# Api Design/System design (Traffic Violation and penalties)

# **1.1** Problem Description / Requirement:

Vehicles have **violations and penalties** associated with it, which owners have **not paid** it since **some/ long** time.

**1.1.1**

Vehicles on road passing via **traffic signal**s is a potential place where **vehicle details** can be captured

* Identify Vehicle’s information at a signal

**1.1.2**

System checks in real time/near real time if there are some **violations records** for the vehicle.

Then it can **notify** **authorities** immediately in various means about it along with a portal feature where **vehicle owners/ drivers** can check violations, **pay fines or appeal** against it.

* Fetch violations done for the identified vehicles and
* Alert the police standing at the signal so Police can catch them then and there

**1.2** Can there be a system to solve this problem?

 • Consider X number vehicles with data such as: **Vehicle Number, Owner of it, when vehicle was purchased, color, Make,** etc.

 • Consider data which is from Government having fields like: **Vehicle Number, Violation Done Violation Date, Penalty amount.**

**1.3** Objectives:

* Need Data storages structures (Use any set of databases)
* Microservices architecture
* API interface
* Application using Spring/Spring Boot, Microservices for the above interface **(or your choice in case you want to develop a prototype - Optional)**.
* It will also help if you can come up with an approach/design on how vehicle details can be captured at a traffic signal.

**1.4 Out of Scope**

**Note: Please mention the assumptions that might take clearly**

**Addition Architecting requirements (potential)**

It would be good to incorporate the following while designing

* How will you handle data from thousands of Vehicles – it should scale without any performance degradation along with elasticity
* How will you optimize your queries on millions of records stored over the time? Depending on the DB you should be able justify your queries. Also, data could be stored for years for legal requirements
* How would you reduce cost for runtime and data storage (in general strategy to reduce cloud cost)?
* Point out the Security challenges and how will you mitigate them
* Point out the Privacy challenges and how will you mitigate them
* Cloud tech stack one can use here – be it AWS/ GCP or any other

**2.1 Identify Microservice**

Vehicles on road passing via **traffic signal**s is a potential place where **vehicle details** can be captured

* Identify Vehicle’s information at a signal

**2.1.1**

Api Name: identifyVehicle() ,

Metod: GetMapping

End Point: <http://localhost:8081/identifyVehicle>

Returned Object: **Vehicle** (VehicleId, VehicleName, purchaseDate, color, make, And

DriverOwner object)

**2.1.2**

Api Name: processVehicle() ,

Metod: PostMapping Consume Govt Api (**http:vahan.com/getViolationDetails**)

End Point: <http://localhost:8081/identifyVehicle>

Returned Object: **Vehicle** (VehicleId, VehicleName, purchaseDate, color, make, And

DriverOwner object)

**2.2 Notify Microservice:**

System checks in real time/near real time if there are some **violations records** for the vehicle.

Then it can **notify** **authorities** immediately in various means about it along with a portal feature where **vehicle owners/ drivers** can check violations, **pay fines or appeal** against it.

* Fetch violations done for the identified vehicles and
* Alert the police standing at the signal so Police can catch them then and there

**2.2.1**

Api Name: violationRecord(int vehicleId)

Method: PostMapping

End Point: <http://localhost:8082/saveUpdateRecord>

Returned Object: **VehicleWithFine**

**Vehicle** (fineId, fineName, fineDate, Amount)

**Assumption:** Consuming Api provide same json of Vehicle Object else use @IgnoreJsonProperties

**2.2.2**

Api Name: notifyVehicle()

End Point: <http://localhost:8082/notifyVehicle>

Returned Object: Boolean

**Assumption:** Phone number and Email address of Driver/Owner is validated at purchase

**2.3 Api Design For Identification and Notification Service**

**2.3.1**

Api Name: identifyVehicle() ,

Metod: GetMapping

End Point: <http://localhost:8081/identifyVehicle> , <http://localhost:8081/processVehicle>

a)**Vehicle.java**

**Datatype:**

int VehicleId,

String VehicleName,

Date purchaseDate,

String color,

String make,

Object DriverOwner

b**)DriverOwner.java**

**Datatype:**

Int licenceId ,

String name,

int age,

String city,

Int phone

c) **IdentifyVehicleController.java , package: controller**

@RestController

@PropertySource(Value=”application.properties”) or @value inside the method

**Method 1: (** <http://localhost:8081/identifyVehicle> **)**

\*public Vehicle identifyVehicleByNo(int vehicleId)

//call service **IdentifyVehicleService** .identifyVehicle() method

**Method2: (**<http://localhost:8081/identifyVehicle>**)**

\*public Vehicle processVehicle()

//call **IdentifyVehicleService** processVehicle() method

d)**IdentifyVehicleService.java , package: service**

\*public Vehicle identifyVehicleByNo(int id)

//call method **VehicleRuleUtil**.getVehicle() method

//Business logic and manipulation if any

//Business logic and manipulation if any

\* public Vehicle processVehicle(String name)

//call method **VehicleRuleUtil**.processVehicle() method

//Business logic and manipulation if any

e)**IdentifyVehicleRepositoryRequestDto.java ,**

**IdentifyVehicleRepositoryResponseDto.java package:dto**

@jsonIgnoreProperties

@JsonInclude

f)**VehicleRuleUtil.java package:rule**

@@Autowired RestTemplate

HttpHeader h= null;

Void init(){

H= new Httpheader();

Header.setContentType(MediaType.Json)

ObjectMapper mapper=new ObjectMapper();

**Method 1: getVehicle(int vehicleId){**

Url=env.getProperty(“vehicle.process.url”+vehicleId)

HttpEntity<String> entity = ne HttpEntity<String>(jsonRequest,header);

Result=template.(“url”,entityName,String.class);

Response=mapper.readvalue(result,vehicleresponse.class)

**Method 2: processVehicle(Vehicle vehicle){**

jsonRequest=mapper.writeValueAsString(Vehicle)

HttpEntity<String> entity = ne HttpEntity<String>(jsonRequest,header);

Result=template.**postForObject**(“url”,entityName,String.class);

Response=mapper.readvalue(result,vehicleresponse.class)

//get the notify phone number

@Autowired KafkaTemplate

kafkaTemplate.send(topic:“TopicName”, message:phoneNumber )

g)application.properties

server.port:8081

vehicle.entry.url= **http:vahan.com/getVehicleDetails**

vehicle.process.url= **http:vahan.com/getViolationDetails** f)AppConfig.java

package: config

//handler for resttemplate

(@configguration, @Bean RestTemplate getTemplate(){template.getMessageConverter()}

getTemplate(){template.getMessageConverter(MappingJackon2HttpMessageConcerter)})

**2.3.2**

Api Name: notifyVehicle()

End Point: <http://localhost:8082/notifyVehicle>

Returned Object: Boolean

**Assumption:** Phone number and Email address of Driver/Owner is validated at purchase and same number is for traffic police

a) **VehicleWithFine**

**Vehicle** vehicle;

Fine fine;

DriverOwner driverowner;

a)**Fine.java**

int fineId,

String fineName,

String fineDate,

BigDecimal Amount

**b)Vehicle.java**

**Datatype:**

int VehicleId,

String VehicleName,

Date purchaseDate,

String color,

String make,

Object DriverOwner

b**)DriverOwner.java**

**Datatype:**

Int licenceId ,

String name,

int age,

String city,

Int phone

c) **NotifyVehicleController.java , package: controller**

@RestController

@PropertySource(Value=”application.properties”) or @value inside the method

**Method 1: (** <http://localhost:8081/notifyVehicle> **)**

\*public void notifyVehicle()

//call service **NotifyVehicleService** . notifyVehicleService () method

d)**NotifyVehicleService.java , package: service**

\*public void notifyVehicleService()

//call service **NotifyTrafficPolice** . notifyVehicleDao () method

e) **NotifyTrafficPolice.java , package:dao**

@Autowired KafkaTemplate

**@kafkaListener(topics:“topicName”, groupid:consumerId)**

**public void notify(int phonenumber){**

**//logic to send notification to traffic police**

**//update notify flag in database with “Y” for no such further processing**

**Database 1: Sql**

**Table:1**

**Vehicle**

integer VehicleId,

varchar VehicleName,

Date purchaseDate,

varchar color,

varchar make,

integer licenceId ,

**Table:2**

**DriverOwner**

integer licenceId ,

varchar name,

integer age,

varchar city,

integer phone

**Database 2: NoSql**

**Collection:**

**Notify**

Ex:

{

notificationId: 101

notified :Y

date: 06-Oct-22

phone:1231231231

}

**3.0** It would be good to incorporate the following while designing

* How will you handle data from thousands of Vehicles – it should scale without any performance degradation along with elasticity:

We can deploy in ec2 with increase capacity

* How will you optimize your queries on millions of records stored over the time? Depending on the DB you should be able justify your queries. Also, data could be stored for years for legal requirements:

In case of SQL we can do sql optimization by:

using joins, instead of Select \* we can particular define column name, avoid select distince/unique , avoid having clause, avoid too many joins use inner query, use index, use limit, put wildcards at end of the query, split big dml operation into small queries, use correct datatypes etc

In case of No-Sql:

use aggregation, get the response in json format in single database call

* How would you reduce cost for runtime and data storage (in general strategy to reduce cloud cost)?

We need to follow lifecycle for external data storage for processed data.

Ex: In s3: move the object to low cost storage after 30 day, move to glacier after 90 days, deep archive after 180 days and delete after 365 days

* Point out the Security challenges and how will you mitigate them

By using spring security, customized oauth2 and jwt for authorized token between microservices without re-authenticating. We can use secret manager to store secrets.

* Point out the Privacy challenges and how will you mitigate them

We can use signed certificates jars, role based access for data security, use of protocols https, ssl, tls , sonar for code coverage and Veracode to mitigate very high/high vulnerable.

* Cloud tech stack one can use here – be it AWS/ GCP or any other

I prefer AWS as it provides more services and options.

**4.0** To build this ‘Traffic Violation and penalties’ application for long term and we will have to design architecture like below:

1. For security purpose who can access the application we have to do proper authentication and authorization we can use oauth2, iam , api gateway
2. Application need to in private network: we can use vpc , subnet
3. When lot of users will try to access we have to deploy our java application in on premises server or EC2 : horizontally/vertical scaling is also required
4. Once instance increased, we can use Load balancer. We can nginx/ Application load balancer (I prefer ALB) also We can have our DNS service.
5. We can move our database to No-Sql in case of non OLTP transaction
6. We can increase read replica if our application is read heavy and accessed through different zones or region.
7. To increase application performance and decrease latency we can use database cache Redis / Ehcache/Memcached
8. To create container image, we can use docker and we can deploy in Kubernetes through proper objects Service->deployment->replica set->node->pods with master slave configuration.
9. If vehicle picture/videos need to be uploaded for validation proof then more storage required, we can use external storage S3 and follow s3 life cycle for cost optimization.
10. Once image/video uploaded in s3 for any kind of violation then corresponding process need to be triggered, we can use lambda.
11. After some period, all the older data need to be analyzed properly and periodically
12. For mobile app uses application need to be developed for android and ios.
13. We can notify Driver and owner through message and mail by using message queue and SES, SNS.
14. If fine amount is not paid then we need to remind and increase penalty with notification we can use SQS, lambda and SNS for this.
15. To monitor services with matrix, we can make dashboard which will monitor and trigger notification in case of any failure by lambda and CloudWatch