CRAFT DEMO

Ride Management System: Driver Module

DATE : / /

Ride Sharing App

* Driver Register Module

Functional Requirements

- Add/Register driver
  - ~~Upe~~ Add Driver address
  - Add Driver identity information
  - Enter Driver car/vehicle details
- For Background Verification:
  - Collect/Verify license number
  - Verify license expiry date
  - Update Verification Status
- OnBoarding
  - Ship tracking device to driver
  - Allow drivers to update his availability

Non Functional Requirement

- Scalability / Scalable
- Availability
- Eventual Consistency
- Low Latency

- To scale the app:

  We will make use of load balancer that will auto-scale the services as and when required.

- To reduce load on Database for writes, we will shard the database using horizontal sharding.

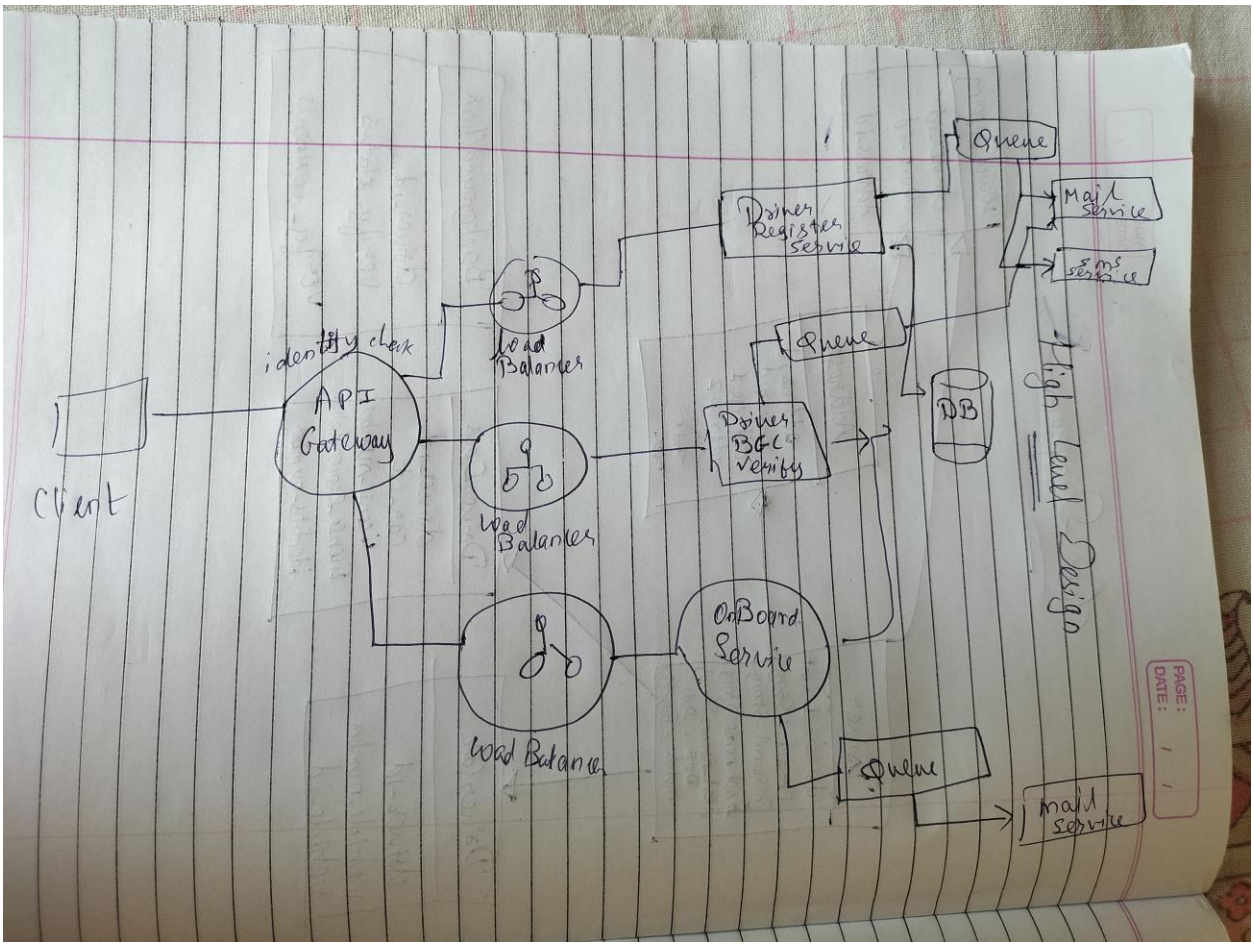
- For Horizontal sharding, we will consider Geo-location as key value

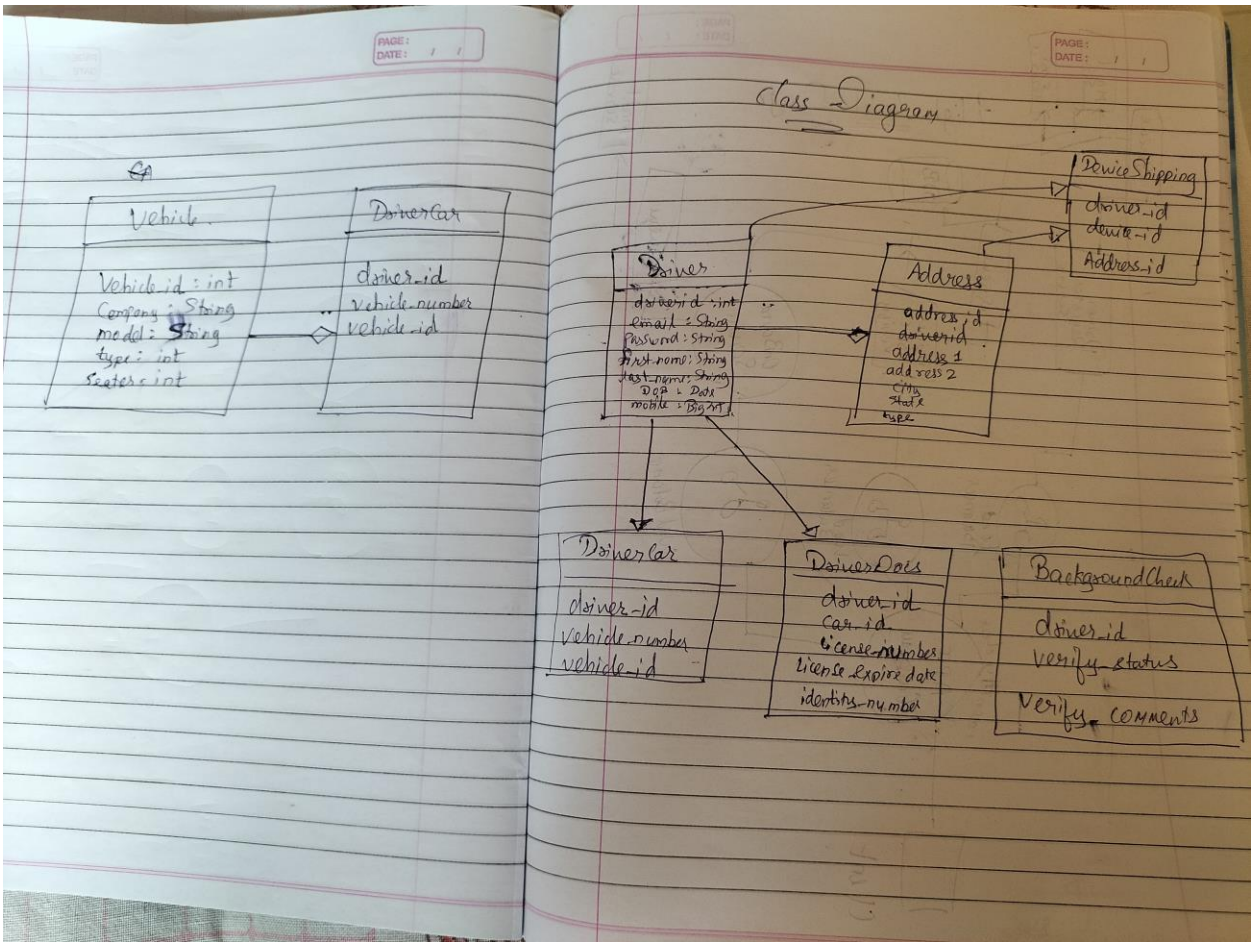
- We will use Master-Slave architecture for read write DB

- As the number of driver registration would be minimal, we would make use of Write Back cache strategy.

- For routing, we will make use of Load Balancer level 4 routing at transport layer

- Queue Service will be used to send Mail notification and can be Sms service.
- As the data releases from Queue, it will be used by Mail and Sms Service.


Client
identity check
API Gateway
Load Balancer
Load Balancer
Load Balancer
Driver Register service
Driver BGC verify
OnBoard Service
Queue
Queue
Queue
Mail service
sms service
DB
mail service
High Level Design
PAGE :
DATE : / /


CA
Vehicle
Vehicle_id : int
Company : String
model : String
type : int
seaters : int
DriverCar
driver_id
vehicle_number
vehicle_id
Class Diagram
DeviceShipping
driver_id
device_id
Address_id
Driver
driver_id : int
email : String
Password : String
first_name : String
last_name : String
DOB : Date
mobile : BigInt
Address
address_id
driver_id
address 1
address 2
city
state
type
DriverCar
driver_id
vehicle_number
vehicle_id
DriversDocs
driver_id
car_id
license_number
License_expire date
identity_number
BackgroundCheck
driver_id
verify_status
Verify_ comments

Tech Stack

- Spring Boot for Microservices
- MySQL as to Maintain ACID properties and also due less numbes of transaction

Although Some features can be implemented using NoSQL (MongoDB) like Driver verification details, address

- For logger → Common Logger is used but for overall logging splunk can be used
- CI/CD → Dockes with Jenkins + Github
- Test → ~~Junit~~ Cucumber for integration and overall testing

Junit for unit testing

- For API Docs → Swagger API

PAGE :
DATE : / /

Design Patterns

→ Decomposition : To implement Single responsibility principle.

For this decompose was done considering Business capability/task

→ API Gateway Pattern : [Integration Pattern]

: UserAuth will act as the Gateway to connect with other Microservices

→ Observability Pattern :

Loggers used to track the user activity in each microservice call