.js* to run the server having all the endpoints
5. Then make post request to the given endpoints for result
6. Postman Collection Link : https://www.getpostman.com/collections/794fe58b7adc10598634
7. Sample requests are shown in the images attached to the doc