Agenda

1) What is an API
2) What is REST
3) Good Practice of REST
4) LLD of Product Service

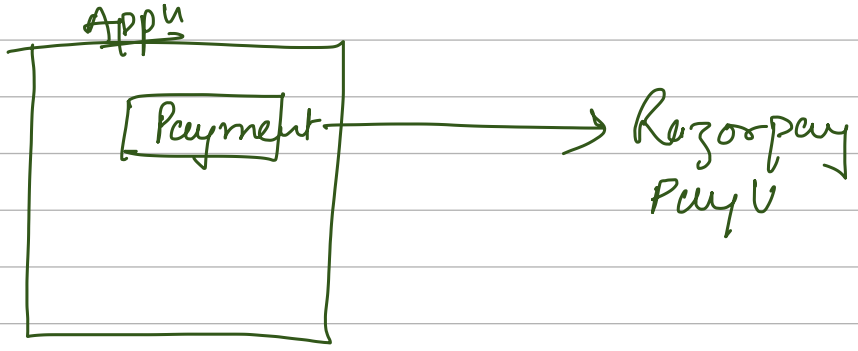
What is an API

A → Application
P → Programming
I → Interface
↳ [Contract definition]

an interface for an App

interface Runnable {
    void run()
}

Thread can be created



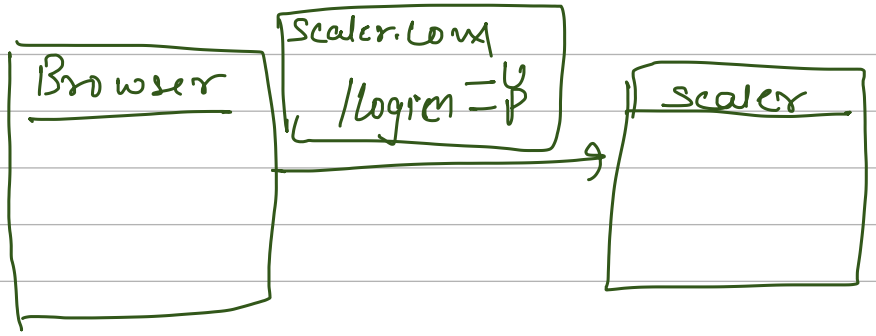
/pay
↳ Auth
↳ srcBankId
↳ destBankId
↳ amount

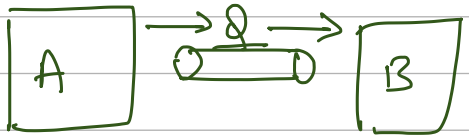
"API" contract

→ Without API can Appn talk to each other

↳ NO

To Talk to 3rd party appn APIs are req.





Whenever 2 Appn are talking to each other, they must have a well defined contract

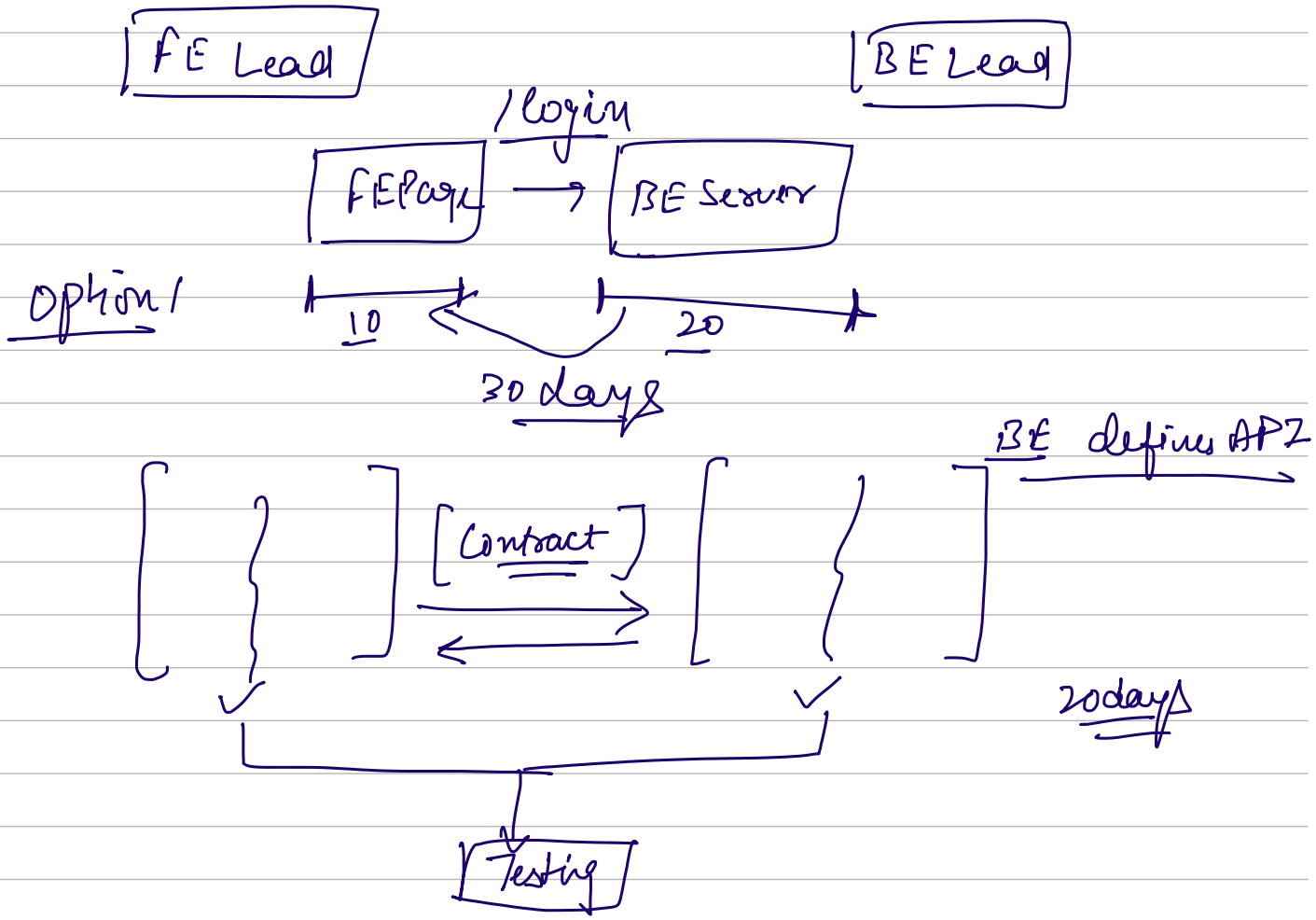
→ where → URL
→ what to send → Request
→ what will received → Response

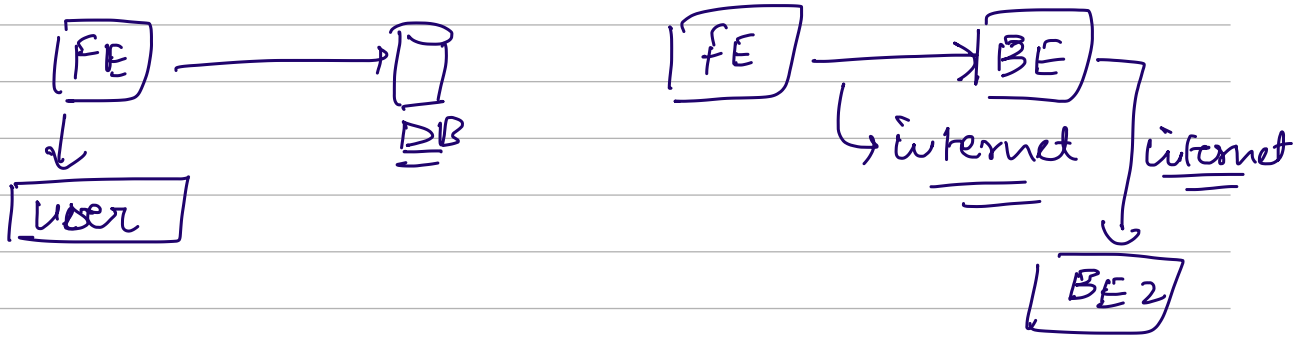
API set of methods / endpoint that are provided by a system for another system to interact

a contract b/w you and 3rd party system

Product Manager

[Responsible for defining what to Built]





Eg User courses → User
↳ Courses

Eg [scaler.com/get-user/1
razorpay.com/make-payment/2id]

REST Api's

RE → Representational
S → State
T → Transfer

→ SOLID principles of APIs
→ Set of good practices around API design
→ how should API be structured

Good Practice

1) APIs should be defined around resources
↓
Entities that are affected by APIs

Eg /say_something_good

/products

/users

API endpoints should be pointing to the name of resource that they are affecting

/users → C Create
R Read
U update
D delete
} VERB

↳ NOUN

/get-user X

Type of Request

GET	Read
POST	Create
PATCH	Update
DELETE	Delete
PUT	Update (Replace complete)

GET /users/{id}

/users?id={id} X

?feild={value} &
↳ filtering

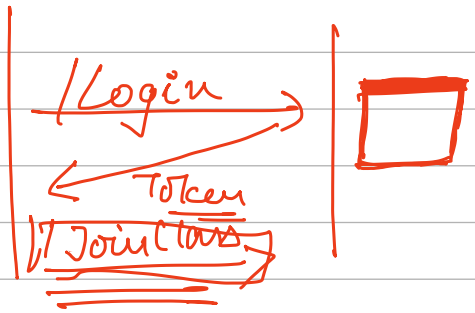
/user?batchId=1

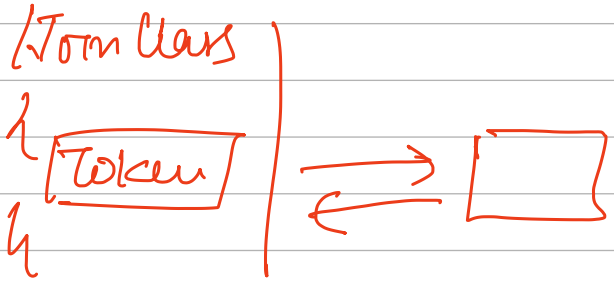
/posts?tags=---

2) REST APIs should be stateless

→ Every req should be self sufficient and should not depend on the prev request that may have been sent

→ ① self sufficient
→ ② Independent





Every req should contain all data needed by server to serve the req.

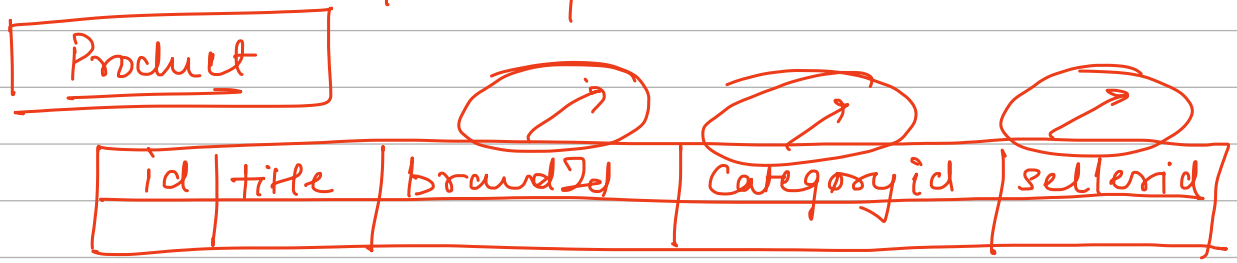
REST APIs

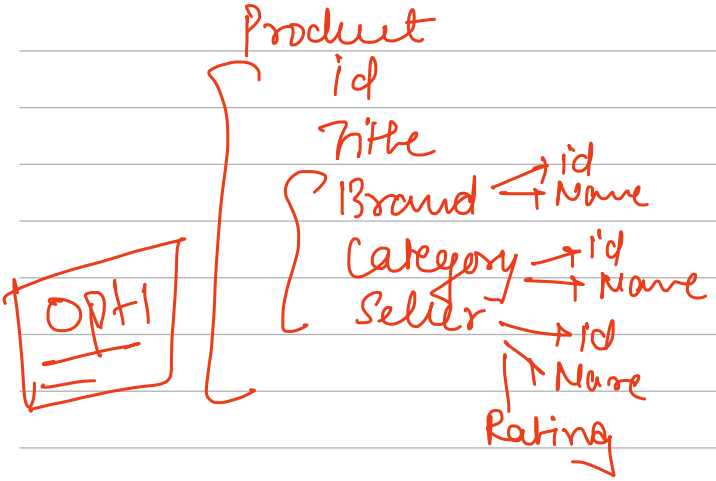
→ Make system very scale able

→ Server is not maintaining any info within it

(3) APIs should not have 1:1 mapping with DB tables

/product/{id}





1 : M

OPT2

Response

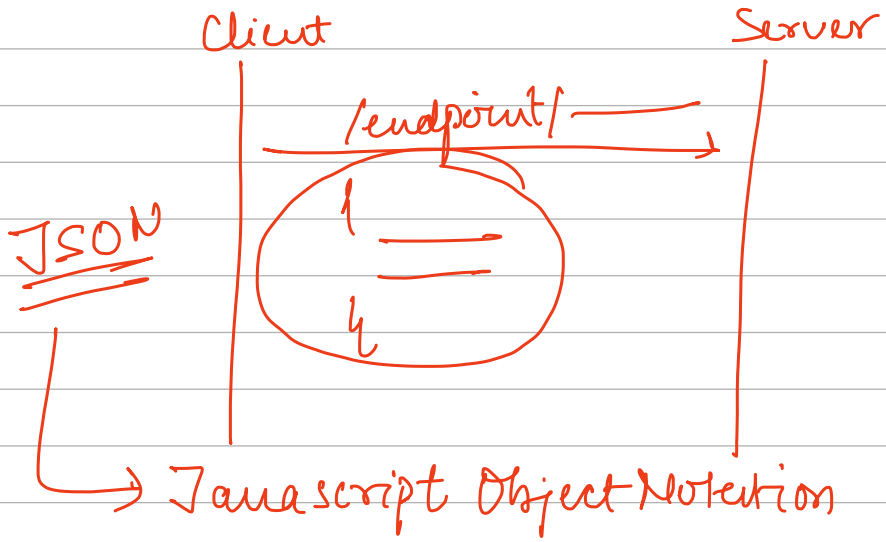
id
title
Brandid
Categoryid
Sellerid

GET /categories/{id}
GET /brands/{id}
GET /seller/{id}

Too Many API calls

Chatty APIs

(4) Data interaction



JSON → Javascript Object Notation

XML

Protos | Protobuf

JSON

```
{ "id" = —
  "name" = —
}
```

XML

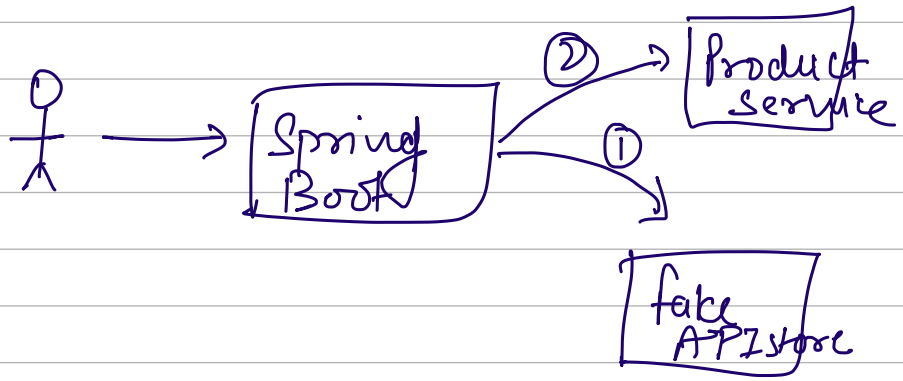
```
<id>
<value> — </value>
</id>
```

LLD for Product Catalog Service

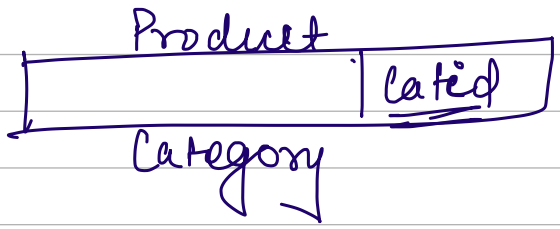
Product
-id
-name
-title
-price
-desc
-Brand
-Category
-ImageURL

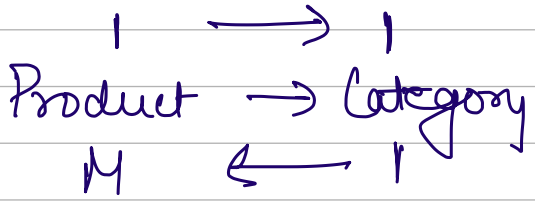
Category
-id
-name
-description
-List<product>

- [Hardware → tools → Cartools →]
- [Electronics → Mobiles → smartphones]



Schema Design





M:1

Assign

1) API Design
2) Rest / vs / SOAP
3) Create your APIs
4) Call Other APIs