1 User Information Not Encrypted in the front end

username, email are all stored in a local variable

```
export default new Vuex.Store({
  state:{
    login: false,
    hasCar: false,
    user: {
      id: '',
      username: '',
      email: '',
      password: '',
      car: {
        id: '',
        type: '',
        licensePlateNumber: '',
        capacity: '',
        otherInfo: '',
        owner: ''
      },
}
```

Some of the informations has stored in local browser. If user's computer gets hacked, there is the chance that username as well as password will leak

2 All Rides Info are stored in front end

```
mounted() {
   loadOpenRide(this.$store.state.user.id).then(
     (response) => {
       if (response.data.length != 0) {
         this.tableData = _.cloneDeep(response.data);
       }
     },
     (err) => {
       alert("Something Went Wrong, May come back later")
     }
   );
 },
```

The thought at the beginning is that by getting all available rides from backend, we filter it in the frontend, which is quite convenient.

But chances are good that users may see some of the orders that does not belong to them, they cannot join though.

- We should limit it into the rule that everyone should has the least privilege in this system
- we can improve that later

3 Unlimited Starting a Ride

We did not restrict the times user can submit a ride

```
     createRide: function(){
         postride(this.ride.destAddr,
         this.ride.date,
         this.ride.time,
         this.ride.numPeople,
         this.ride.isJoinable,
         this.ride.otherInfo,
         this.$store.state.user.id).then((response) => {
           alert("Publish Info successfully")
           console.log(response.data)
           this.$router.push('/availableorder')
         }, (err) => {
           alert(JSON.stringify(err.response.data))
         })
     },
```

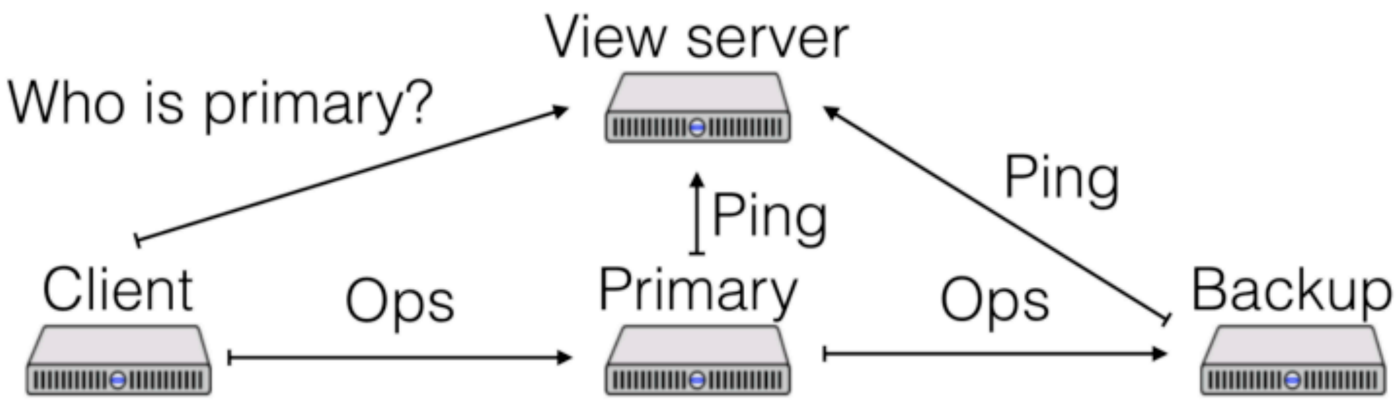
there might be a chance that users can submit many orders at the same time and overload our servers

- we can, however, check at the frontend if the user request too frequently. If yes, we should block and cool it down
- also we can check for the start time and end time of a ride

4 Single Point of Failure

right now, we only got one server, the availability is quite low. We should do some system design that makes replicas for the server and thus, it can provide a reliable service.

The structure is quite like



the basic design is we get View Server (VS) to inform the client who is the primary and vs can route the trafiic.

- we consider each server as a Finate State Machine, the input is ops from client, the output is the state server can maintain
- Once the primary is down, we can take it over using backup

ref:

1: https://users.cs.duke.edu/~chase/cps512/internal/7-replication.pdf