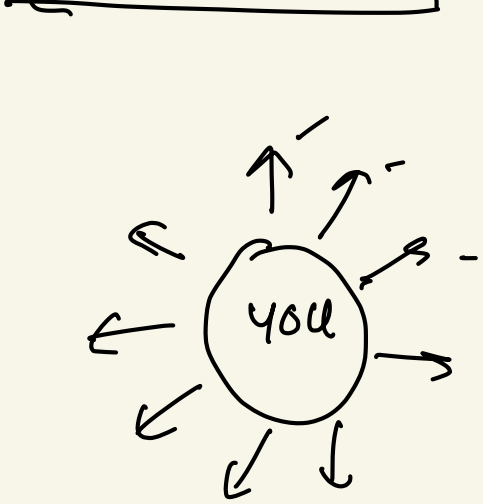
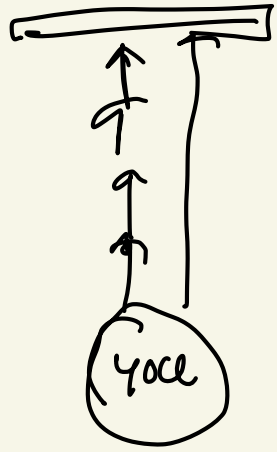
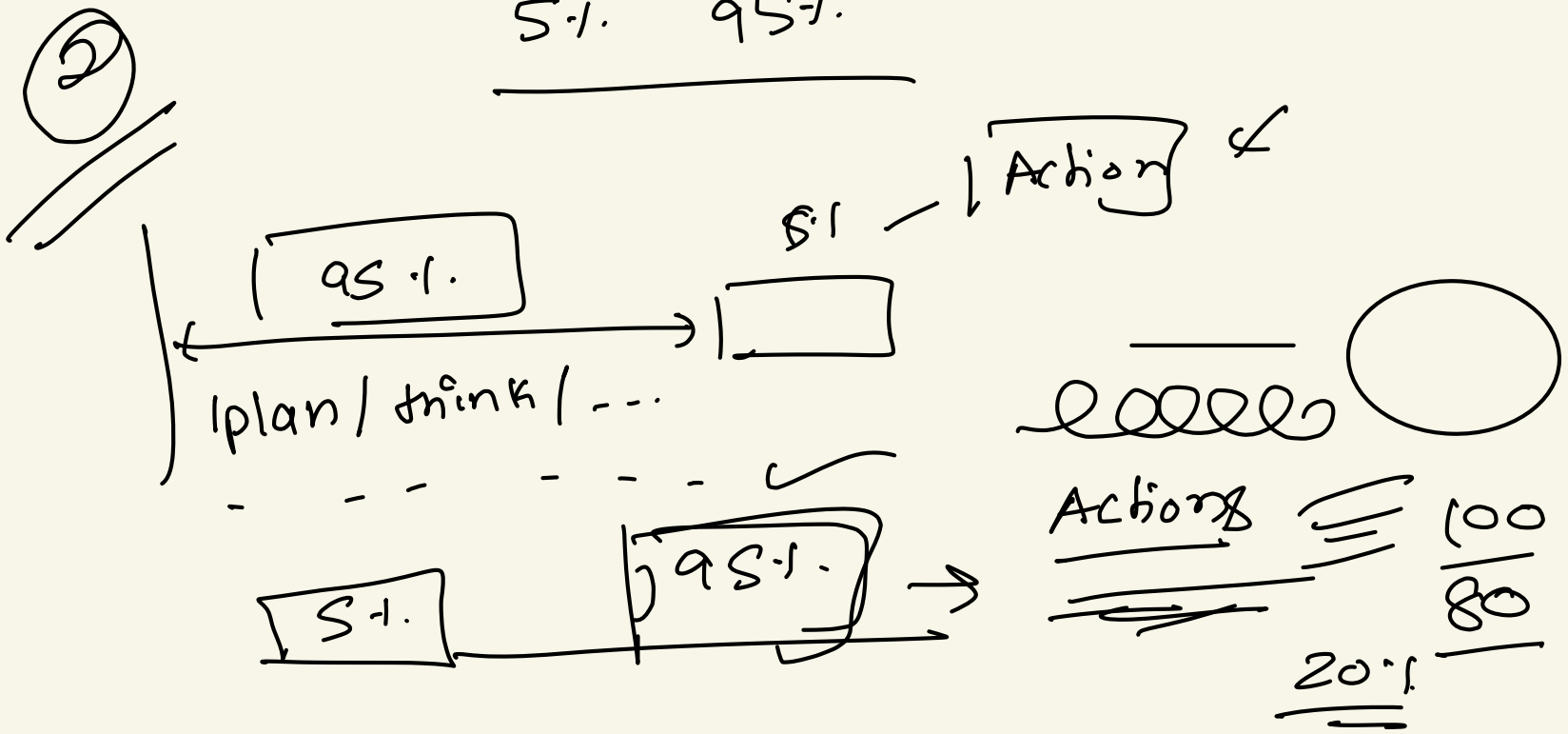
Focus on One thing

Limited = 100



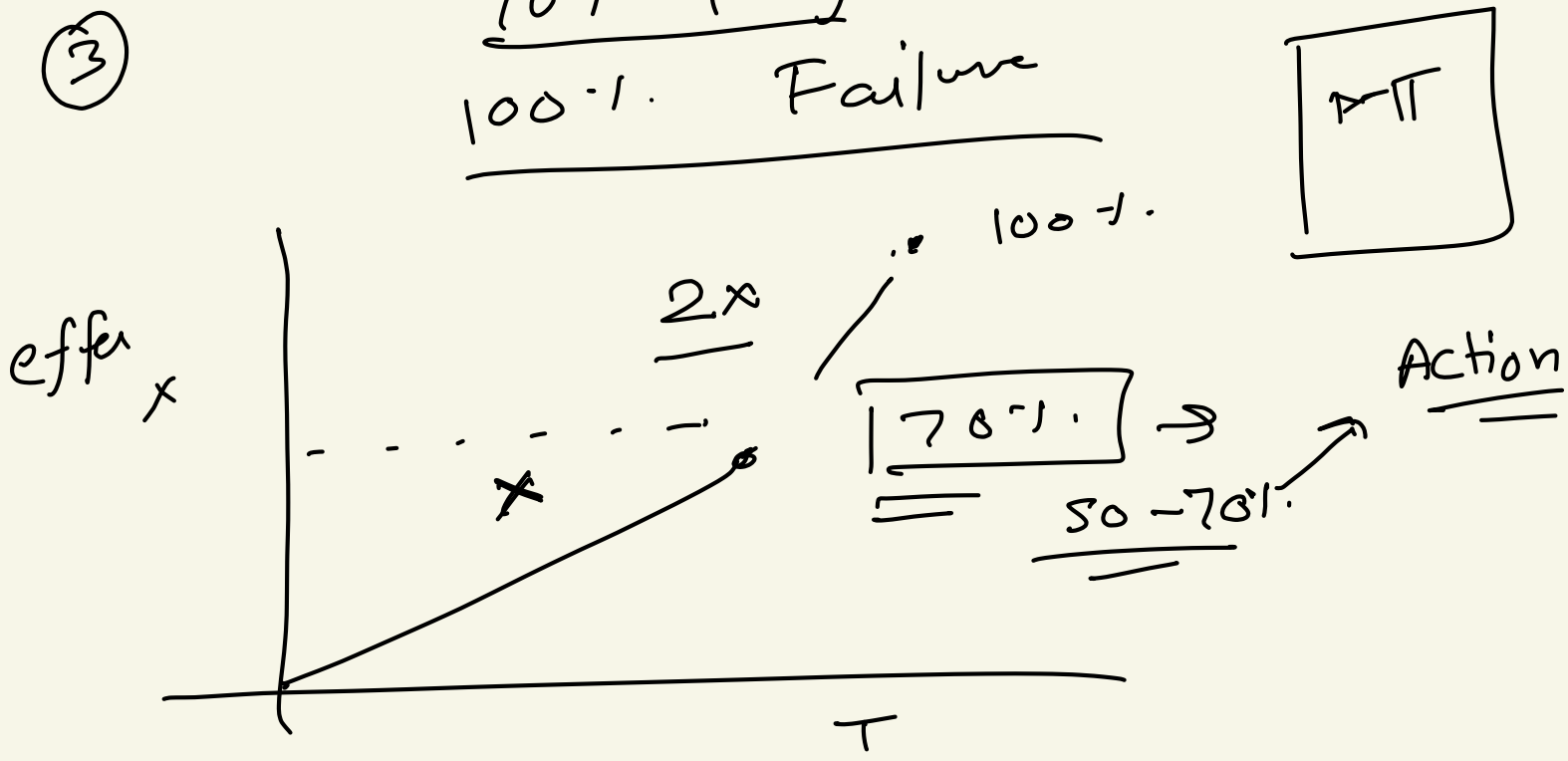


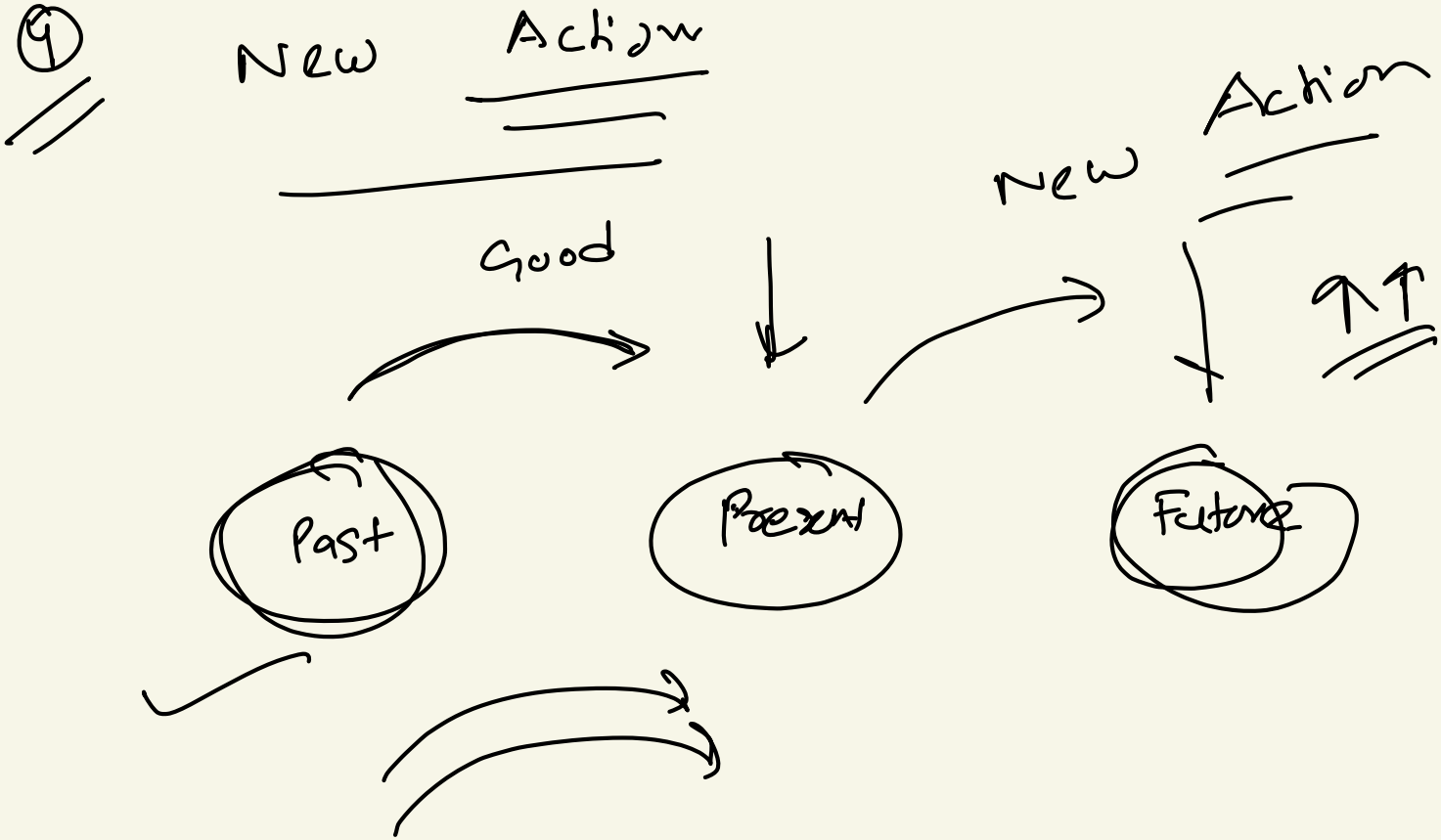

2
5·1. 95·1.
Action
5·1
95·1.
plan / think / ...
5·1.
95·1.
Actions
100
80
20·1

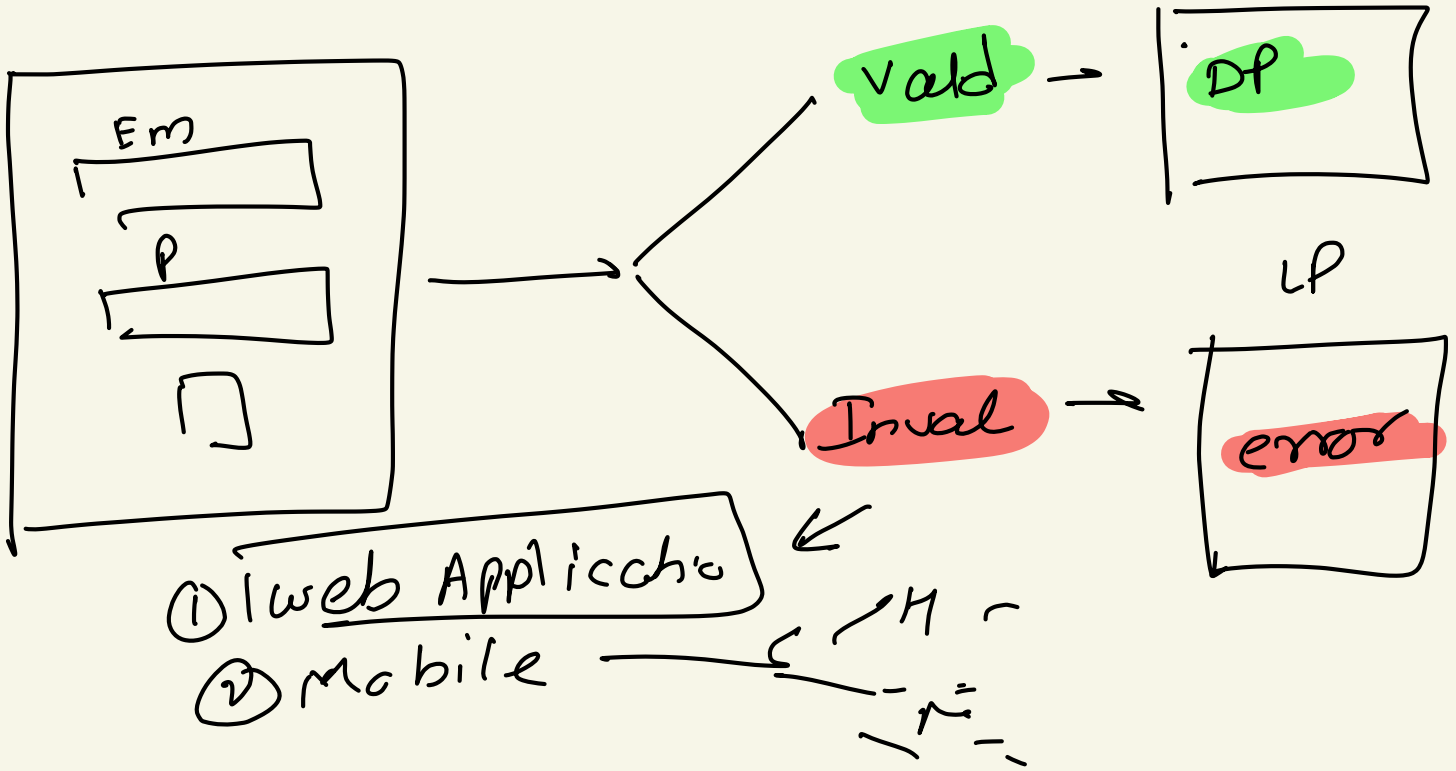
(3)

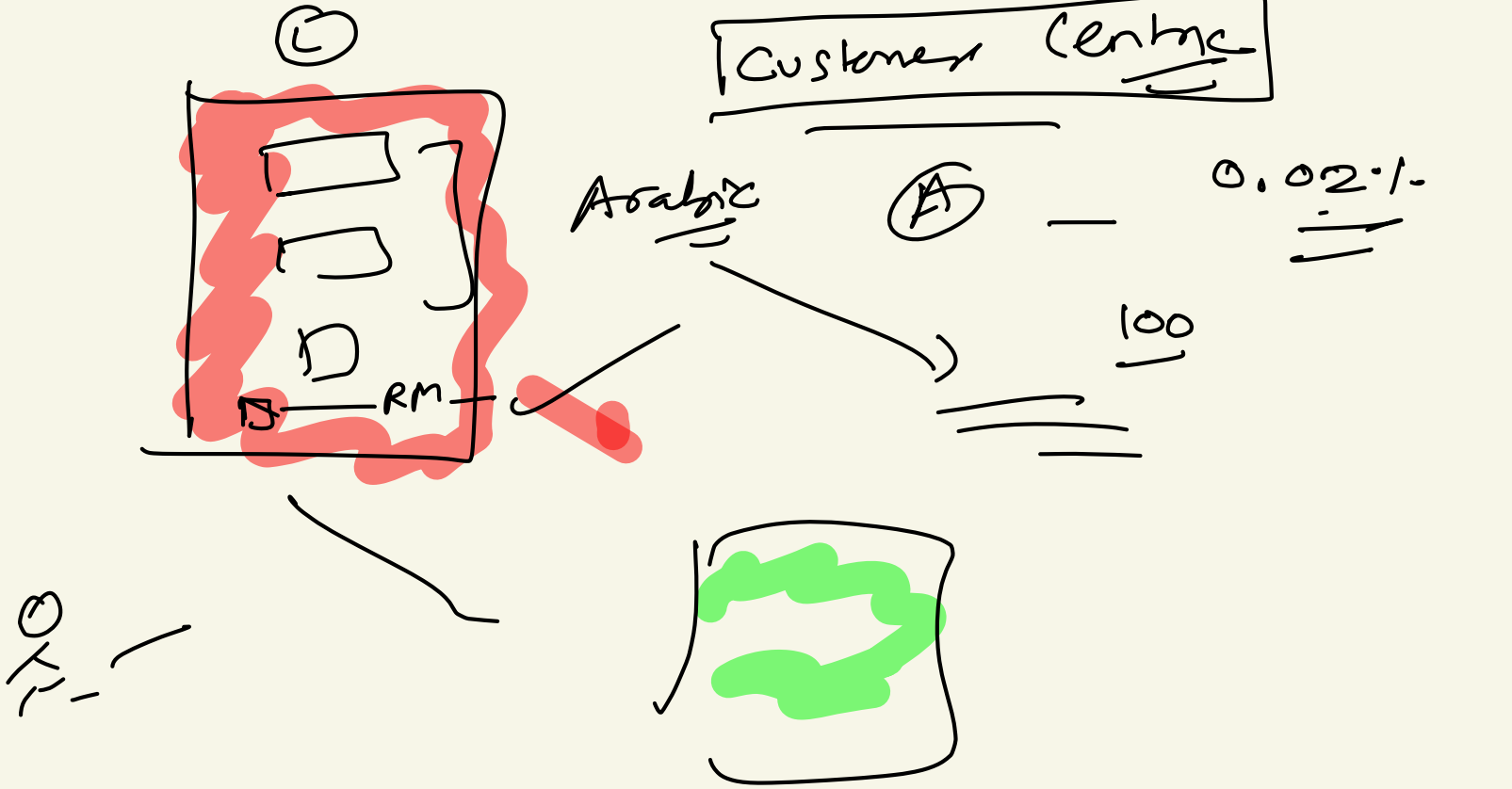
70-1. perfect

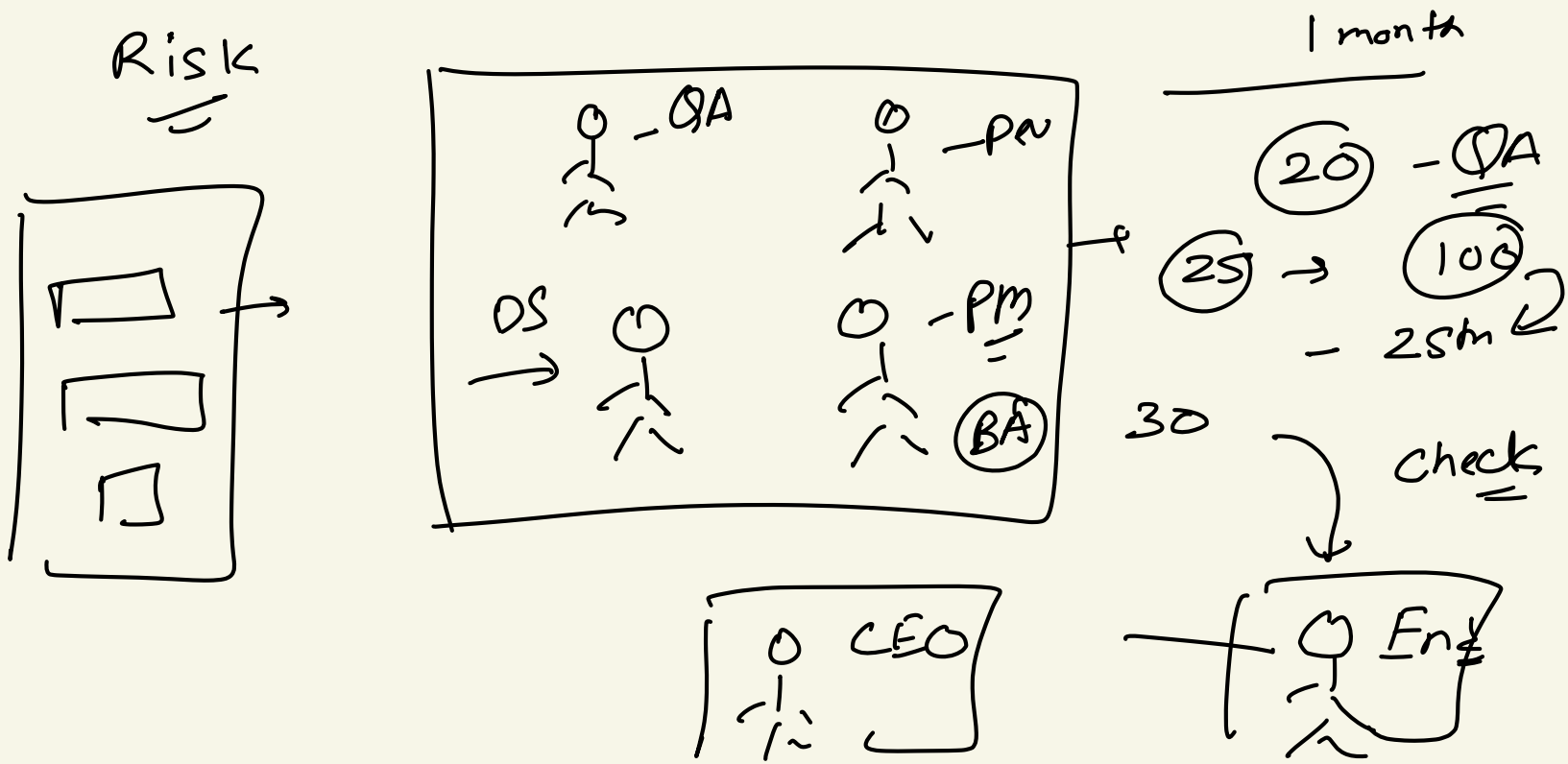
100-1. Failure




9
New
Action
Good
New
Action
Past
Present
Future

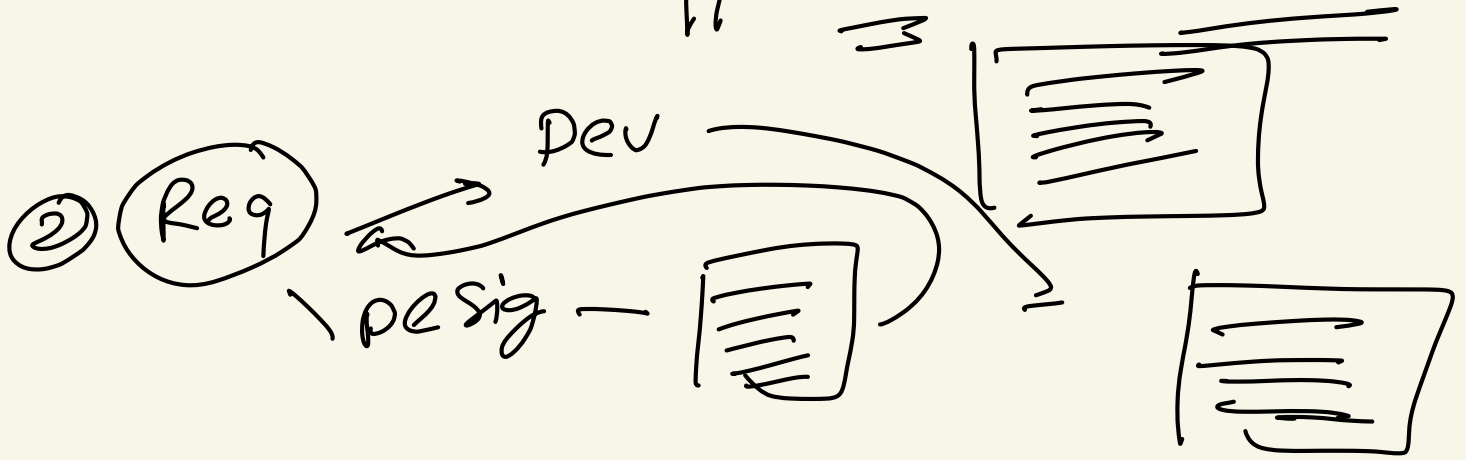

Em
P
Valid
DP
LP
Inval
error
① Web Applicchio
② Mobile

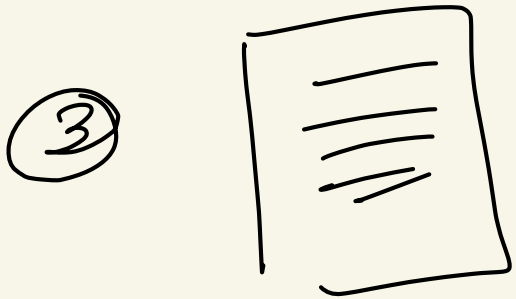

Customer Centric
Arabic
A
0.02%
100
RM


Risk
QA
PM
OS
PM
BA
1 month
20
QA
25
100
25m
30
check
CEO
Ent

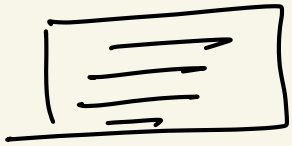
① BA / PM → talk to Customer

↳ Req / SRS (Soft Req Spec)





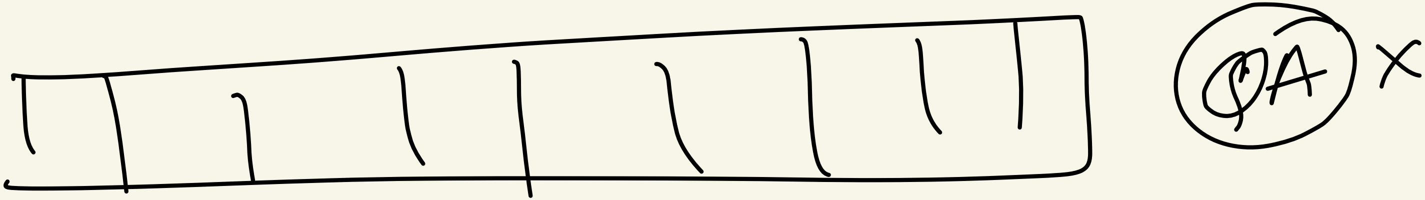
QA → Prevent Detect

Req →  — Tester] QA

→ Prevention + Detect

QC → Detect ← - Quality control

Defects



Metro'

(-1) check

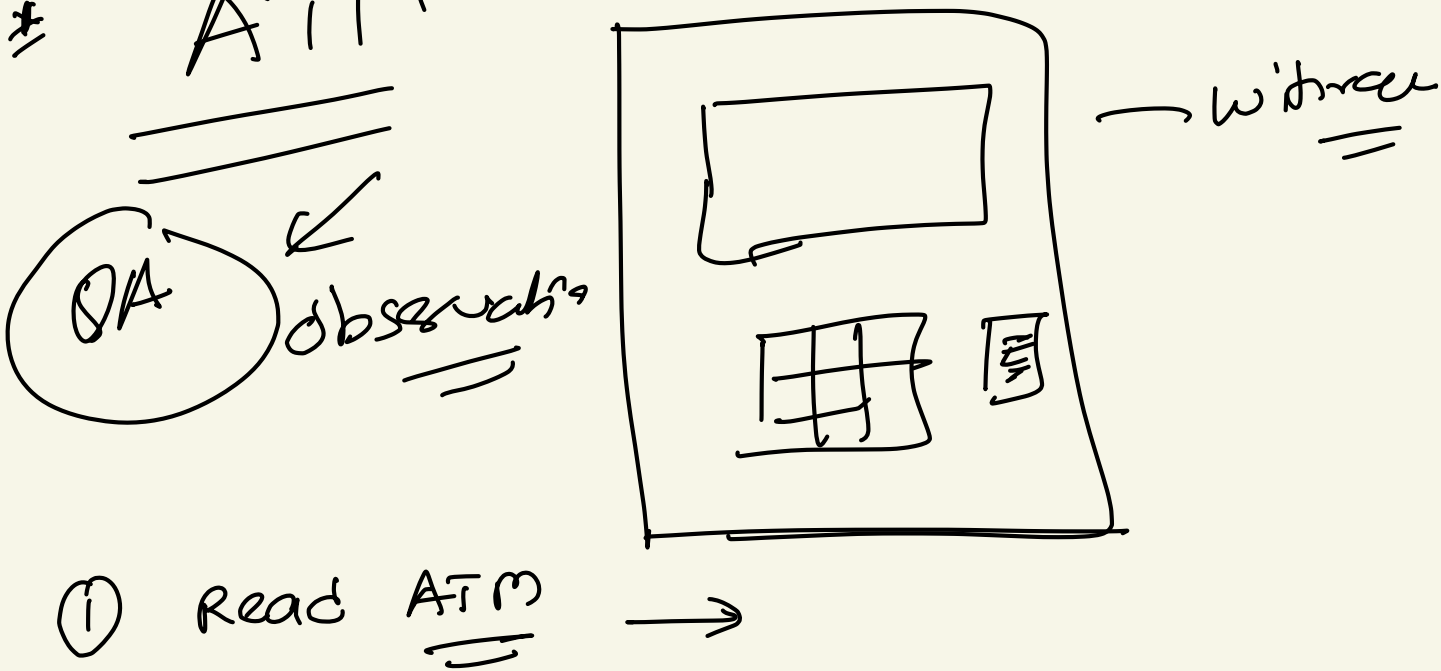
delete

QA — Prevention + Detect — QC

Read → ① Review
② Test plan
③ Test case

Execute
checklist
① TC
② - — —
③ ———
④ Bugs

* ATM



① Read ATM →

ER → Requirement

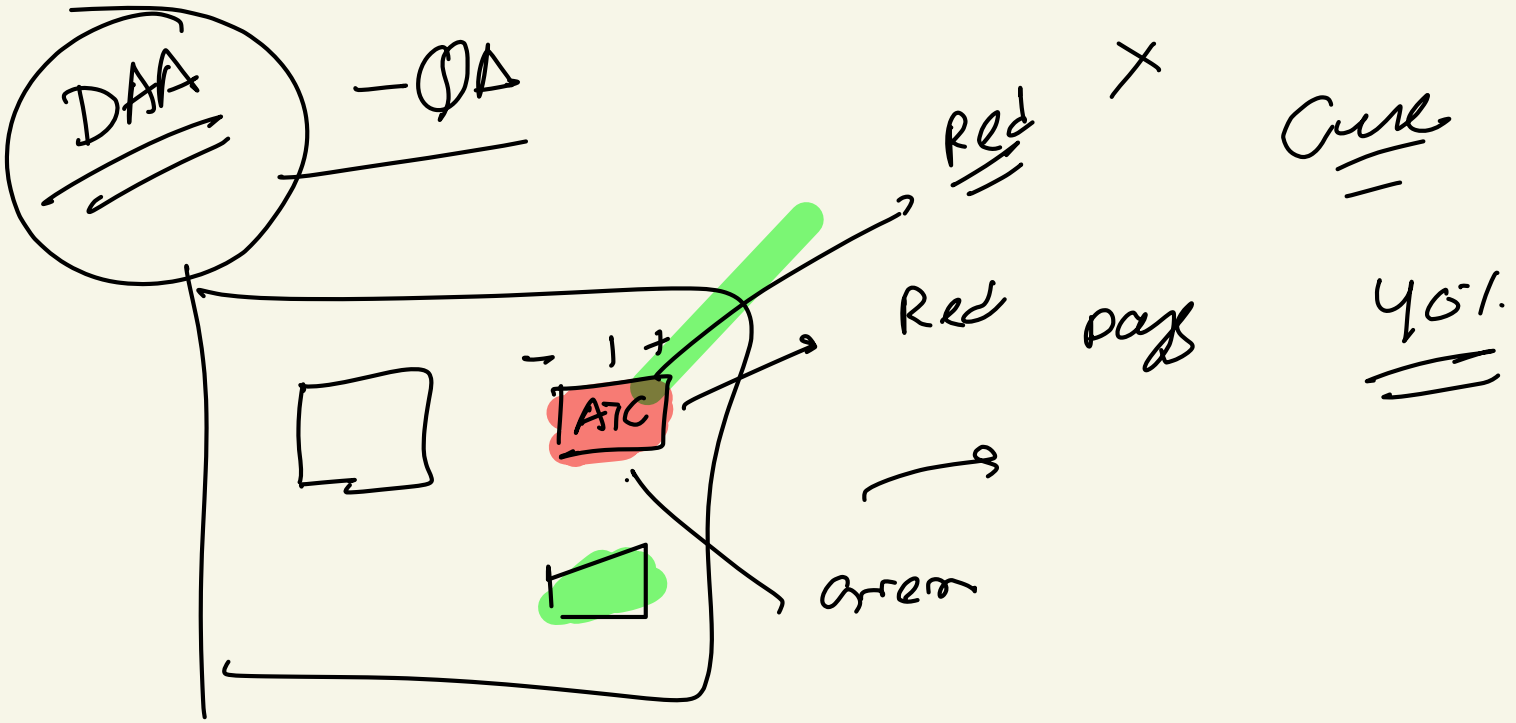
Swiss — IP

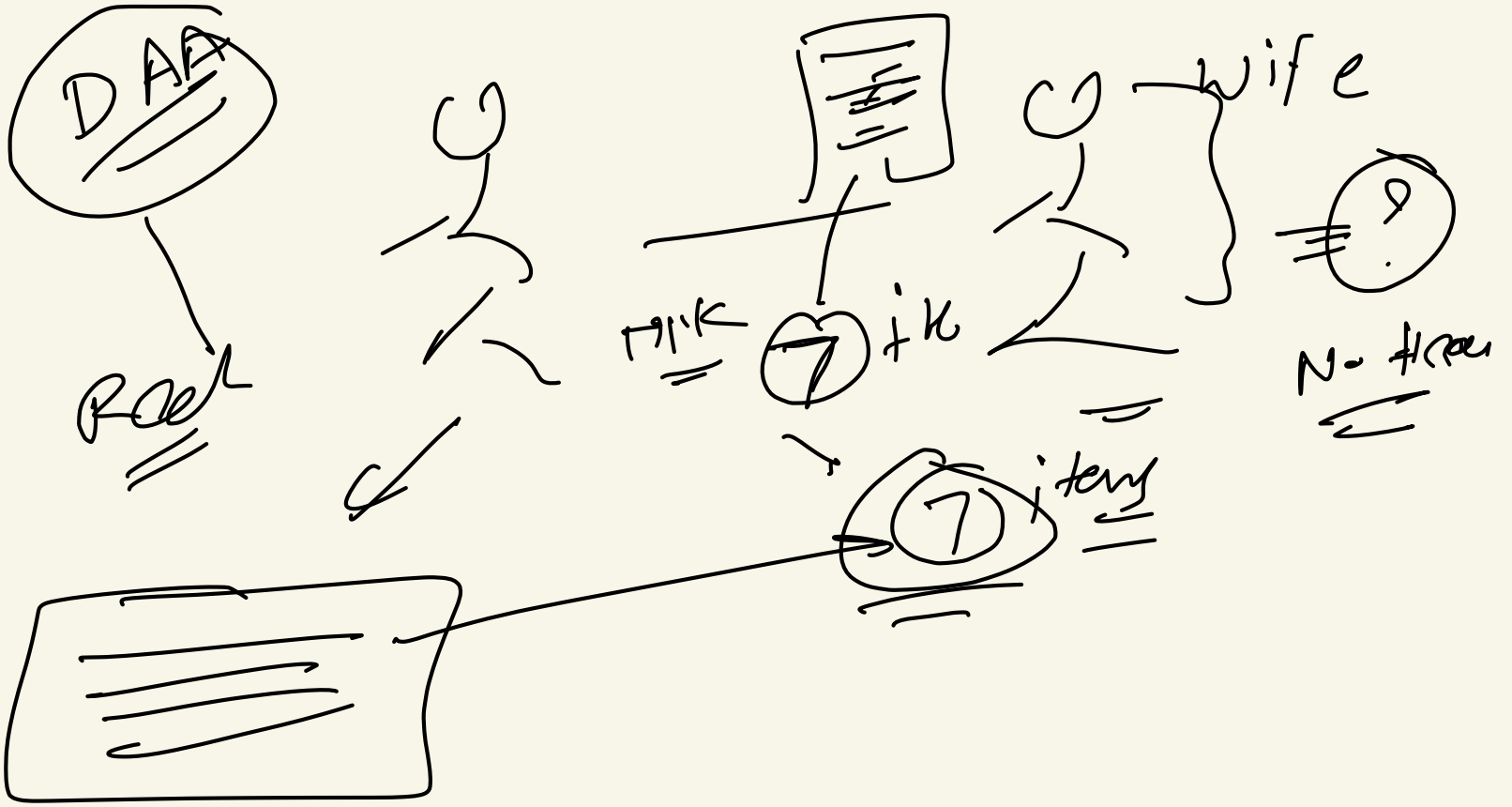
Stokes

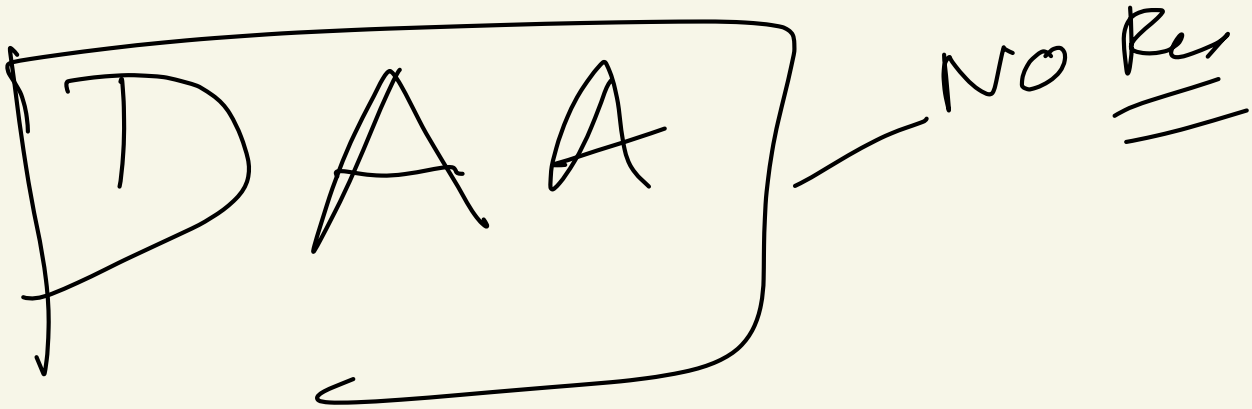
DAA

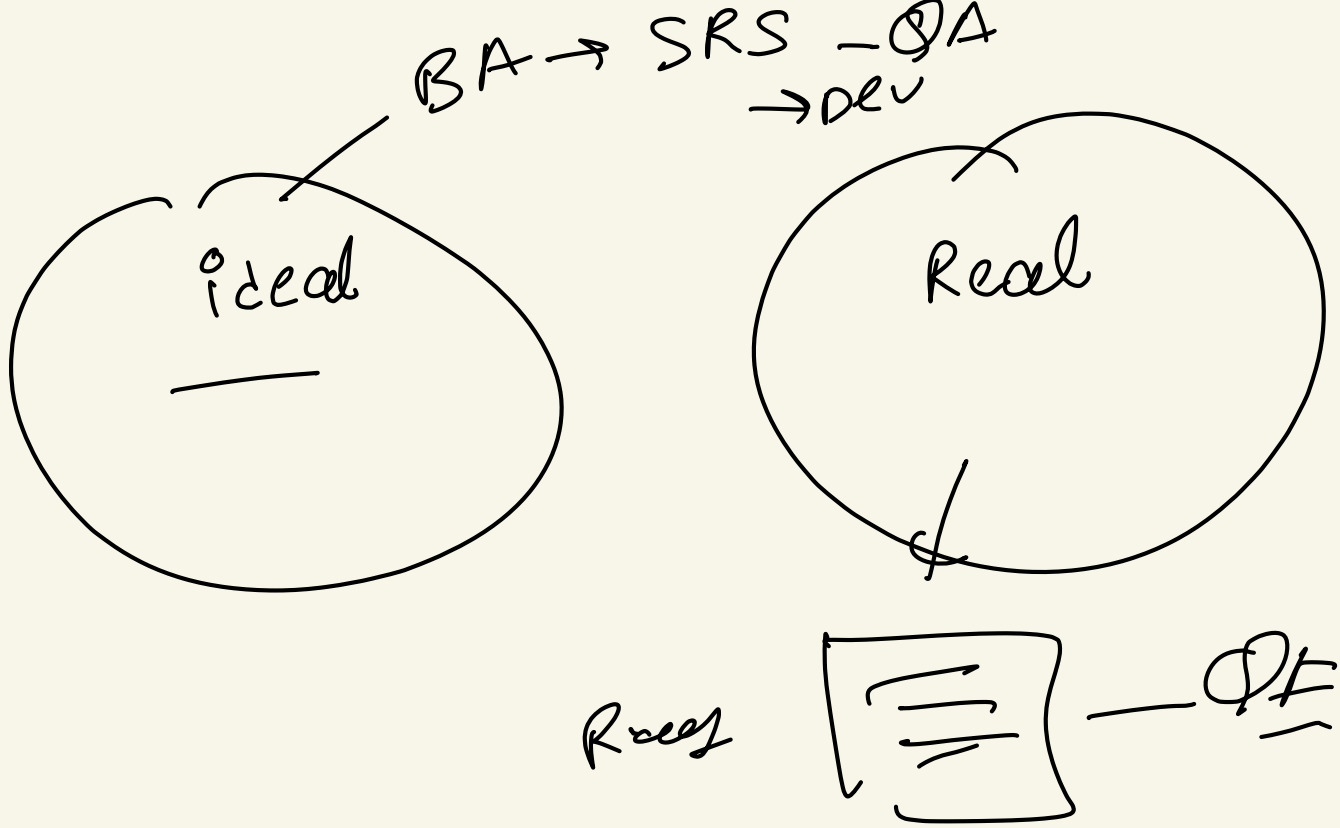
SA

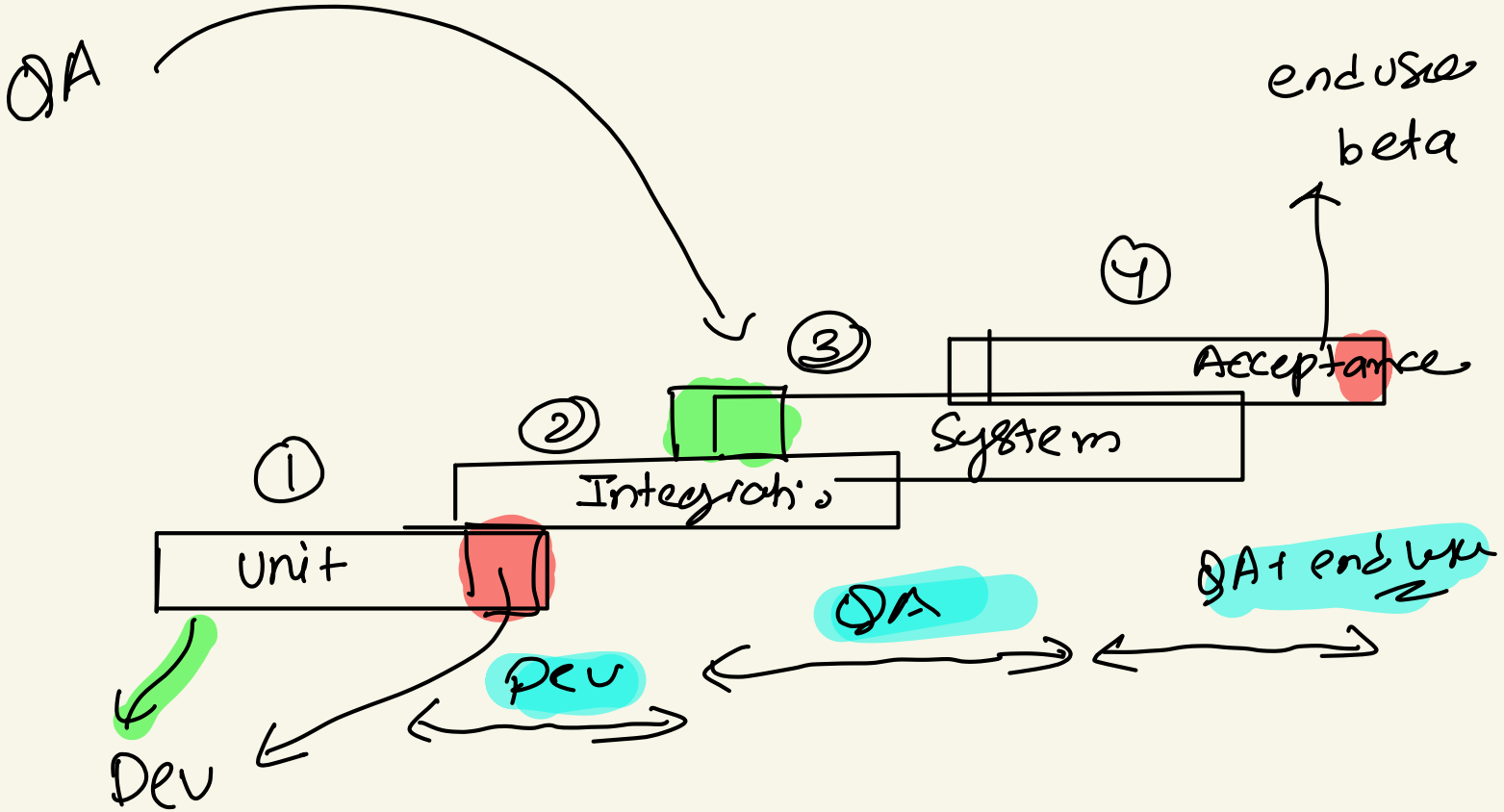
Don't Assume Anything

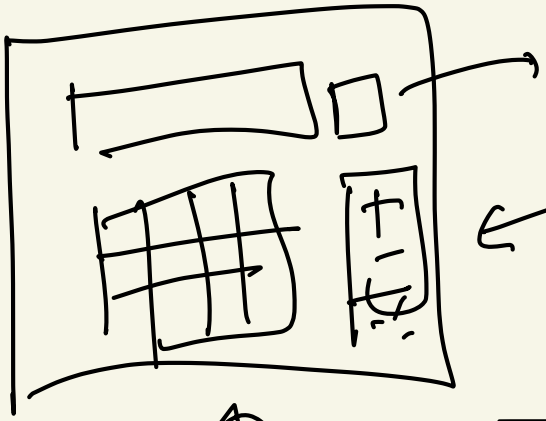

DAA
- 1 +
ATC
Red
Red
pays
Green


DAD
wife
?
milk
jik
jeans


DAA
NO Re


BA → SRS → QA
→ Dev
ideal
Real
Reqs
QE


QA
end user
beta
4
3
2
1
Acceptance
System
Integration
Unit
QA + end user
QA
Dev
Dev



unit = 2+2 (Sum)
=
divide
multp

unit Sum

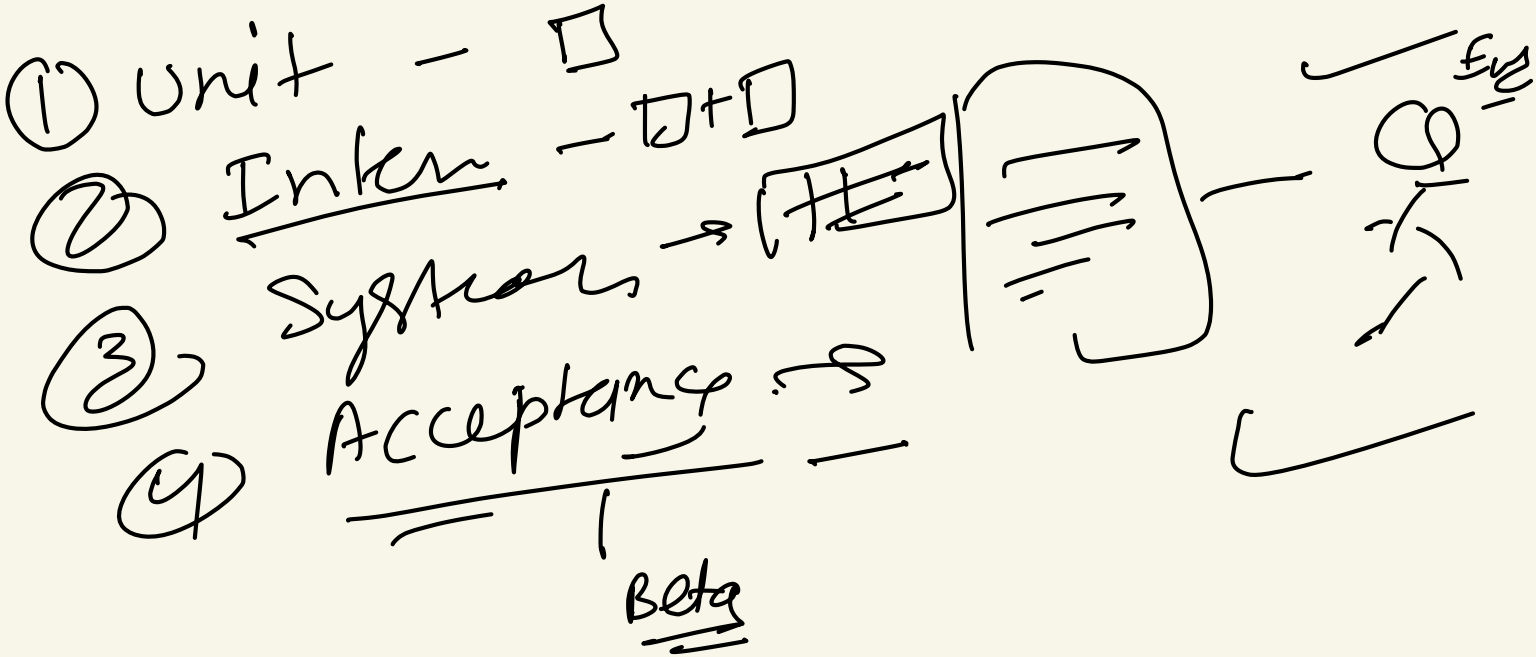
(1)

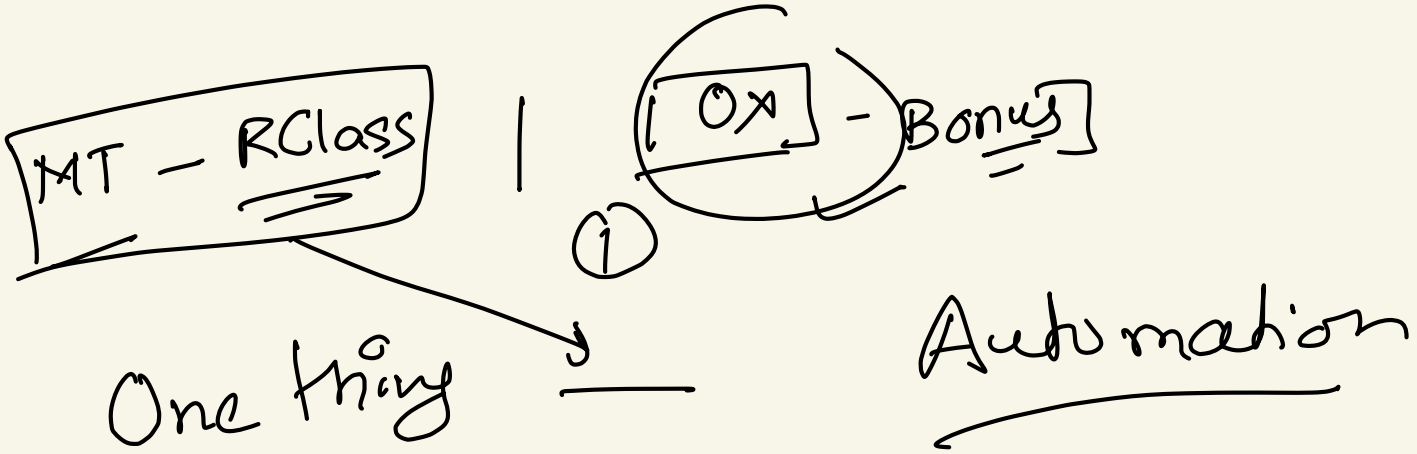
(2)

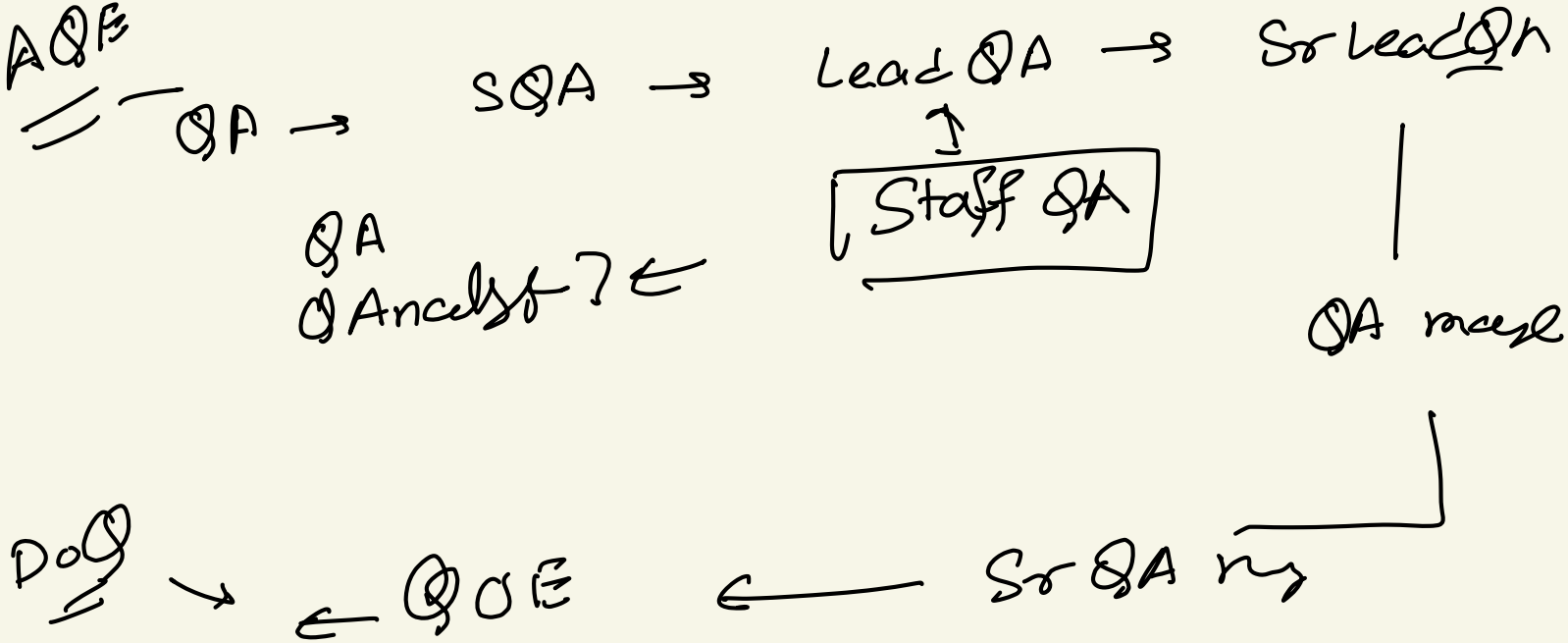
Sunt duke — Insp

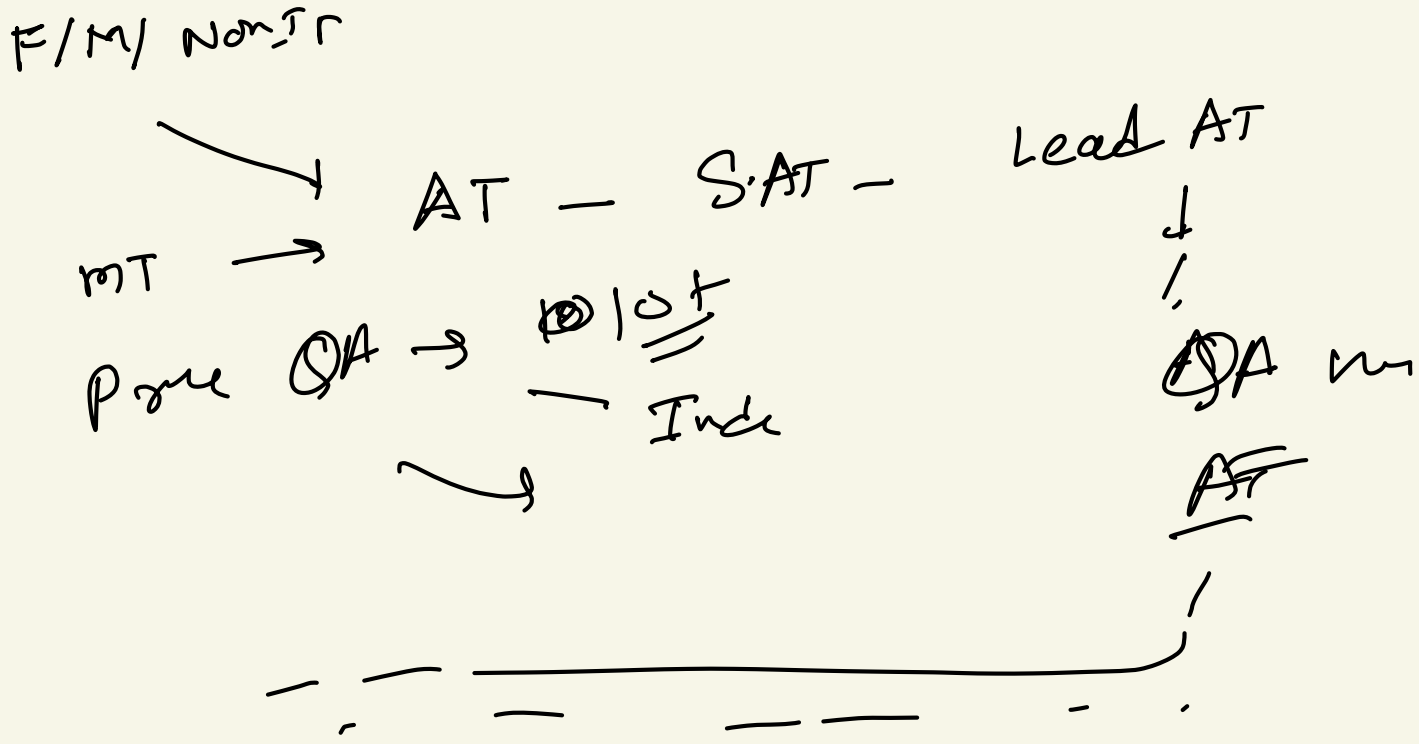
System — QA

Finde — Accept


1) Unit
2) Inter
3) System
4) Acceptance
Beta
End


MT - RClass
OA - Bonus
One thing
Automation


AQE
QA
SQA
Lead QA
Sr Lead QA
Staff QA
QA
QA Analyst
QA manager
DoQ
QOE
Sr QA mg


F/M/ Nonsr
AT — S.AT —
Lead AT
mT
Pure OA →
Incr
AT

① Function /v Non

② Levels of testing

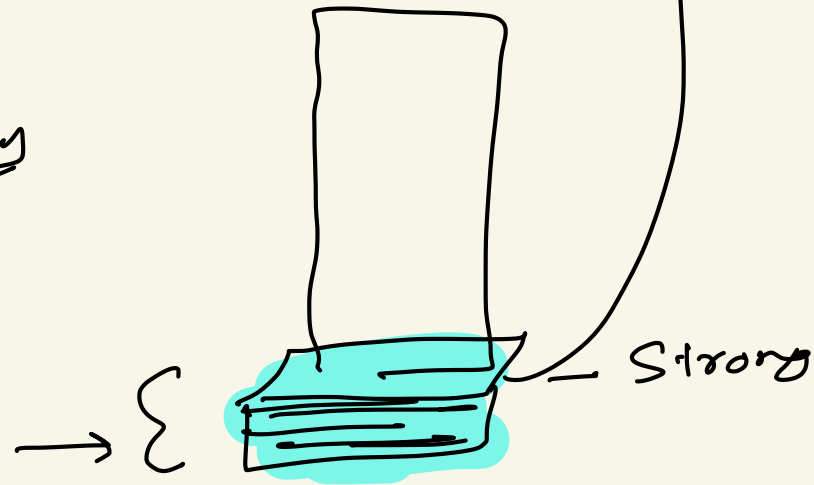
③ Testing pyramid

= ?

MTRC

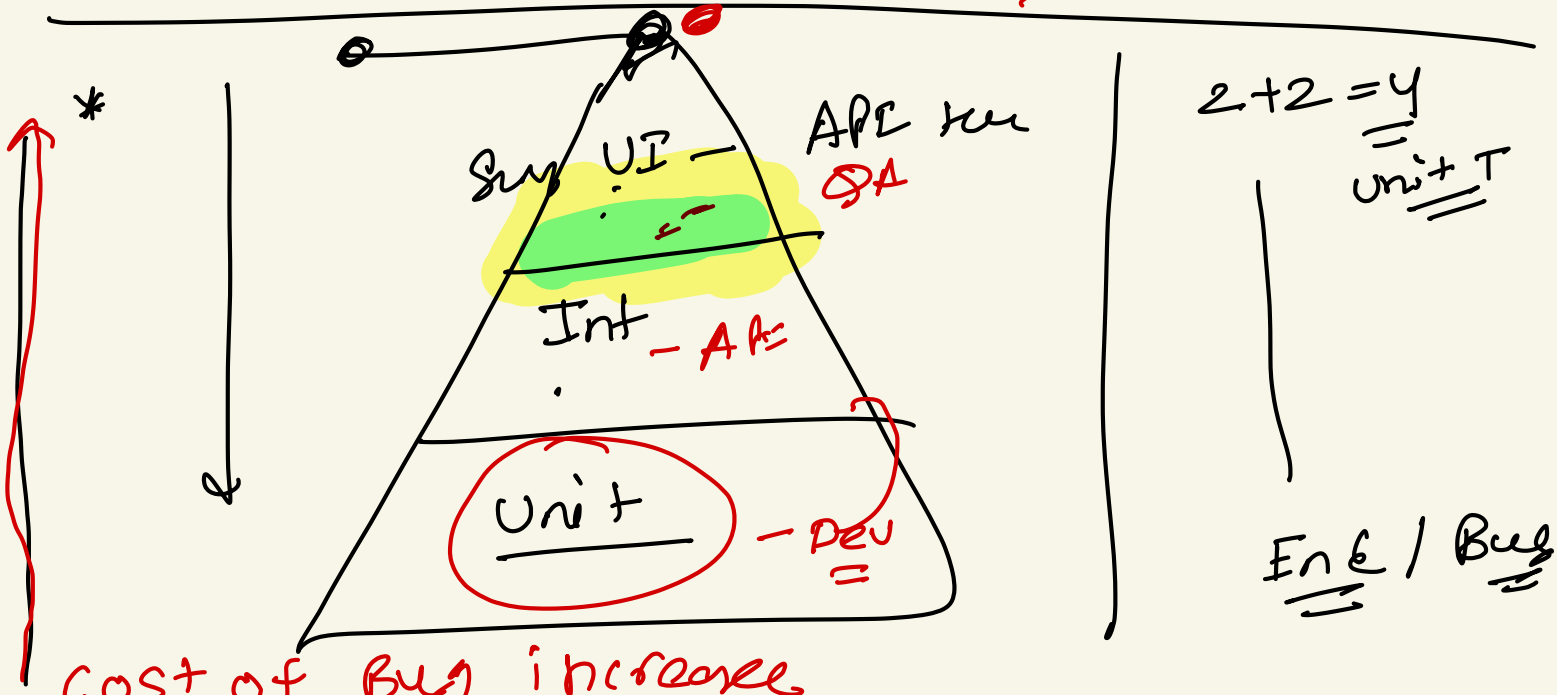
AT

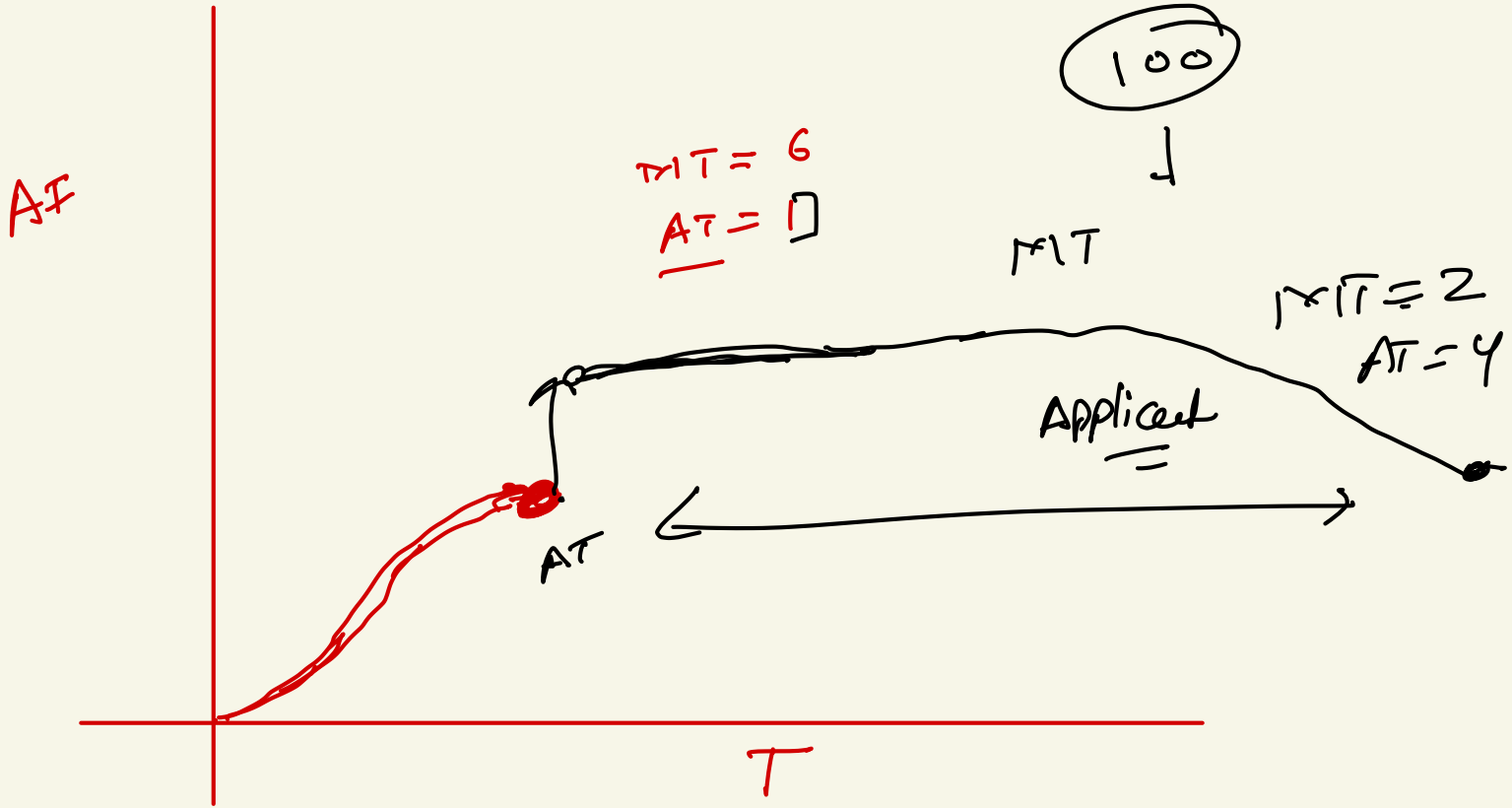
10 Story

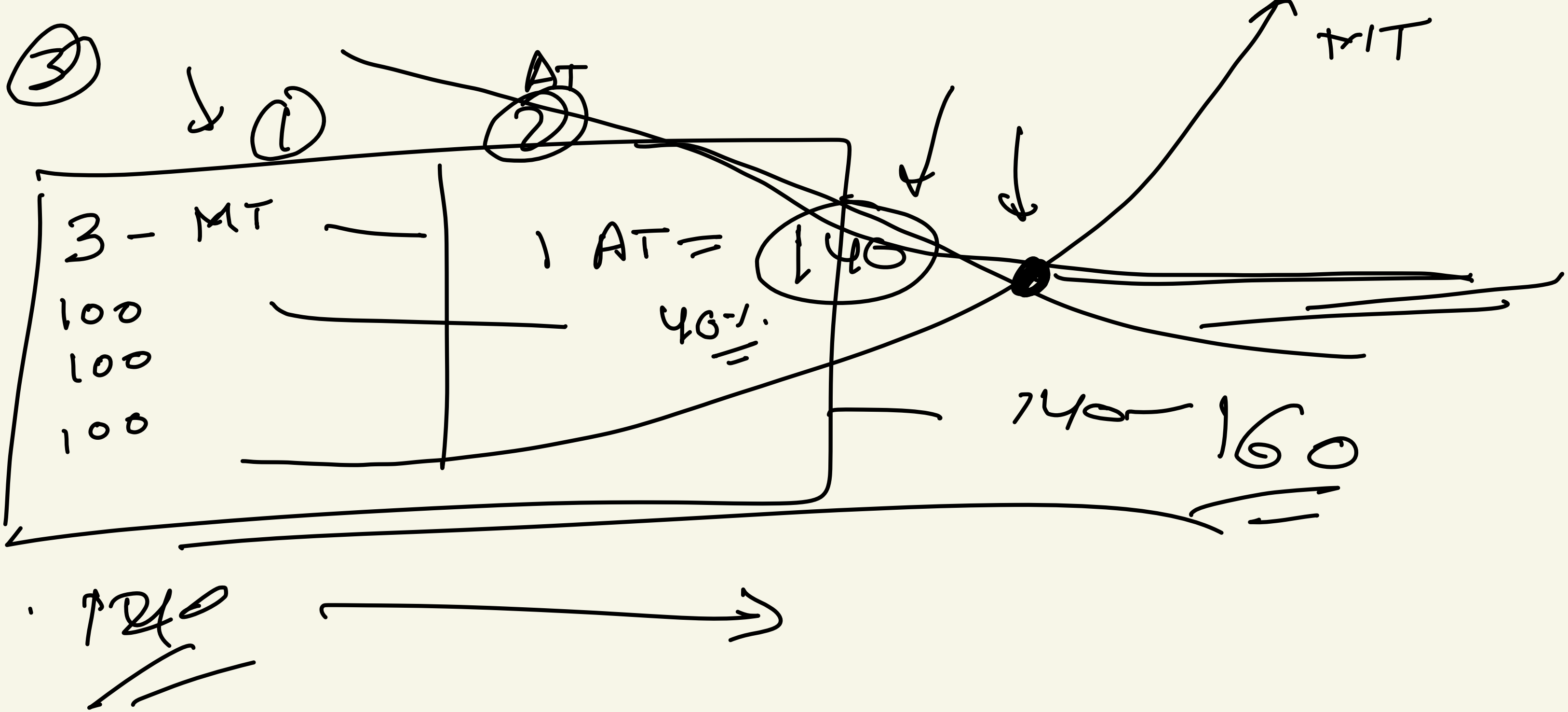


MTP — 100% clarity

end — :C




AI
100
MIT = 6
AT = 10
MIT
MIT = 2
AT = 4
Applicat
AT
T

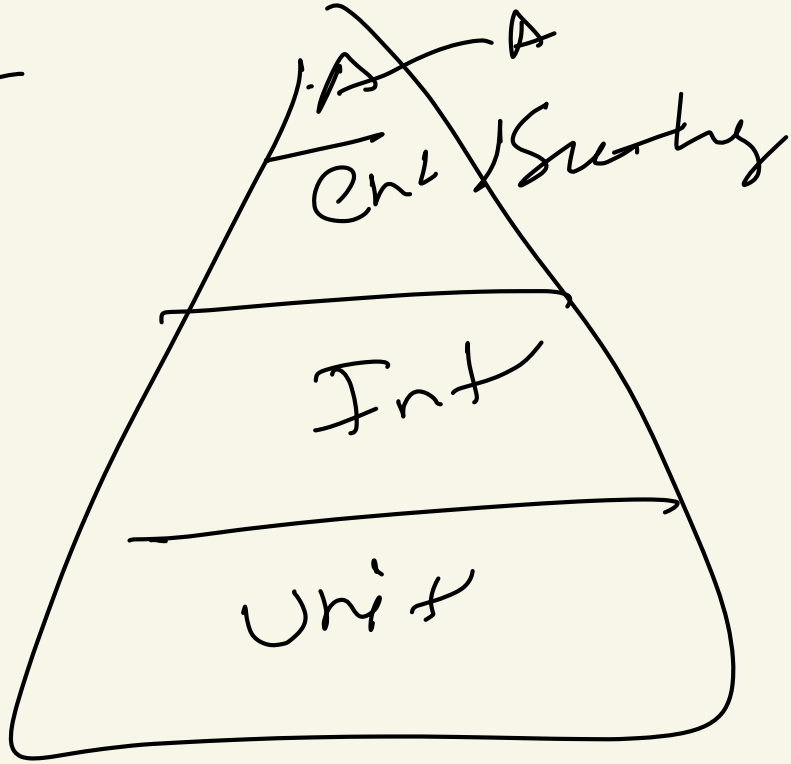


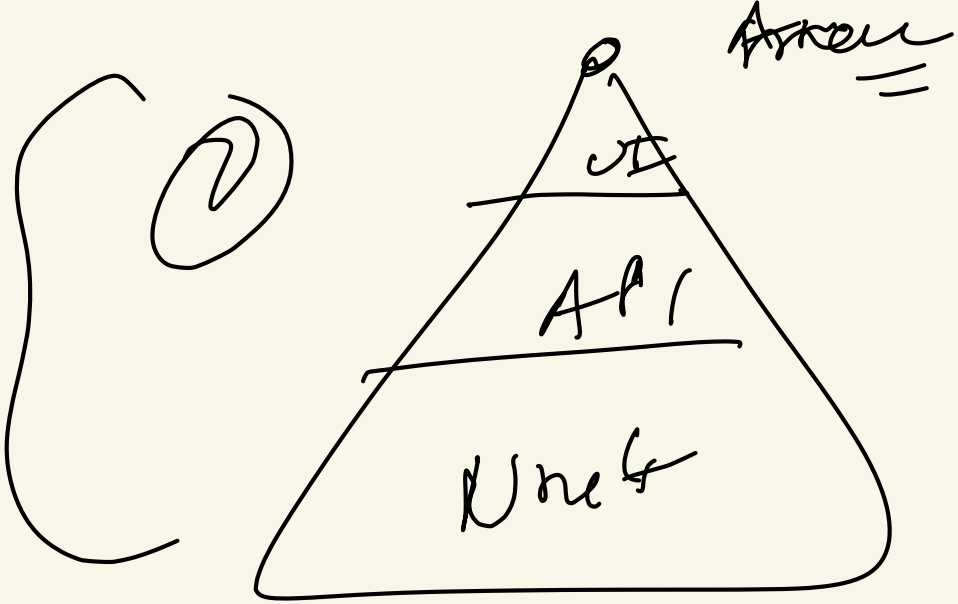
3
1
2
AT
MT
3 - MT
100
100
100
1 AT = 140
40%
140 - 160
140

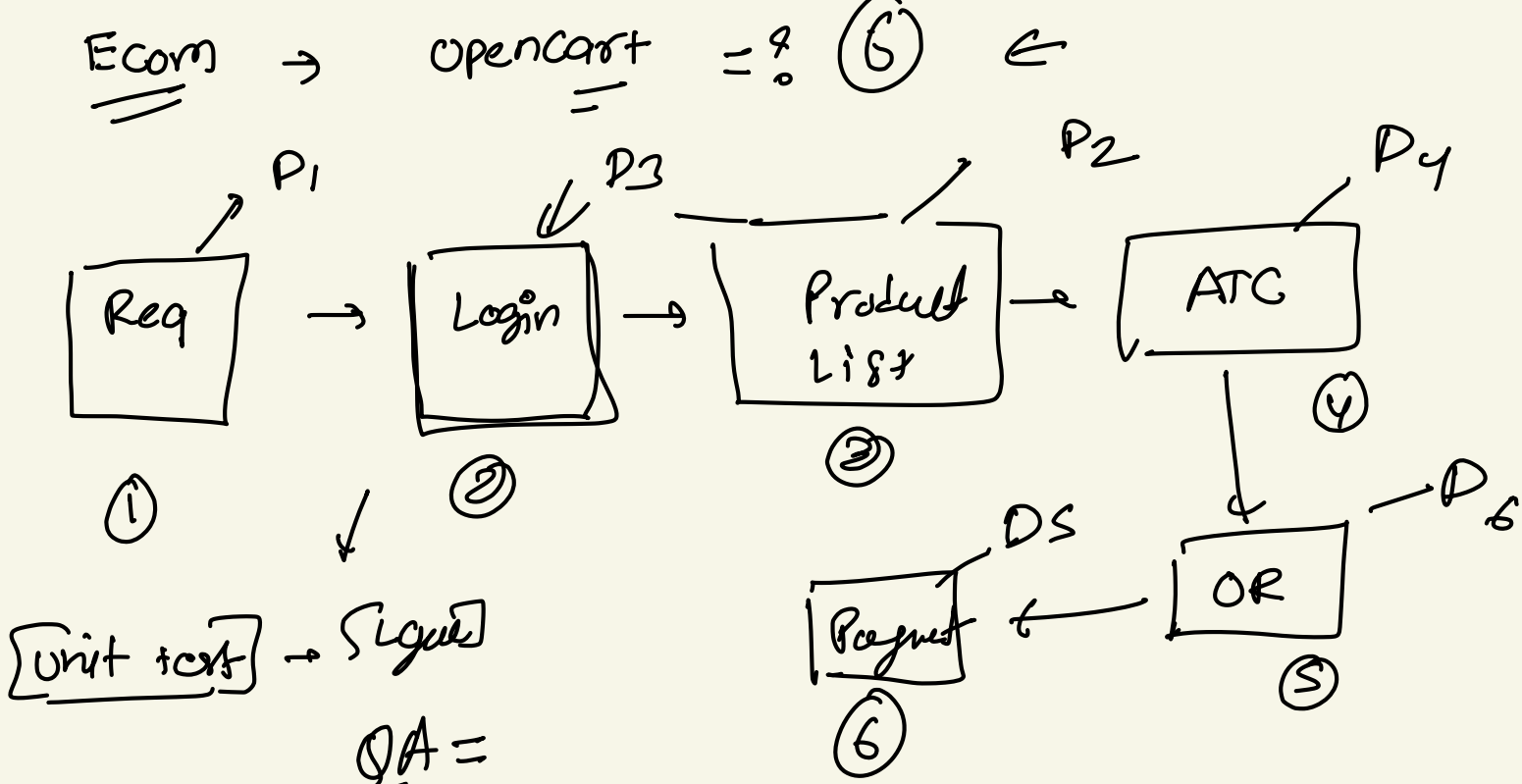
Testing

Automation

UF

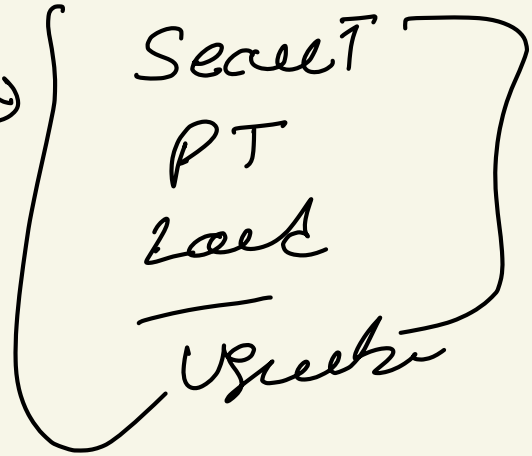



UI
API
Unix


Ecom → opencart =? (6) ←
P1
P3
P2
P4
Req
Login
Product List
ATC
1
2
3
4
Unit test
Sanity
QA =
P5
P6
Request
OR
6
5

System → FT + NFT

Funcionales

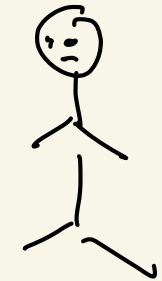


* → Automation fast

① → Automating Scripts

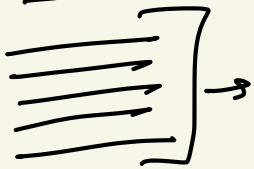
② Project -(STLC) →

↳ Req
Test plan
Test case
- - -
Bug

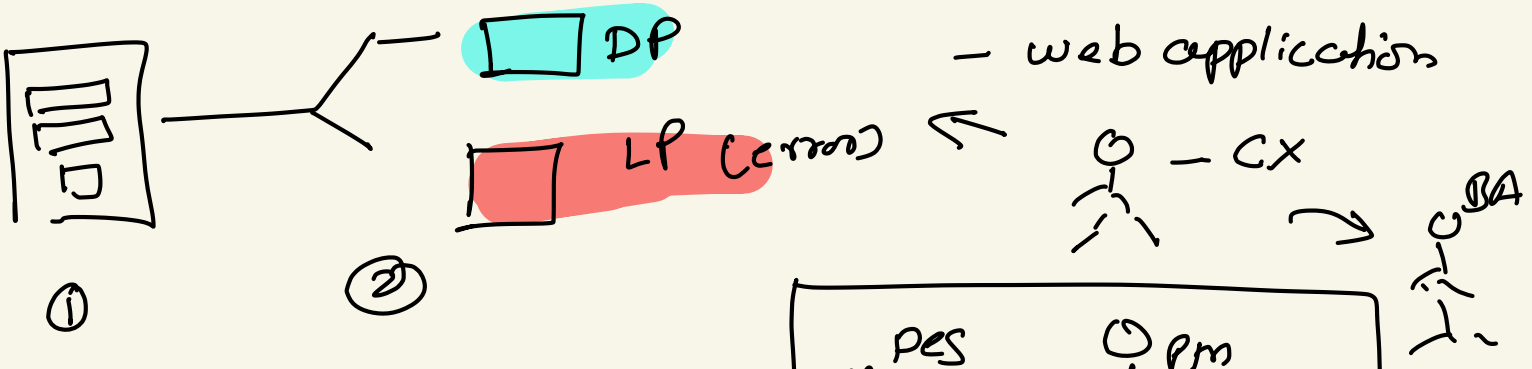


TC1= Login Succe
TC2= Login Failed
with Incorrect

1000



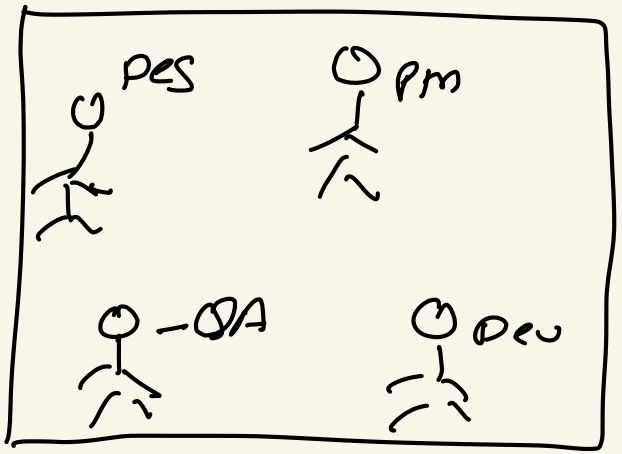
→ (20) → P0 / Repeative
stable



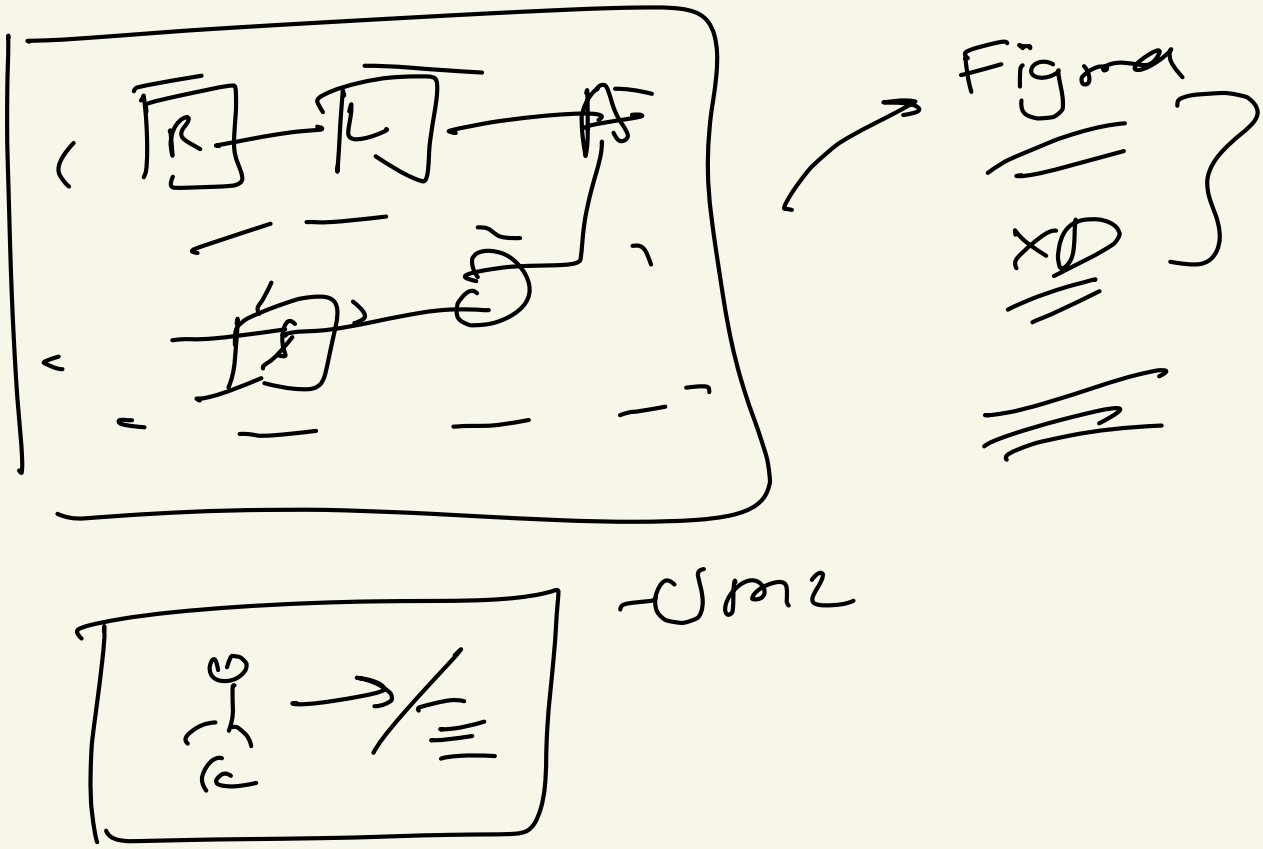
SDLC →

① Req - BA

② Plan



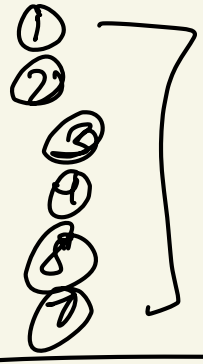
idea → Plan → Req → develop → test → Ensure


Figma
XD

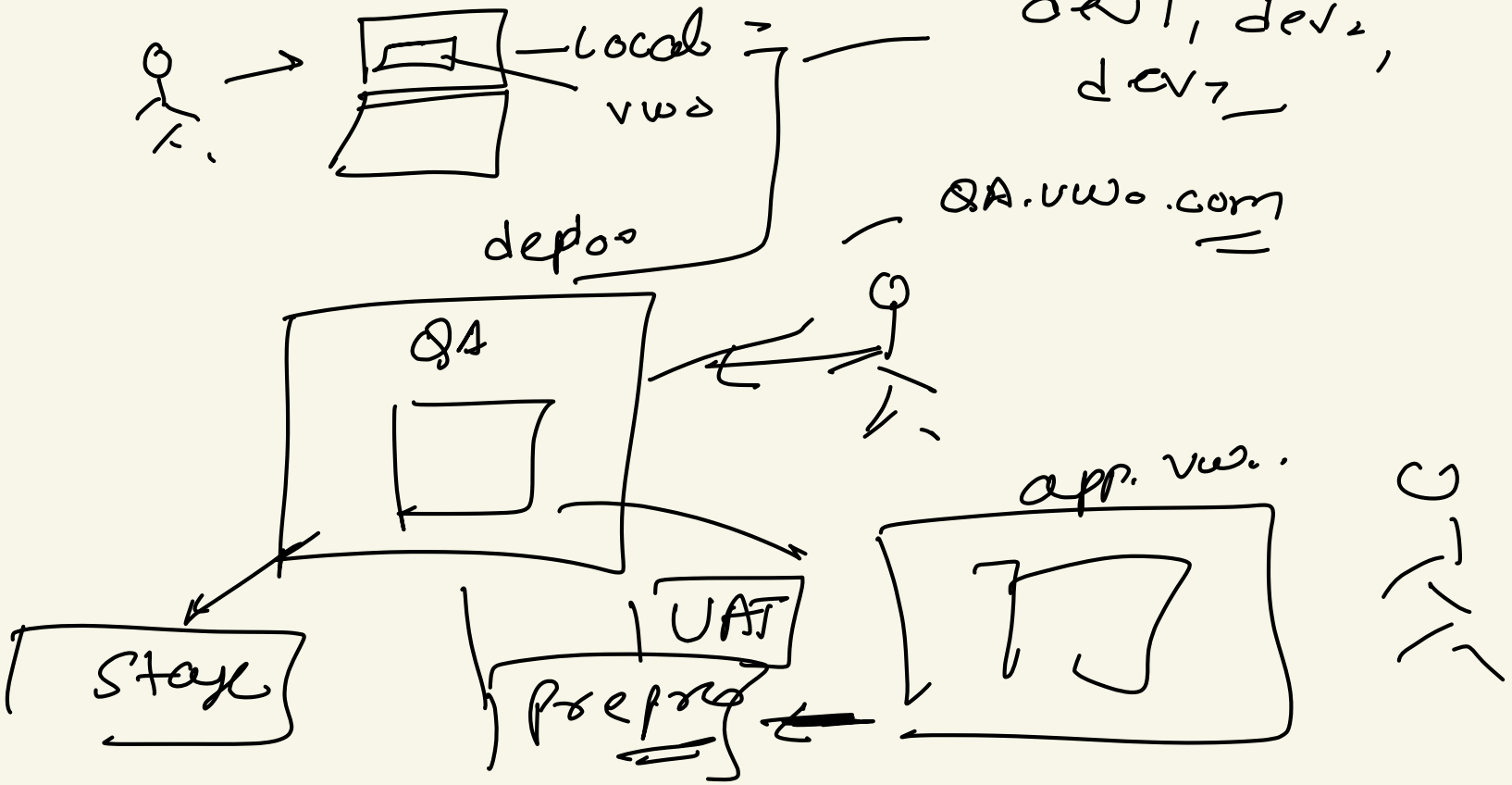
SDLC — Testing — STLC = ⚠

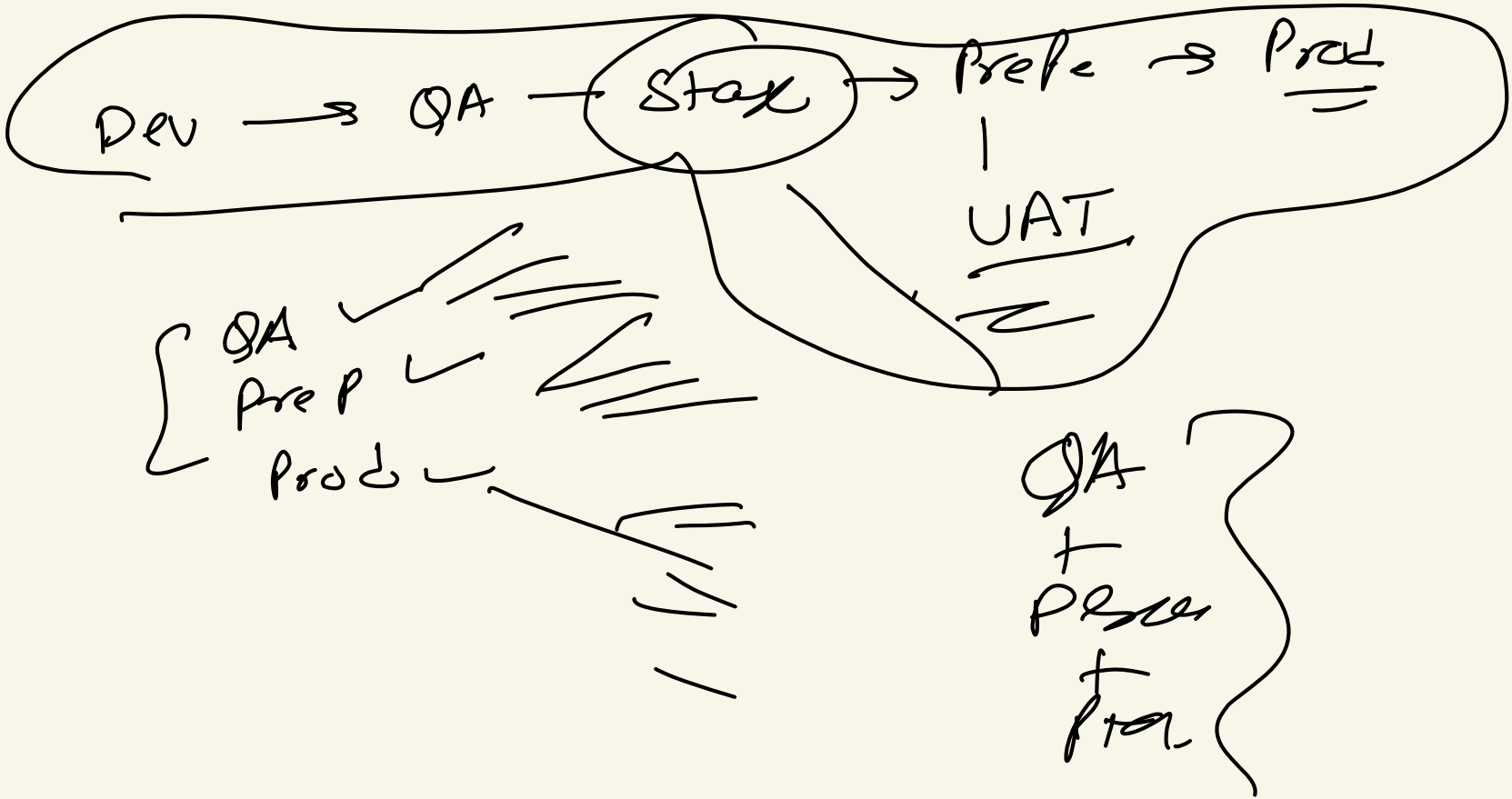
STLC] Test-Plan → Req

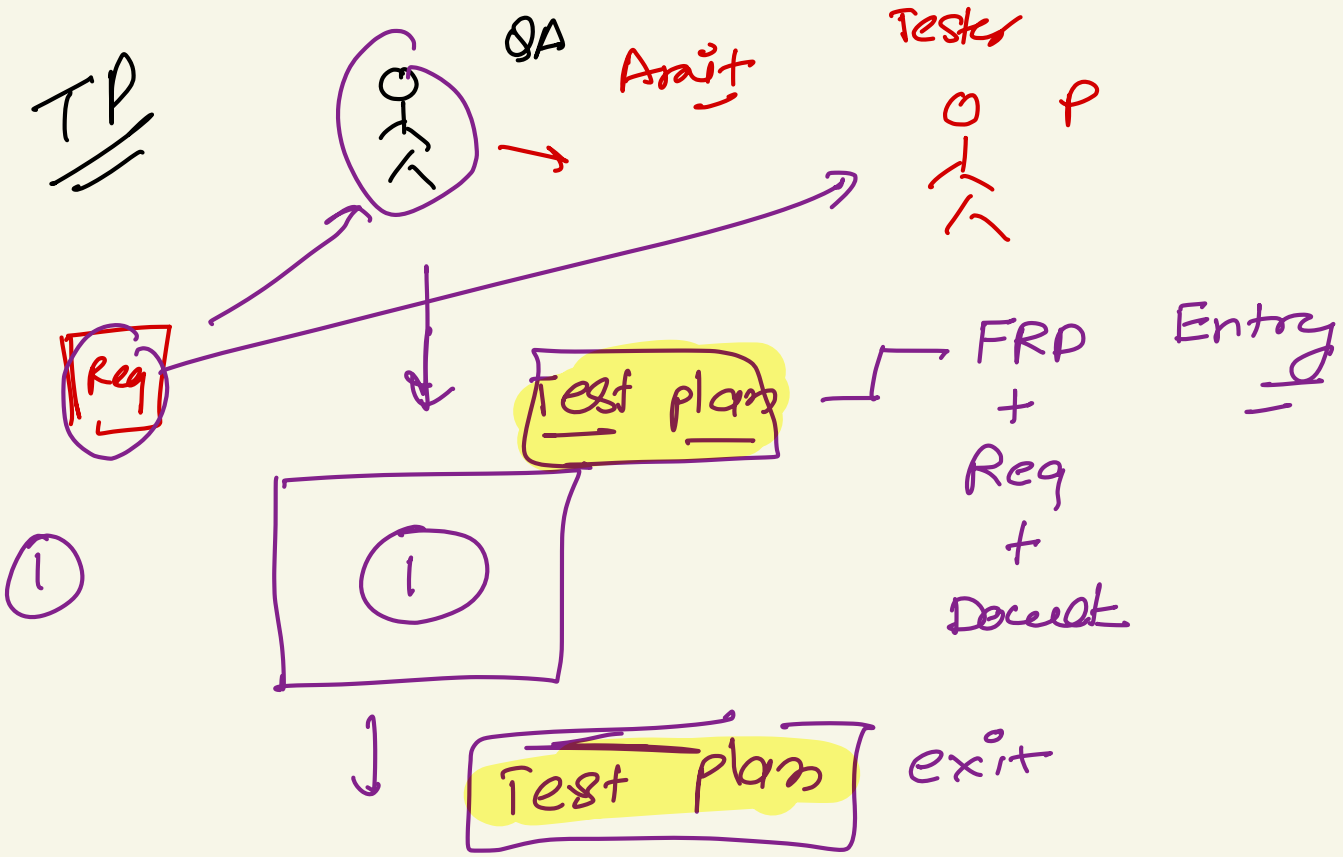
MI #1

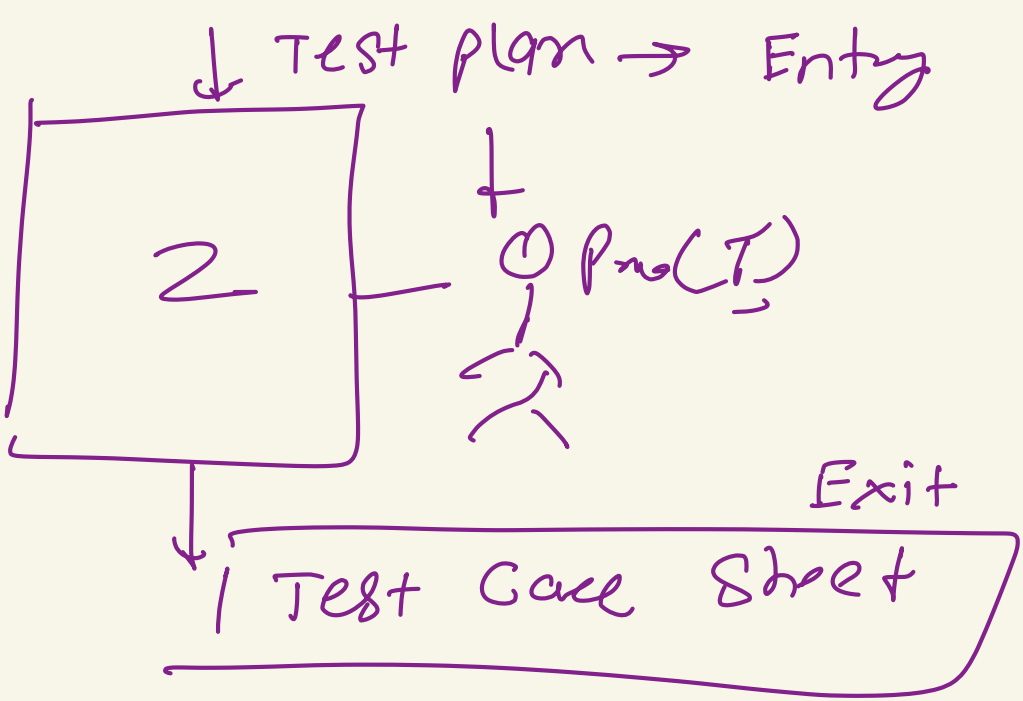


Login → Dash (app.vwo.com)


dev1, dev2, dev3
localhost
vwo
QA.vwo.com
deploy
QA
app.vwo.com
UAT
Stage
Prepro


Dev
QA
Stax
Prepe
Prod
UAT
QA
Prep
Prod
QA
+
Prep
+
Prod

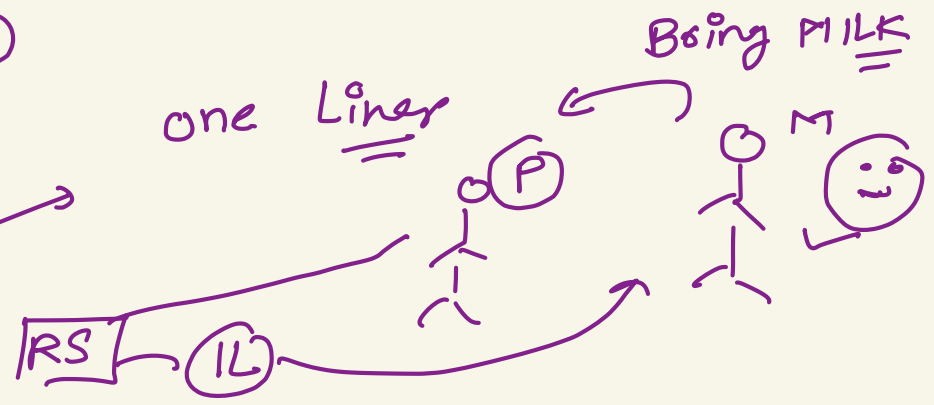

TP
QA
Analyst
Tester
P
Req
Test plan
FRD
+
Req
+
Docum
Entry
1
1
Test plan
exit


Test plan → Entry
2
Proc(T)
Exit
Test Case Sheet

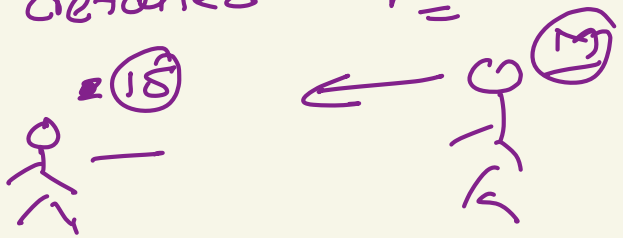
VWO - com(VWO)

Test Scenario → one Liner

Bring 1L Milk



Test Case → detailed steps

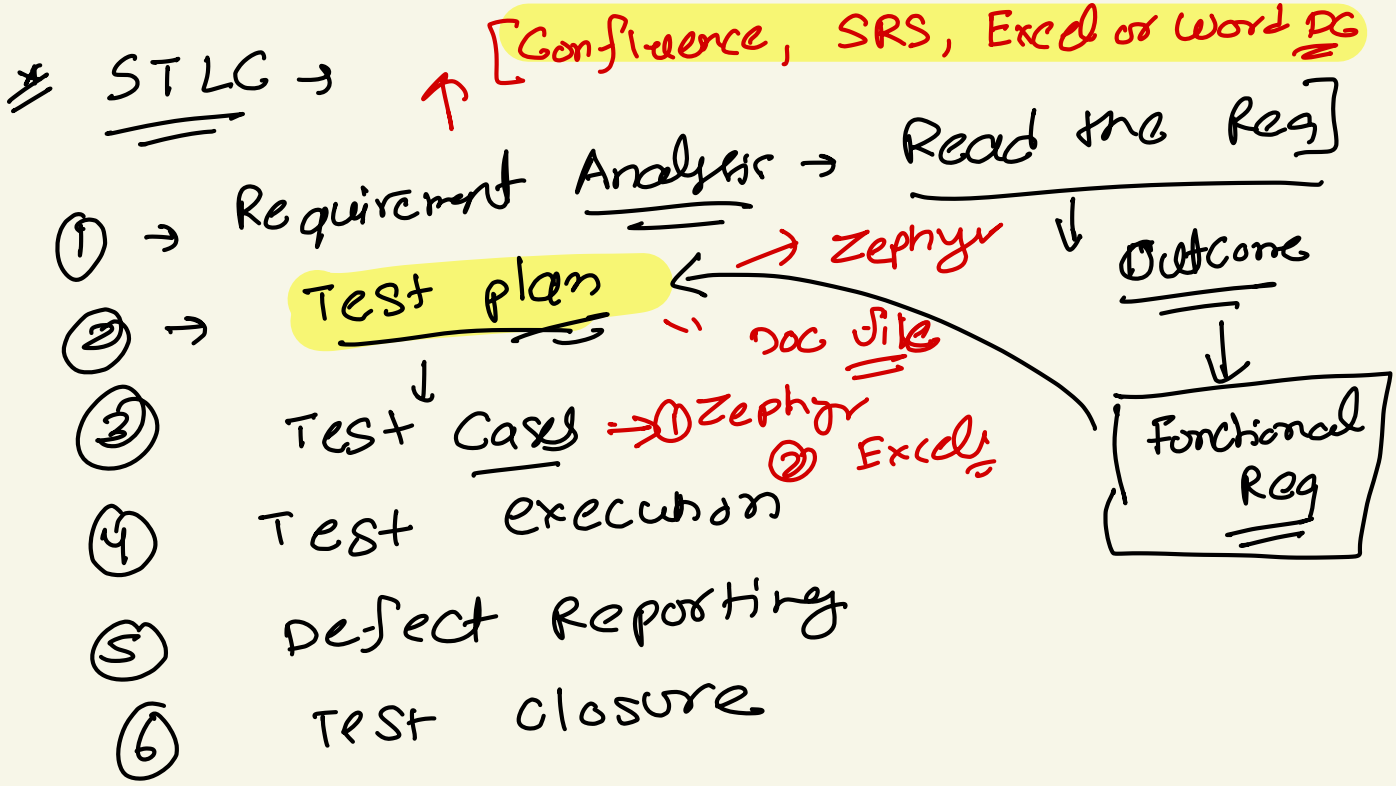


① go to Blue R

② ask for ML

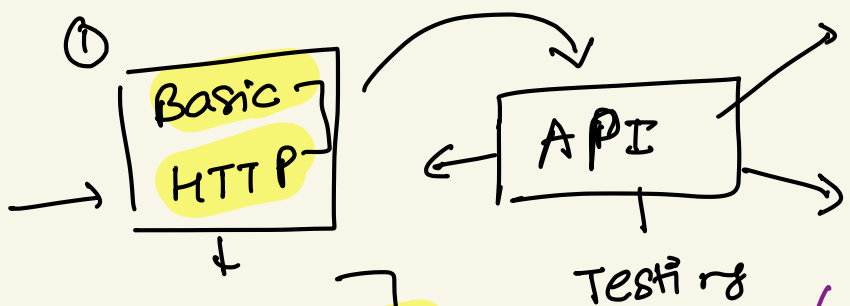
③ 1L, Cow - 70 Rs.

④ 3 1L M - (White + 2 ices?


* STLC →
[Confluence, SRS, Excel or Word DG
Read the Req]
1 → Requirement Analysis →
2 → Test plan
Zephyr
doc file
3 Test Case → 1 Zephyr 2 Excel
4 Test execution
5 Defect Reporting
6 Test closure
Outcome
Functional Req

API →

FB, Twitter – API



Manually

① Req A
② Testplan
③ testCase
④ Execution
⑤ defect
⑥ deploy

Automation

[ECE, IT, CSE
MT

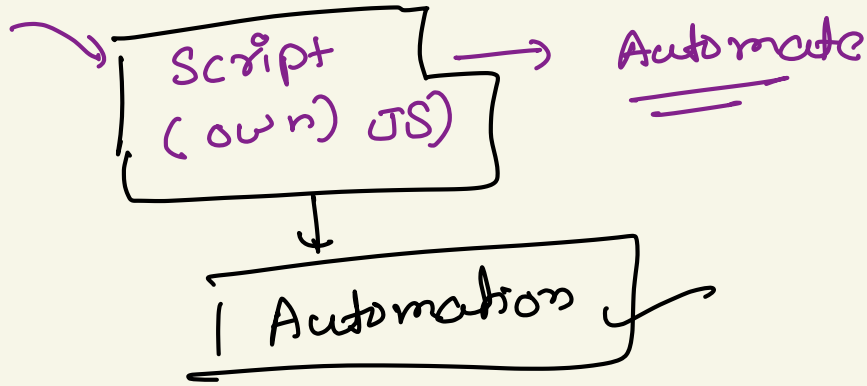
① Postman → Develop → testing

Swagger

Rest clients

⋮

(1) Postman → Manually + Postman → execution



API Testing → Manually → Postman

→ Automate Some scripts

② Rest Assured (Java lib)

→ API Automation

→ Postman can do Automation

Why we using RA?

① Duplicacy

② you can't do everything in Postman

③ Limited Automation

API Testing → ① Manual + execute manual scripts

② Automate

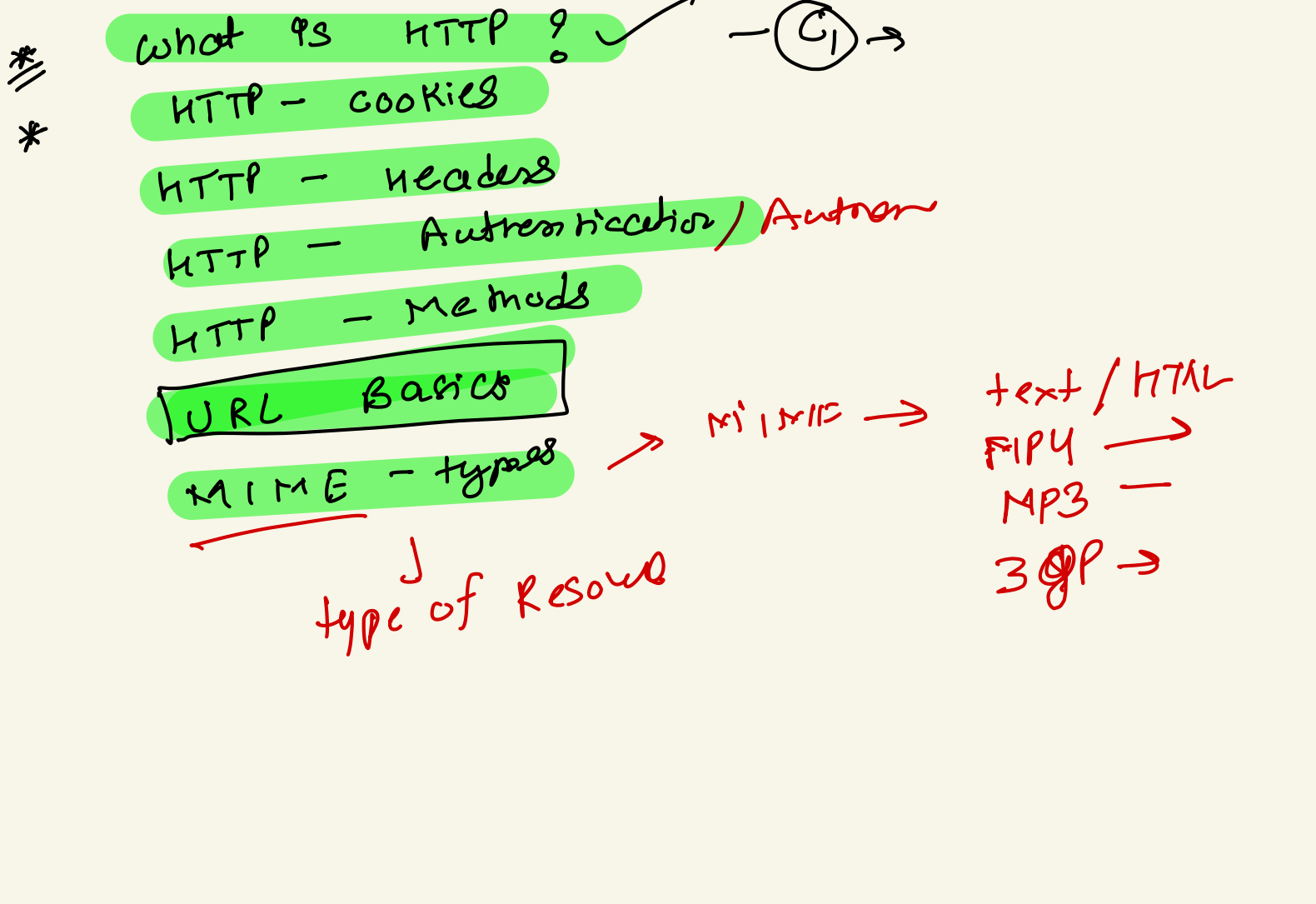
- Basic API / Postman
- Advance = Rest Assure

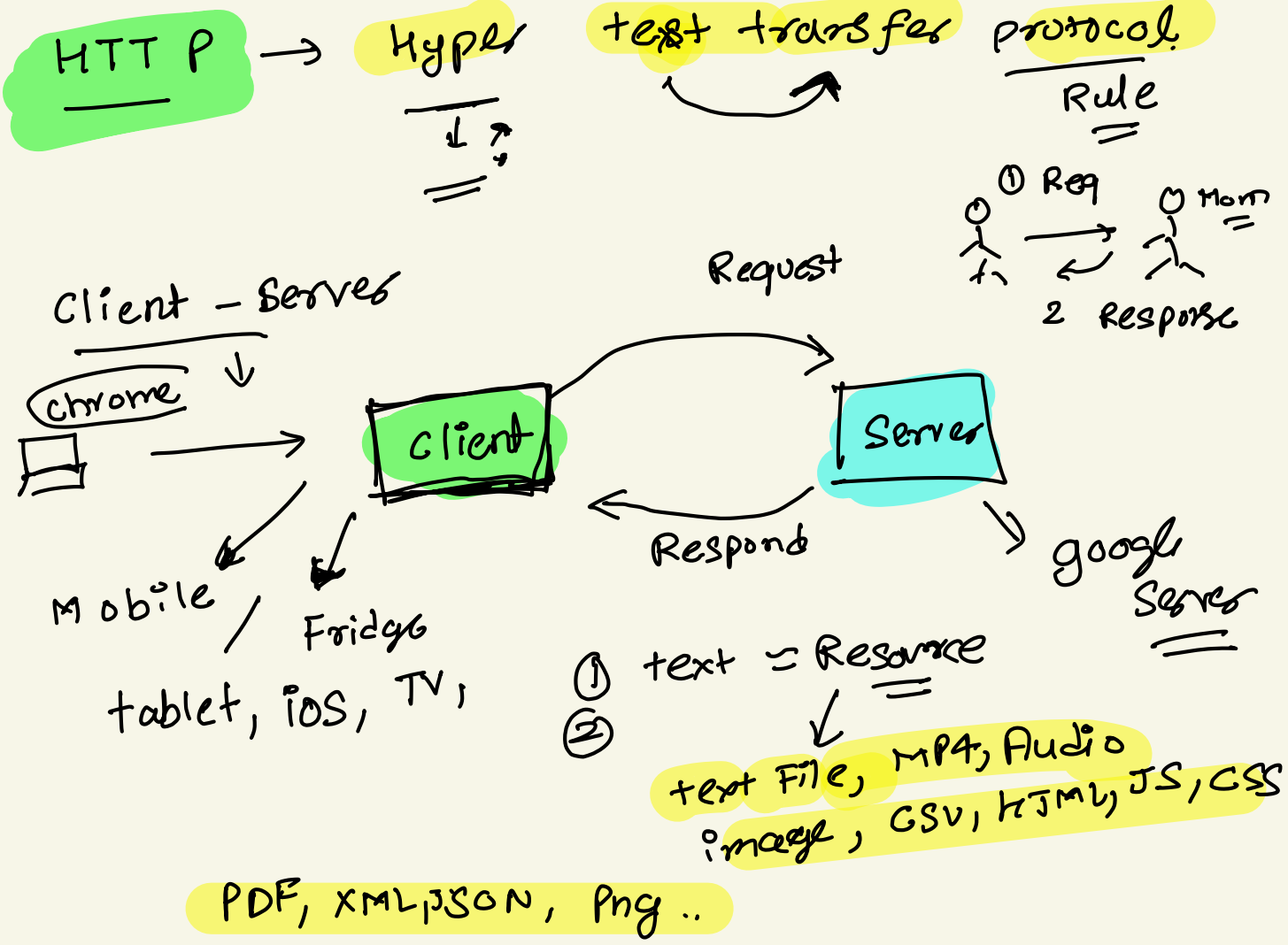
① Basics of HTTP

② API → API testing

③ CRUD → Basic CRUD → Postman

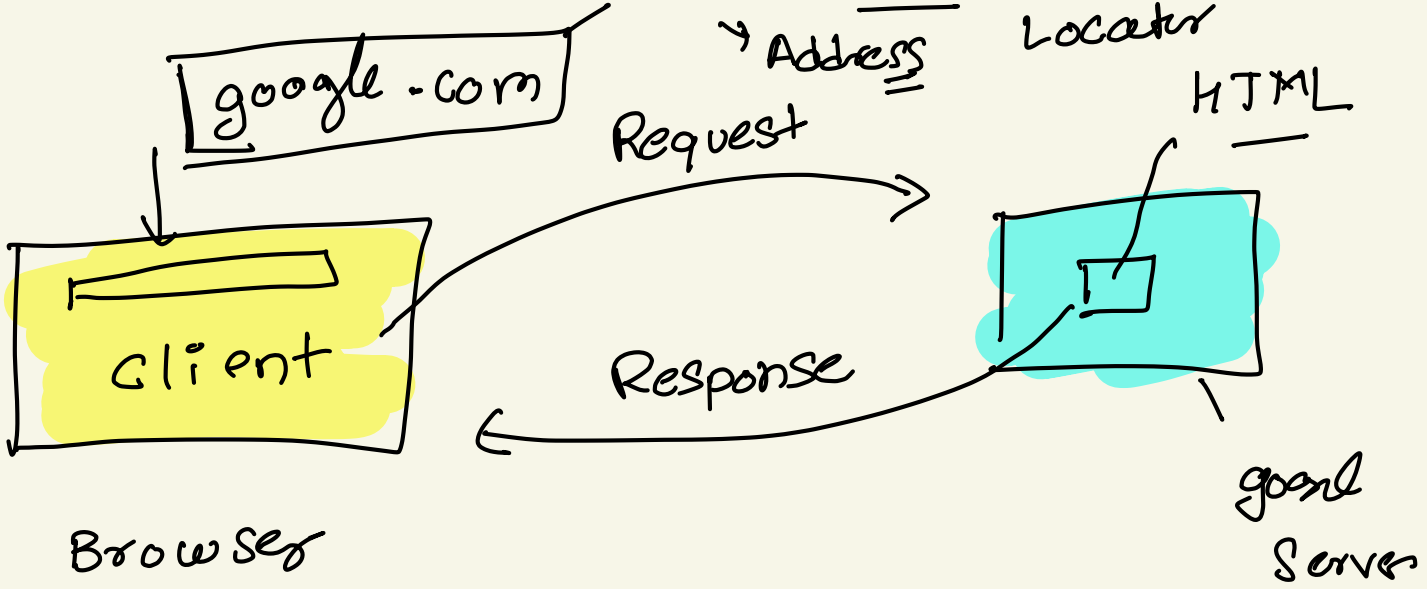
④ Basic test case also with Postman


what is HTTP ?
HTTP - cookies
HTTP - Headers
HTTP - Authentication / Authorization
HTTP - Methods
URL Basics
MIME - types
MIME
text / HTML
FIPY
MP3
3gp
type of Resource


HTTP
Hyper
text transfer protocol
Rule
① Req
Mom
2 Response
Client - Server
Chrome
Request
client
Server
Respond
Mobile
Fridge
tablet, iOS, TV,
google Server
① text = Resource
②
text File, MP4, Audio
image, CSV, HTML, JS, CSS
PDF, XML, JSON, Png ..

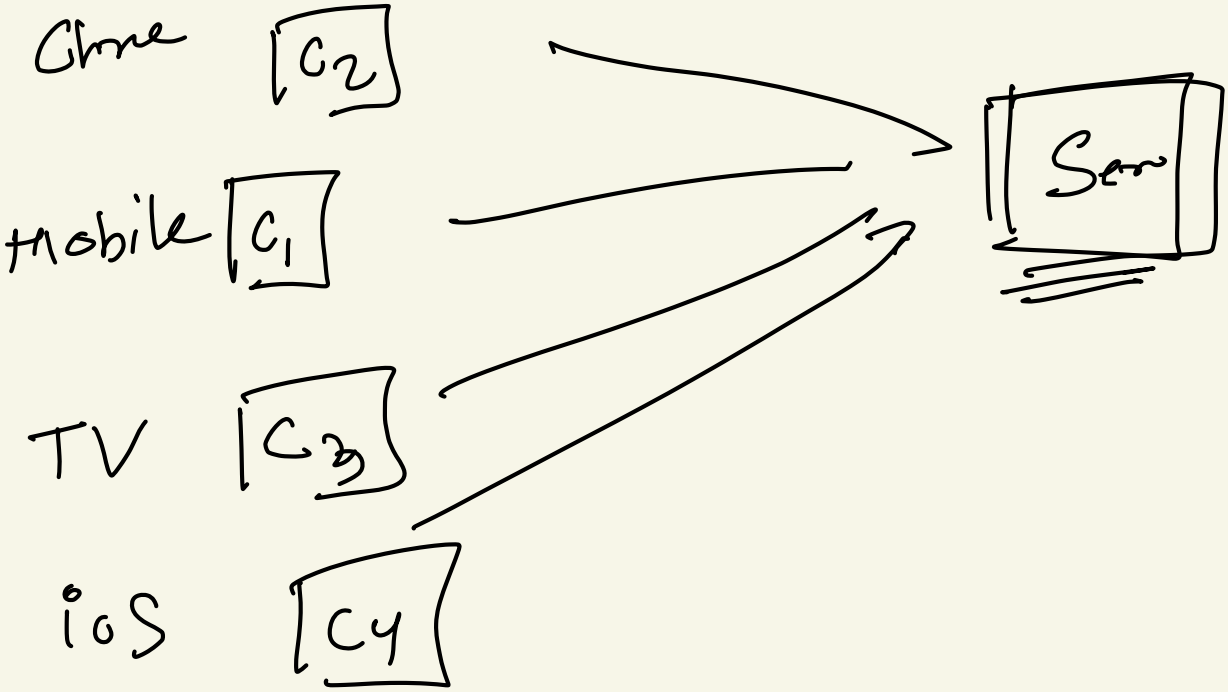
Cleor

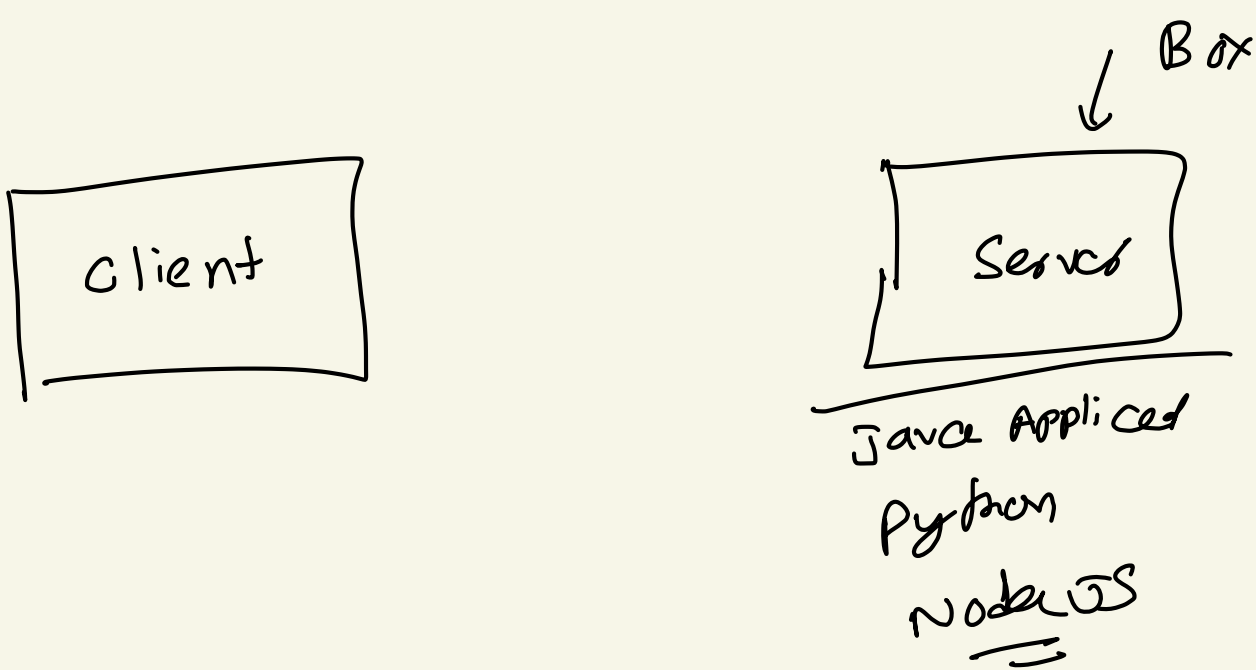
URL → Uniform Resource Locator
→ Address

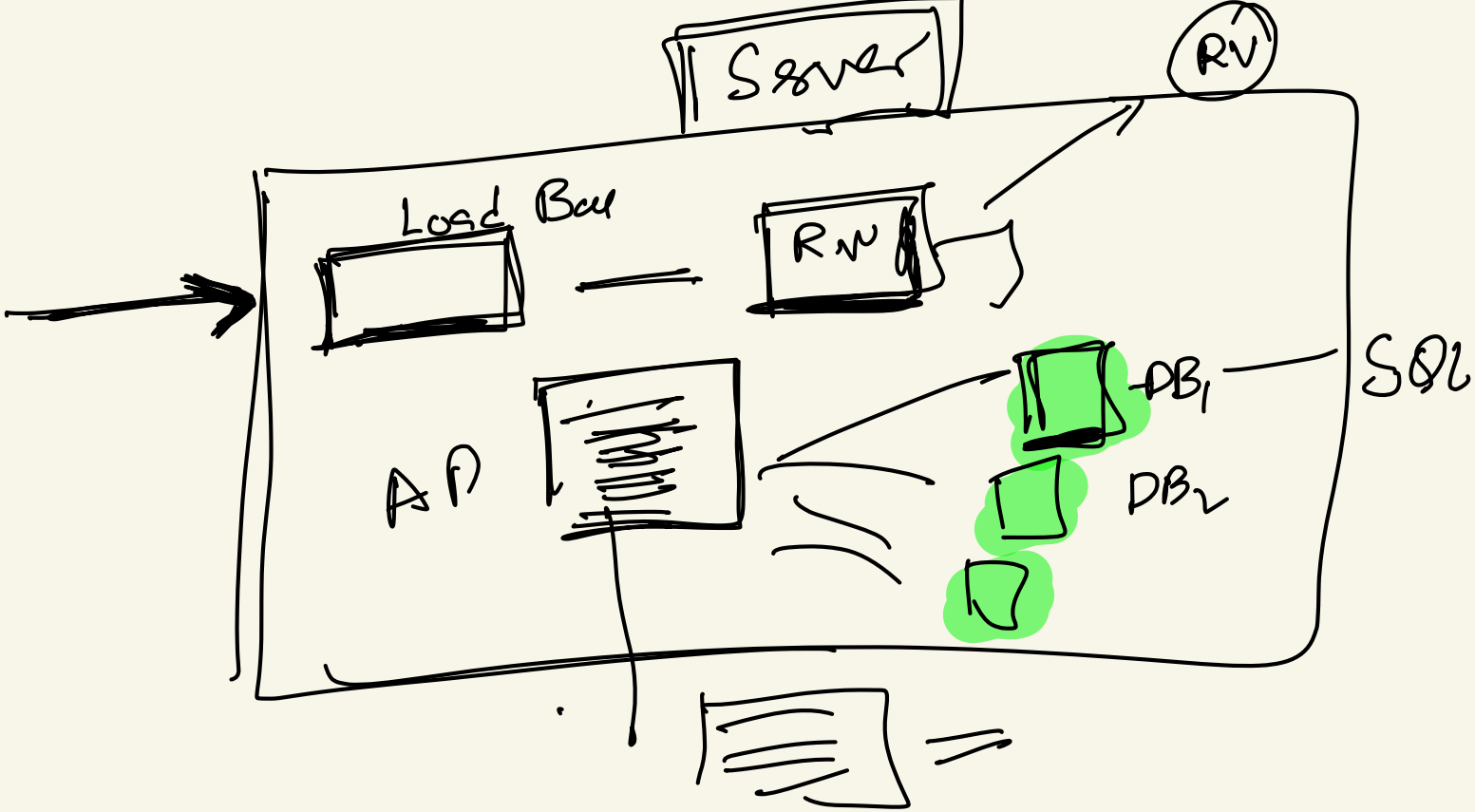


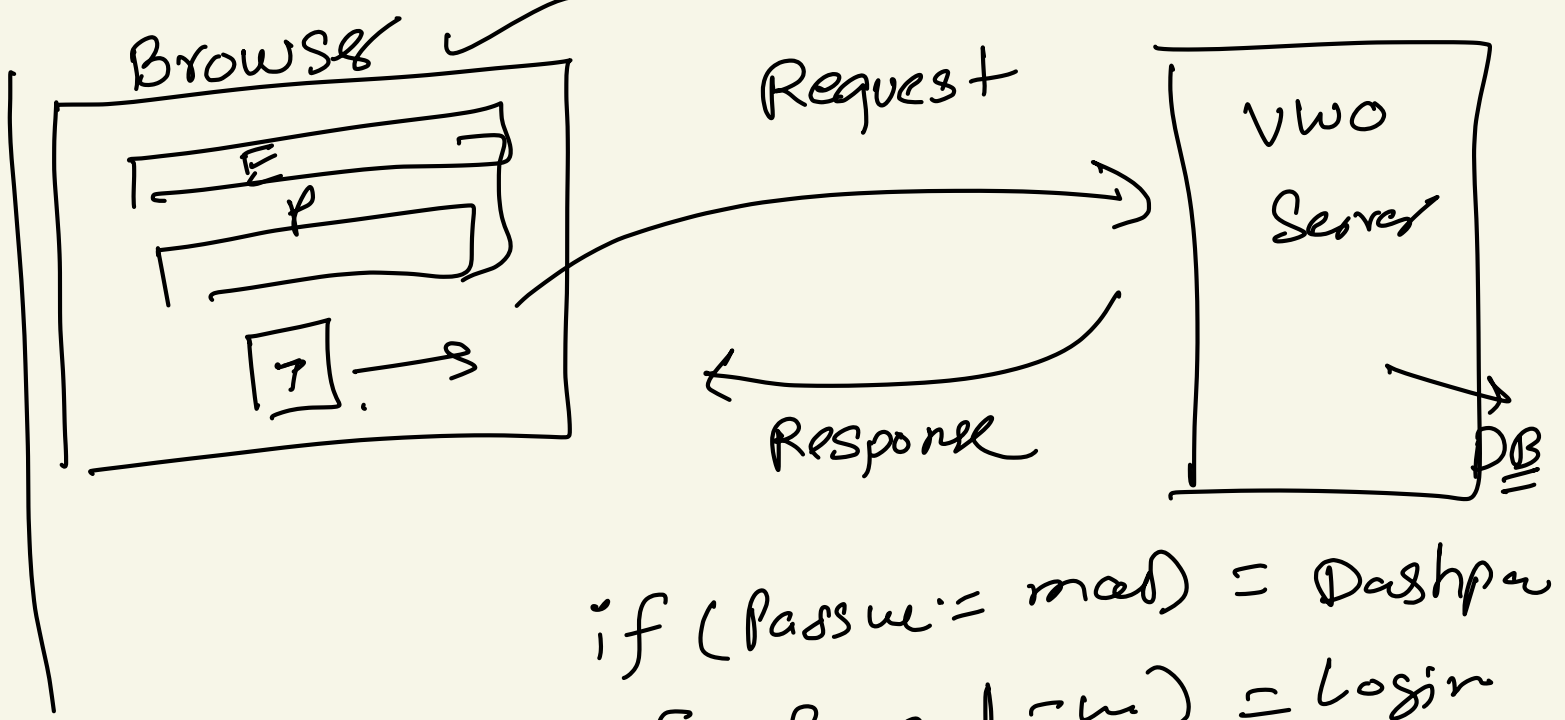
① → HTML
② JS, CSS, Image

Protocol — ① client will req to Server
② Server will respon Client


Chrome
C2
mobile
C1
TV
C3
iOS
C4
Server


Box?
Client
Server
Java Applications
Python
NodeJS


Server
RV
Load Bal
RV
DB1
SQL
DB2
AP



if (Passw == mod) = Dashpa

if (Pass != m) = Login

HTTP → protocol used to transfer data

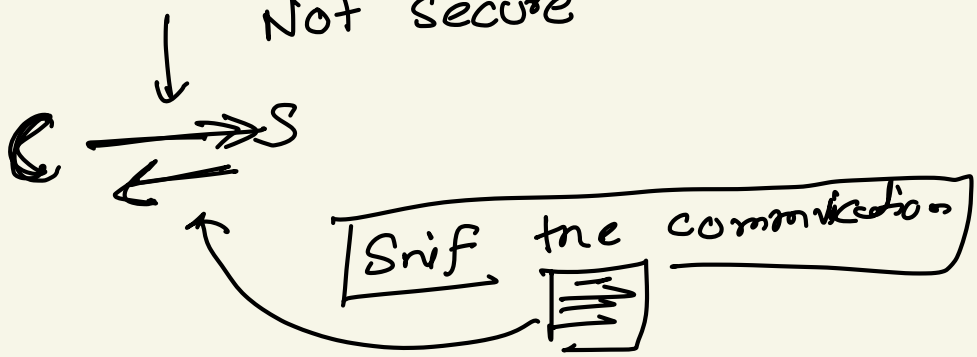
Between Client / Server

Resource

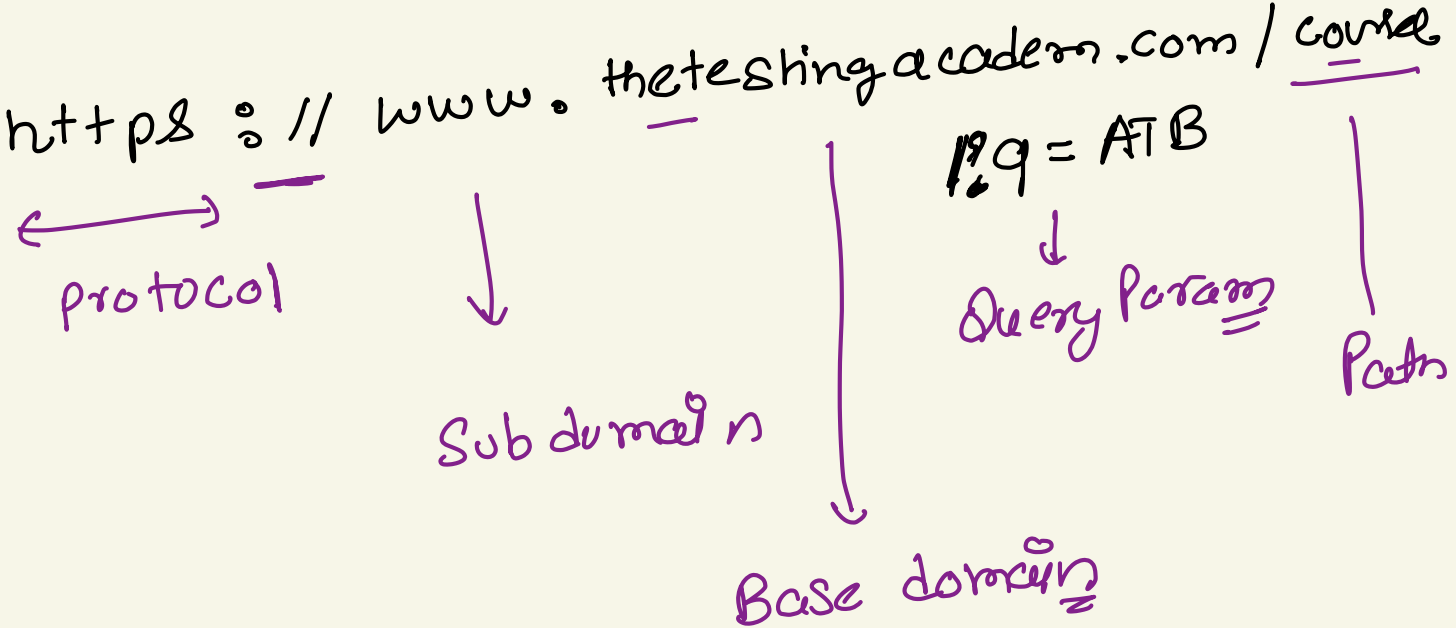
- * HTTPS (secure)

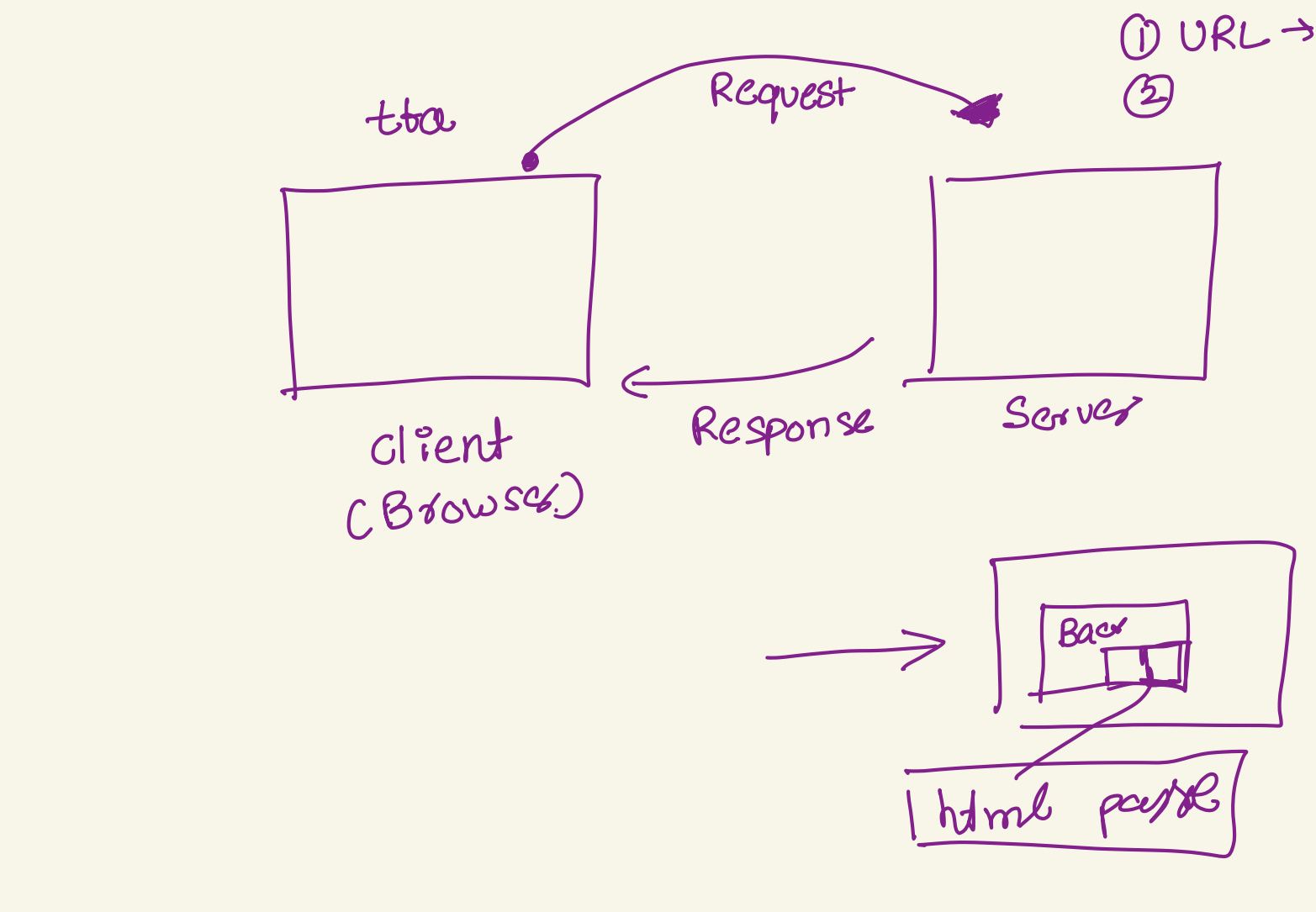
HTTP → HTTPS (2.0)

encrypted



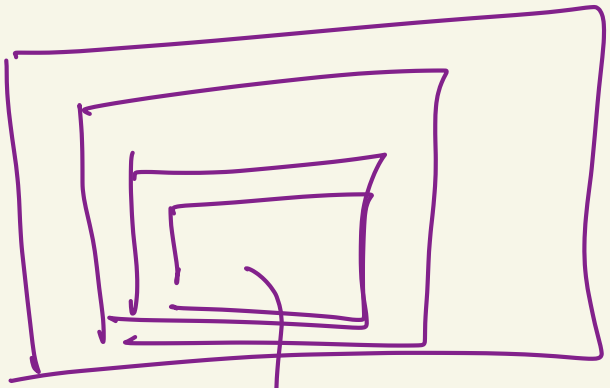
URL → Uniform Ressource Locator (Ressce over internet)




① URL →
②
tba
Request
Response
Server
client
(Browser)
Back
html page

https://www.google.com/a/b/c/?q=ATB

Sending to client

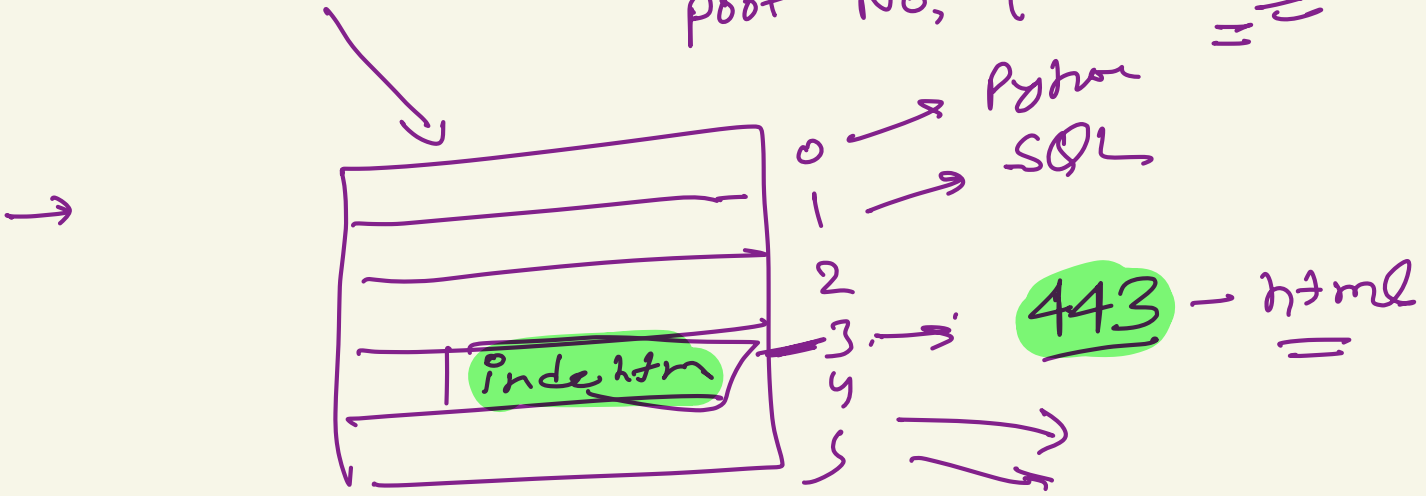


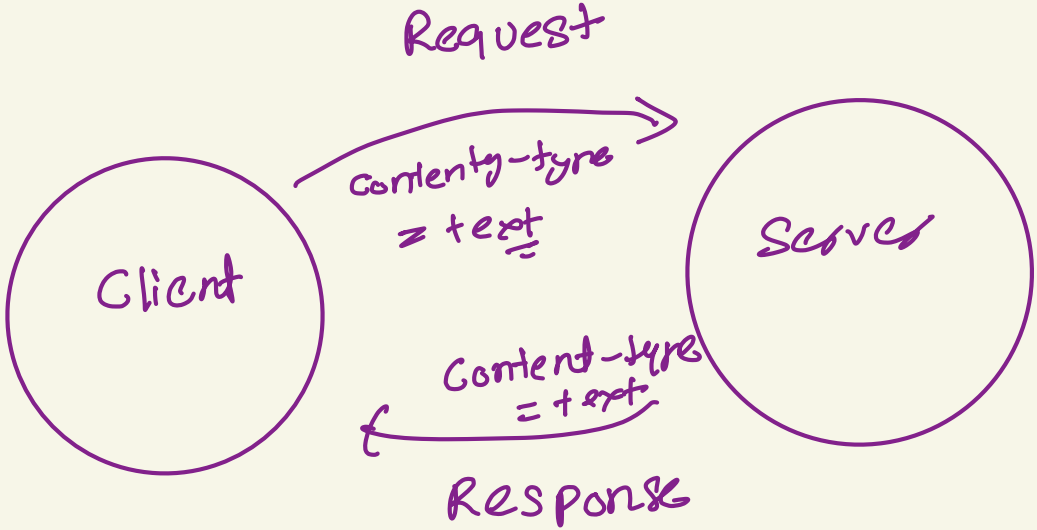
q = ATB

Search of ATB on a html

https : appsrwo.com : 443 / #login

port No, (0 - 65K)





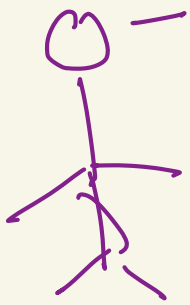
① URL

② Additional info (K = V)

→ Header

→ Content-type = text/plain

Header



— Head (Meta) — Basic information to send as

K = V

URL + header

C ⇄ S

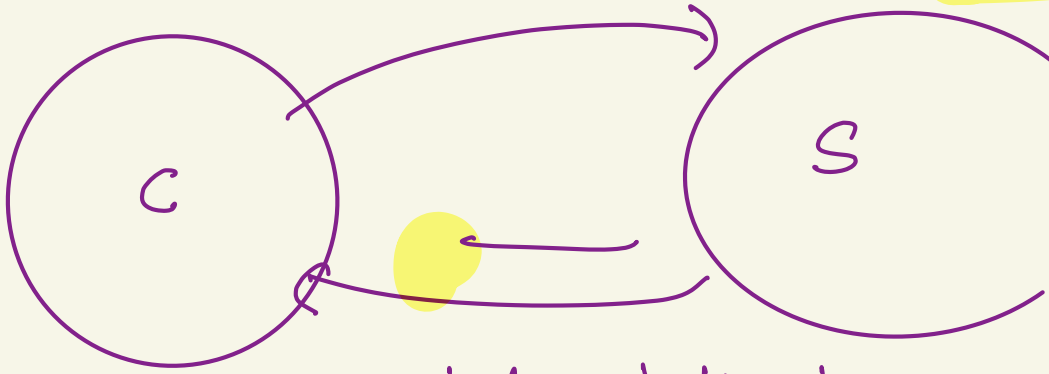
Header

URL + Header (K:V) → Content-type

User Agent

Authen

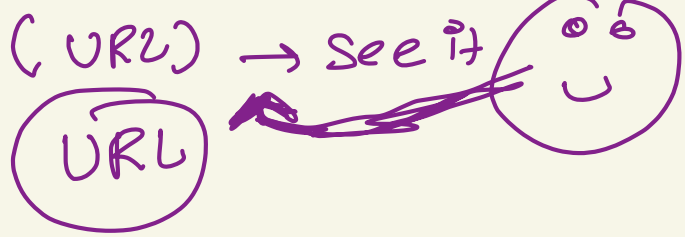
Cookies



data + header

Response
(HTML) + header

Header — QP, (URL) → see it



↓

Chrome dev tools

↓

→ debug (Header)

Req = URL + QP + Header

Authentication

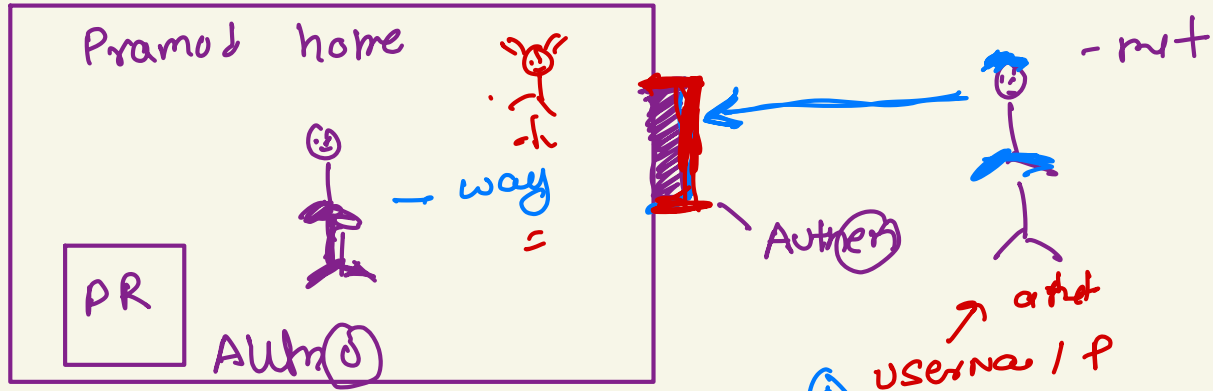
↓

who are you?

VS

Authorization

(Permission ⊗)

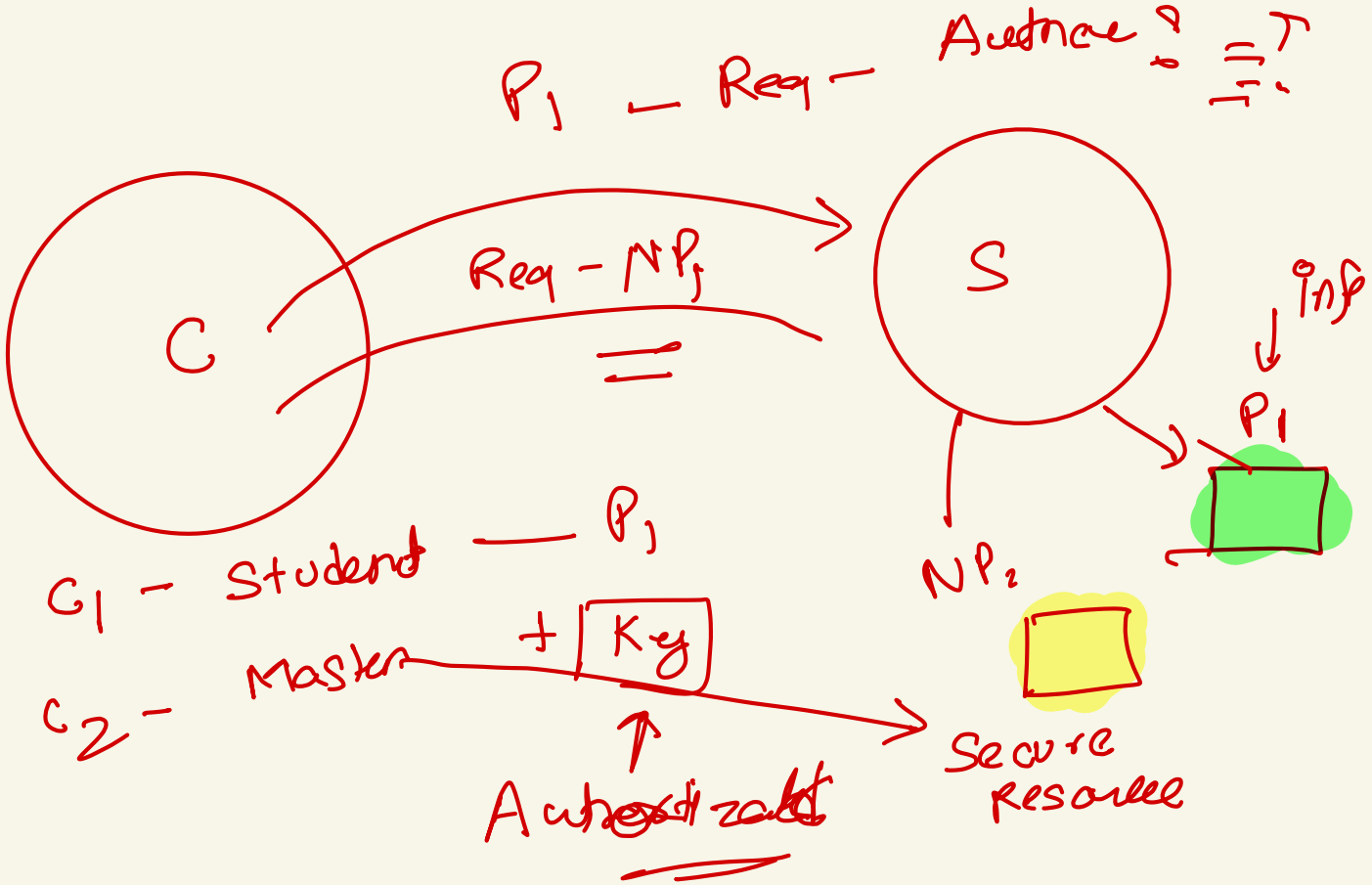


① username / P

② API token:

③ XYZ key

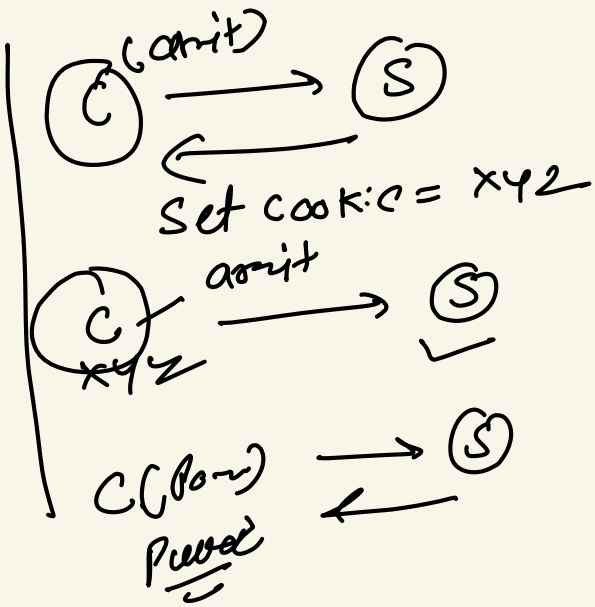
④ token (oAuth2)

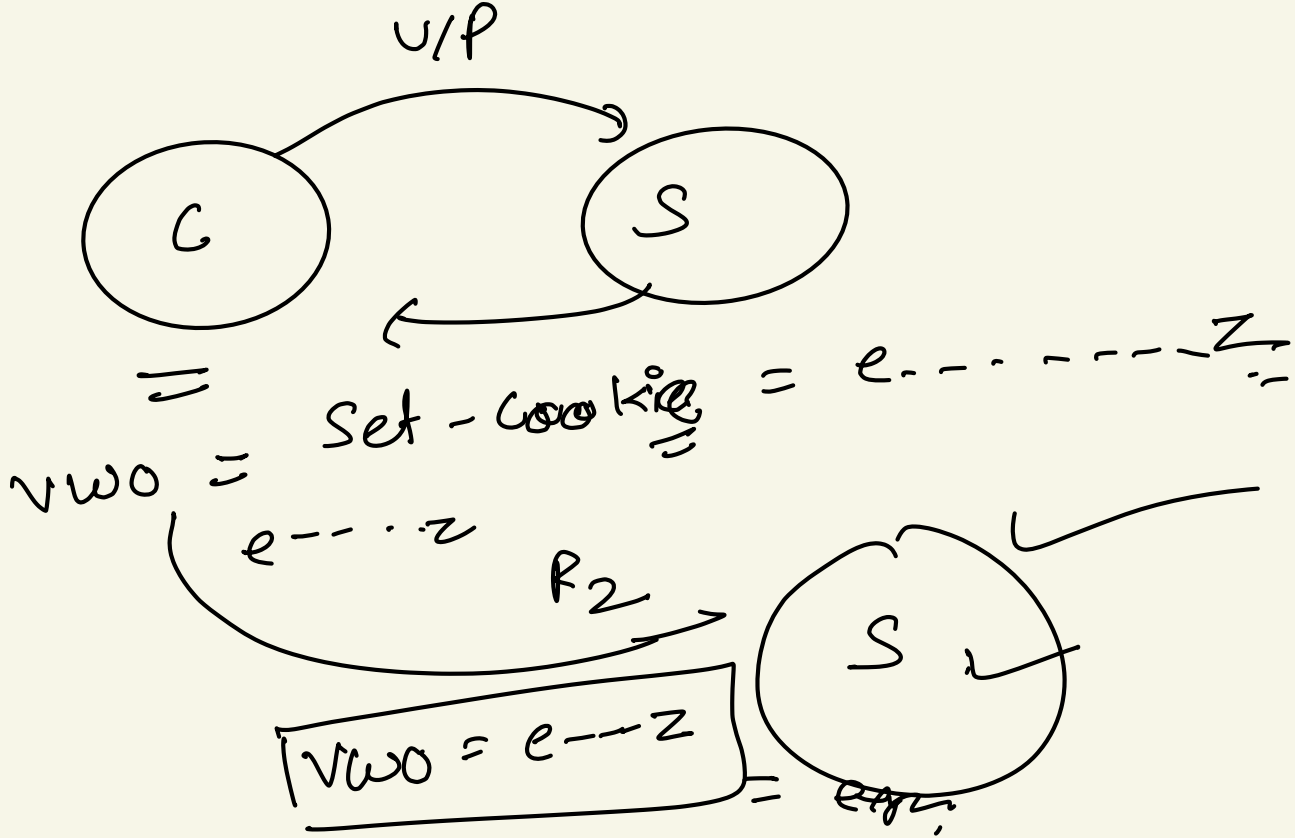

P1 — Req — Audience ? =?
C
Req - NP1
S
Inf
P1
C1 - Student — P1
C2 - Master + Key
NP2
Secure Resource
Authorization

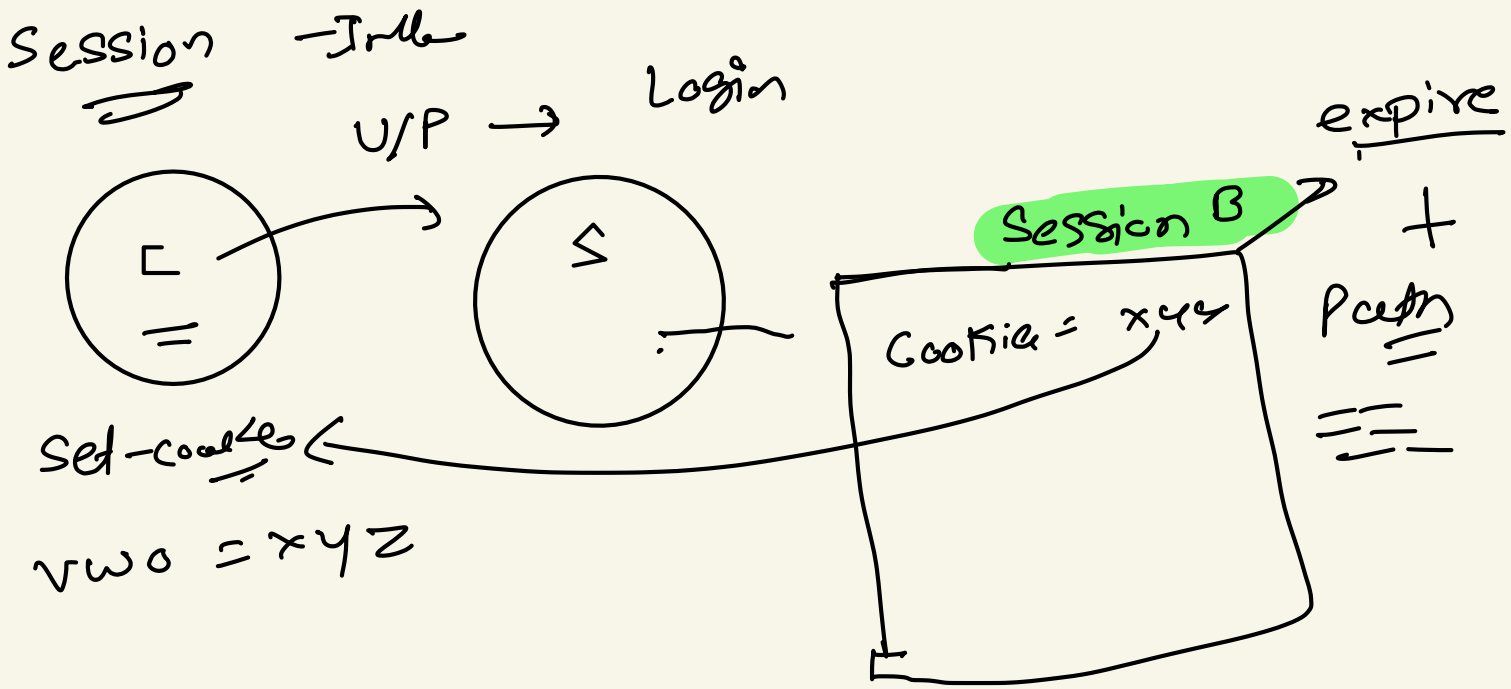
HTTP - cookie → (K=V)

Key = value

1. Session management
2. Tracking →
3. Personization

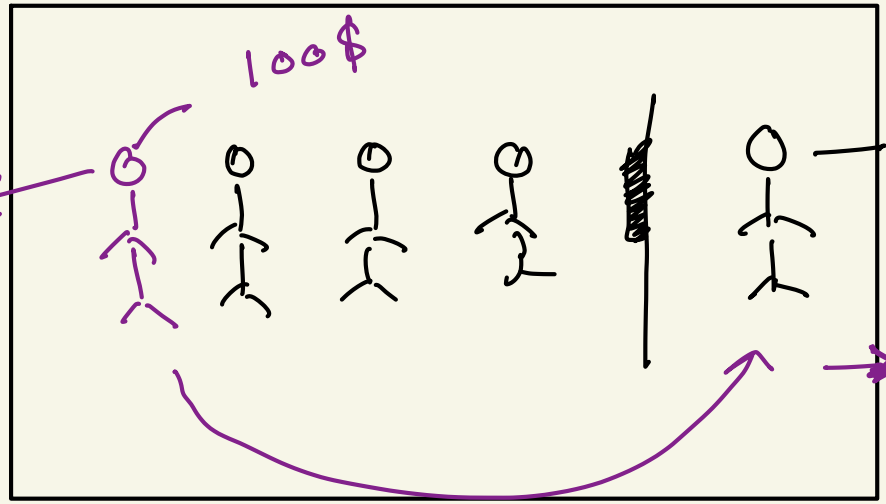



U/P
C
S
Set-Cookie = e-----z
VWO = e---z
R2
S
VWO = e---z

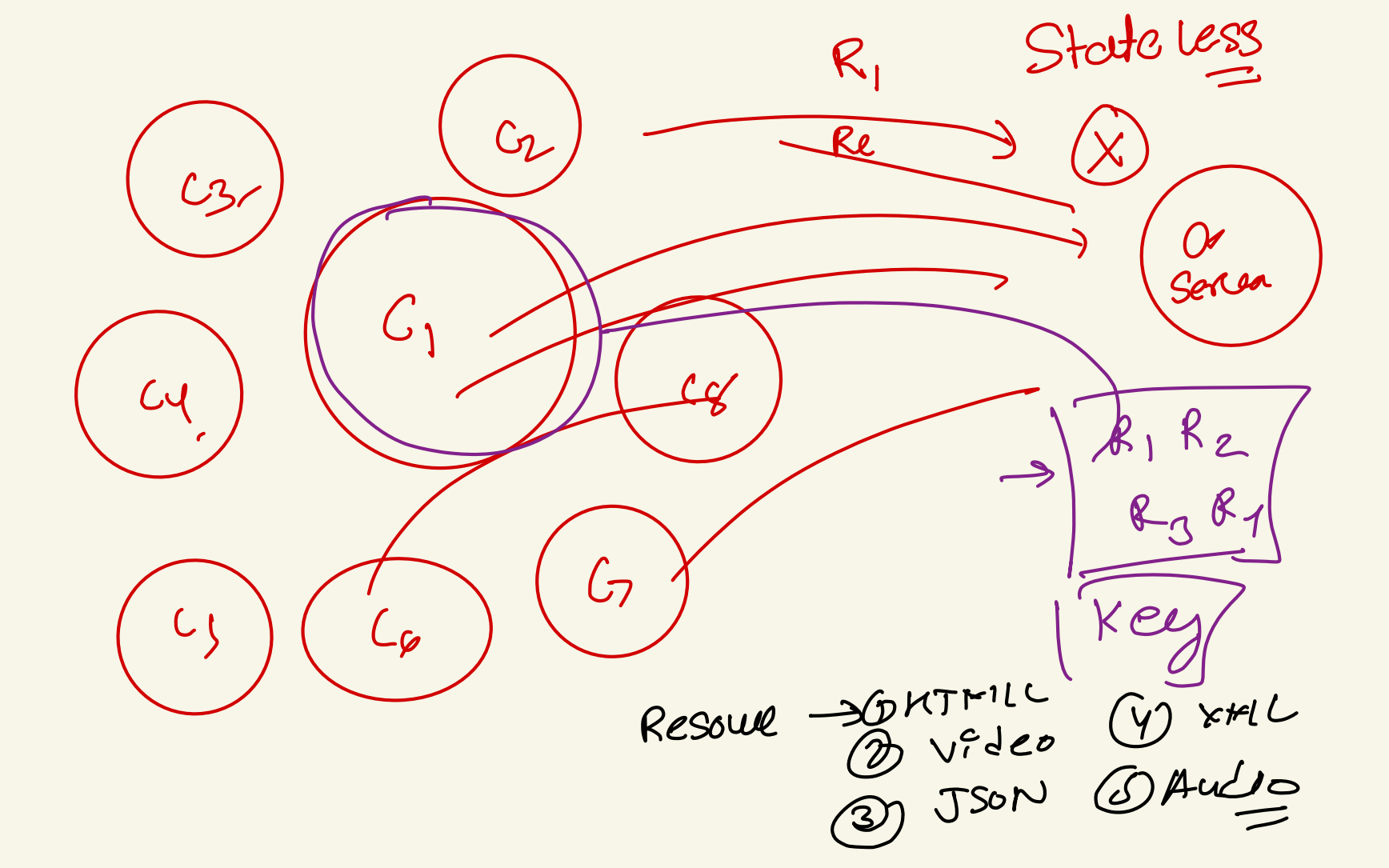

Session
U/P →
Login
expire
+
Session B
Cookie = xyz
Path
Set-cookie
vwo = xyz

Why we need Authorization

cookie
API key
JAuth2.0
Bearer



1. U/P
2. API
3. Key
4. Pass

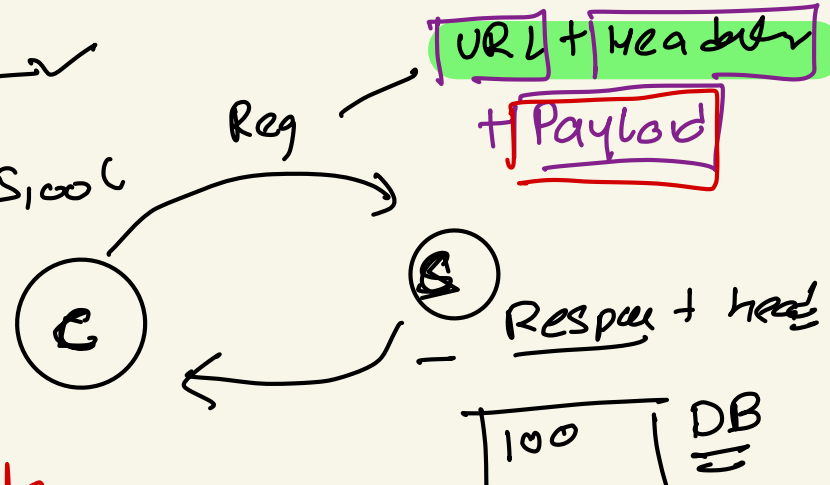

Stateless
R1
Re
X
Or Server
C1
C2
C3
C4
C5
C6
C7
C8
R1 R2
R3 R1
Key
Resour
1 HTML
2 Video
3 JSON
4 XML
5 Audio

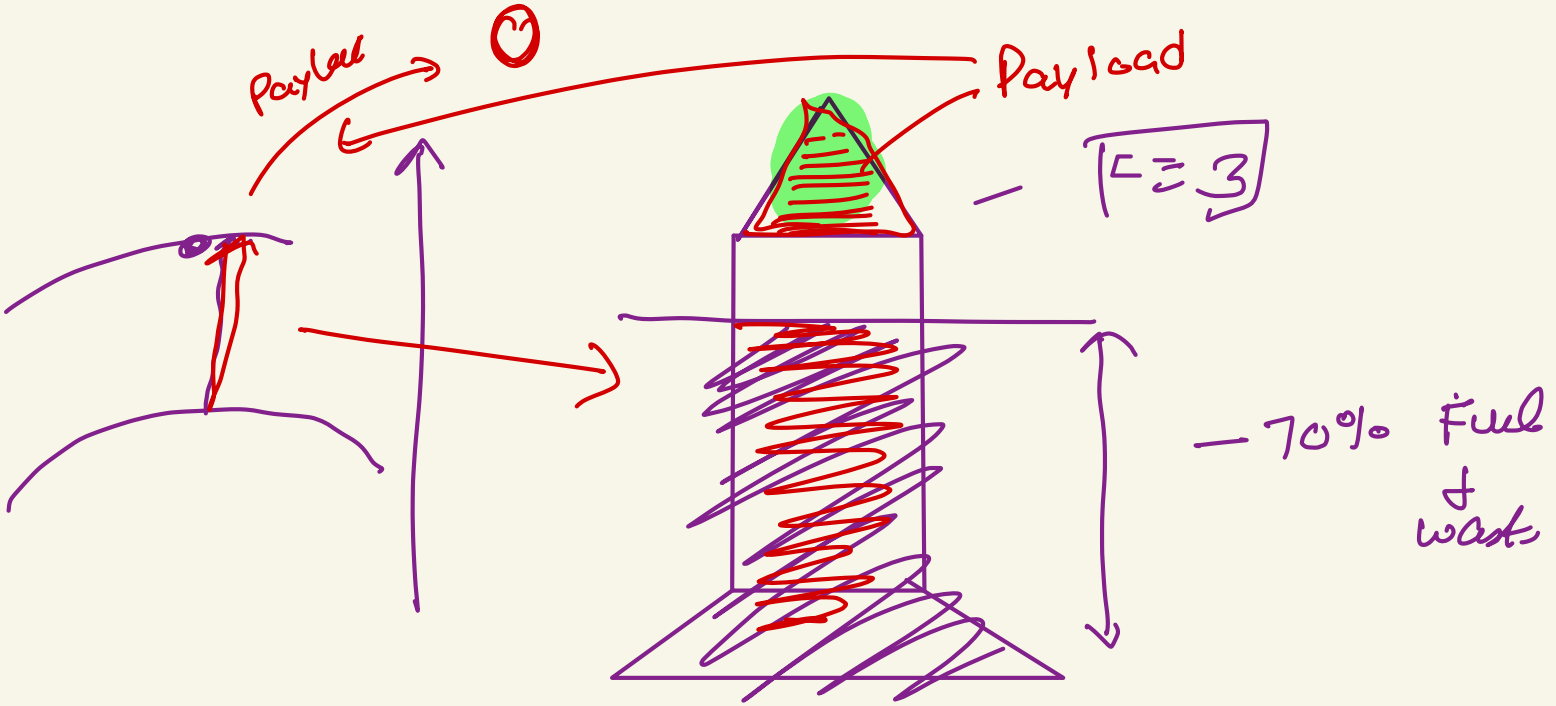
HTTP - METHODS

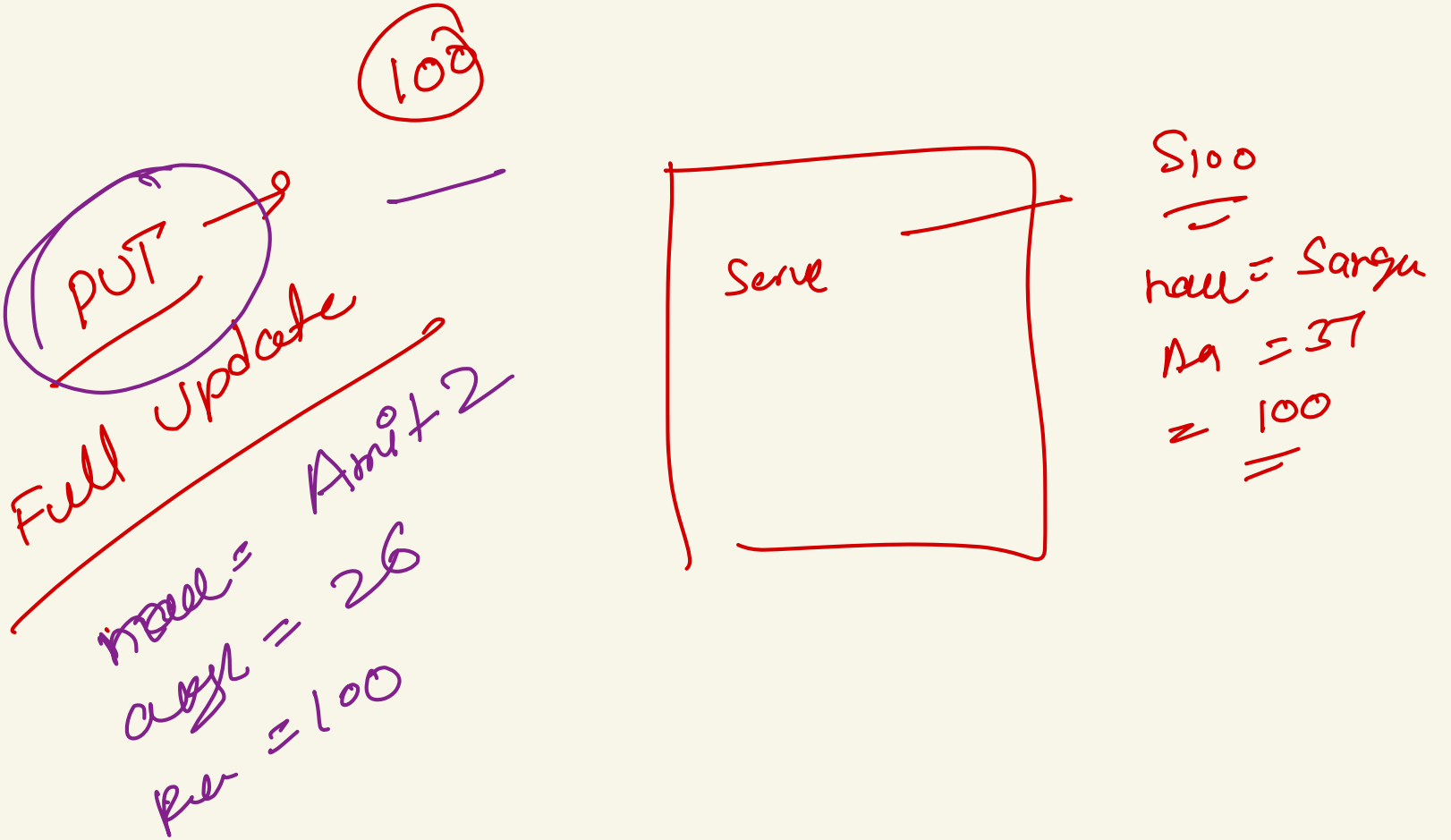
1. Get → Fetch → S_1 ~ S_{100}
2. Post → Creation → S_{101}

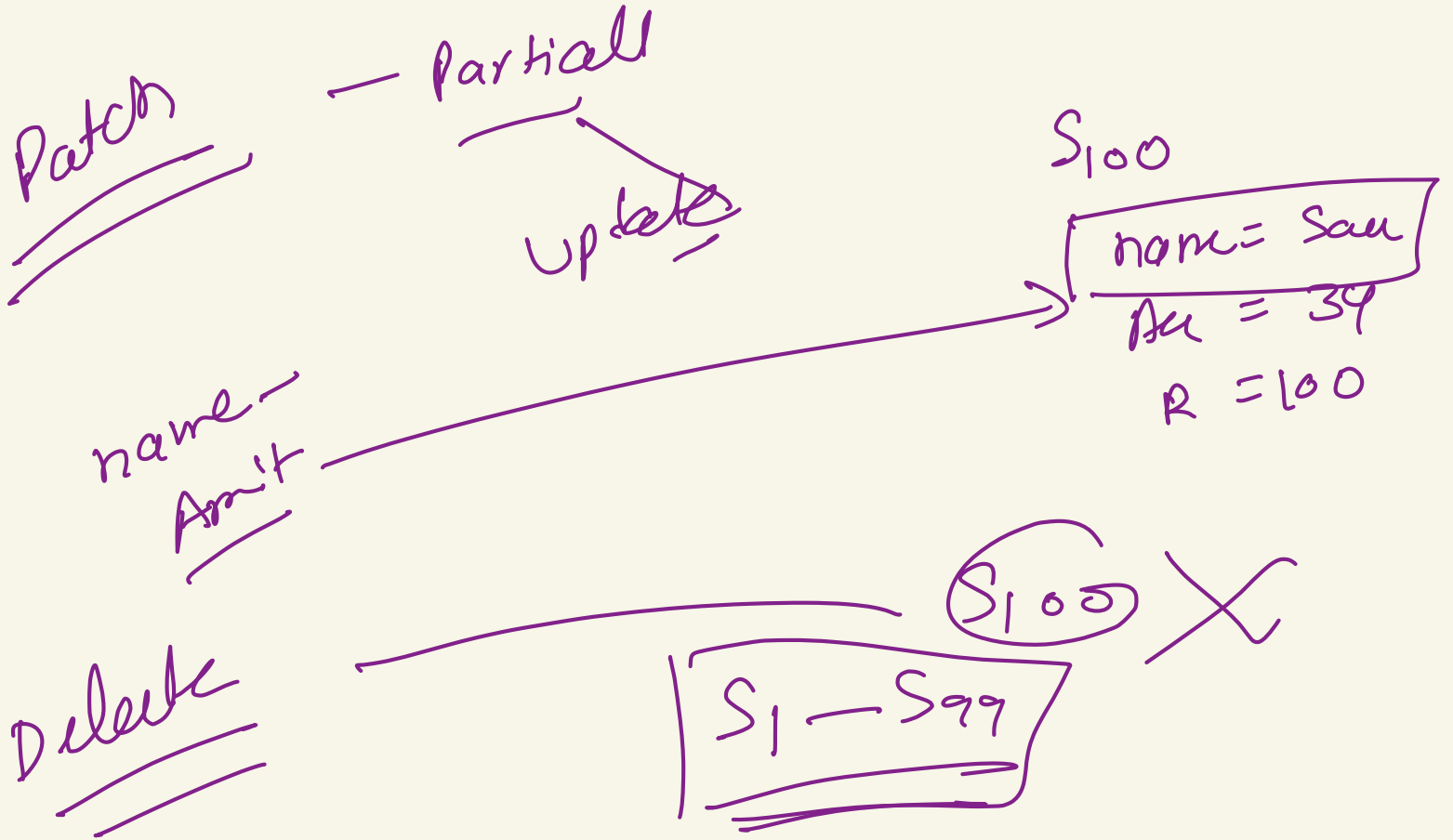
```
name = Amit
age  = 34
R.  = 101
```

3. Put → Full Update
4. Patch →
5. delete

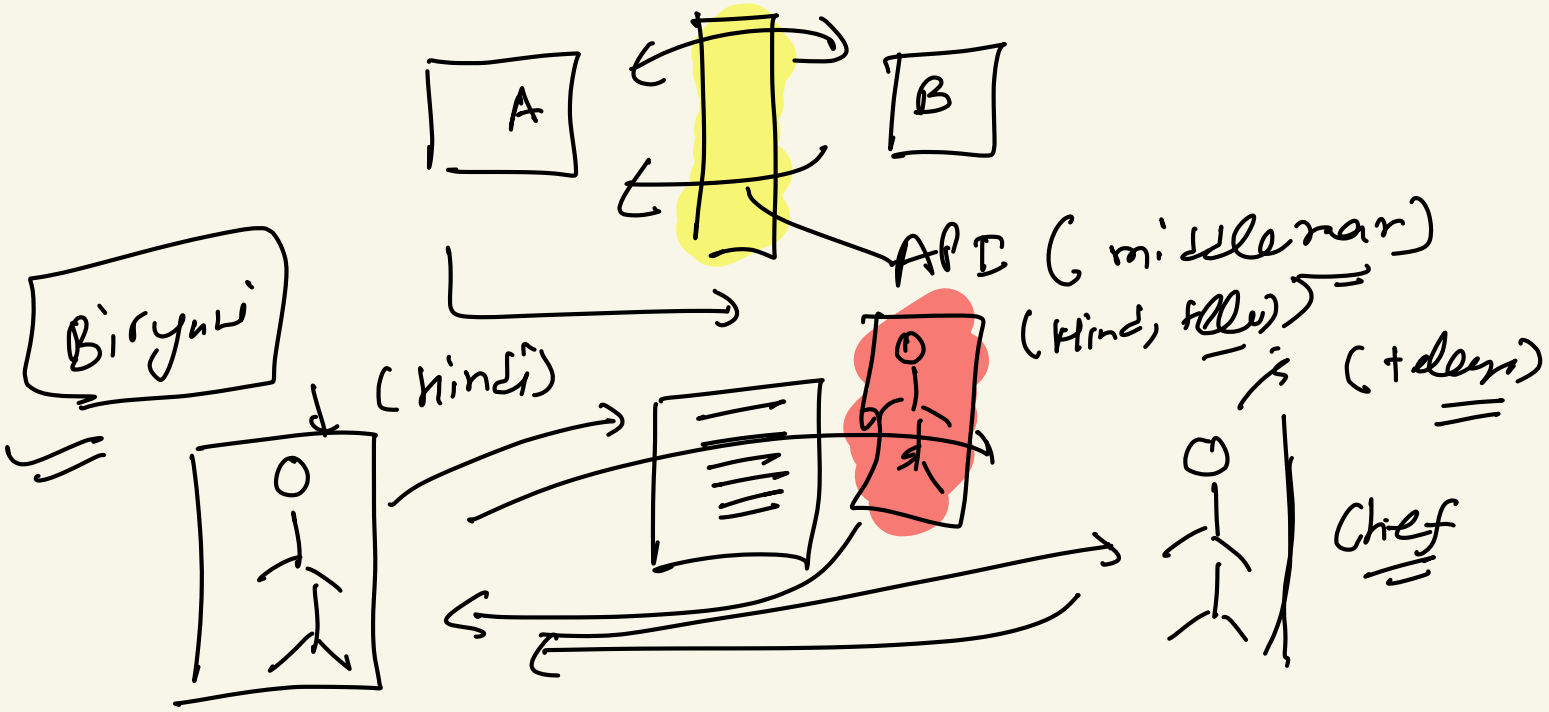


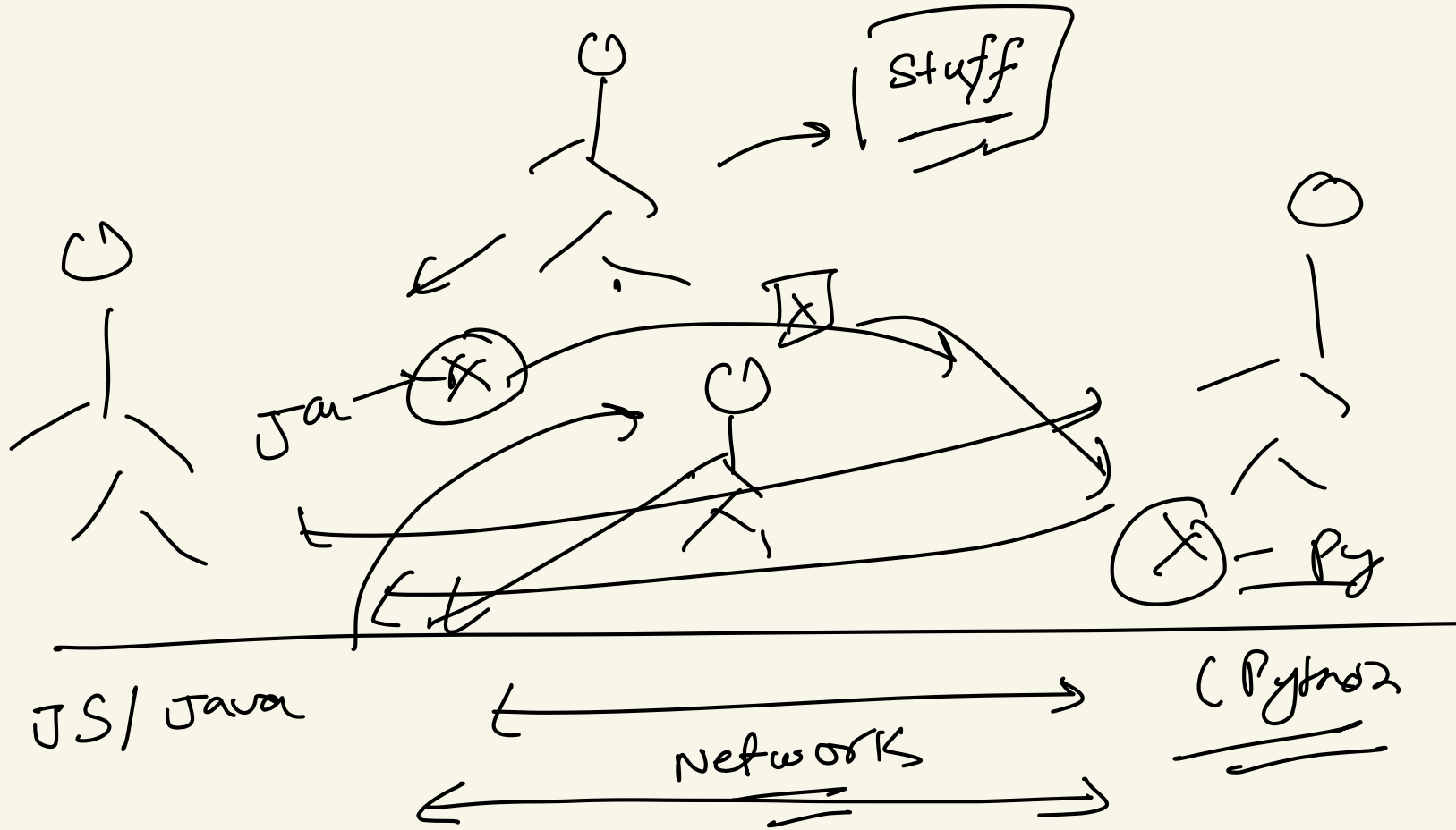

Payload
Payload
F = 3
70% Fuel
+
waste

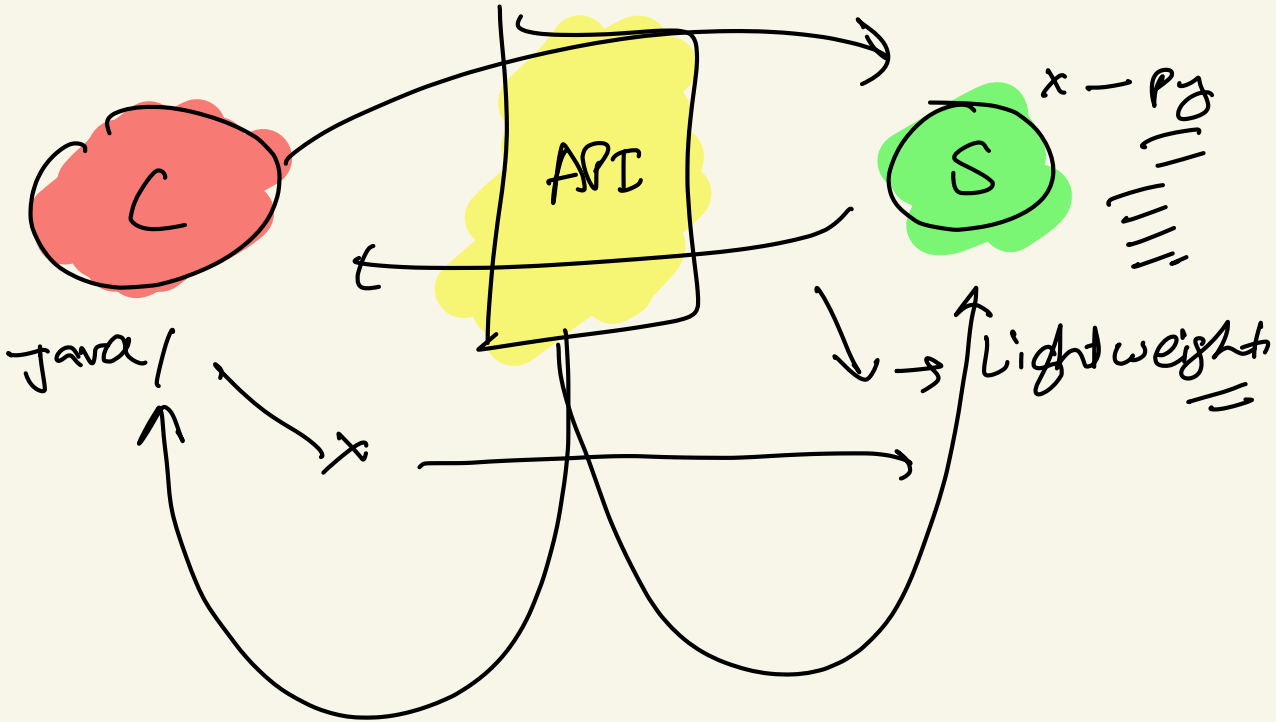

100
PUT
Full Update
name = Amit2
age = 26
id = 100
Serve
S100
name = Sargu
Ag = 37
= 100

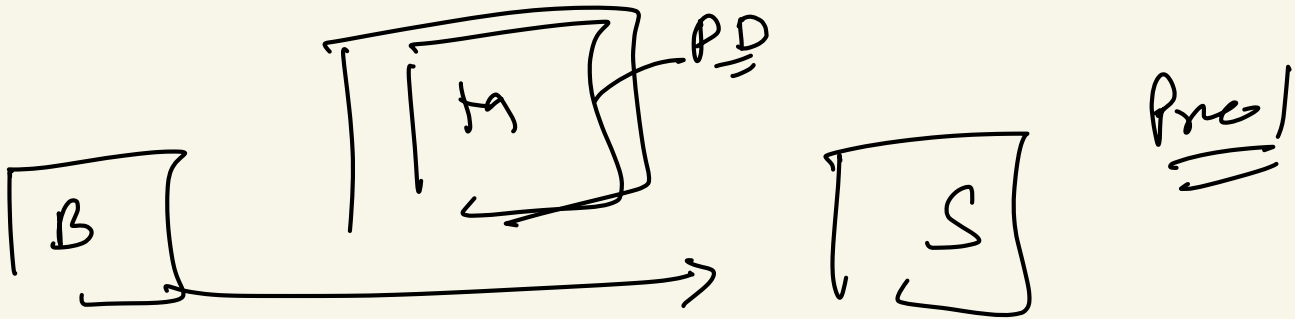

Patch
— Partial
Update
S100
name = Sam
Age = 39
R = 100
name-
Amit
Delete
S100
X
S1 — S99

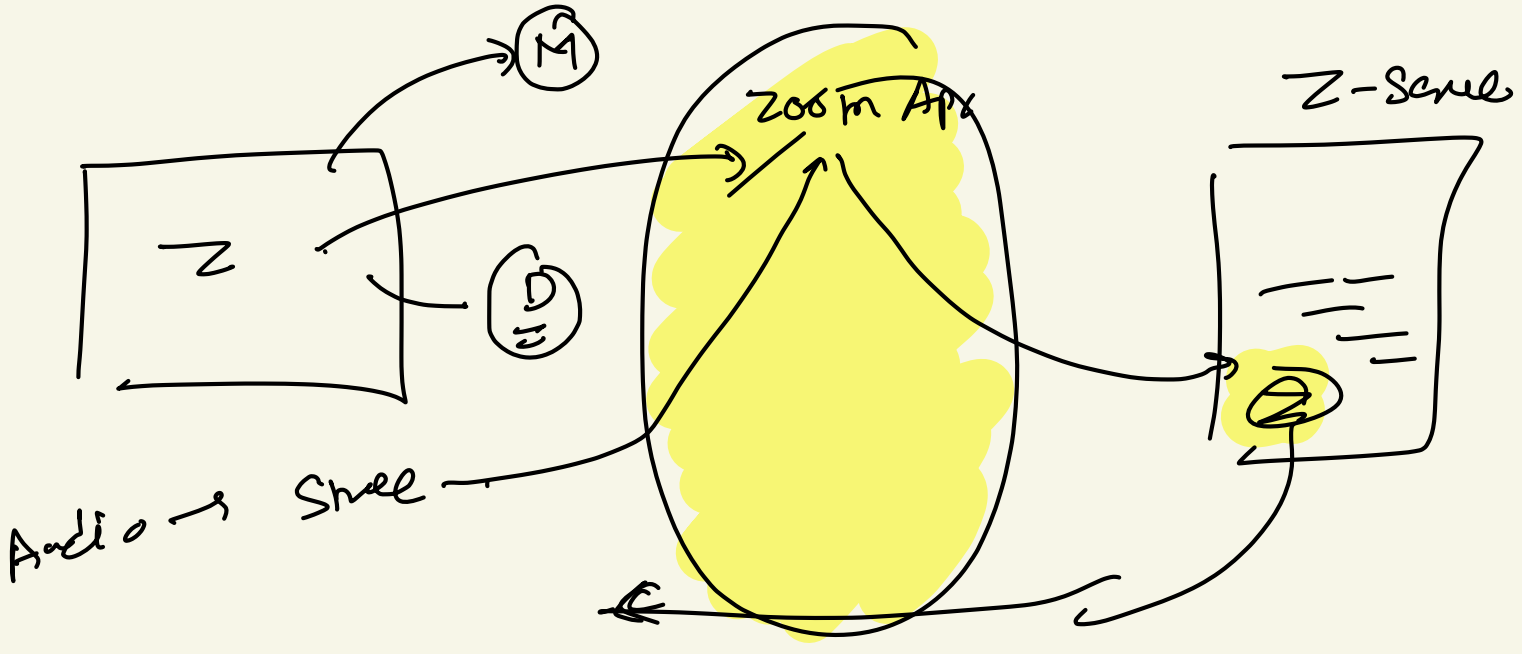
API → Application prog. Interface

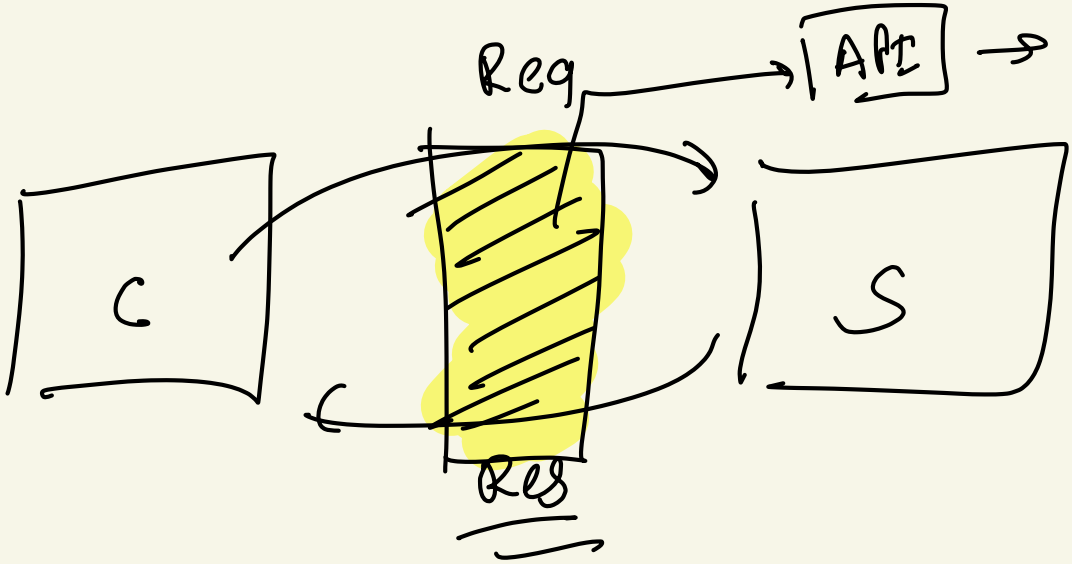


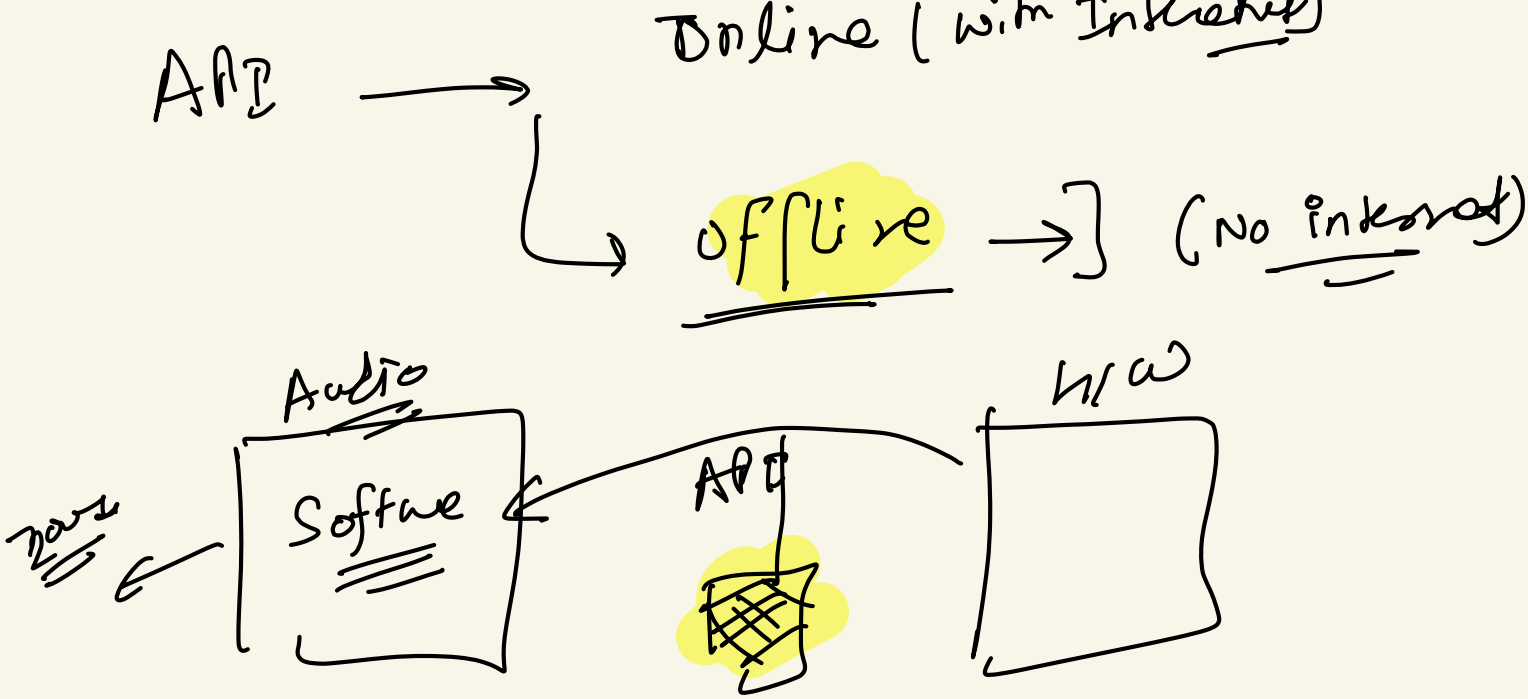

Stuff
Java
Py
JS/ Java
Network
CPython2


C
API
S
x — pg
java
Lightweight


B
M
PD
S
Proc

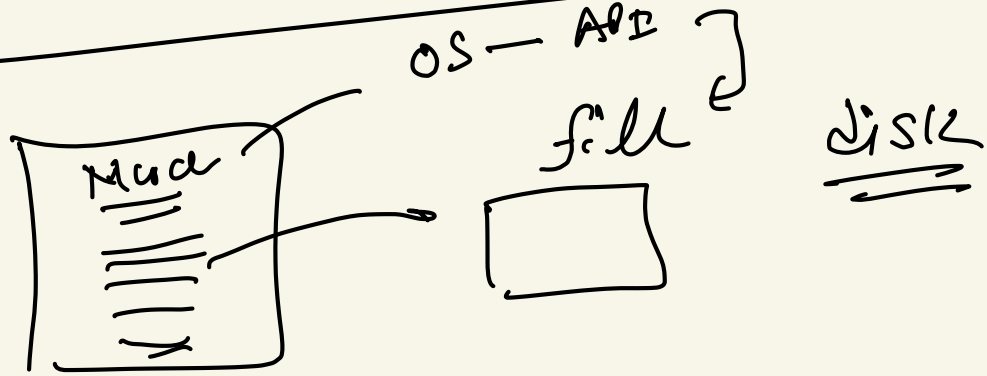

M
Zoom API
Z-Server
Z
D
Audio + Share

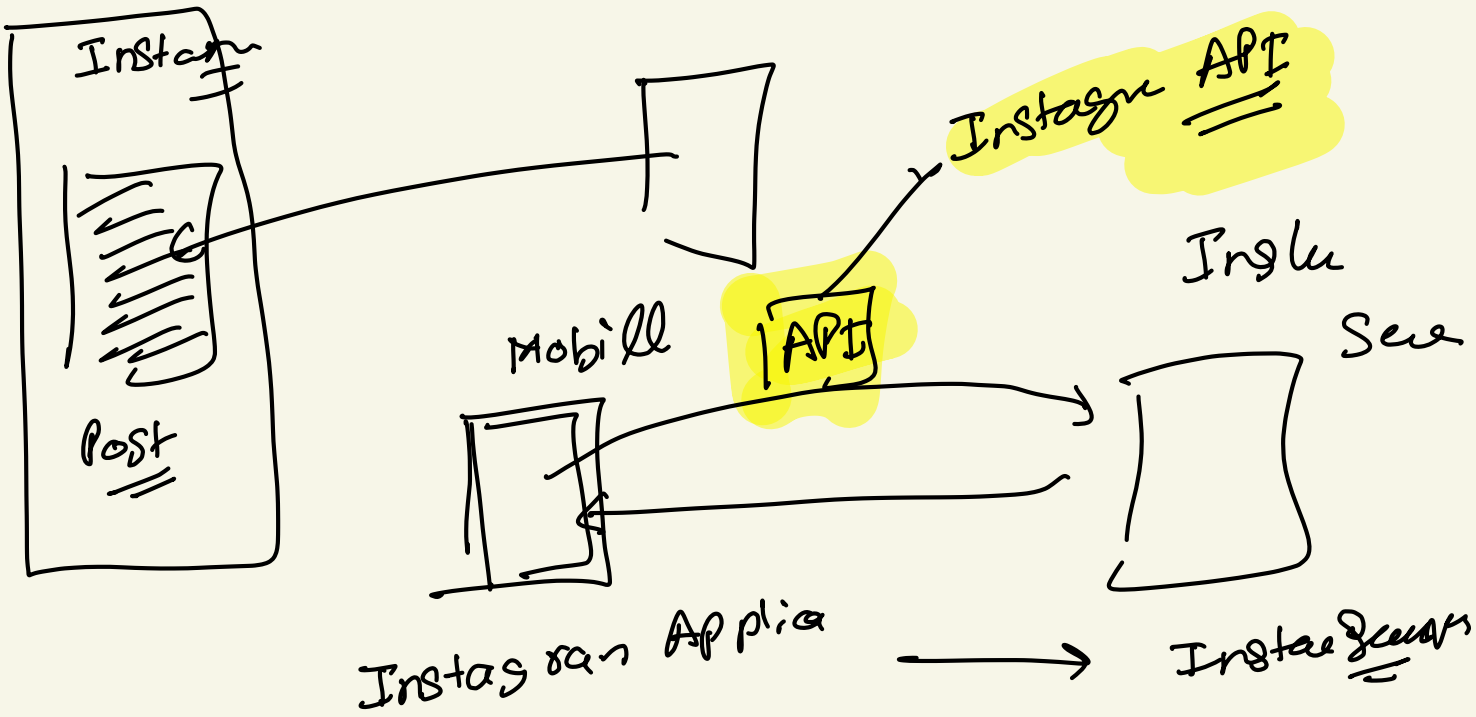

Req
API
C
S
Res


API
Online (with Internet)
offline
(No internet)
Audio
Software
H/W
API

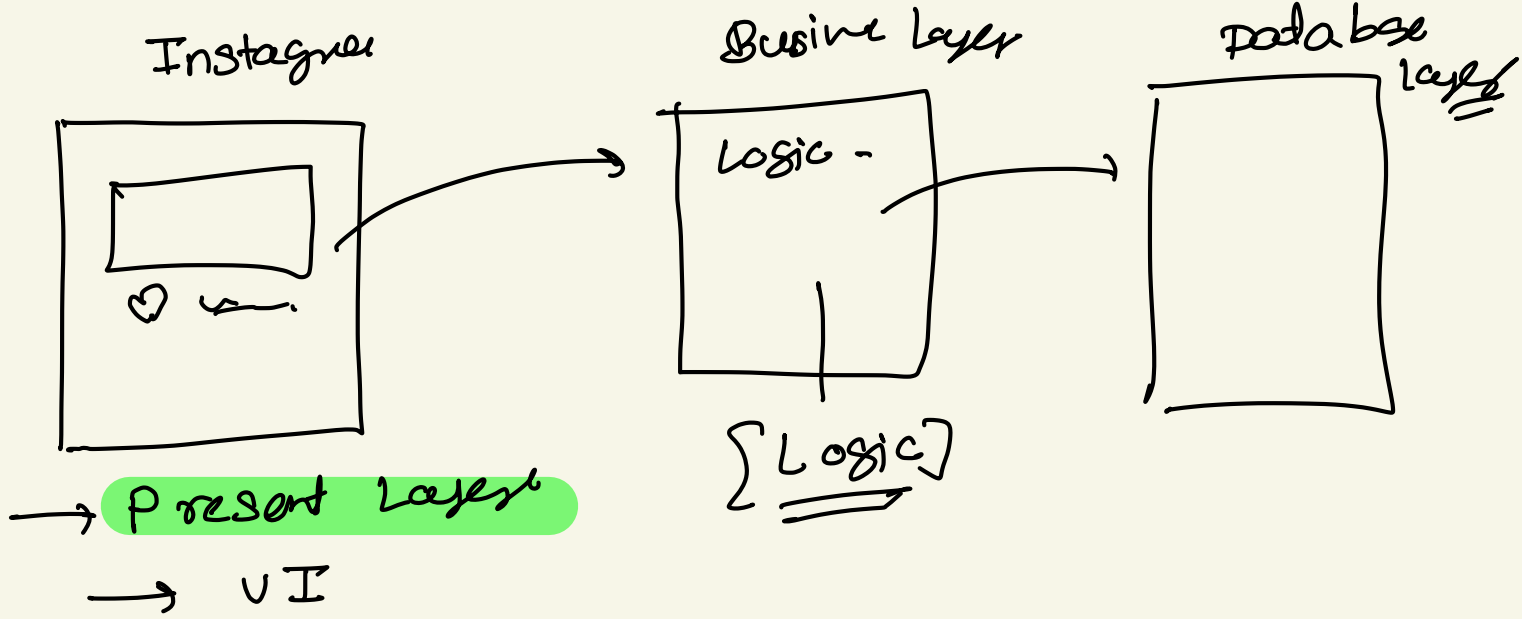
web Service → API which on Internet

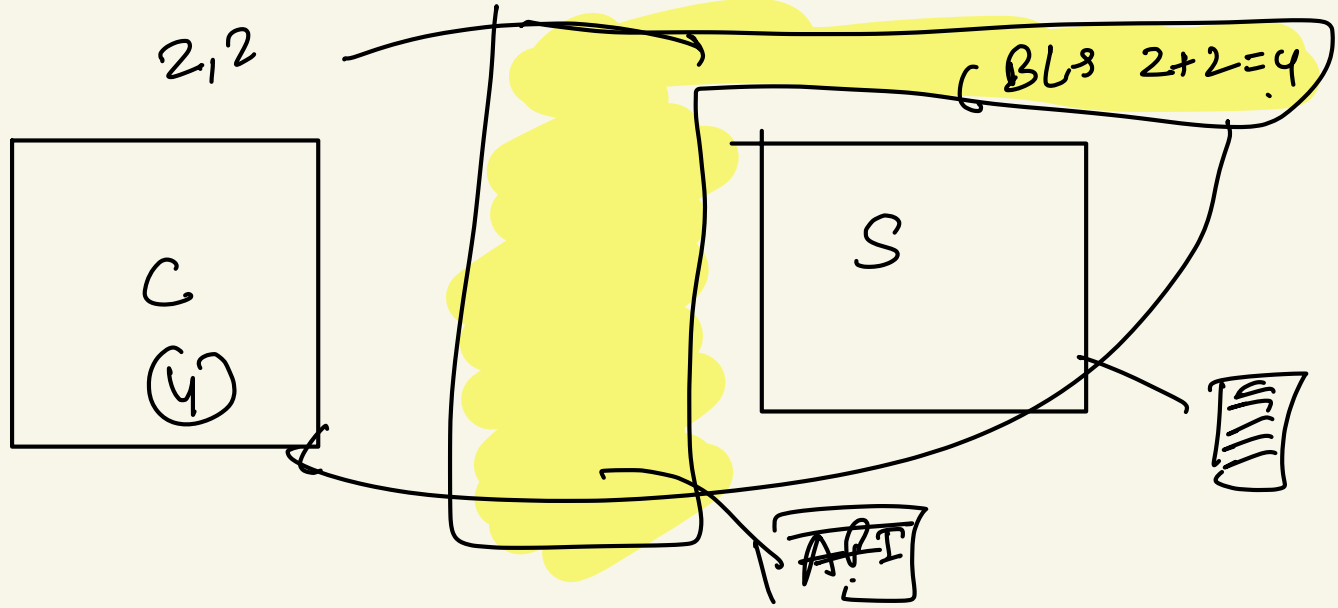
if not → API (not webservice)

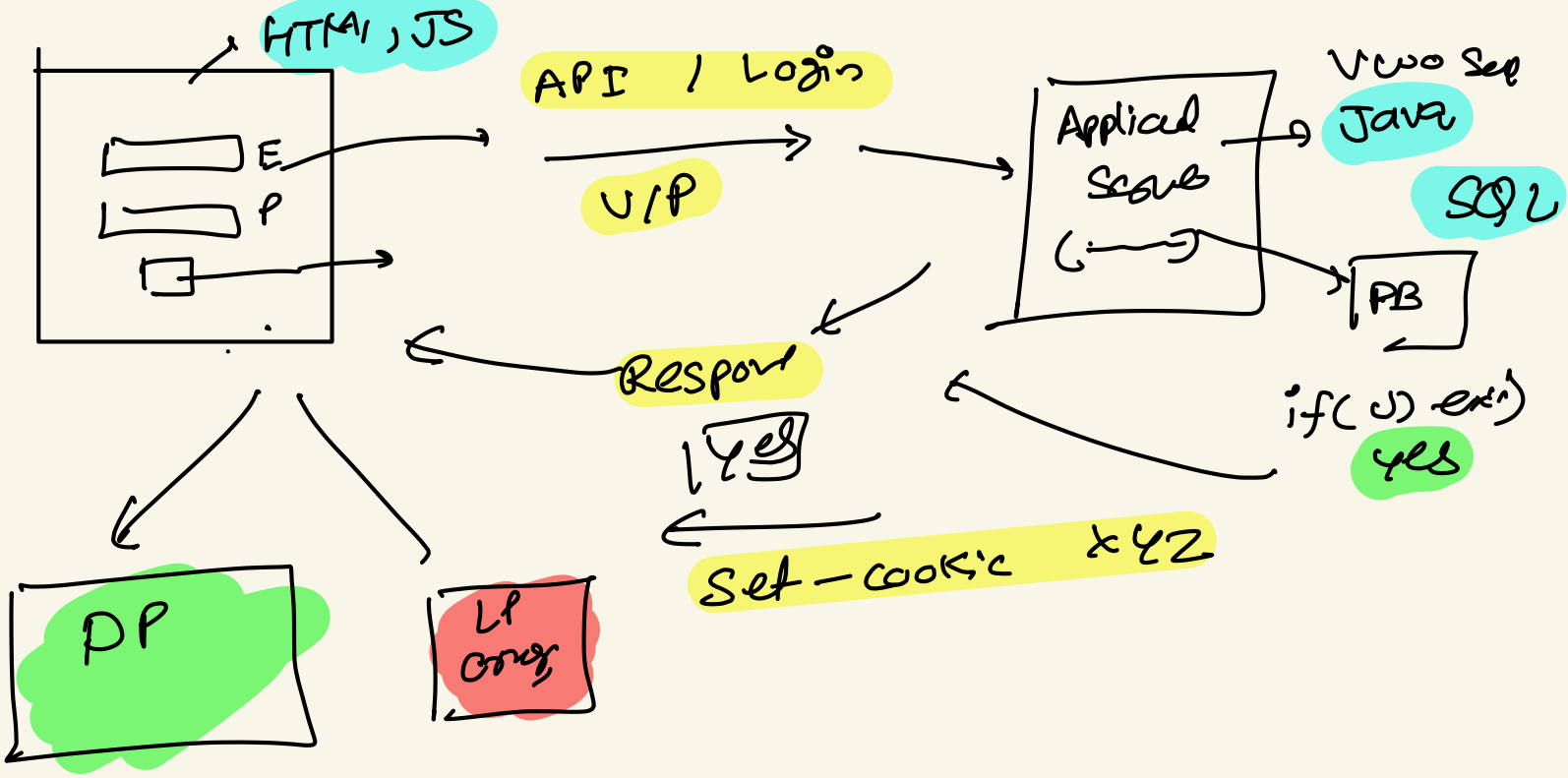


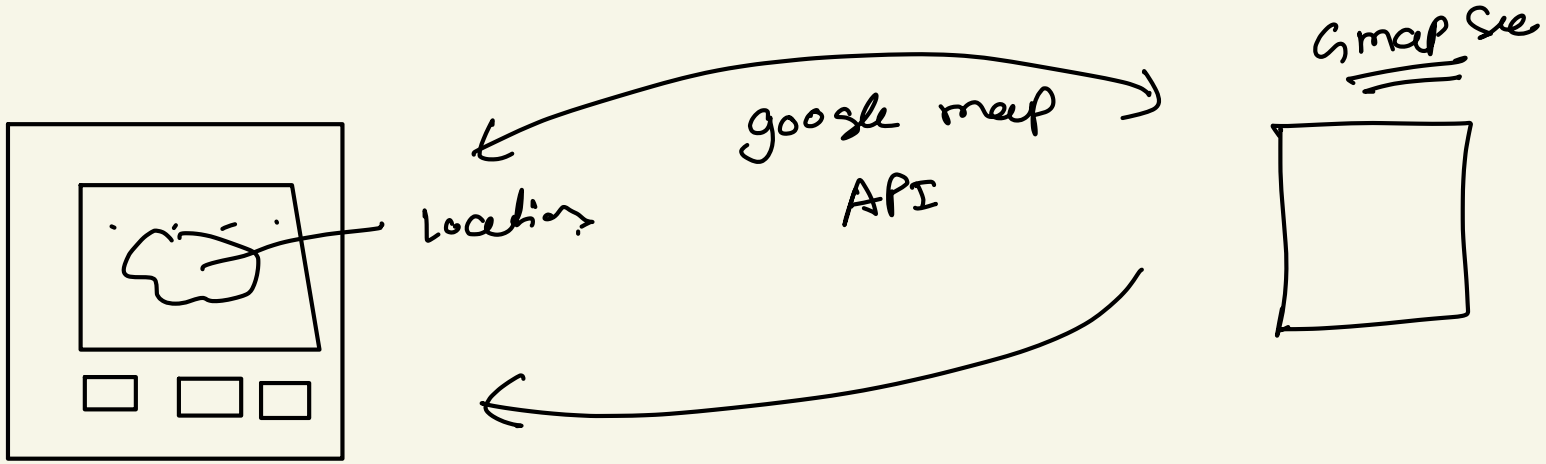

Instagram API
API
Post

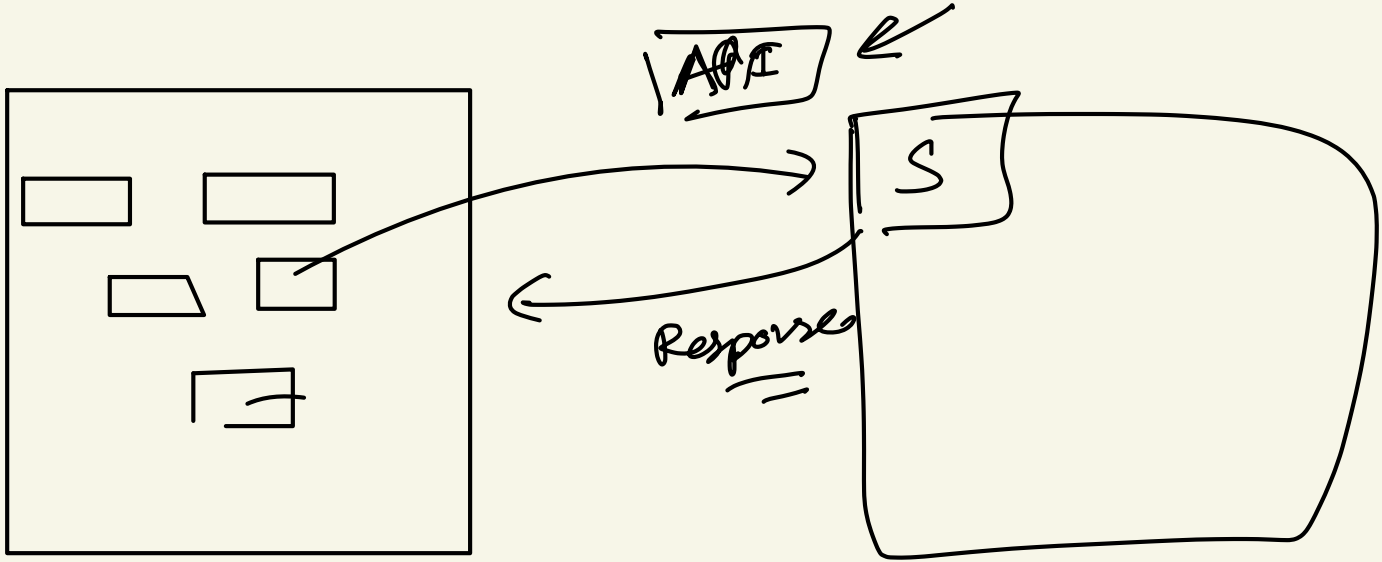
web application / Model (3L) ← CS ←

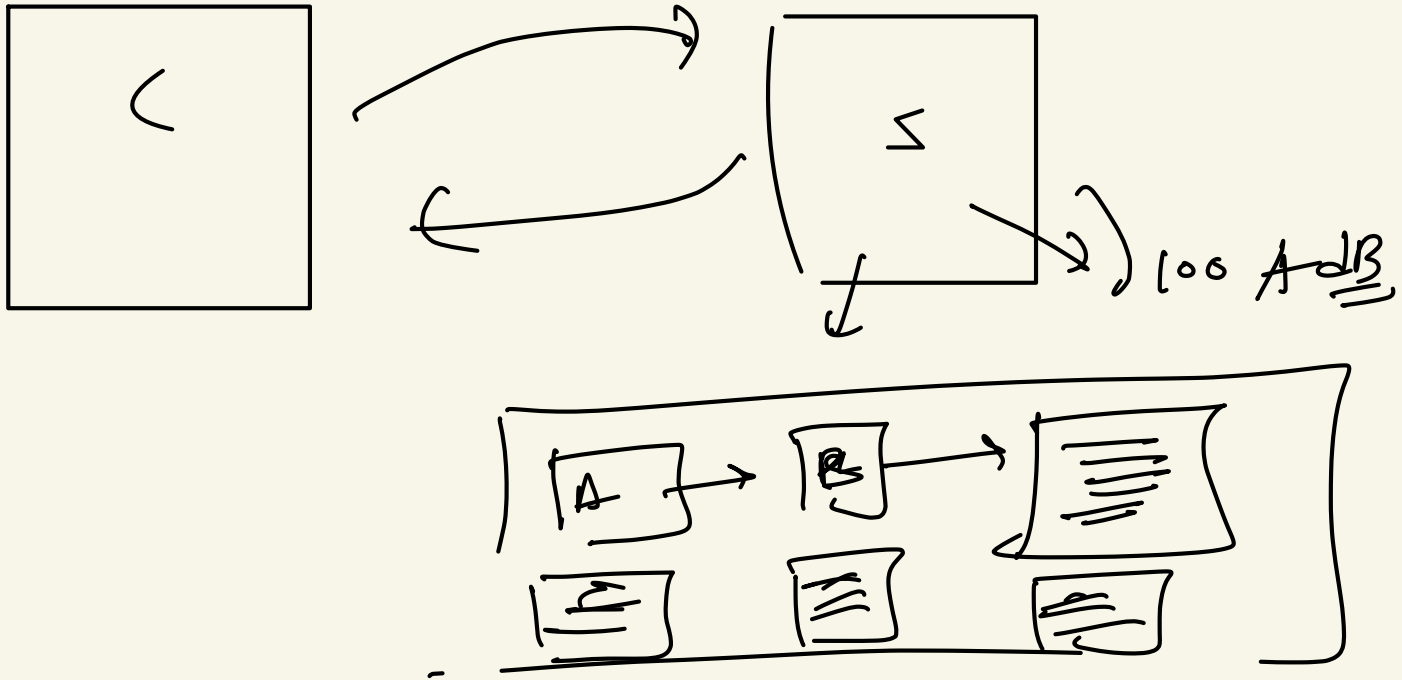


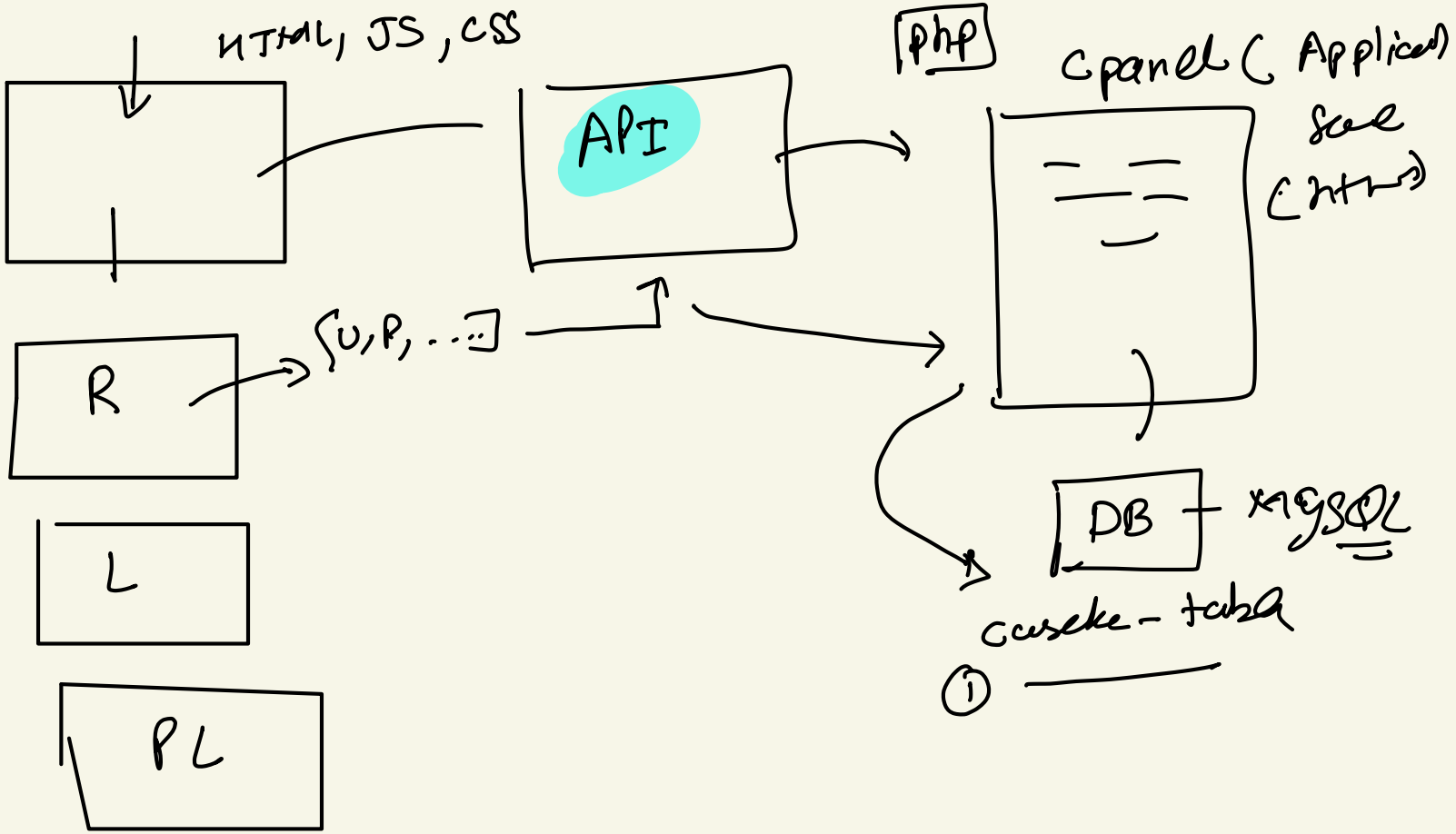

2,2
BL: 2+2=4
C
4
S
API

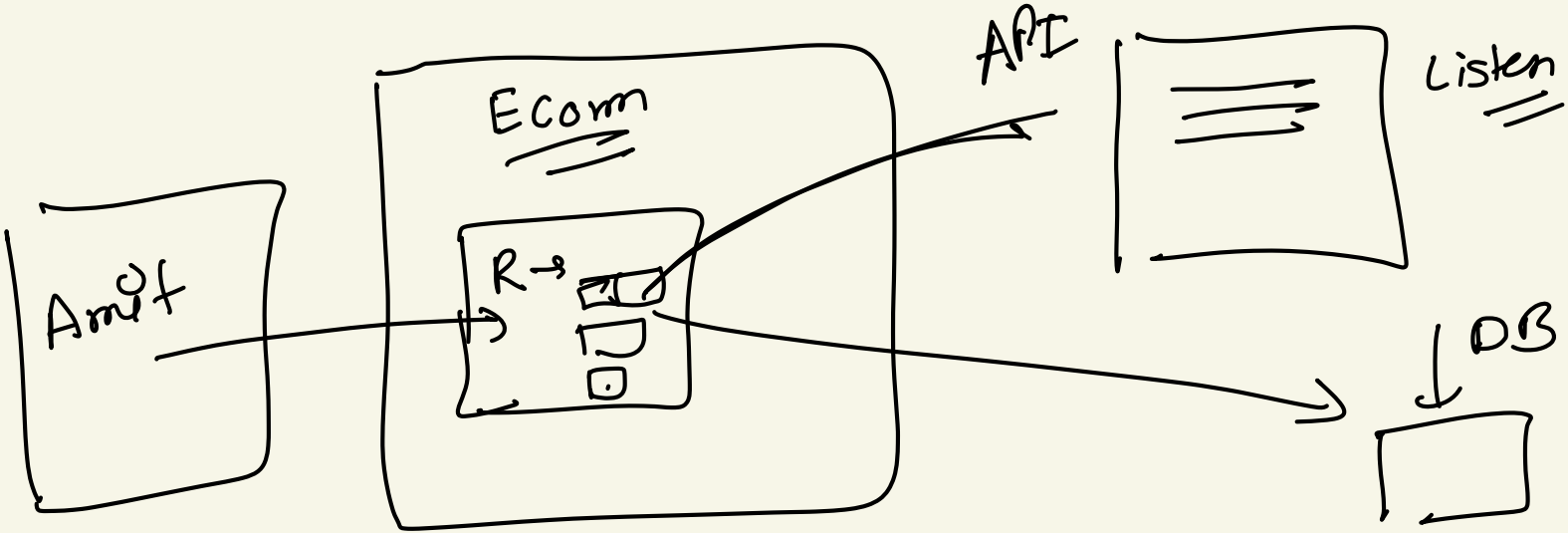

HTML, JS
E
P
API / Login
V/P
Applicat
Scrub
Vuo Scr
Java
SQL
PB
if( ) exi
yes
Respon
yes
Set-cookie xyz
PP
LP
Comp

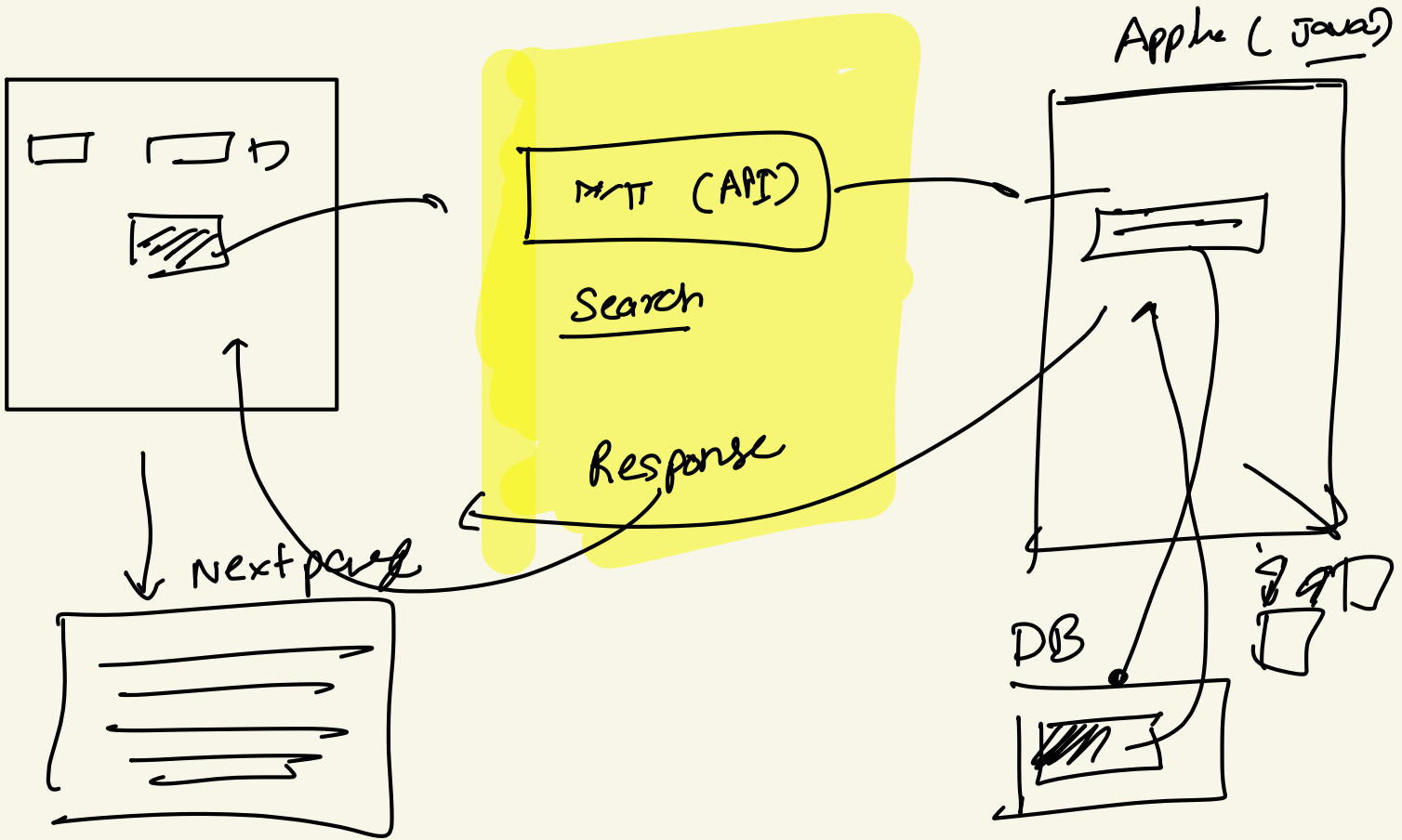

Gmap Src
google map
API
location


API
S
Response

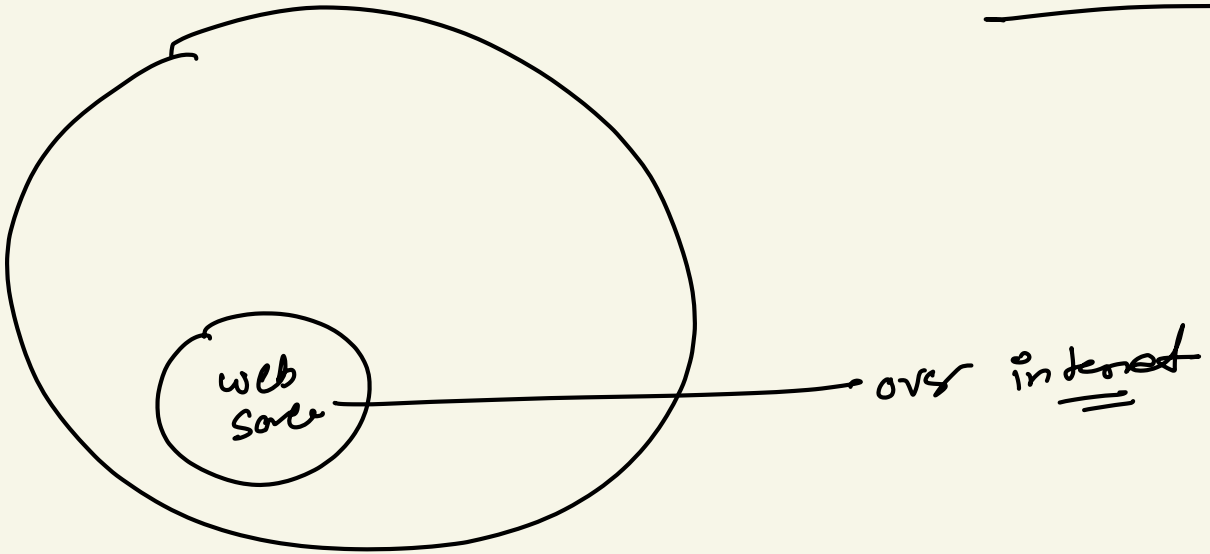

C
S
100
A
B

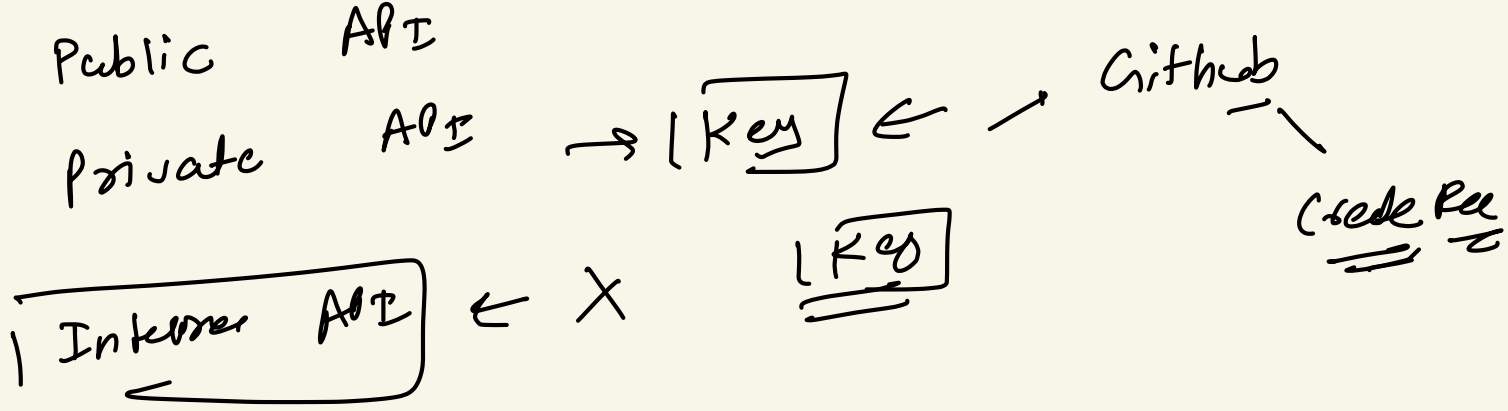

HTML, JS, CSS
API
php
cpanel (Applica
Scal
R
{U, P, ...}
L
PL
DB
MySQL
cuseke - tabla
1

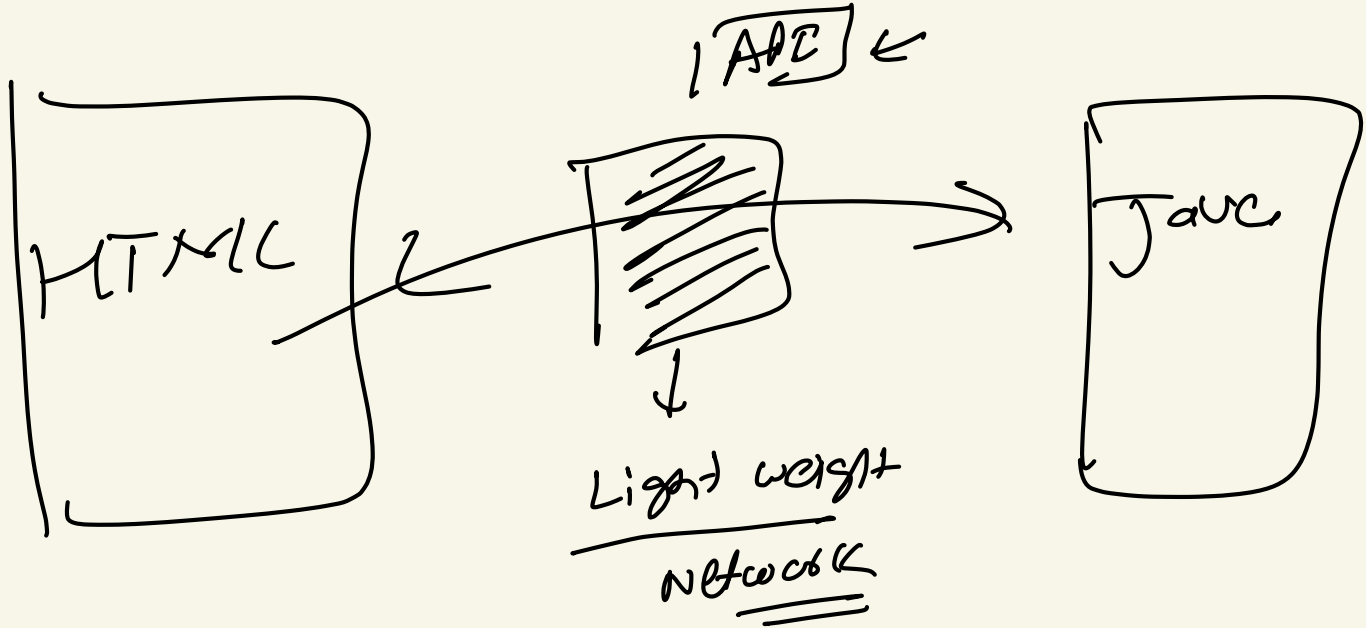

Ecom
Amit
R
API
Listen
DB


Apache (Java)
HTTP (API)
Search
Response
Next page
DB

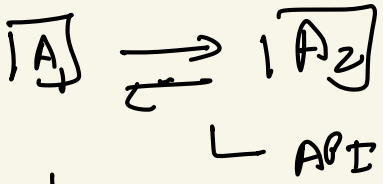
API - API → offline
online → web Service




Public API
Private API
Key
Github
Code Repo
Intruder API
X
Key


API
HTML
Java
Light weight
Network

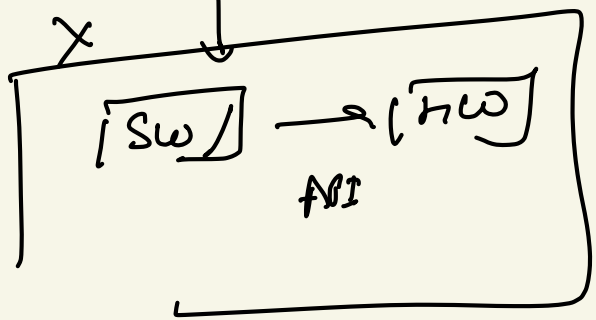
API →



offline

online

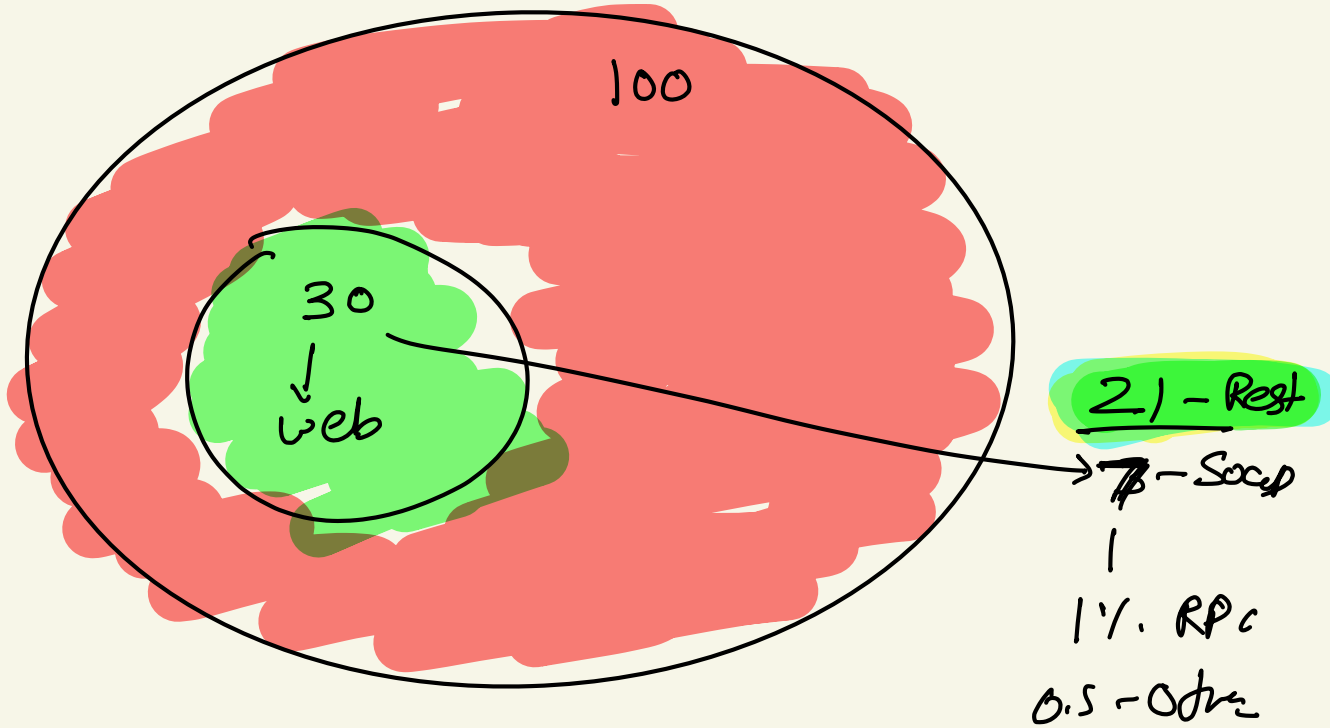
web service



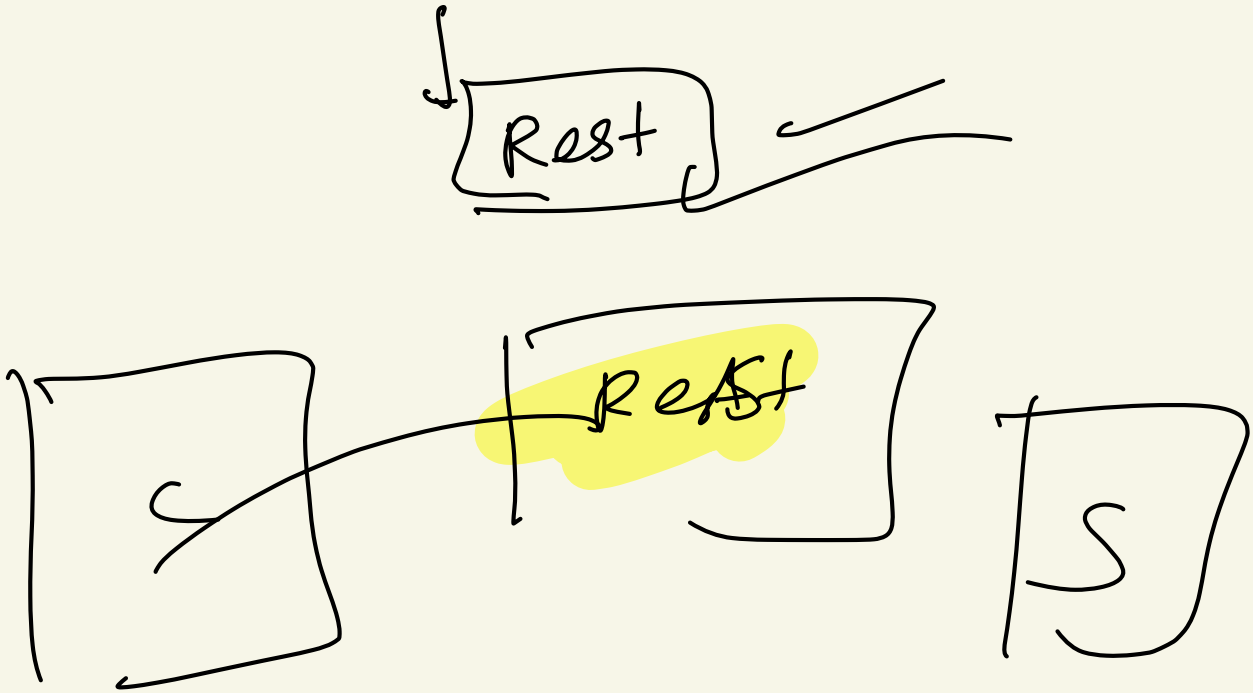
REST, 70%

Rest API

SOAP, XMLRPC ...Others

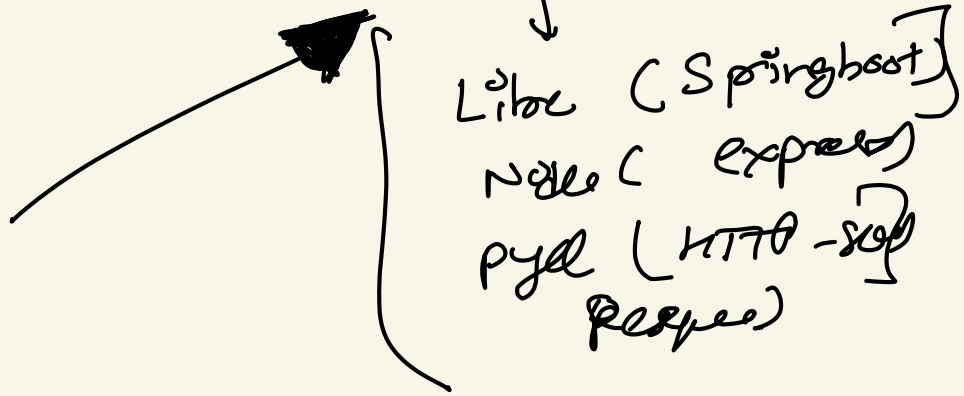

100
30
web
21 - Rest
- Soap
1% RPC
0.5 - Other

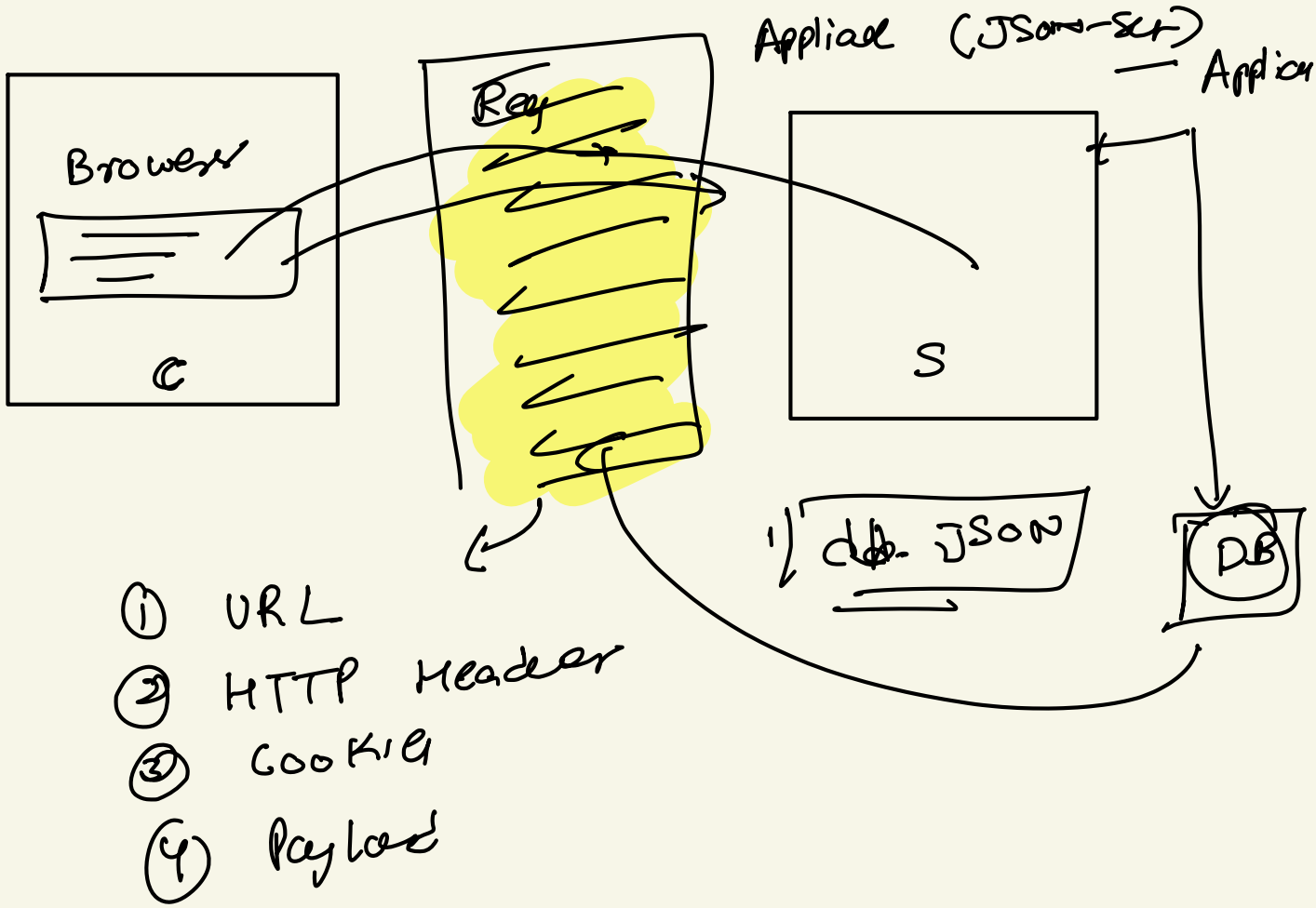
web- appln / mobile cell



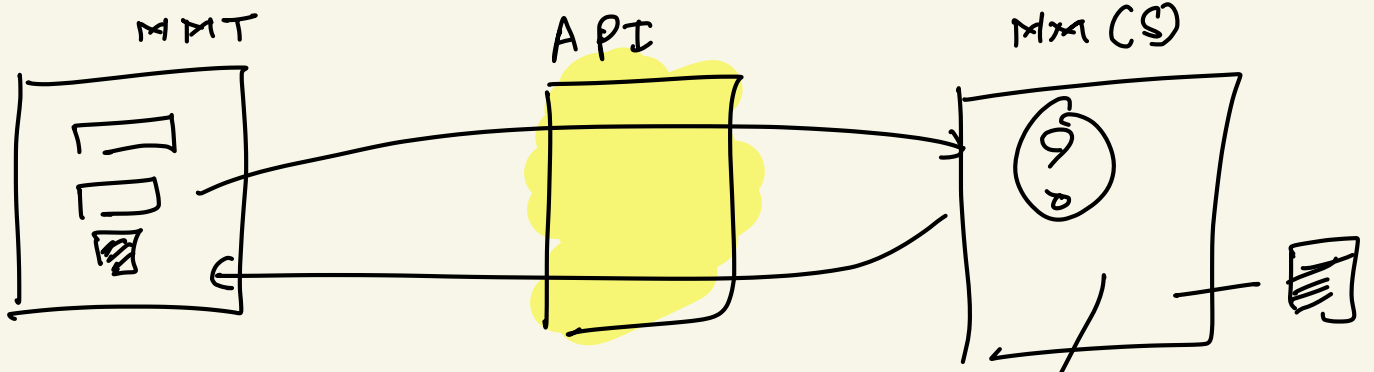
1. API → Rest
→ SOAP

who API — ? (delivers)




Applicat (JSon-Sch
Applica
Ray
Browser
C
S
db. JSON
DB
① URL
② HTTP Header
③ Cookie
④ Payload

①



① → Search ✓

② → Booking

Req → URL + Send data

↓

Payload

[makeMyTrip.com Server

→ Del-By

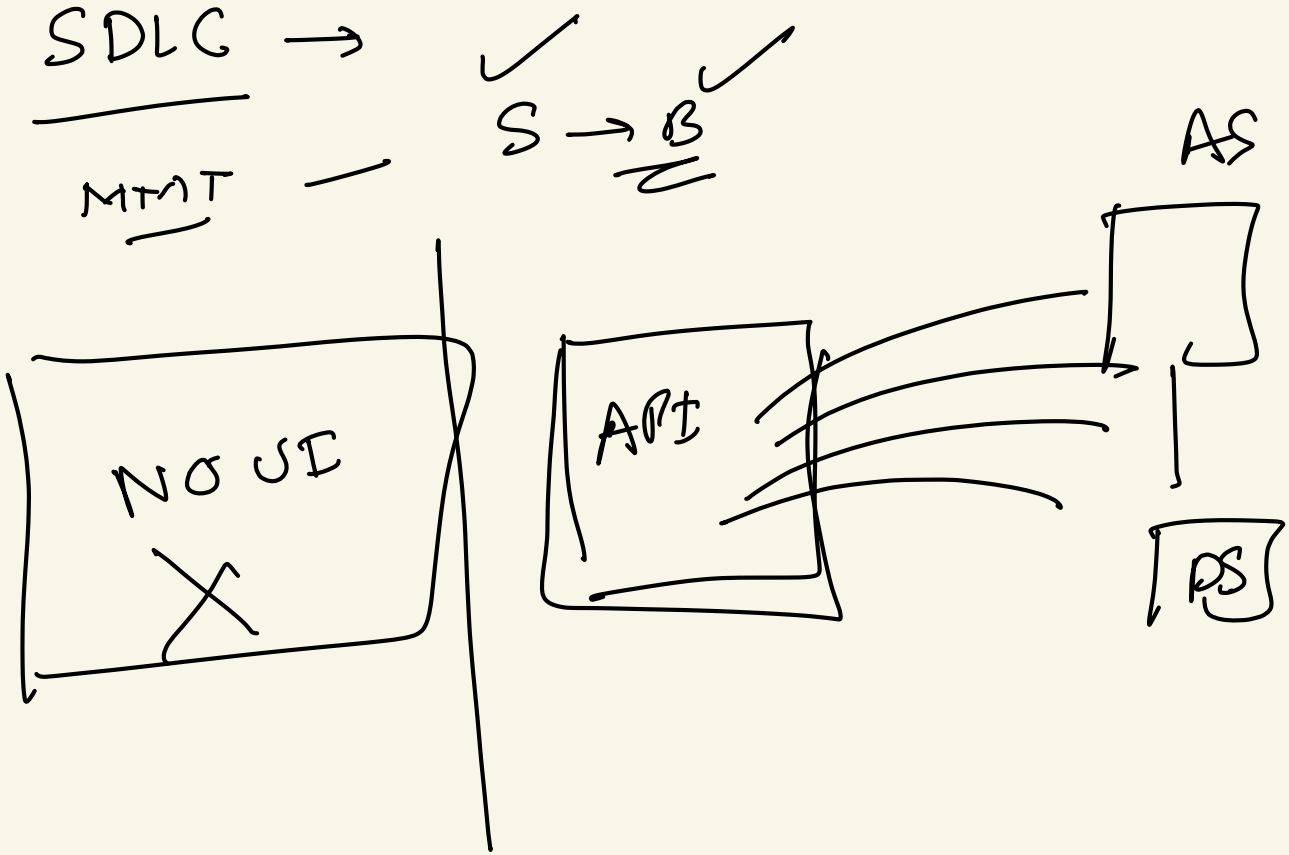
10 Aug-2023]

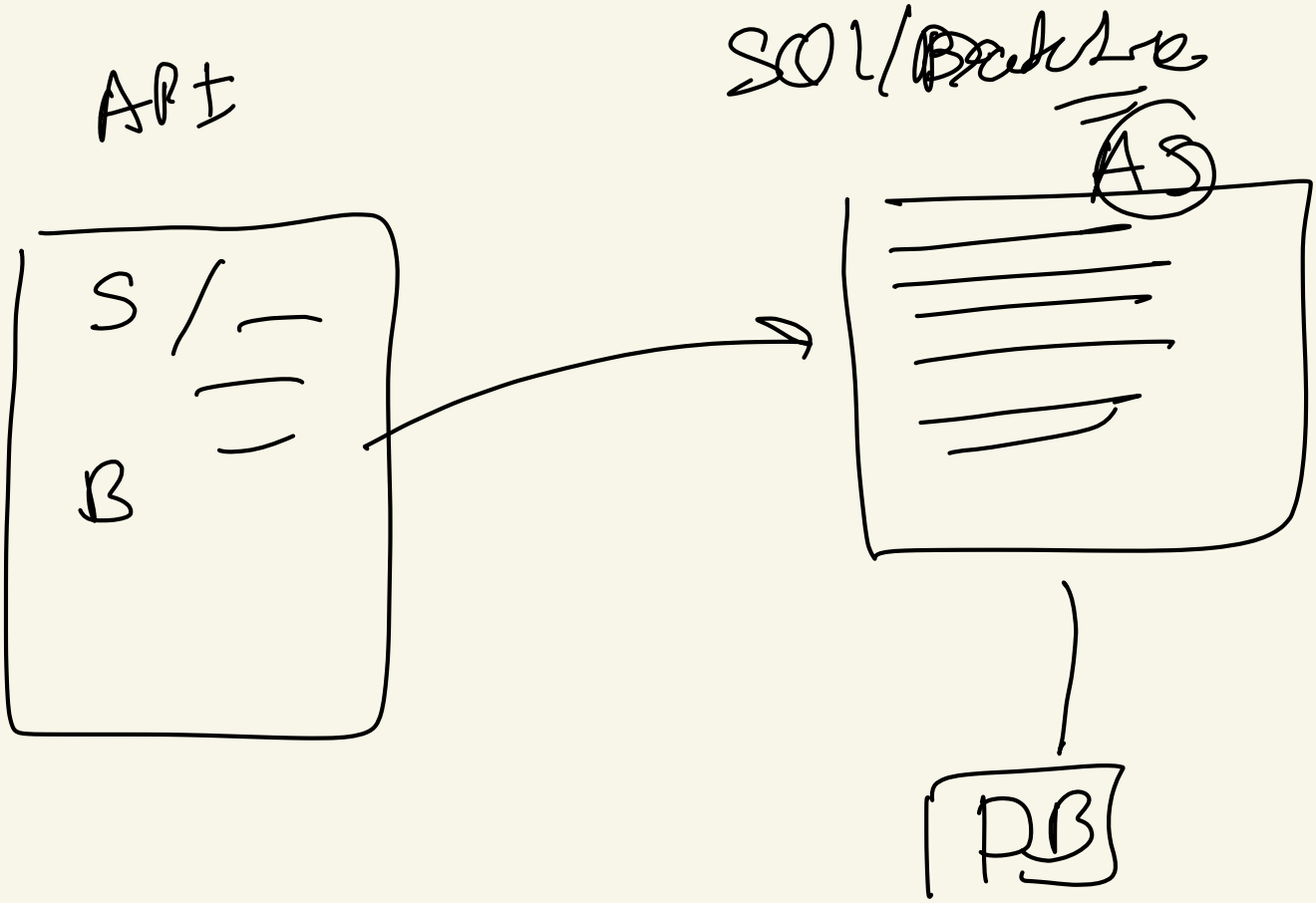
TC → STLC — TP

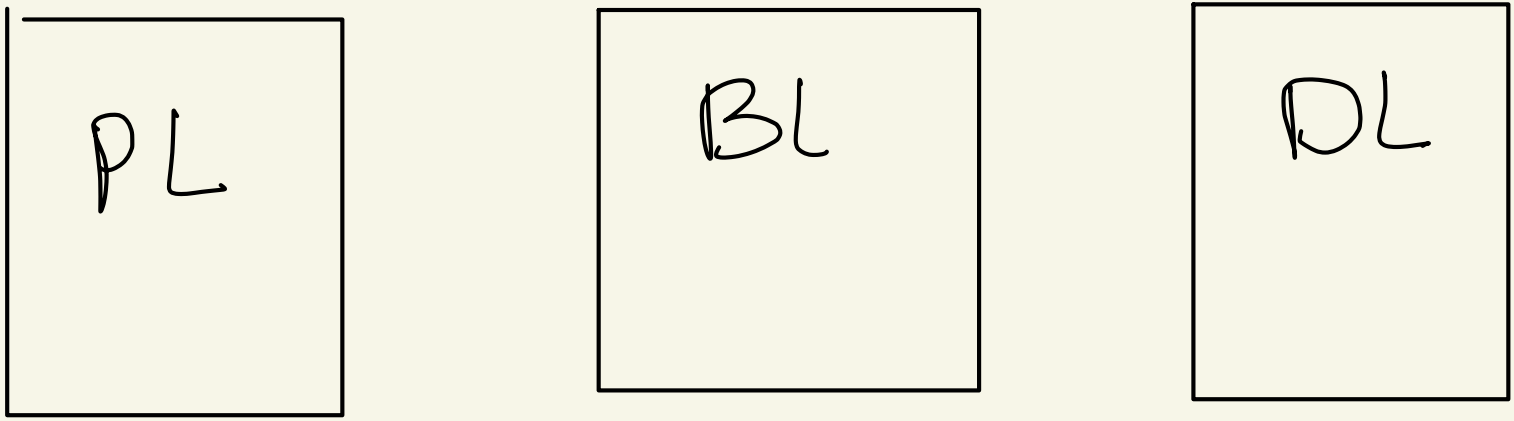
① → DE-Ba → 0 - (Null zero)

② → dob 2027 → % (Invalid) →

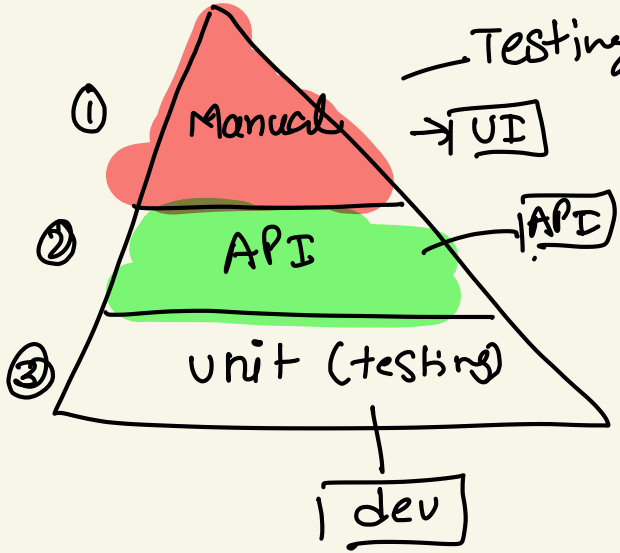
③ → Valid / Invalid


SDLC →
S → B
MTMT
AS
NO UI
API
PS


API
SQL/Backend
AS
S
B
DB


PL
BL
DL

Restful Booker :



Testing Pyramid

① → UI

② → API

③

① API → middle (Fun/Ri)

$A_1 \rightarrow A_2$

② why API

③ what test ? ✓

④ Rest / Soap) ?

BS / win → UI → Business Logic → Fail

UI → API

No UI

① test Fast

② Save time

③ early bugs

④ API is easy

what to test ?

Q3 ←

① Functionality I/V → TC + Data validation

② ↳ Request, Response → time

↘

Non-F → Performance (1000/s)

③ Error - handling → → 404

④ Security → 1 UI
2 UI → UX ← (J)

API – Manual → Automation ←

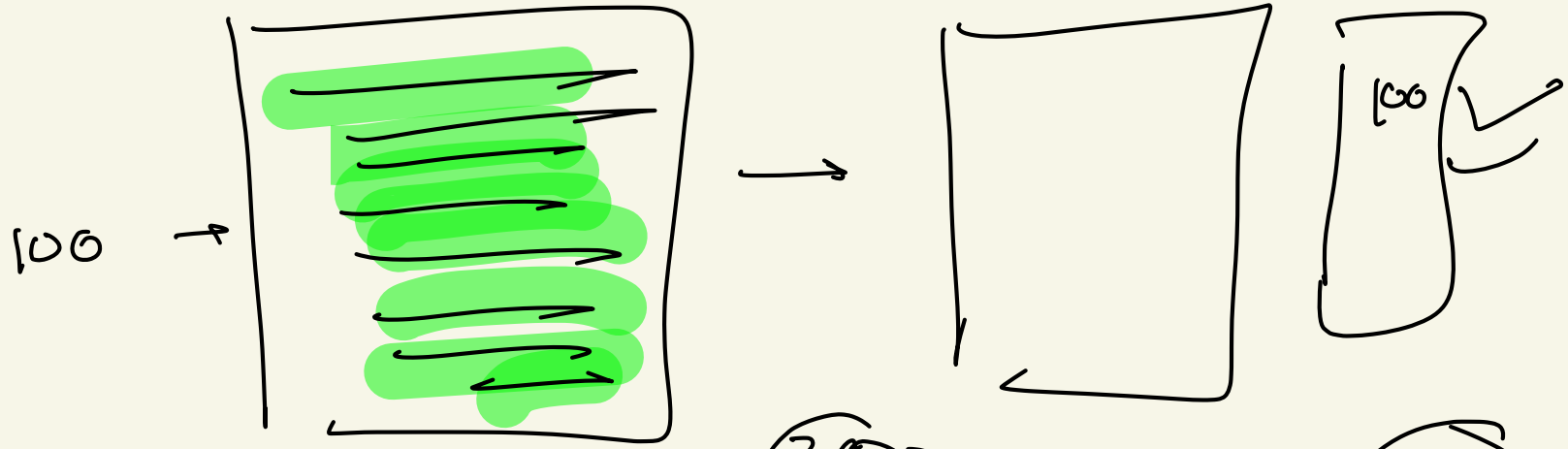
STLC →
0 – Req – Analysis
1 Test plan
2 Test cases → 100 →
3 Execute → Postman
4 Bugs → Fix
5 Test Closure

News 20 TC = 120 (100 Note)

Automated

Test → Inva V ee

Jenkis → QA



REST-ASSURE (2.0) + JAVA

Login → U/P — Login

Ex

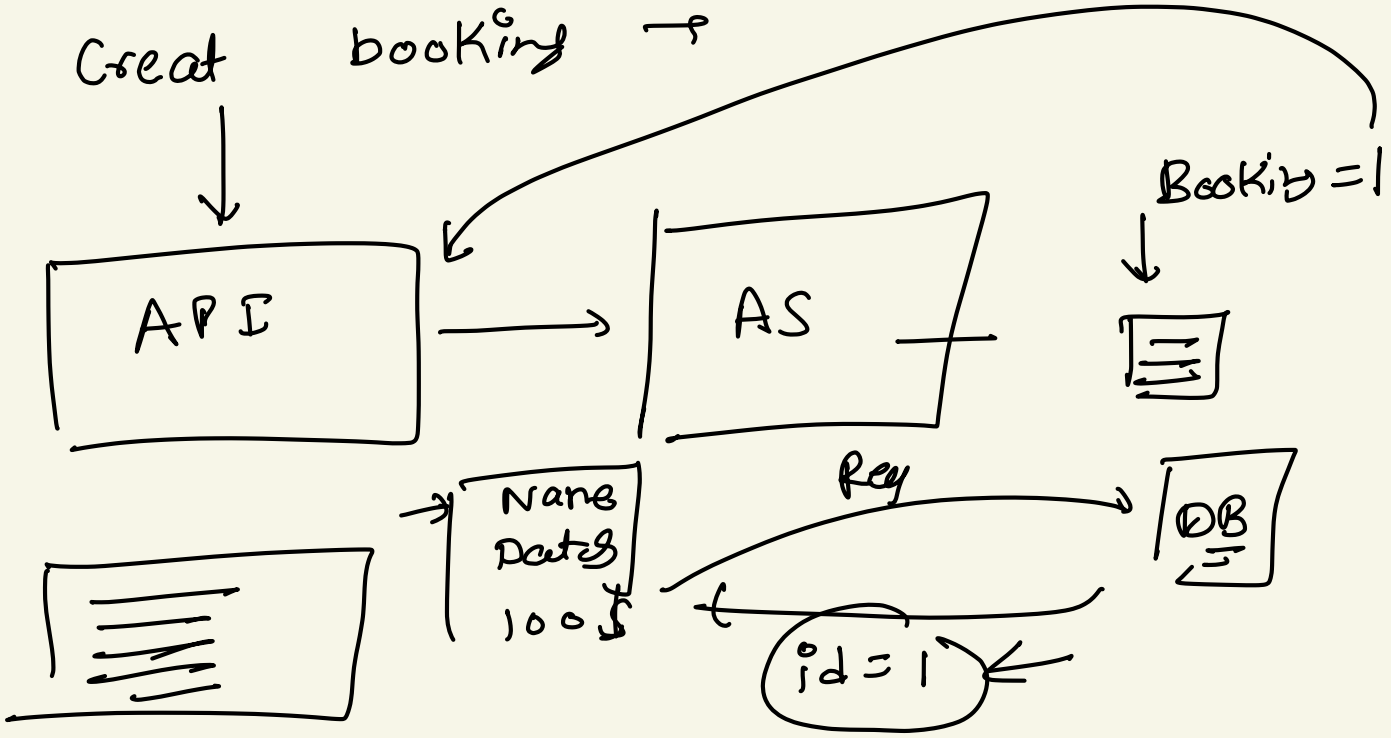
TC1 → valid — U/P → 200 OK

TC2 → In U/P → 400 error

↓

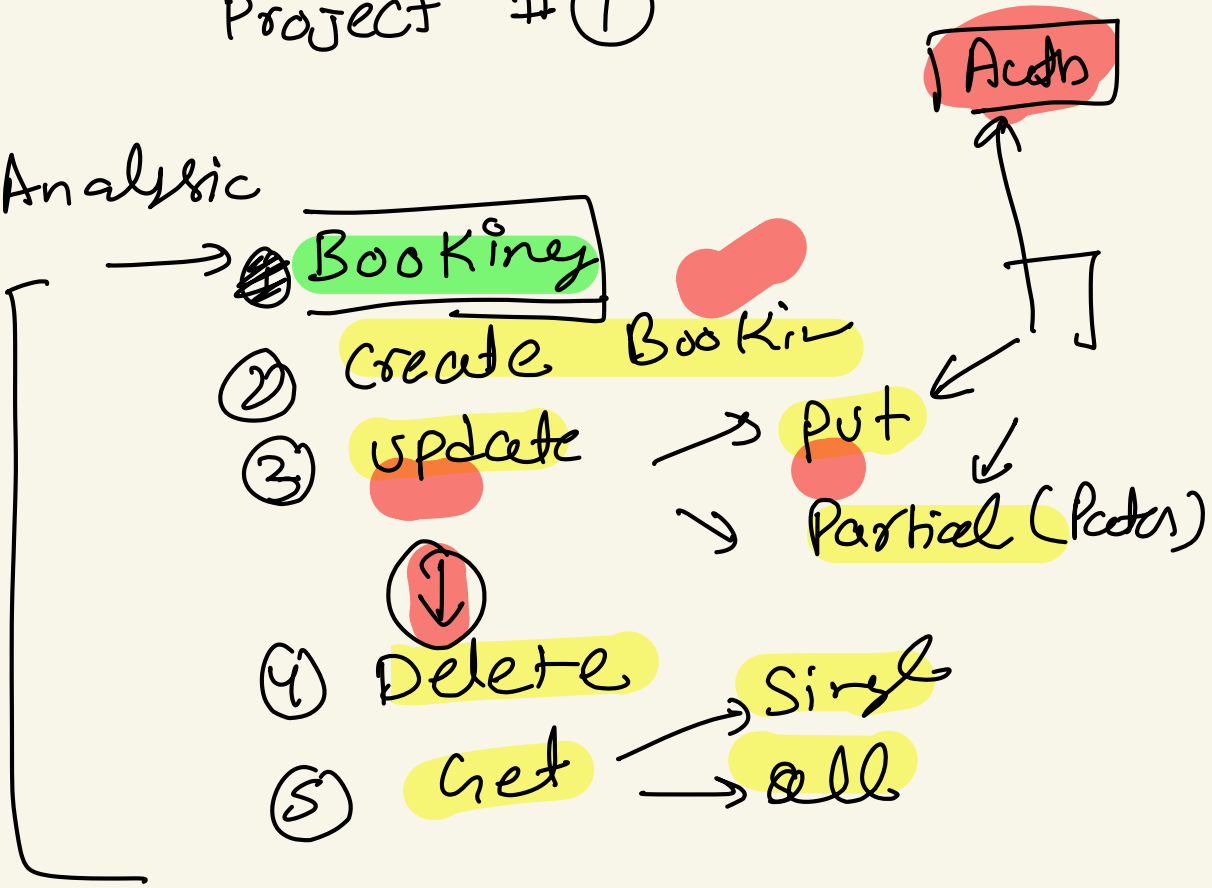
Restful Booker → API (NO UI) → UI ←

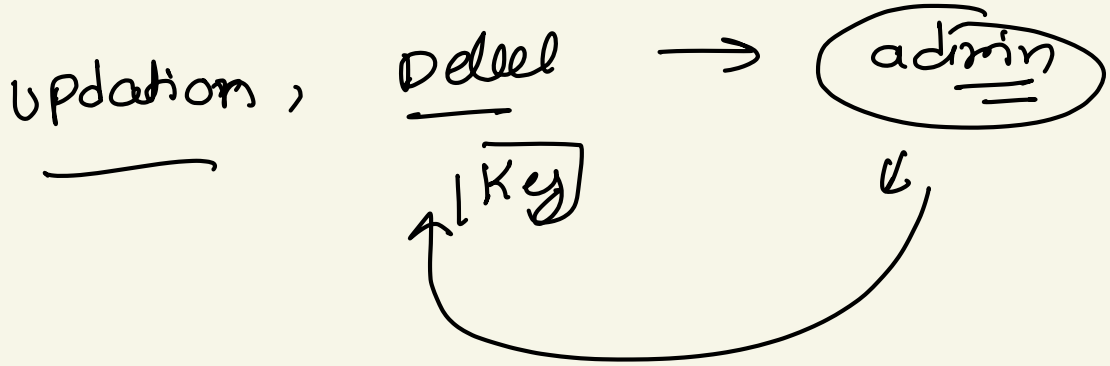
1. Creat booking →



Project #(1)

(1) Req- Analysis

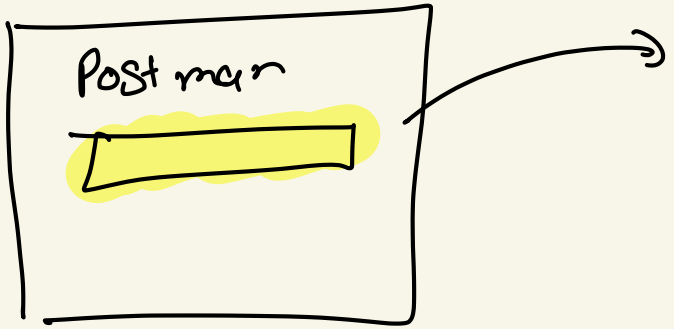



Updation ;
Delete
admin
Key

Rest API → HTTP METHODS →

1. Create Booking → POST
2. delete → delete + Auth → id
3. updation → Partial (Patch + Auth →)
   → Full (Put)
4. Get a Booking → single → id
   → all → id = x
   → GET

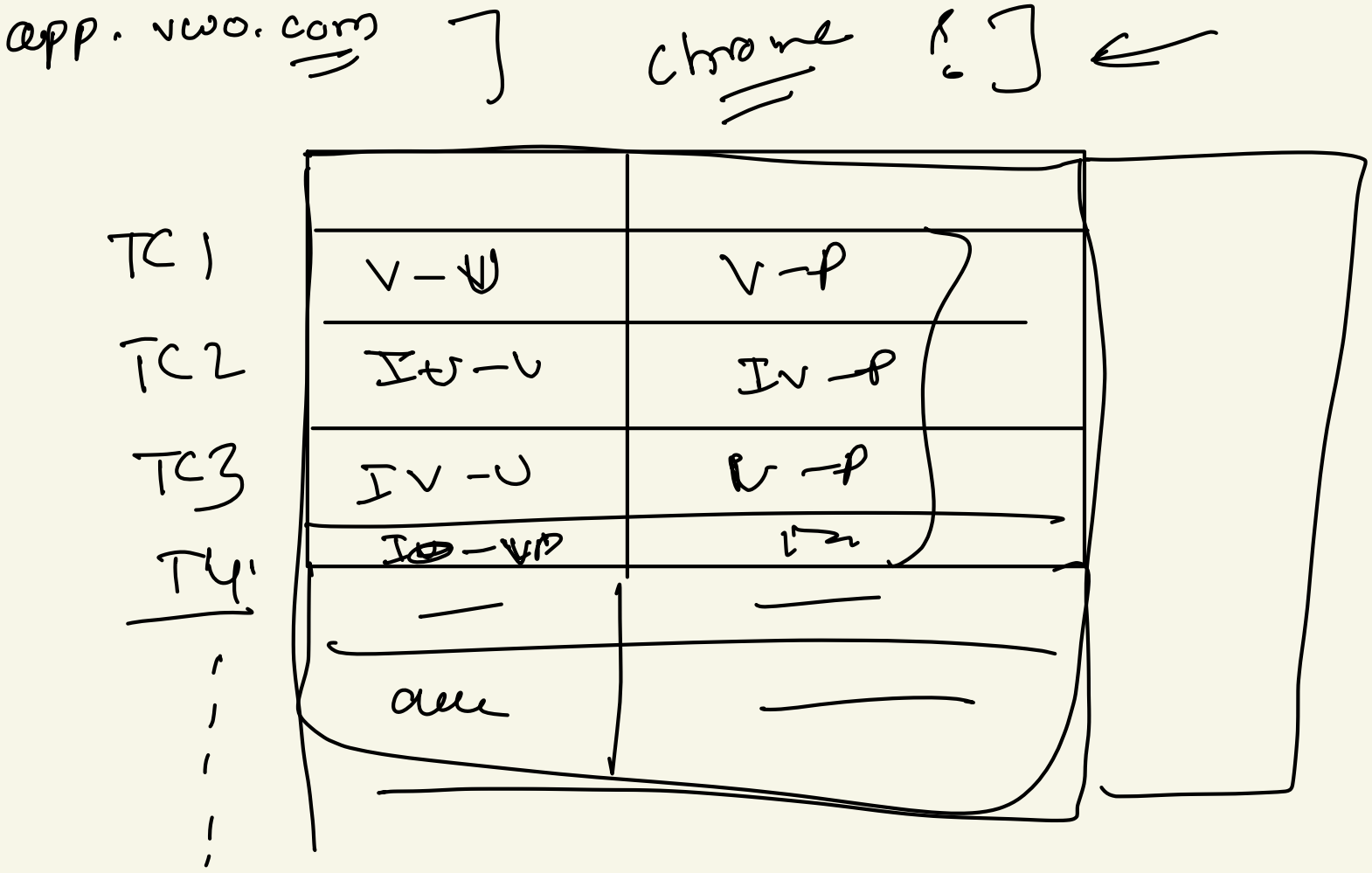
1. Create Booking →

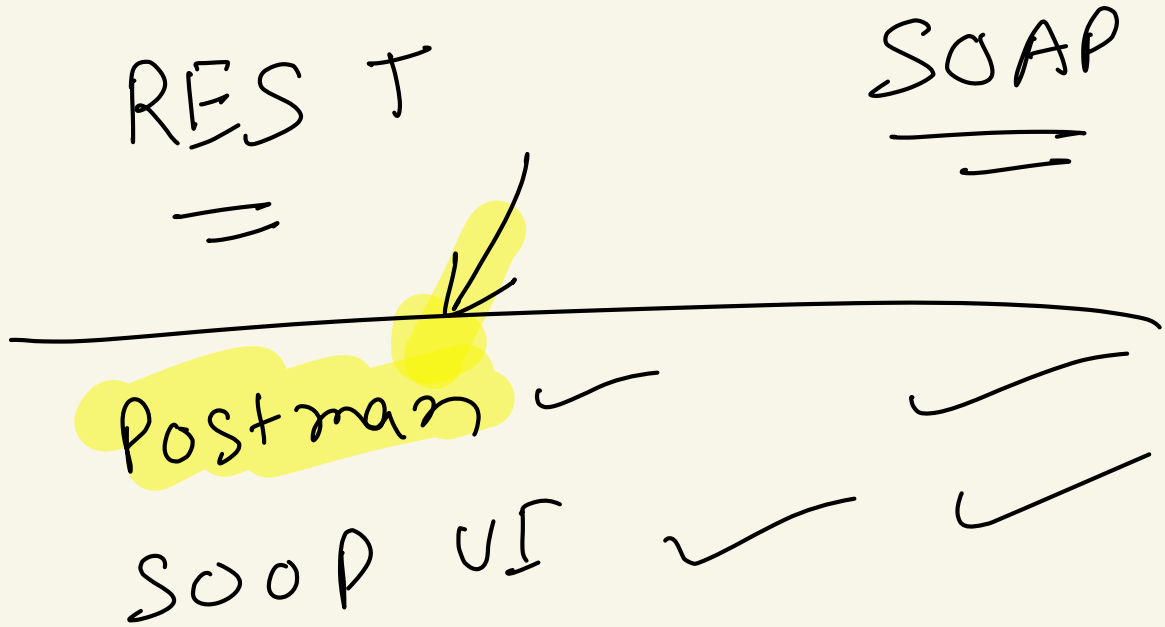


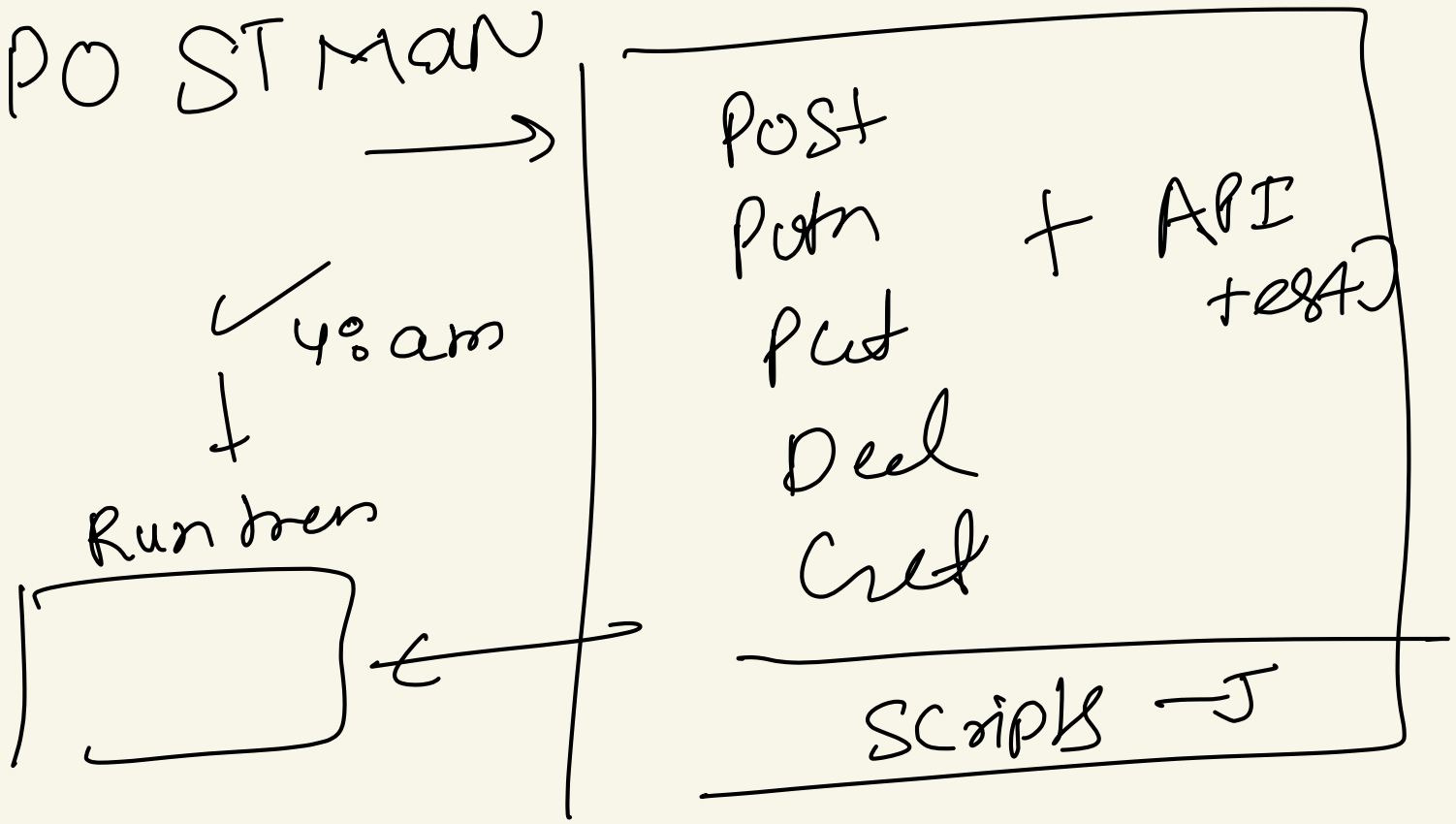
Valid

Invalid

→ → URL + Payload + headers


app.vwo.com
chrome
TC1
TC2
TC3
TC4
V-W
V-P
IV-U
IV-P
IV-U
IV-P


REST
SOAP
Postman
Soop UI


POSTMAN
Post
Patch
Put
Del
Cred
+ API
test
4:00 am
Runners
Scripts

C — create ⟶ POST

R → Read ⟶ Get

U → update ⟶ Put/Patch

D → Delete ⟶ Delete

QA Lead → ① Test plan – QA Lead

② test case

manual testing → ③ Postman – executed

④ Bug – Fix

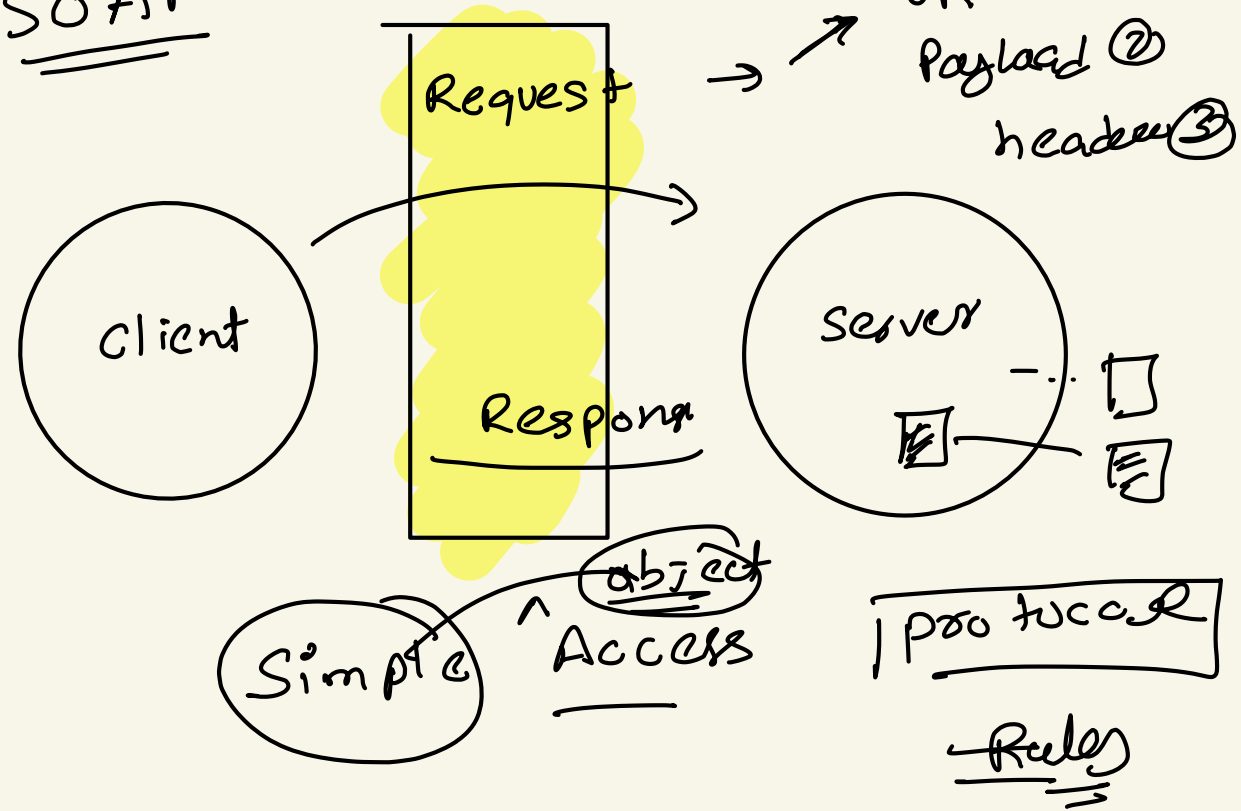
⑤ Test – de → Acc 100

20

API Testing → → MT
→ AT → Scripting

REST / SOAP / XML-RPC

→ SOAP



1. Soap → Microsoft → Restricted +
Secured
heavy
visit

① XML — ~~JSON~~ — XML
[tag Based language]
XML — tag
HTML

② Post — C → S
— J → Get, Post, Put, Delete
→ unissued

XML → name: Pramod

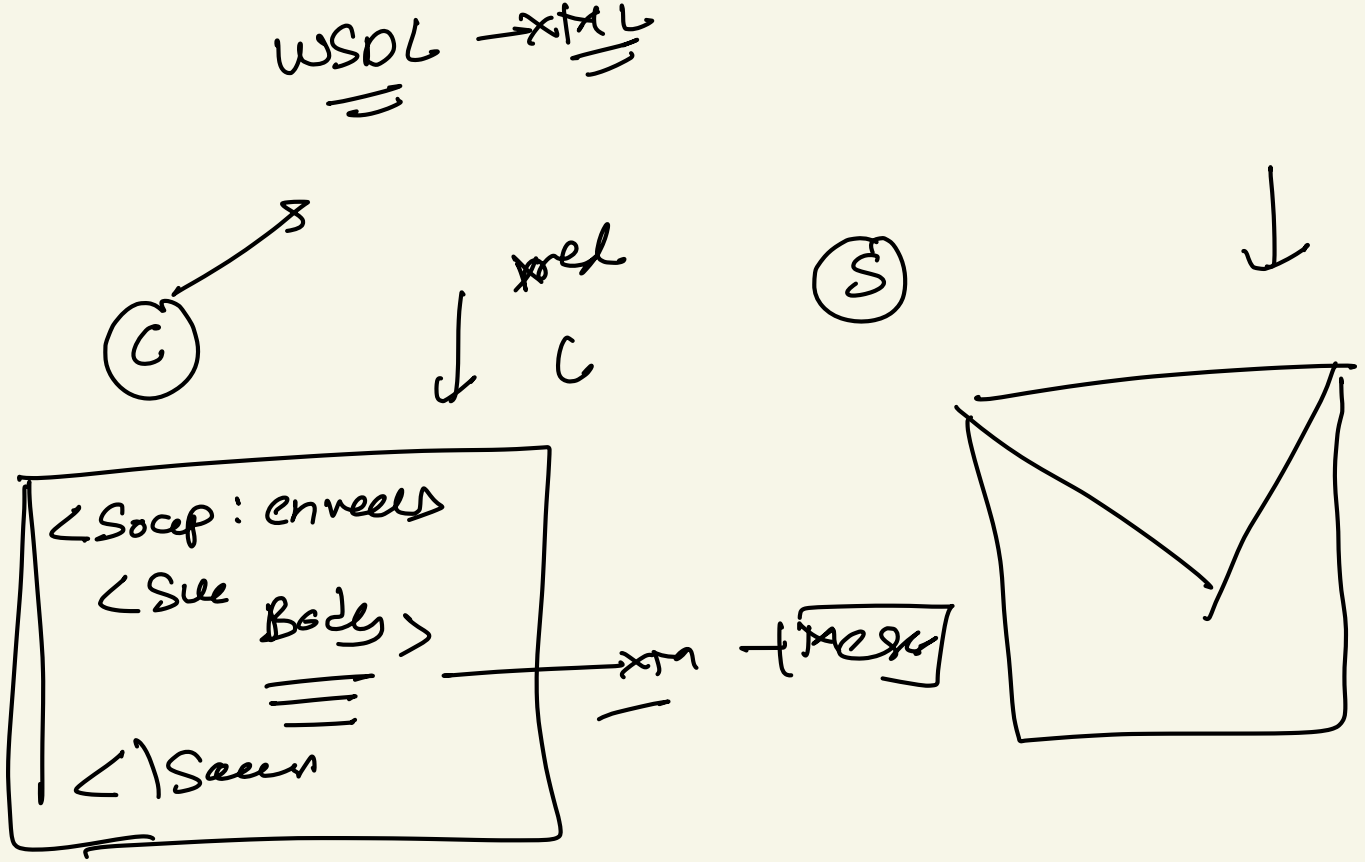
<?xml 1:0>

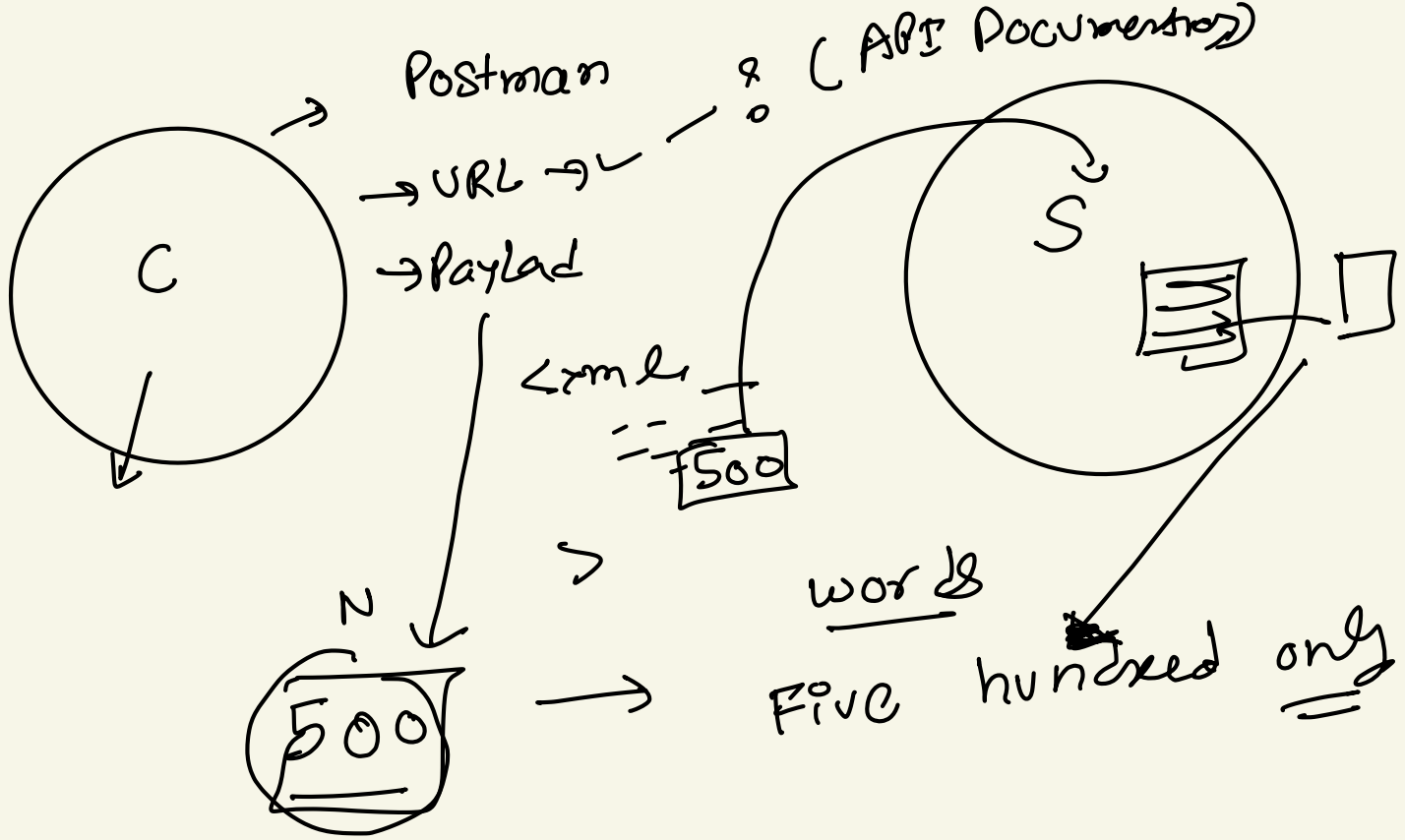
<name>

Pramod

</name>

</xml>


WSDL
C
S
<Soap:Envelope>
<Soap:Body>


Postman
& (API Documentation)
C
URL
Payload
S
<xml
500
N
500
words
Five hundred only

ER = AR

① 500

② -1

③ Blank

④ wrong URL

⑤ without header

⑥ → @

⑦ 123abc

$, Font, &

⑧ → wrong xml

⑨ → Screen(Title)

⑩ → 100_!

⑪ → 100,00

⑫ 100.01

⑬ :), Afg