Agenda.

→ Authentication

→ Authorization

→ RBAC (Role based access control)

→ How Authentication flow works. (tokens)

⇒ What is Authentication?

Scaler.com/academy ✓

⇒ Anyone can access without telling who they are.

Scaler.com/meetings/ ——

⇒ To access this page we need some kind of permission

⇒ leetcode.com/problems ⇒ No need to tell who we are.

——

——

——

——

——

——

⇒ Solve a problem. ⇒ Can't solve a problem without telling who we are.

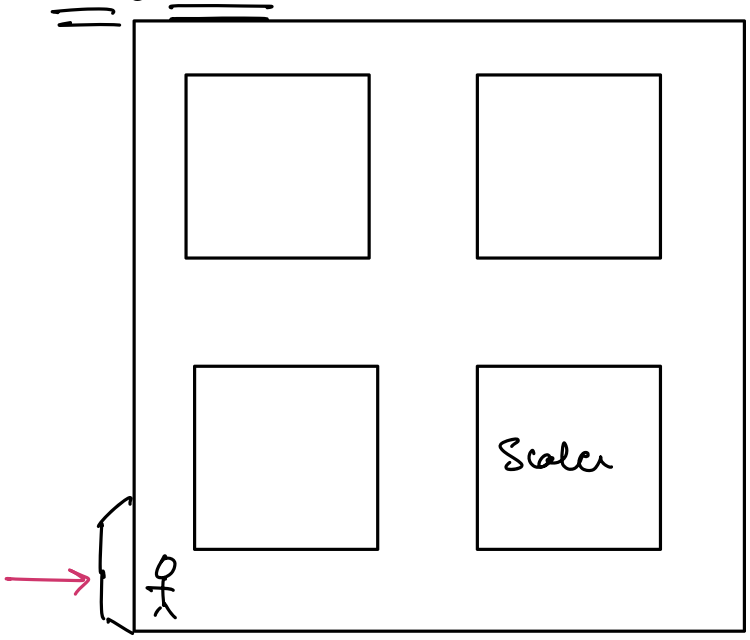
Scaler.com.

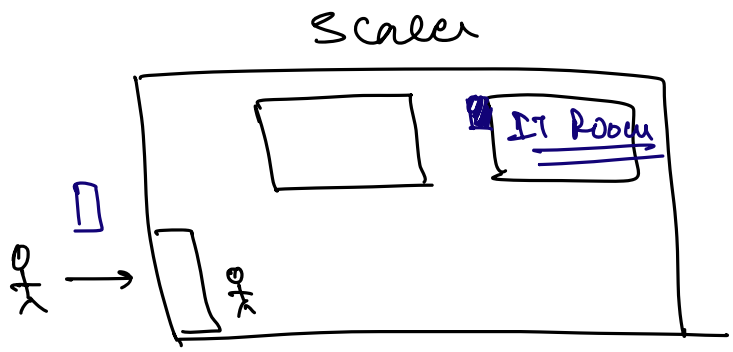
Scaler.com/meetings/ ——— } To access this URL we need some extra permissions.

Scaler.com/admin/ ———

Only users with admin role associated can access this page.

Tech Park.





⇒ Anyone with a valid Id card will be able to enter inside the office. But to enter inside a specific meeting room we may need some extra access/permissions.

AUTHENTICATION.

⇒ Telling the server who you are.

⇒ Process using which server identifies who you are?

AUTHORIZATION.

Authentication

+

Do you have an appropriate permission

scaler.com/admin/ ——

Authorization ⇒ Authentication + Roles/Permissions

Roles.

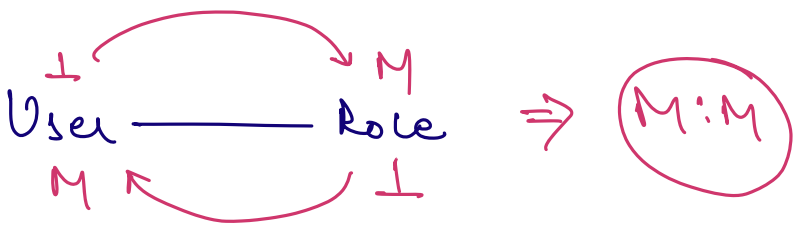
Authentication ⇒ Email / Phone No + Otp.

RBAC — Role Based Access Control.

users

roles

Id	Value
1	STUDENT
2	TA
3	MENTOR
4	INSTRUCTOR



⇒ mapping table

user_roles.

user_id	role_id
1	1
1	2
1	3
⋮	⋮

scaler.com/admin/ ——

↓

user_id

⇒ Movie Booking Platform.

	Authentication	Authorization
(1) See the availability of a particular show	✗	✗
(2) Book a ticket	✓	✗
(3) Cancelling a ticket	✓	✓

⇒ scaler.com | topics | courses | ——

↳ No Authentication.

if user is trying to watch a recording.

⇒ Authentication

How Authentication flow works.?

↳ Email + password

↳ Phone + Otp.

Whenever any user wants to verify themselves.

⇒ (id) ← Email
Phone

⇒ Some way to verify if that id belongs to you or not.

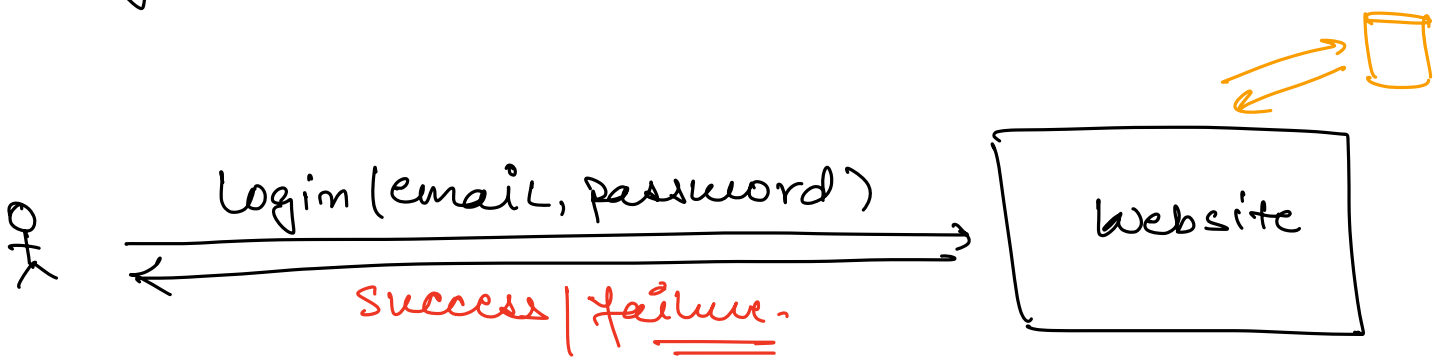
OTP Password

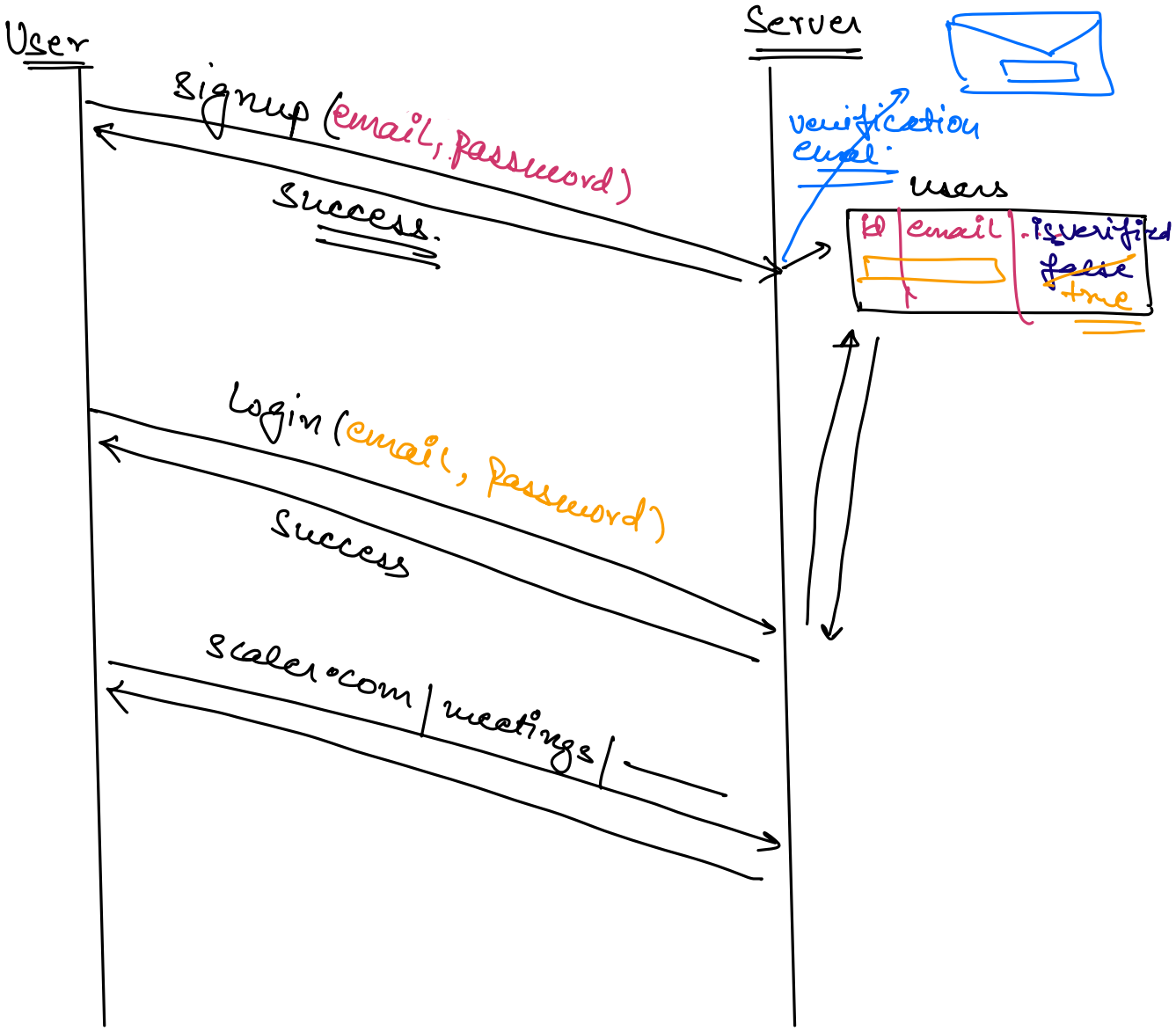
① SignUp(email, password)

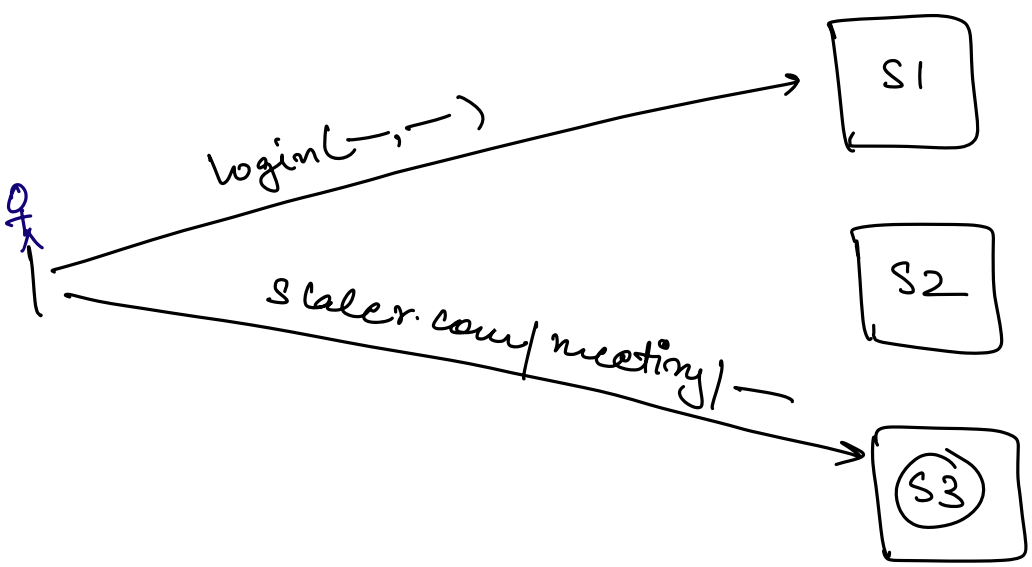
⇒ Verify by sending a verification link to check if the given belongs to the user or not.

⇒ Creates a user with the given credentials.

② Login.

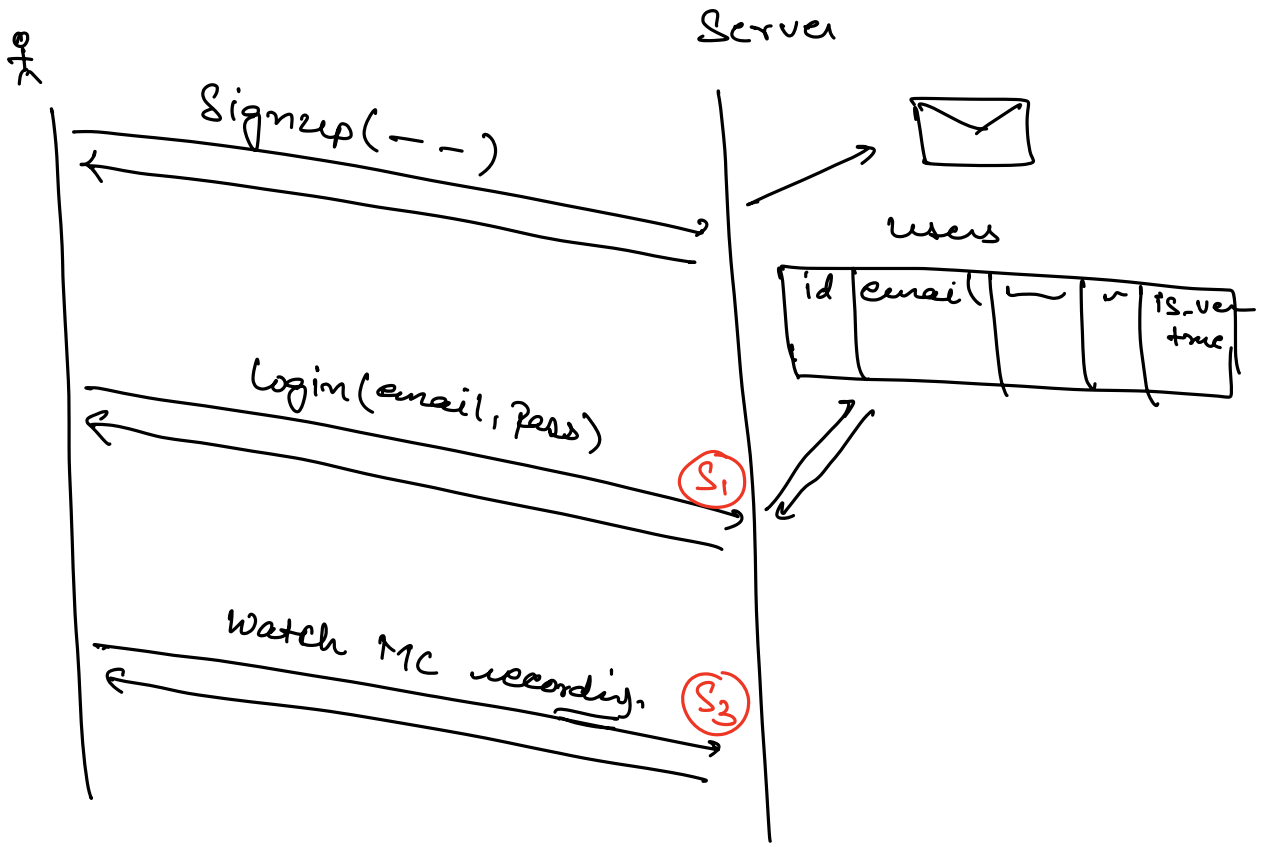



User
Server
signup (email, password)
success
verification email
users
id
email
isverified
false
true
login (email, password)
success
scaler.com/meetings/


S1
login(—,—)
S2
scaler.com/meeting/—
S3

⇒ Every request should be self sufficient.

⇒ Stateless.

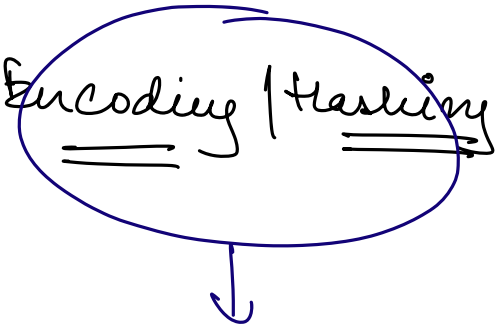


users

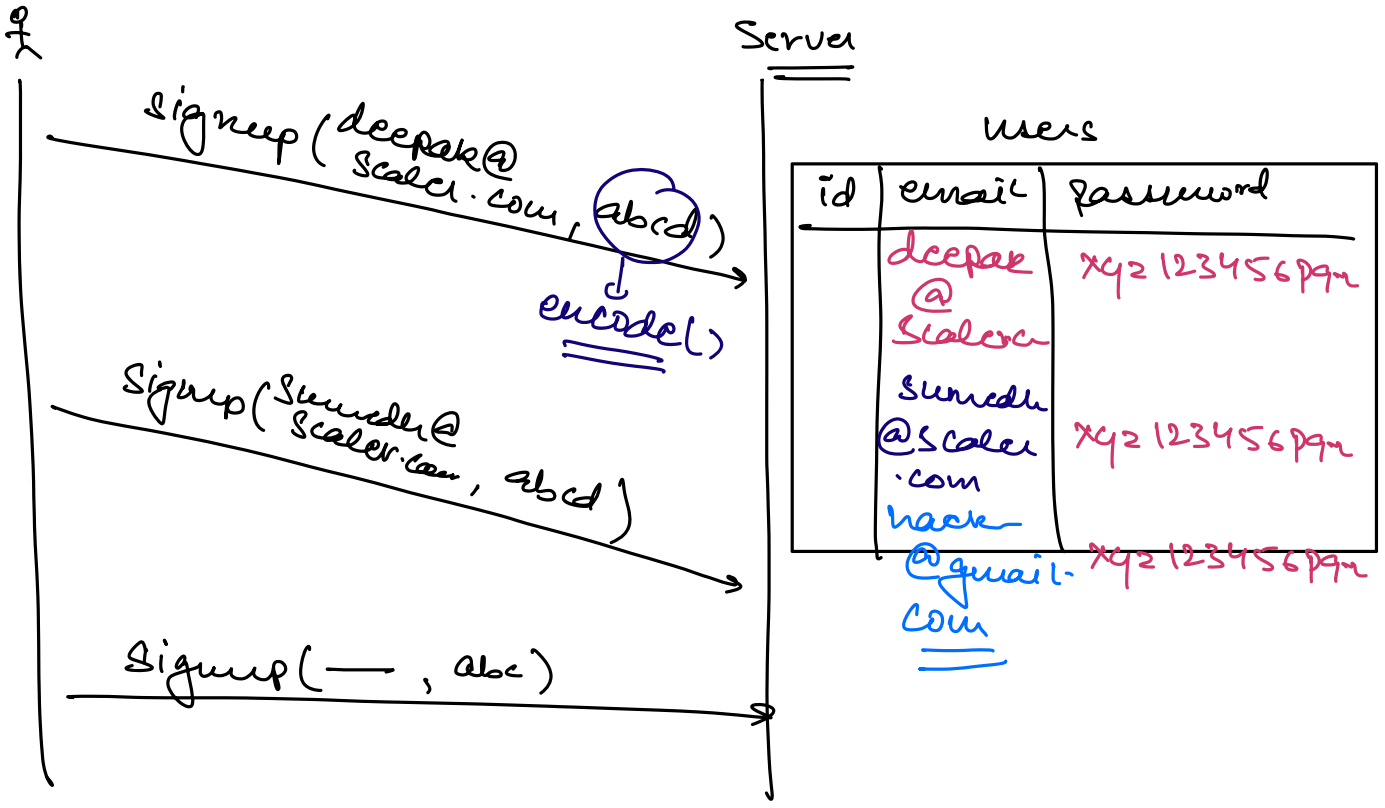
id	email	password	— — —— — - - -

⇒ What if the DB gets Compromised ?

Soln : Databases shouldn't store the password as it is in the plain text.



Same value leads same encoded | hashed value



⇒ Same password gets same hash.

⇒ Create a sample account with most common passwords

hacker@gmail.com

abcd.

⇒ Hashing + Salting

BCryptPasswordEncoder < encode(—) , matches(—)

encode (abcd) = $2a123xyasq ————

encode (abcd) = $2apqmno - - - - -

Signup (email, password)

BCryptEncoder.encode (password)

↓

Save in the DB

Login (email, password)

x = db.getPassword (email)

y = encode (password)

if (x == y) {

login success ✗

}

else

failure

}

⇒ matches

login (email, password)

x = db.getPassword(email)

BCryptEncoder.matches(x, password)

Is it possible to generate (x) encoded value from the password.

from DB ↓ = user is passing ↓

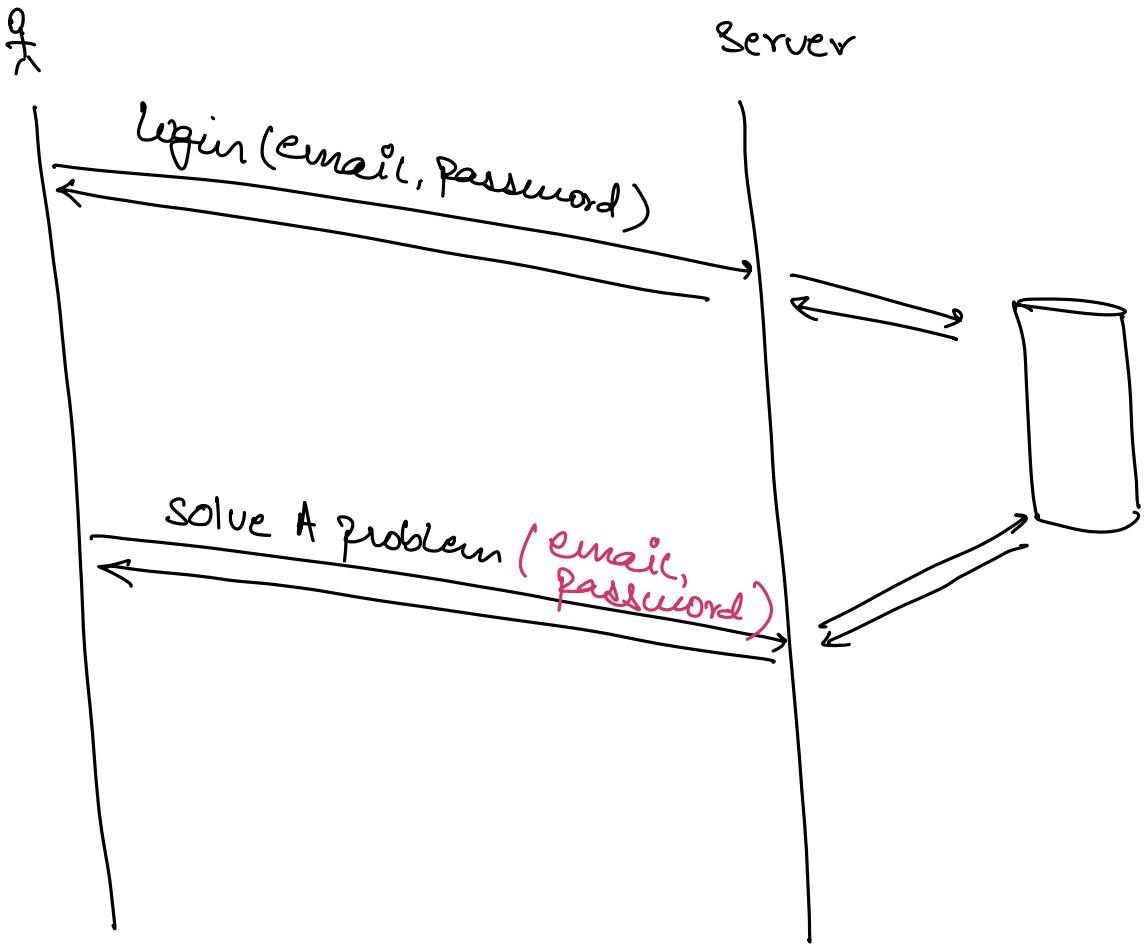
matches (x, password)

↓

Returns true if (x) can be generated from password or not

Token.

Servers & HTTP protocol are stateless.



⇒ Because HTTP is a stateless, we need to send all the details required to authenticate with every request.

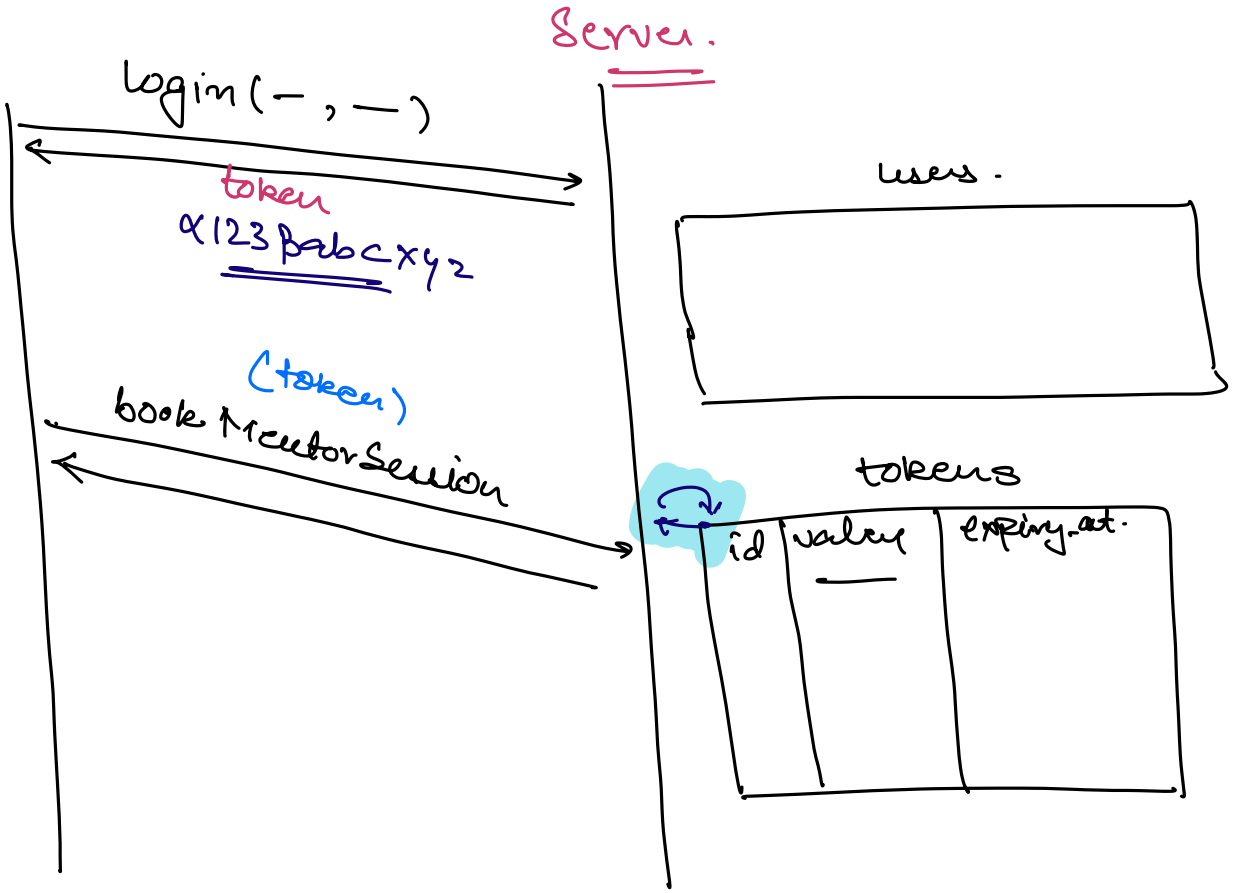
⇒ Server with authenticate the request by making a DB call.

Costly operation.

⇒ Cache. ✓

- Costly $$
- Sync overhead.

Token

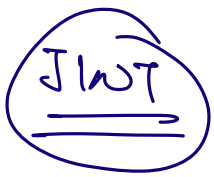


⇒ If email & password matches, server generates a token for that user & sends it in the response & save it to DB.

⇒ To validate the token, server still needs a DB call.

time consuming.

⇒ What if somehow token contains all the information required to validate itself, then do we need a DB call?

⇒ 

Next Class.