

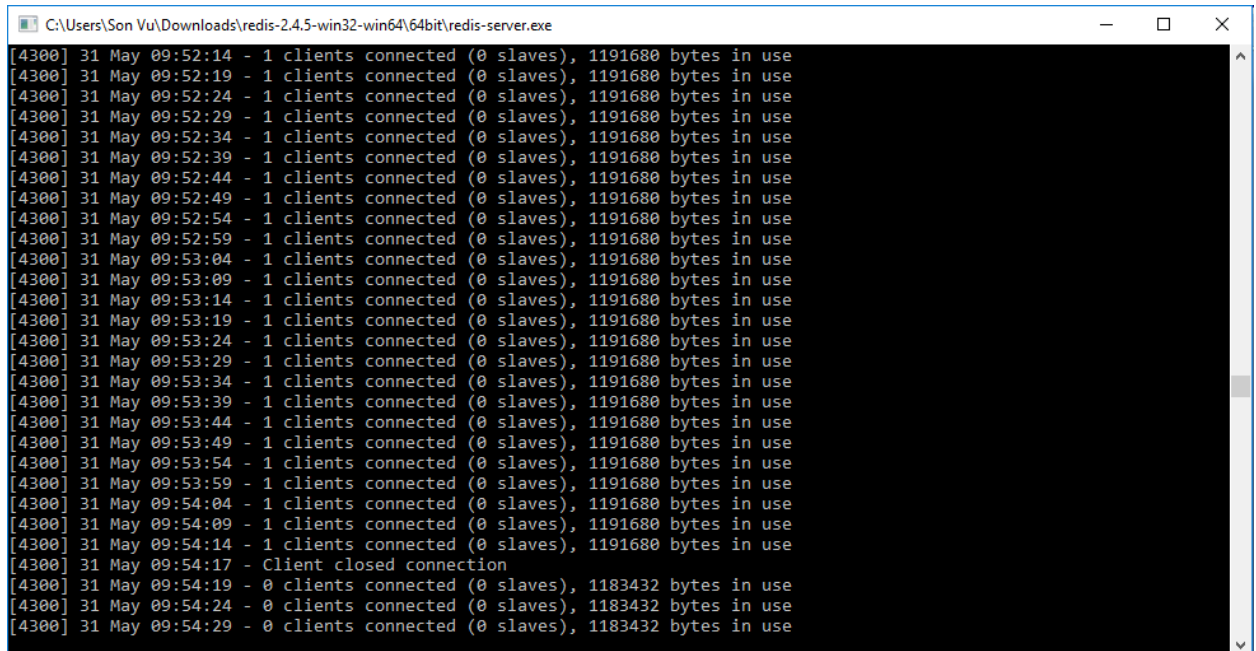
UBER BACKEND TEST

I. Prerequisite

1. Node.js `-v > 6`.
2. NPM
3. Redis
4. Git clone <https://github.com/VictorVuSon/uber-service>
5. Git clone <https://github.com/VictorVuSon/uber-client>

II. Setup

1. Run redis-server: host 127.0.0.1 – port 6379



The screenshot shows a Windows command prompt window titled "C:\Users\Son Vu\Downloads\redis-2.4.5-win32-win64\64bit\redis-server.exe". The window displays a series of log messages from the Redis server. Each line starts with a timestamp in brackets, followed by the date and time, then a status message indicating the number of clients connected, the number of slaves, and the number of bytes in use. The logs show a sequence of connections and disconnections, with the number of clients fluctuating between 0 and 1. The number of bytes in use starts at 1191680 and increases to 1183432 after a client disconnects.

```
[4300] 31 May 09:52:14 - 1 clients connected (0 slaves), 1191680 bytes in use
[4300] 31 May 09:52:19 - 1 clients connected (0 slaves), 1191680 bytes in use
[4300] 31 May 09:52:24 - 1 clients connected (0 slaves), 1191680 bytes in use
[4300] 31 May 09:52:29 - 1 clients connected (0 slaves), 1191680 bytes in use
[4300] 31 May 09:52:34 - 1 clients connected (0 slaves), 1191680 bytes in use
[4300] 31 May 09:52:39 - 1 clients connected (0 slaves), 1191680 bytes in use
[4300] 31 May 09:52:44 - 1 clients connected (0 slaves), 1191680 bytes in use
[4300] 31 May 09:52:49 - 1 clients connected (0 slaves), 1191680 bytes in use
[4300] 31 May 09:52:54 - 1 clients connected (0 slaves), 1191680 bytes in use
[4300] 31 May 09:52:59 - 1 clients connected (0 slaves), 1191680 bytes in use
[4300] 31 May 09:53:04 - 1 clients connected (0 slaves), 1191680 bytes in use
[4300] 31 May 09:53:09 - 1 clients connected (0 slaves), 1191680 bytes in use
[4300] 31 May 09:53:14 - 1 clients connected (0 slaves), 1191680 bytes in use
[4300] 31 May 09:53:19 - 1 clients connected (0 slaves), 1191680 bytes in use
[4300] 31 May 09:53:24 - 1 clients connected (0 slaves), 1191680 bytes in use
[4300] 31 May 09:53:29 - 1 clients connected (0 slaves), 1191680 bytes in use
[4300] 31 May 09:53:34 - 1 clients connected (0 slaves), 1191680 bytes in use
[4300] 31 May 09:53:39 - 1 clients connected (0 slaves), 1191680 bytes in use
[4300] 31 May 09:53:44 - 1 clients connected (0 slaves), 1191680 bytes in use
[4300] 31 May 09:53:49 - 1 clients connected (0 slaves), 1191680 bytes in use
[4300] 31 May 09:53:54 - 1 clients connected (0 slaves), 1191680 bytes in use
[4300] 31 May 09:53:59 - 1 clients connected (0 slaves), 1191680 bytes in use
[4300] 31 May 09:54:04 - 1 clients connected (0 slaves), 1191680 bytes in use
[4300] 31 May 09:54:09 - 1 clients connected (0 slaves), 1191680 bytes in use
[4300] 31 May 09:54:14 - 1 clients connected (0 slaves), 1191680 bytes in use
[4300] 31 May 09:54:17 - Client closed connection
[4300] 31 May 09:54:19 - 0 clients connected (0 slaves), 1183432 bytes in use
[4300] 31 May 09:54:24 - 0 clients connected (0 slaves), 1183432 bytes in use
[4300] 31 May 09:54:29 - 0 clients connected (0 slaves), 1183432 bytes in use
```

2. Open user-service project -> npm install -> npm start

```
Microsoft Windows [Version 10.0.16299.431]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\Son Vu\WebstormProjects\uber>npm start

> Sompo-User-Service@0.0.1 start C:\Users\Son Vu\WebstormProjects\uber
> nodemon server/index.js --exec babel-node

[nodemon] 1.17.5
[nodemon] to restart at any time, enter `rs`
[nodemon] watching: *.*
[nodemon] starting `babel-node server/index.js`
App listening on 3000!
```

3. Open user-clien -> open rider.html -> open driver.html

Name	Date modified	Type	Size
.idea	5/31/2018 9:56 AM	File folder	
css	5/30/2018 2:22 PM	File folder	
images	5/30/2018 3:06 PM	File folder	
js	5/31/2018 9:43 AM	File folder	
driver.html	5/30/2018 3:33 PM	Chrome HTML Do...	2 KB
rider.html	5/30/2018 2:36 PM	Chrome HTML Do...	2 KB
UBER BACKEND TEST.doc	5/31/2018 9:50 AM	DOC File	28 KB

rider.html

Request your trip

Please enter information

From:

To:

Submit

driver.html

Your picking trip

List requesting trip

III. Idea and solution

1. Technologies

Nodejs for backend.

SocketIO for realtime.

Redis for storing database.

2. Solution

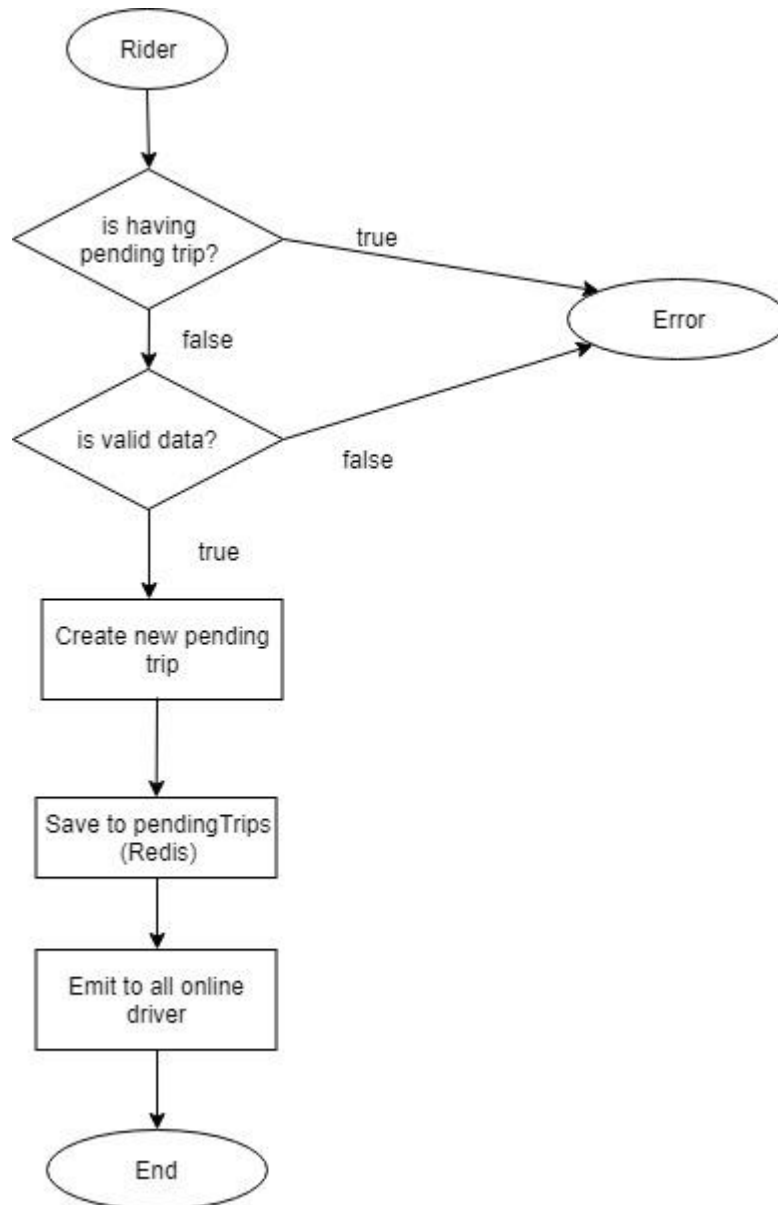
- 2.1. When an user requests a ride, the request goes out to the nearest highest rated driver. The driver then receives a notification upon which he can choose to respond or ignore. If he chooses to ignore, the request will go to the next nearest driver.

2.2. When an user requests a ride, the request will go out to all the drivers in the proximity. Whoever accept first will be awarded the ride.

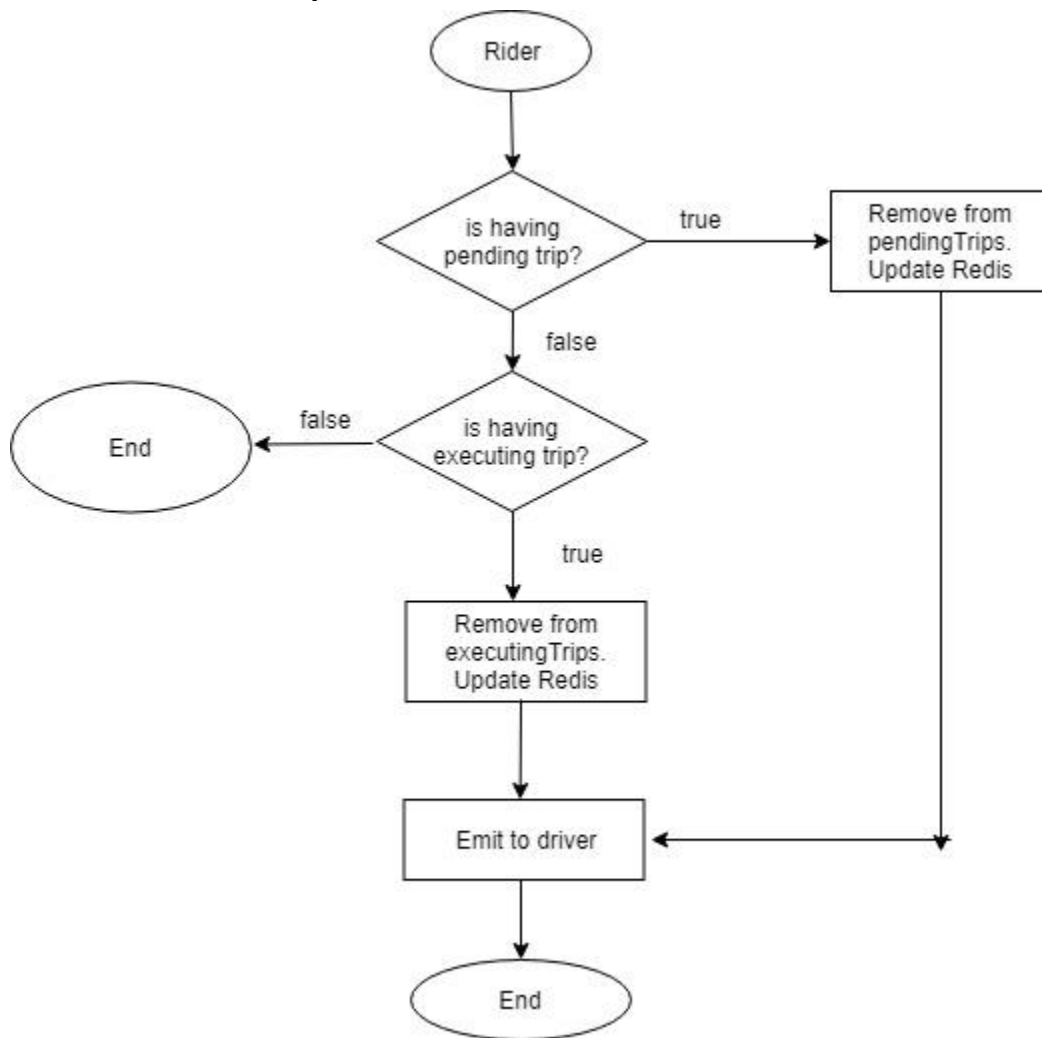
- **This demo is emulating the method 2.2.**
- **Server listen to 2 namespaces of socket: '/riders', '/drivers'**
- **Two namespaces of sockets will communicate to each other for handling request trip, pick trip, cancel trip ...**
- **Redis will create a client for storing data including two main keys: 'pendingTrips', 'executingTrips'.**
- **Redis uses hash set type for storing data then we can take the power of hash set type.**

3. Block diagrams

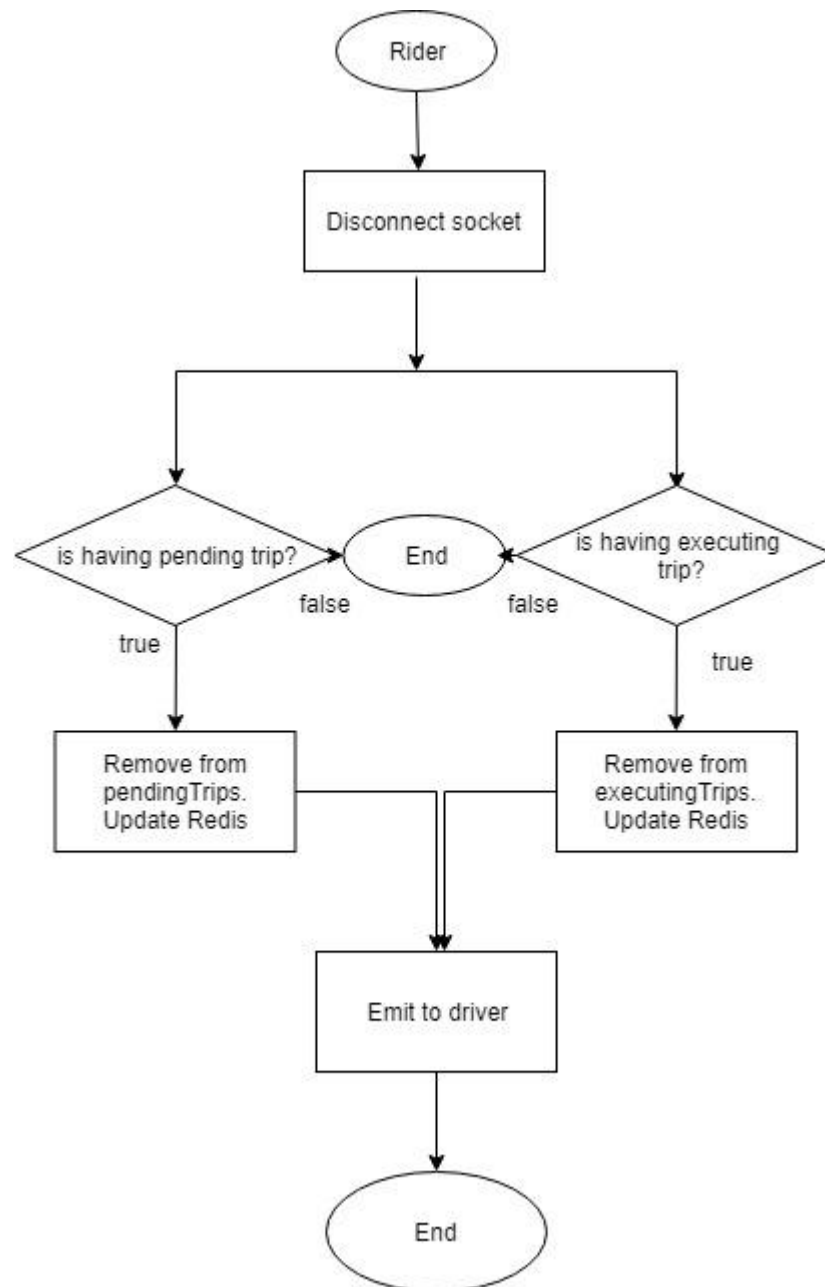
a. Rider requests a trip.



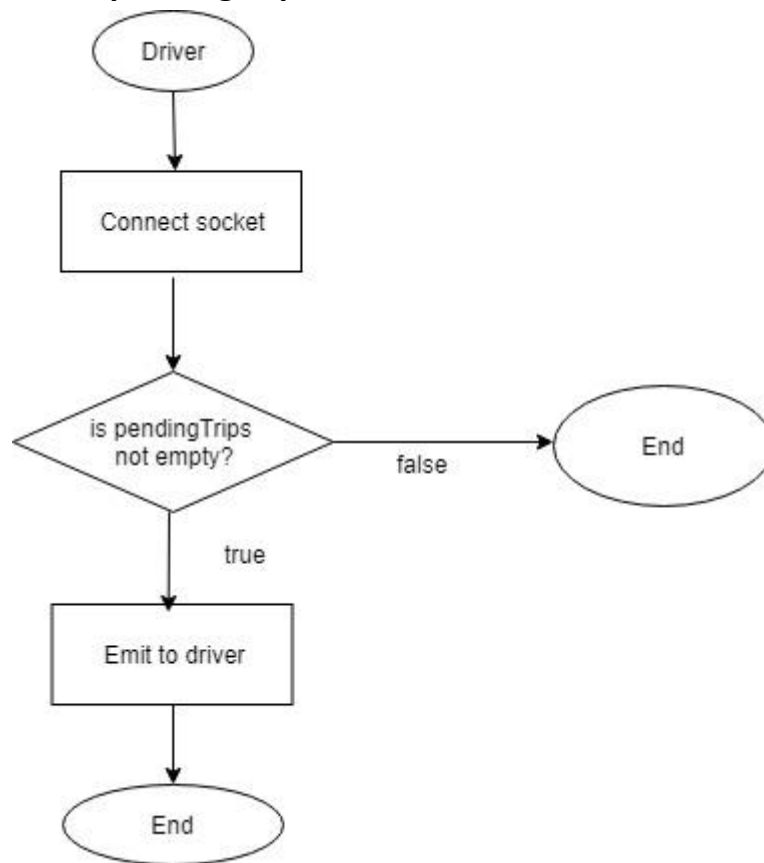
b. Rider cancel a trip.



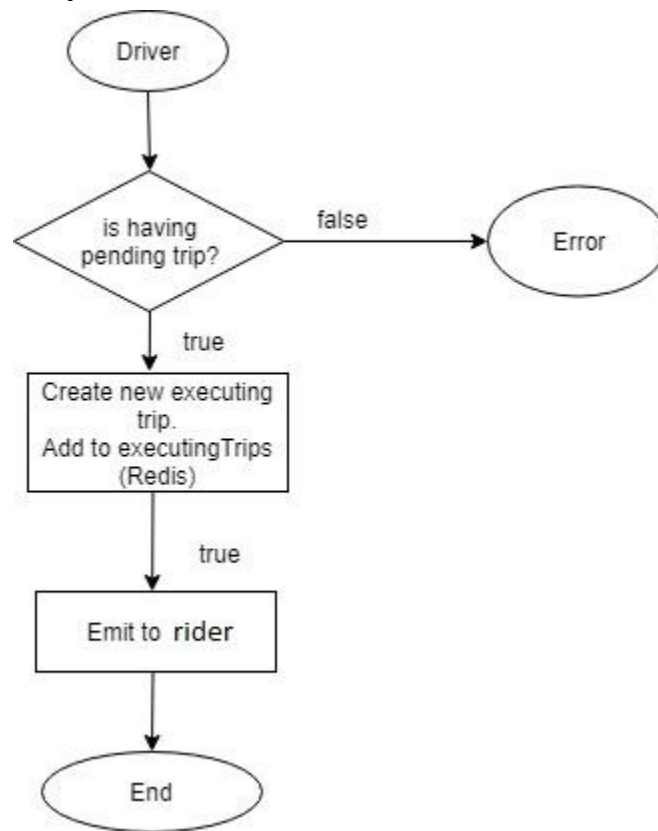
c. Rider disconnects.



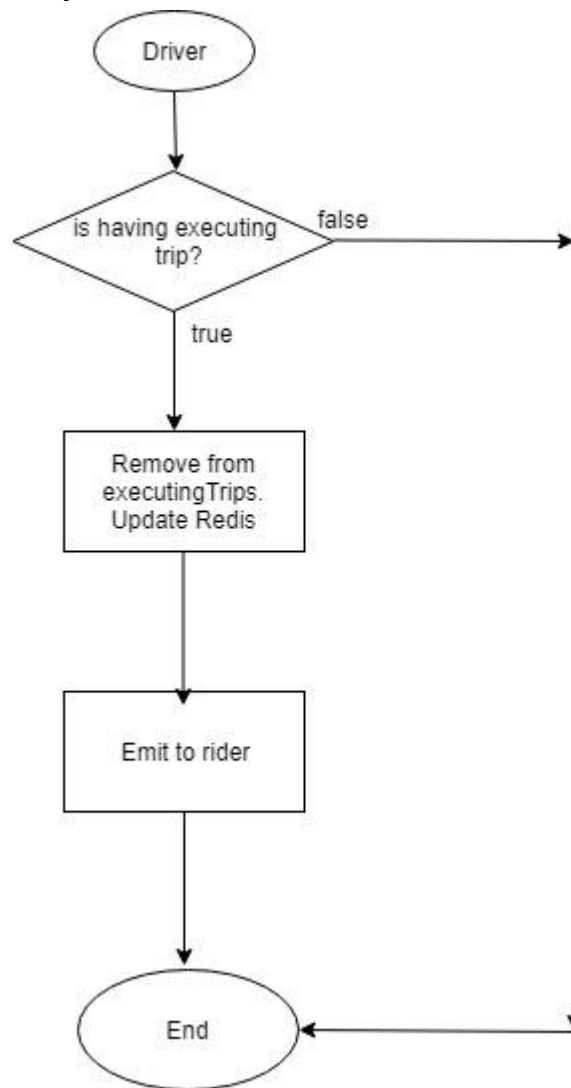
d. Driver loads pending trips from server.



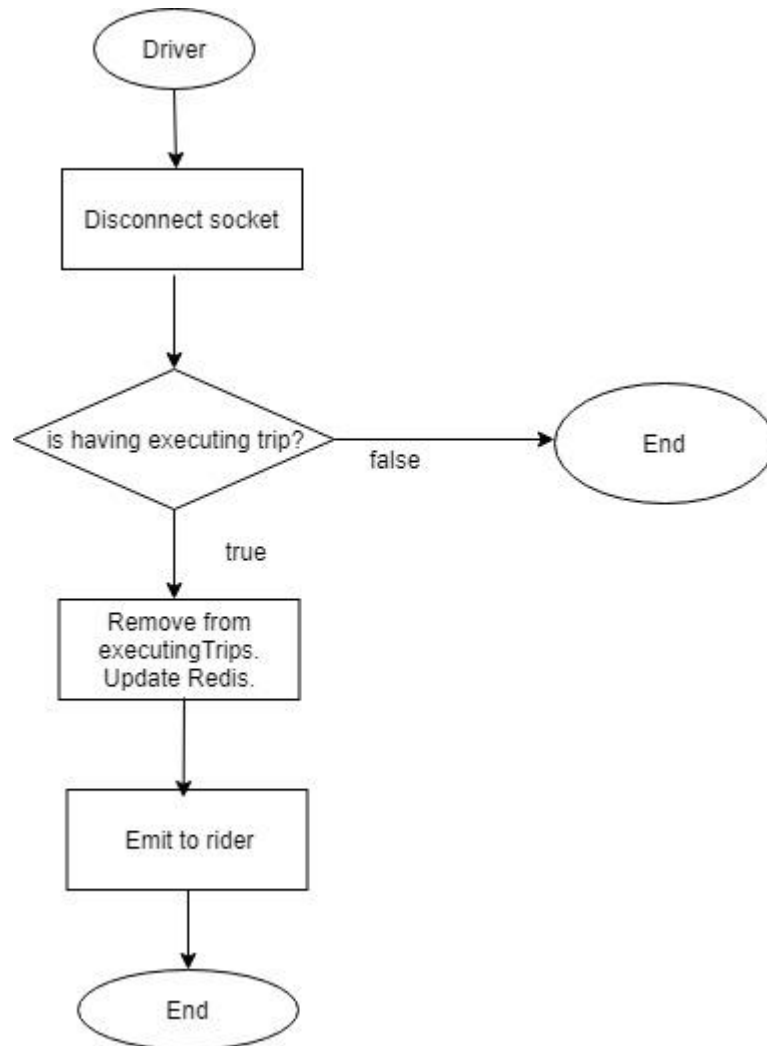
e. Driver picks a trip.



f. Driver cancels a trip.



g. Driver disconnects when having a picking trip.



IV. Conclusion

This is basic application to emulate basic uber app working.

- In this demo, I use socket io for execute realtime. I also seperate socket in to 2 distinct namespace: one for rider and another for driver.

By this method, we can easily hanlde when we hit a bunch of event from client and easily make communication between rider and driver.

- To analize database storing, We should use Redis for store the data which need to be access quickly. A recomendation is use hashset data type which is a power full type for process a big and heavi data.

- Moreover, We can also use Mongo DB for storing some necessary data. Anyway, to make higher performance, we can use synchronization method to get data from redis to mongo db.

--- Translation ---

Xin chào anh chị,

Đây là bài test mô phỏng request trip của tôi. Demo này mô tả một số chức năng cơ bản trong quá trình request và cancel trip, giao tiếp giữa rider và driver.

Trong demo này, tôi không nhấn mạnh về chức năng. Tôi chỉ muốn nhấn mạnh về ý tưởng cũng như kỹ thuật mà tôi đã sử dụng. Sau đây tôi xin được trình bày sơ qua.

Backend server được viết bằng Nodejs. Cơ chế giao tiếp giữa rider và driver bằng socket io được chia thành hai namespace riêng biệt cho rider và driver. Điểm này giúp cho việc handle event được rõ ràng và dễ dàng implement cũng như maintain hơn.

Trong demo tôi cũng đang sử dụng Redis để lưu dữ liệu tạo cho việc truy xuất và lưu lại nhanh hơn. Tôi sử dụng data type hashset một loại tối ưu performance nhất trong Redis.

Về tương lai, tôi sẽ phát triển hệ thống như sau:

- Nodejs service + Redis + Mongo DB: realtime logic.
- Nodejs service + Mongo hoặc PHP + MySQL: bussiness logic(payment, login,)

Tôi sẽ trình bày rõ hơn về lợi thế, ưu nhược điểm, vì sao chọn công nghệ, vì sao phân tầng service như vậy nếu như có cơ hội phỏng vấn và làm việc cho công ty. Tôi xin cảm ơn.