Authentication

1. Authentication? ✓
2. Authorization? ✓
   - ABAC ✓
3. How auth works
   - Signup → How to store passwords ✓
     - BCrypt
   - login.
     - Intro to JWT

Authentication

1. scaler.com/events
   - → anyone can visit this page
   - → scaler may not even know their name.

Public

2. Register for event
   - → anyone can register.
   - → BUT they have to give their info
   - → KYC
   - → unless Scaler knows who you are, you can't register.

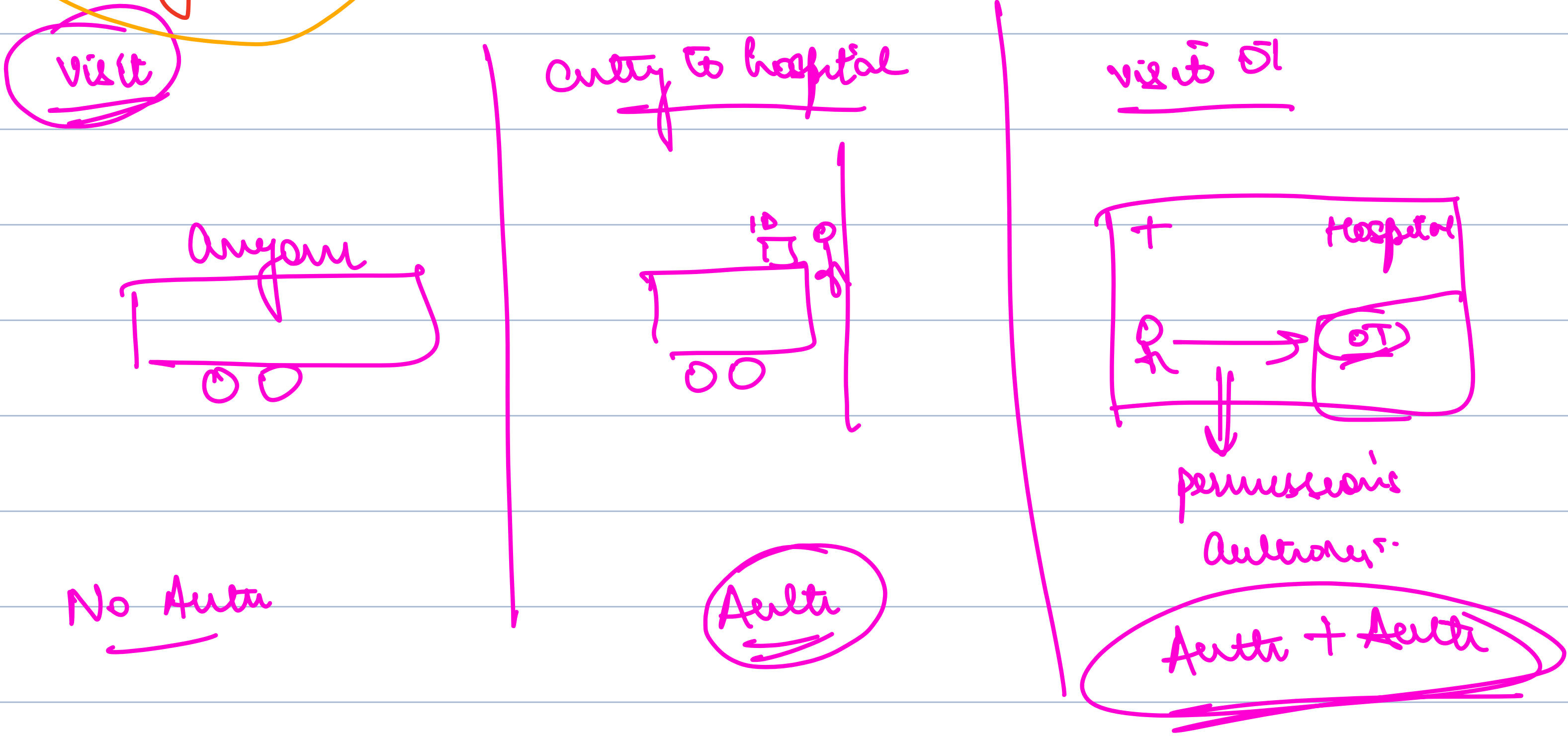
Authentication (telling who you are)

③ Scaler.com/admin

→ not everyone can visit
→ tell who you are
+
{ you should have necessary permission }

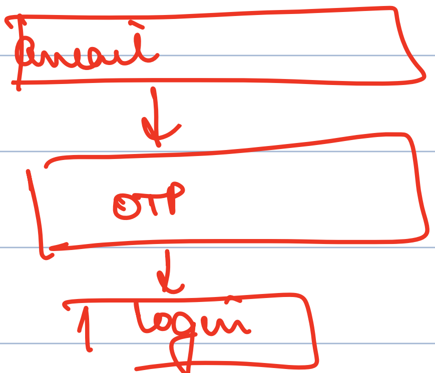
Authorization

Authentication
+
Having Permission



How do we authenticate

{
→ email, password
→ mobile#, OTP
→ Google | FB | Github
}



How do we authorize?

RBAC
- → Role
- → Based
- → Access
- → Control

users

id	email	pass

roles

id	Name
1	ADMIN
2	MENTOR
3	MENTEE
4	INSTRUCTOR

⇒ Put RBAC at diff API endpoints

```
user_id
⇒ method()
{
    if (user has xyz roles) {

    } else {
        throw Exception()
    }
}
```

user_roles

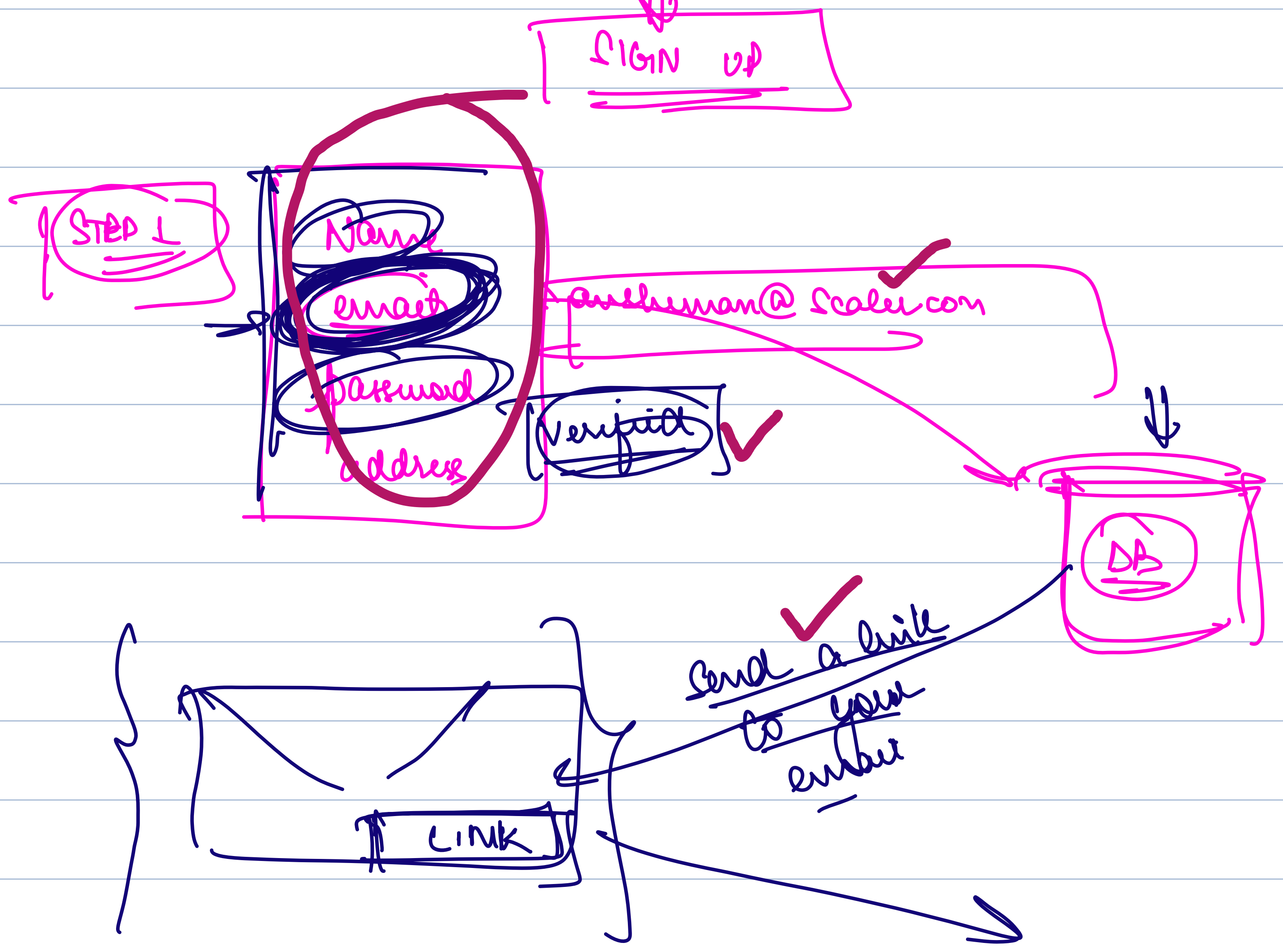
user_id	role_id
1	1
1	2
1	3
2	1
2	3
3	2
4	1
4	2

admin	[1, 3]

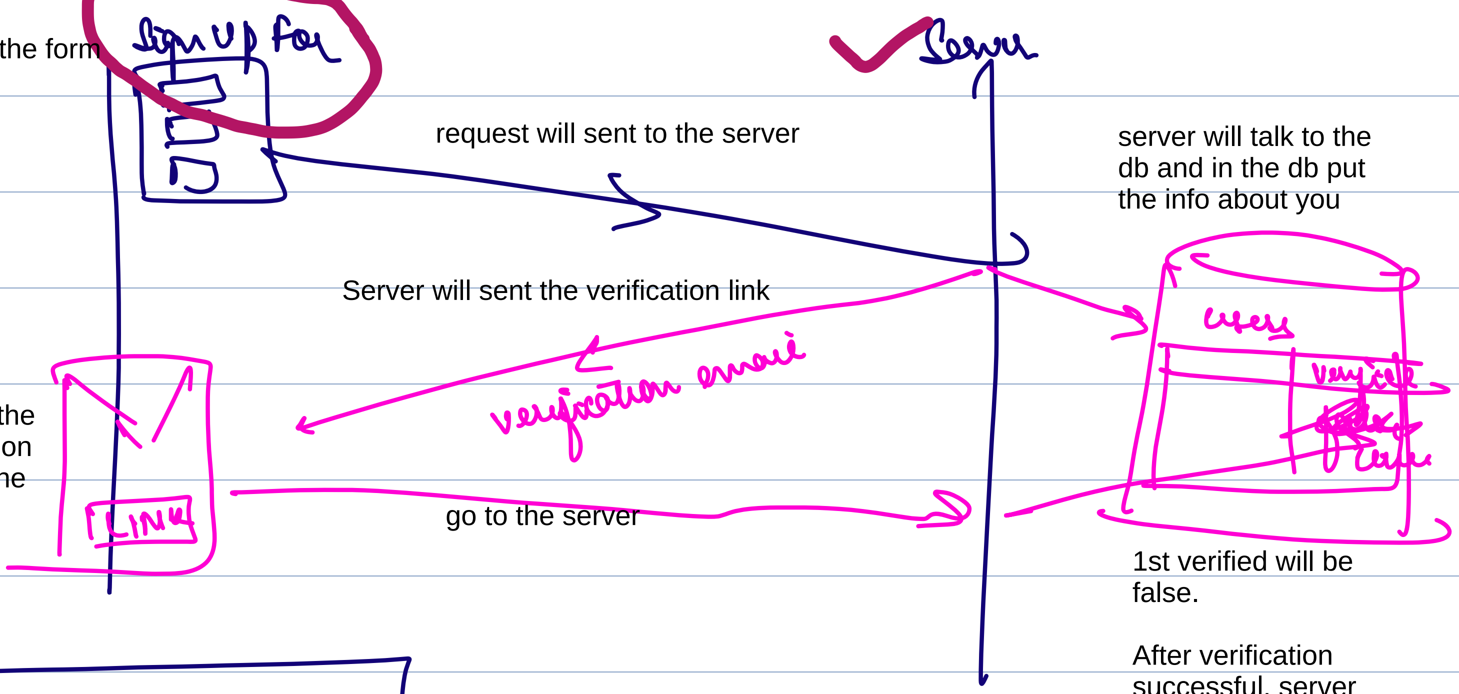
for every user, defined some roles and for every method make it accessible to some roles

How authentication works

① Apply for a way to be authenticated



Step1: fill the form



request will sent to the server

server will talk to the db and in the db put the info about you

Server will sent the verification link

Click in the verification Link in the mail

go to the server

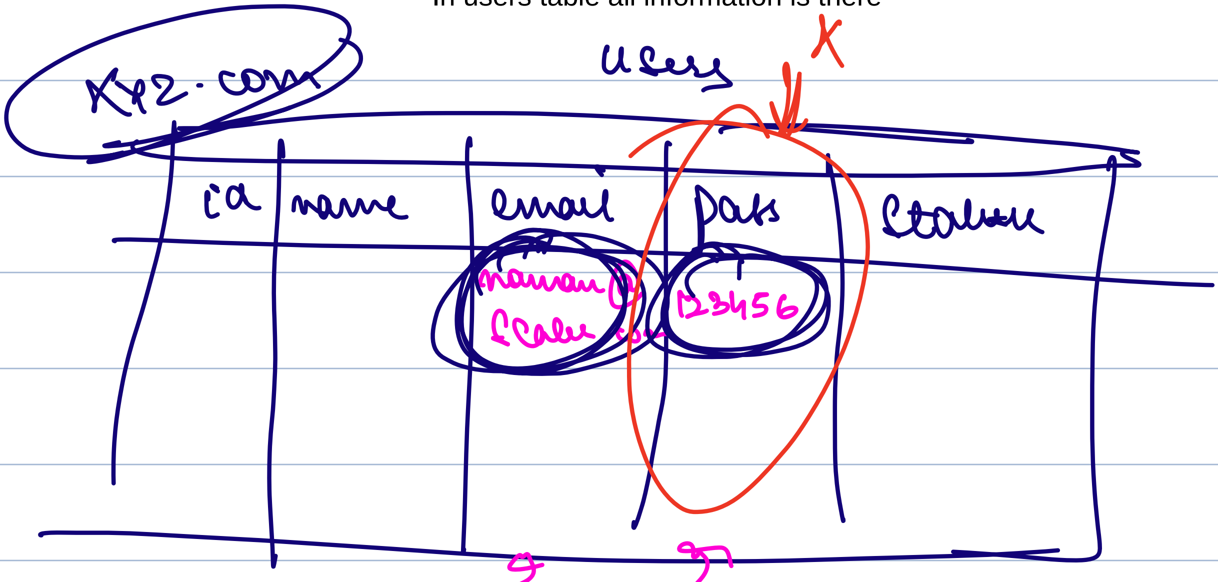
1st verified will be false.

After verification successful, server will map this as true

Email + Pass

Mostly you sign up via email & password

In users table all information is there



login (email, pass)

At the time of login, you will pass email & password, if these are matched with the table you are allowed access else not

Storing passwords as plaintexts is a terrible idea!

Storing passwords as a plain text is a terrible idea coz
1. if DB is leaked then hacker can see the passowrd and login id
also users can use same password on many website so at once many websites can be hacked.

⇒ DB leak : people have same pass at multiple places.

⇒ your emp can leak

2. Employees who got fired can leak

✱ Hashing

Rather than storing the password like this we can use HashMap

123456 → hash(123456) → abcdef

abcdef

hash(123456)

↓

xyzabc

login (email, pass)

email   hash(pass)

Hash function compute the same hash value everytime for the same key

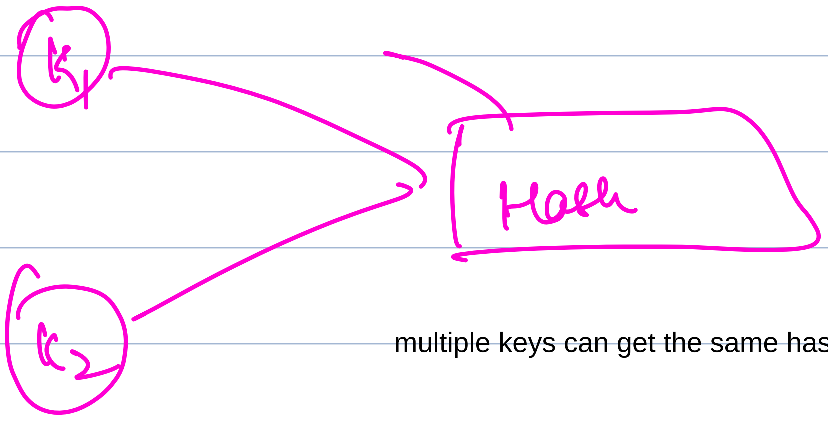
Yes ✓

⇒ hashing same key always gives same value.

Hashing is also not a good idea coz many people use same common password and multiple keys can get the same hashvalue

hash(naman) → XY 123

getting original val from hashed can't possible.



multiple keys can get the same hashvalue

When I sign up, I just have to give in DB email and Hash of password

login(email, pass)

in DB if I have [email, hash(pass)]

BAD Idea

→ many people have some common pass.

"Qwerty"

"password"

Hashing is also not a good idea coz many people use same common password and lets suppose db is leaked.

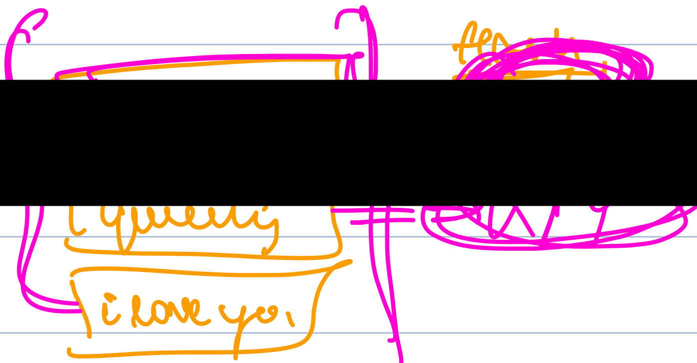
Hacker has a folder with password like "Password", "Qwerty". And suppose hashkey for Qwerty is :Alpha beta gamma and if many ppl have similar passwords they get to know about many #of users

Case

DB gets leaked

Hacker creates multiple accounts with common password.

[redacted]



Is better to use Hash + Salting. Salting is assigning some random values.

get to know many :

[redacted]

Hash + Salting ⇒ Some random value

I am storing hash of password which is a combination of password & current time. Here random value is current time and hash of password + currtime is alpha beta gamma

naman@scaler.com → hash(password + currTime)

Lets say Naman is trying to sign in and his password is password

password

↓

αβγ def

Lets say Sakshar is also there and Sakshar is storing password, which is "password". So his random value will be Hash(password + current time). It can't be that both Naman and Sakshar ended up creating the same password coz the current time of Sakshar would be different than that of Naman. So lets say his hashvalue comes out to be abc12abc.So hacker cannot find all of the ppl with the same password

Sakshar

password → hash(password + currTime)

↓

abc12abc

different

But the problem is how you will login since the current time when naman tries to login will be different than when Naman signed up so how will you login?

We have special libraries/ Algorithms k/a Bcrypt

login(naman@scaler.com, password)

hash(password, currTime)

Special libraries (Algorithm)

BCrypt Password Encoding Algo

BCrypt allows to encode a password

① encode(password) → αβγ123

encode(password) → ab12ab12

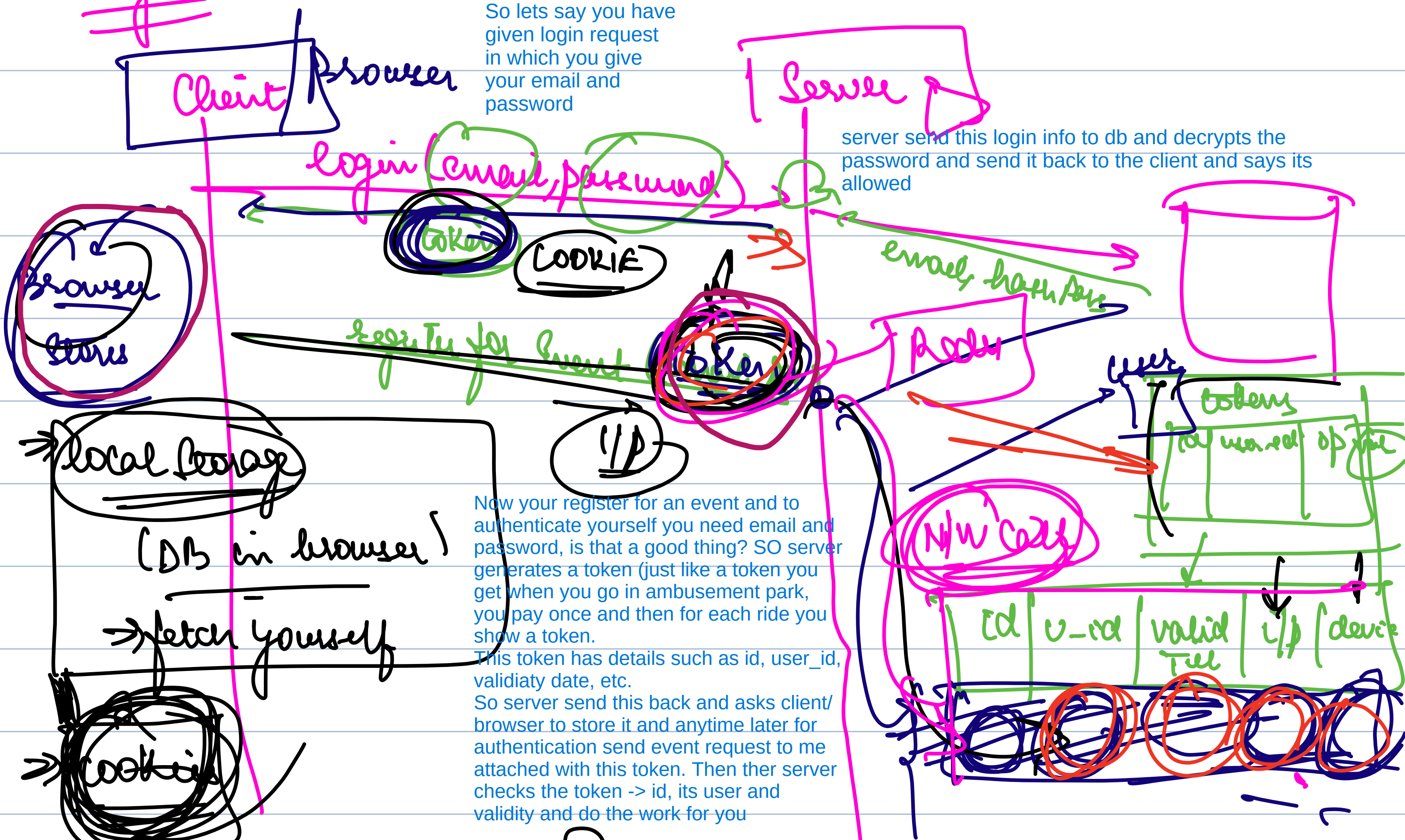
It allows gives another function in which you can verify given a string alpha beta gamma, could that string have been generated by this password

② verify(αβγ123, password) → true

verify(ab12ab12, password) → true

So BCrypt will encrypt the password and later when given on login, it will check whether encrypted password was generated by the password given at the time of sign up

login



So lets say you have given login request in which you give your email and password

server send this login info to db and decrypts the password and send it back to the client and says its allowed

Now your register for an event and to authenticate yourself you need email and password, is that a good thing? SO server generates a token (just like a token you get when you go in ambusement park, you pay once and then for each ride you show a token.
This token has details such as id, user_id, validiaty date, etc.
So server send this back and asks client/ browser to store it and anytime later for authentication send event request to me attached with this token. Then ther server checks the token -> id, its user and validity and do the work for you

→ every future req will automatically also get ~~all cookies~~ attached.

Now the question is where the browser store this information, it stores in local storage like Db in browser and everytime it needs info it will fetch but the problem is you have to fetch yourself. There is a counter path is cookies. Cookies in every future request will automatically also get all cookies attached

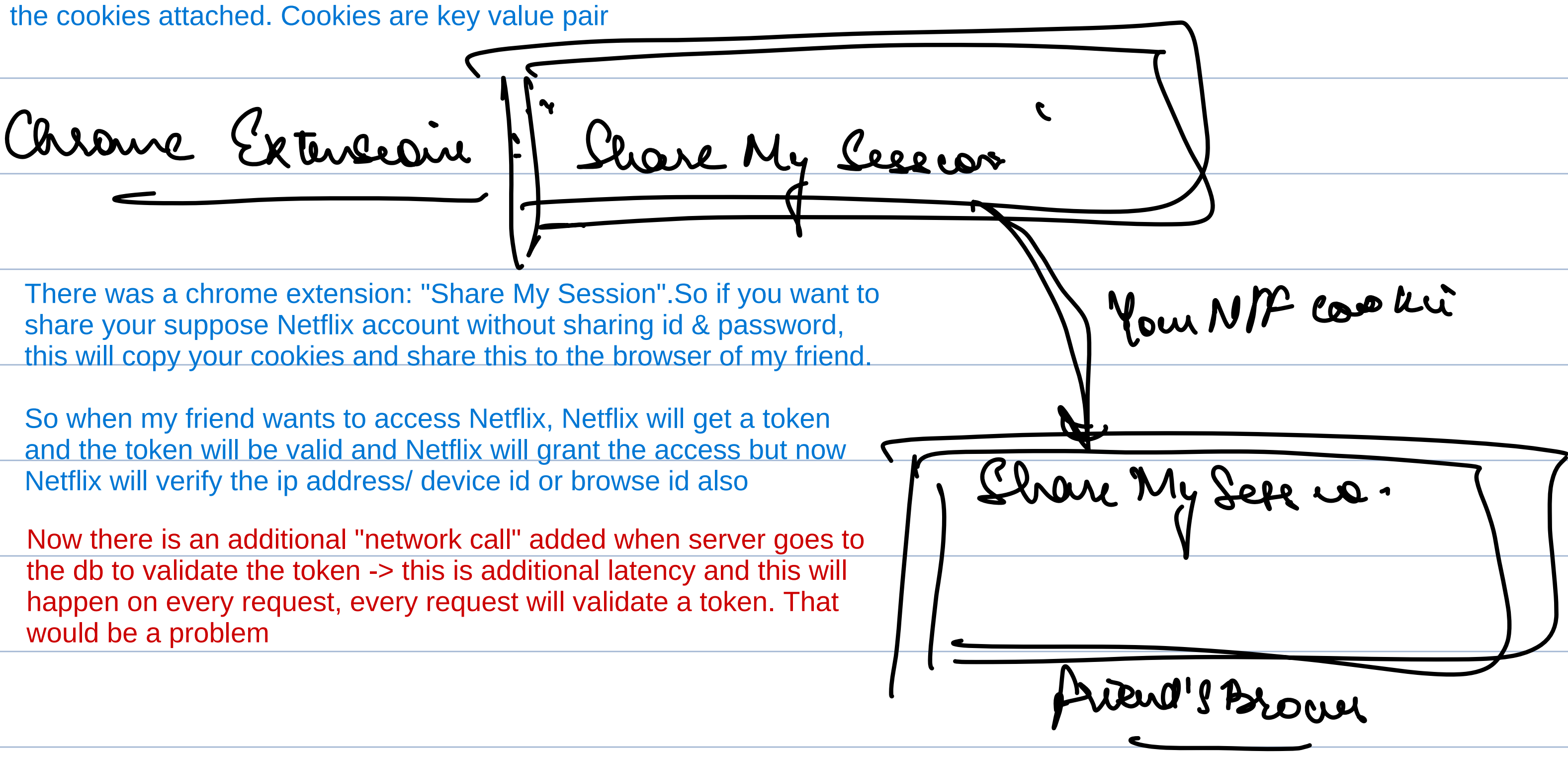
Now you donot have to write code to fetch cookie, Automatically every future request send from that browser to that url will get all of the cookies attached. Cookies are key value pair

→ logout from all devices

→ login limit

coz of this token system only you can access only 2 accounts from different machines as in case of Scaler.

Also if your password gets hacked, there is a feature logged out from all devices. and it automatically log out from all these devices, how? using this token system



There was a chrome extension: "Share My Session".So if you want to share your suppose Netflix account without sharing id & password, this will copy your cookies and share this to the browser of my friend.

So when my friend wants to access Netflix, Netflix will get a token and the token will be valid and Netflix will grant the access but now Netflix will verify the ip address/ device id or browse id also

Now there is an additional "network call" added when server goes to the db to validate the token -> this is additional latency and this will happen on every request, every request will validate a token. That would be a problem

How can we make token validation fast -> using cache. And token is a 50 bytes string and for 1 billion users logged in it would be 1B * 50 bytes = 50 GB -> 50 Gb can be stored in the RAM

How can we make token validation free

→ Cache

50B string

1B

1B * 50B

→ 50GB

→ Token validated itself

What if token would be able to validate itself

JWTs

JSON Web Tokens

Build a token in a way that within itself token had every info to validate itself

Lets say you had to build a token
in a way that within itself

→ what if ip, device, valid till, u-id

→ were stored in token itself

What if ip address, validity, user_id were stored in the token itself

Sat → JWT & OAuth