Dessign a middleware library

↕

Design a route handler

Q: Are we looking for any db interactions ?

No

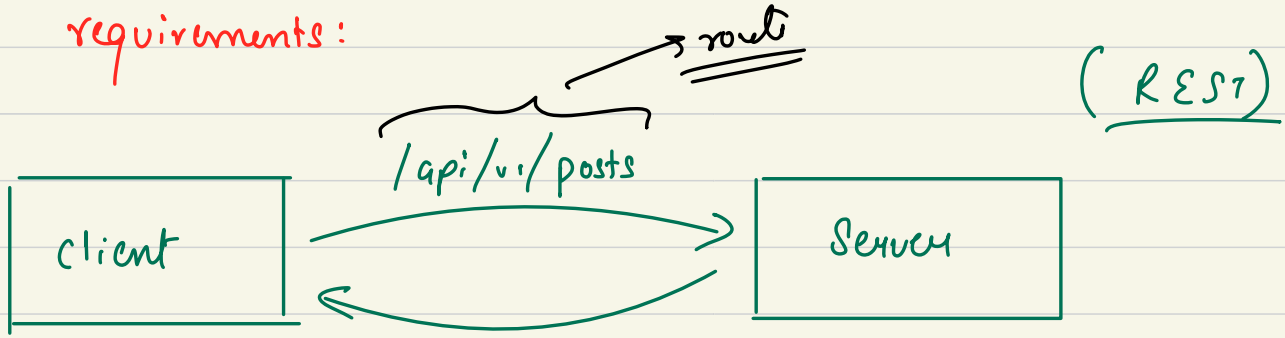
Q: Are we looking to setup REST API's for comm??

No, (console based is ok)

Q- Are we looking for class impl or feature impl is also reqd?

Class structures are preferred.

feature requirements:



(REST)

→ Route handlers should be taking the incoming Request and validating the request body

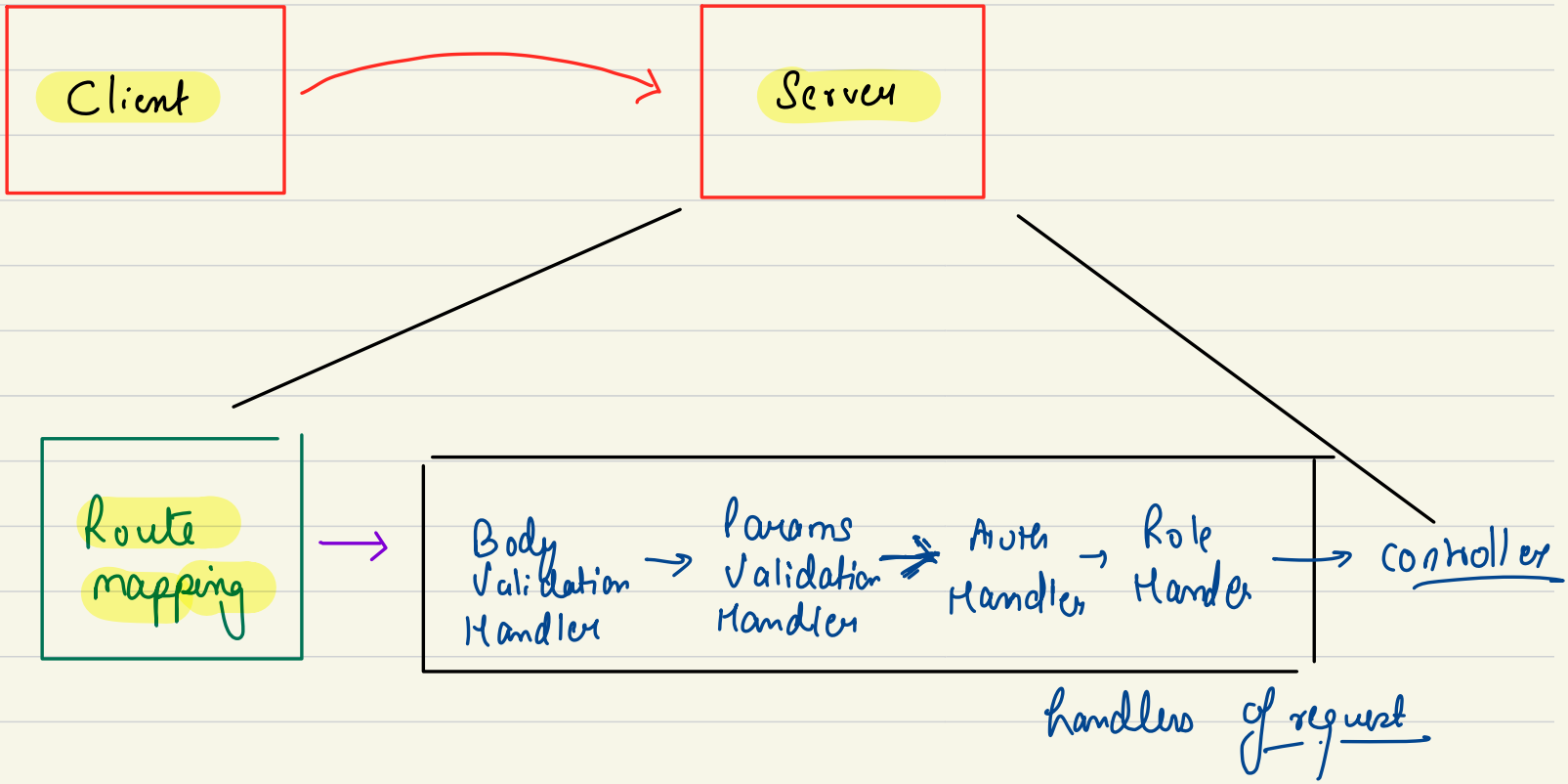
→ Route handler should be able to validate request params.

→ Route handlers after validating body & params should be capable of doing Authentication.

→ Once auth is done, route handlers can also do authorization.

→ If the user is authorized then we continue to access the controller layer.

→ Some route handlers might be in specific order & some might not.


Client
Server
Route mapping
Body Validation Handler
Params Validation Handler
Auth Handler
Role Handler
controller
handlers of request

Route mapper is a module that is responsible for mapping the incoming http req to it's corresponding handlers

// assume we have a single handler func

```
handler (Request req) {
    → validate body
    → validate params
    → auth
    → authorization
    → controller
}
```

↓

↗ Direct violation of SRP.

→ an improvement would be to put all these logics into separate fun:"

→ More preferably in separate Classes.

Note→ a route handler is capable of modifying request object before next handler gets it.

↗ a handler should be capable of returning the resp midway without calling remaining handlers.

↗ every handler should be able to execute some logic & call the next handler expected.

the logic might be executed before or after calling the next handler.

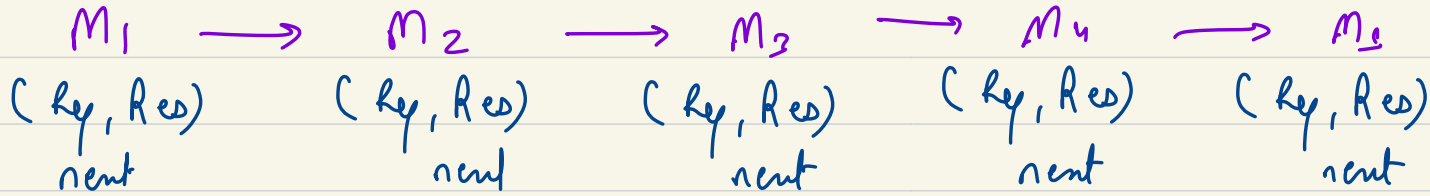
→ handler arrangement can be diff for diff routes

→ In most common terminologies, we call these route handlers as middlewares / interceptors / filters etc

middlewares → Most common term

middlewares → Express.js

middleware are chain of functions where each function has access to the Req, Res & the next function in chain. (dll like structure)



→ Create an OR based filter.

→ Category

(C1 → C2 → C3 → C4 .....)

→ Design pattern which helps in implementing middlewares.

↳ CHAIN OF RESPONSIBILITY (COR)

every handler

- logic
- access to the next handler