Chapter 13) Ref, Populate & mayht process of writing APIs

⇒ Before writing any APIs, cover all the corner cases or what can go wrong

API i) Connection Request review by user.

→ Receiver's End

POST / request / review / {accepted / rejected} : requestId
POST / request / review / {accepted / rejected} : requestId.

dynamic status

otherwise you are not authorized to review that request.

API

⇒ Receiver should be logged in to review the request (userAuth middleware)

↓

logged in user

→ ~~@~~ logged in user == receiver's Id

→ status == ignored > nothing changes (can't accept or reject)

→ status should be always interested.

→ ~~@@~~ status should be valid (enum : accepted, rejected)

→ requestId ie. user must be present in DB as well as logged on the system. (like implicit don't write)

→ proper checks (otherwise attack on APIs)

API Validation Checking

name : Rohan Sharma
email : rohan.sharma@gmail.com
password : "RO@9Hs2#"
Id) 69902bf12dCdab8(
10908Le3

interested →

Priya Verma
priya.verma@gmail.com
Pr!8va@4
Id) 69902c6c2cda5
8cL0908Leb

↓

she can either accept / reject.

userRouter

→ GET v/s POST
Thought process is different.

→ Security Guard ⟶ Mansion
(developer) (Database)

POST API

→ some random data ⟶ in database
→ check each and everything before putting data into the database.
→ save > second last line of API. (POST)

GET API

→ only send the data which is not sensitive (to the client)
→ attacker can even destroy and sold your company data to competitors.
→ user should be authorized
→ Data leaks can be very dangerous.
(for all API, user should be logged in)

API I) i) get all the requests (pending) for the logged in user
ii) get all the connections
iii) display this feed > so the can left swipe or right swipe.

API 1) getting all the pending requests and displaying them

→ data is being fetched from database

- loop to display (poor way of handling this)
- creating : relation b/w table / (ref: (row reference) in b/w two collections.
  ↳ then populate ~~whatever~~ fromUserId to whatever data we need in them (firstName, lastName).
  ↳ not overfetch the data as well as not send any sensitive information).
  ↳ read about ref and populate (their relation) (joins) similar) /

API 2) // get all the connections which are either accepted by you or others of yours in which you are interested /

1) fromUserId > loggedInUserId /
   toUserId > interested profile /
   status > accepted /

2) fromUserId > someone interest in me /
   toUserId > loggedInUser (// console.log
   status > accepted / // .toString
   // use comments
   // destructuring should be good,

) first get all the accepted)
then loggedInUser should be either sender or receiver. use $OR operator