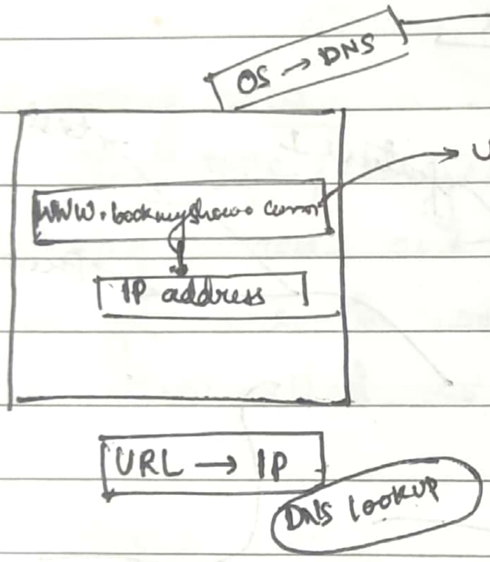
Tanay Bomb
CN
Book

How Web Works??

Q→ What happens when you write www.bookmyshow.com

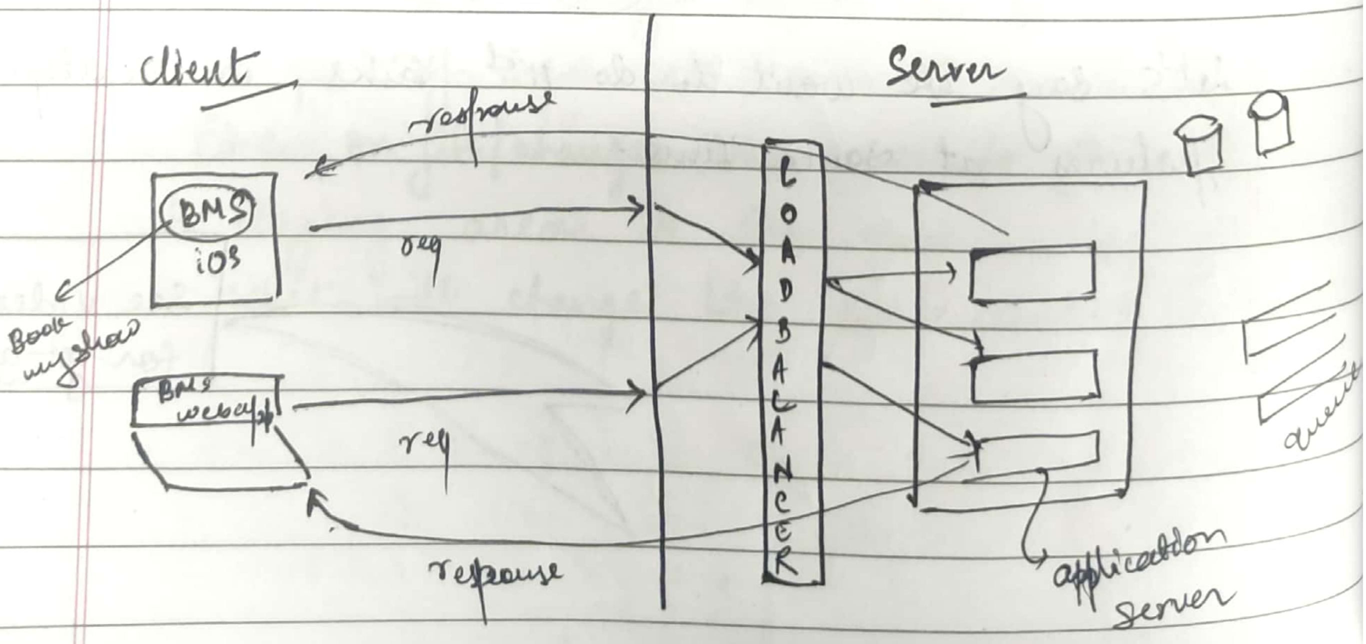
Computer Network → It is a group of Interconnected Computer Systems

Internet → It is network of computer networks.



→ OS have local DNS
for resolution of IP we first lookup onto local DNS

If not found
we look into ISP DNS
} Domain name Service



AB Testing

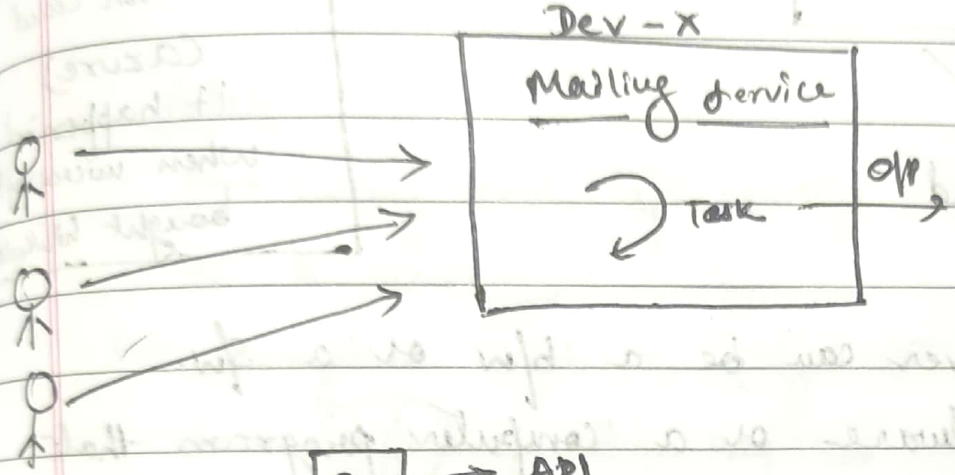
Walkat frontend

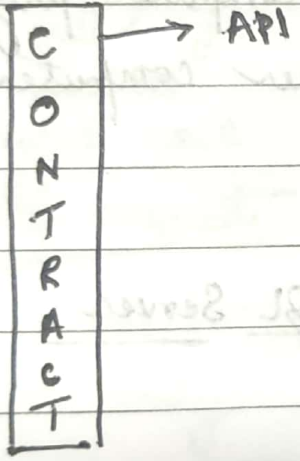
- optimize app
  - minification
  - compression
  - lazy loading
  - not serve everything in one go
  - optimize images
    - compress
    - fetch correct size image

API.

What is an API ??

↳ application package/programming interface

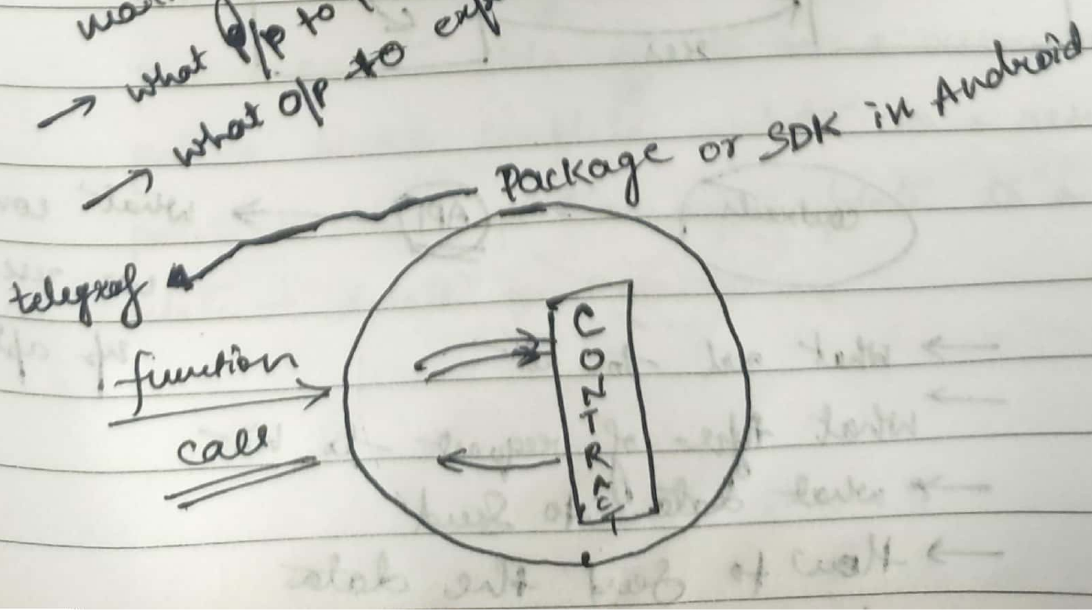




How to call the mailing service?

→ what i/p to provide

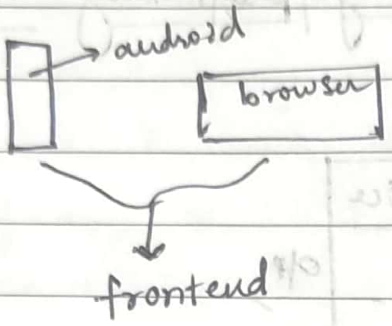
→ what o/p to expect



Client Server Architecture

Client → any user facing interactive end

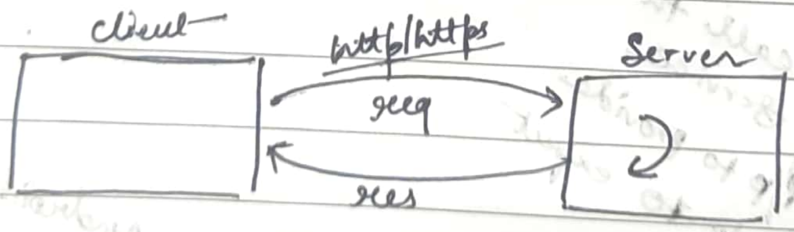
Server → backend



LinkedIN is the first social media platform deployed on public cloud (azure) it happened when microsoft bought LinkedIN

Server → Server can be a h/w or a s/w.
It is a hardware or a computer program that provides a service to another computer program or a different machine.

MySQL → MySQL Server



contracts → API → what conventions we use for setting up apis.

→ what url to hit
→ what type of request to hit
→ what data to send
→ How to send the data
:

Conventions of API

→ REST
→ SOAP
→ gRPC

(REST) → convention

Representational state Transfer

Rules

↳ Every real life entity is expected to be represented as a resource.

↳ Everytime with a Restful API request we have to sent type of the request.
methods

↳ Dedicated URL'S

→ http requests

Get, POST, put, patch, delete

- HTML only supports Get and post request

In Rest conventions

Get → retrieve info about a resource.
post → create side effects on a resource.
put → make complete update to a resource.
patch → make partial ~~upd~~ update to a resource.
delete → delete a resource.

Get	Post
↳ In get request data is send in URL, that means, it gets saved in our history, we can log it, cache it etc.	generally data is not expected to be sent in URL, rather than in request body/payload.

3 ways to send data

→ Request params → /movies/blackpanther
used generally to send unique identifiers

→ Query params → /categories/electronics? company = Samsung & order = desc & filter = price
used generally for filtering purpose

→ Request Body →

email :
pass :
regions :

payload / body

↳ In Rest conventions, data / messages sent apart from ~~data~~ URL are sent in JSON.
SOAP uses XML, grpc uses protocol buffers

Rest API for a blog

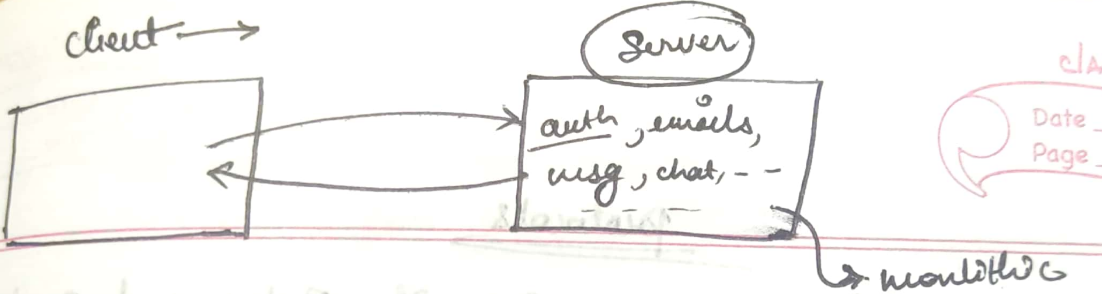
CRUD → Read, update, Delete, create

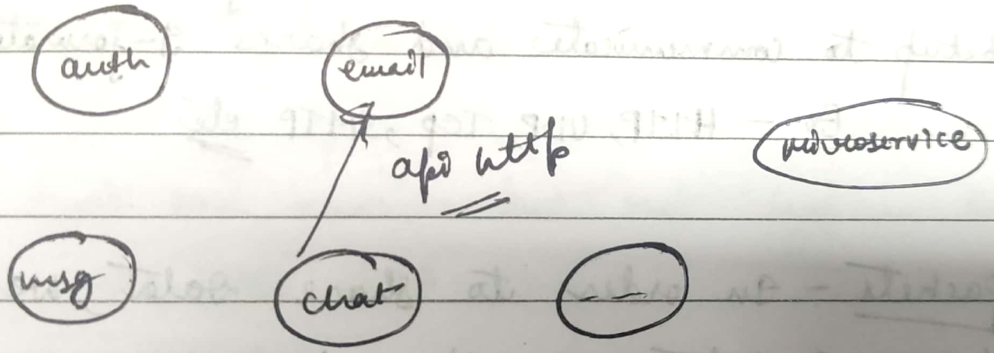
CREATE → POST

/blogs

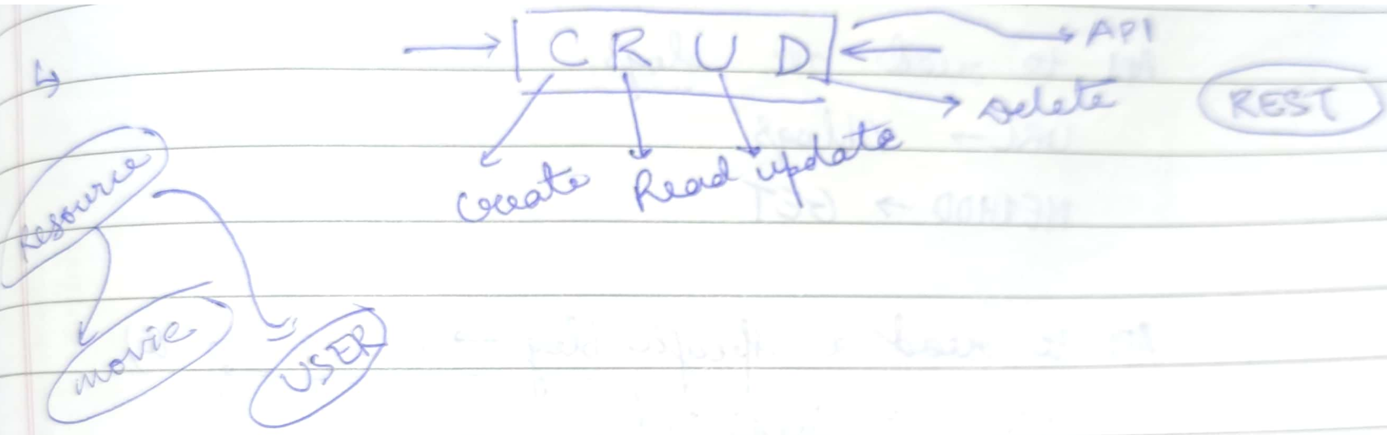
```
req body → {
    name: '',
    desc: '',
    created: ''
}

res body → {
}
```


client →
Server
auth, emails,
msg, chat, - -
monolithic


auth
email
microservice
api http
msg
chat
--

Protocols


C R U D
API
delete
REST
Create
Read
update
Resource
movie
USER

Medium → Create a blog

→ Read a blog

→ Update a blog

→ Delete a blog

Create a basic blog app

C → create

R → Read

U → Update

D → delete

REST

API to create a blog → REST

→ Creating a resource

URL → /blogs

Method → POST

→ body params → {

title: " ",

desc: " ",

:

}

API to read all blogs

URL → /blogs

METHOD → GET

API to read a specific blog → (using blog id)

URL → /blogs/:id

↳ indicates variable value

ex → /blogs/2

/blogs/3214963

Method → GET

API to delete a blog →

↳ URL → /blogs/:id
Method → DELETE

API to update a blog →

↳ URL → /blogs/:id
Method → put/patch
body params → { title:
⋮
}

→ or slugs

* We use friendly id's nowadays for improving Search engine optimizations. We Identify unique piece of Text from the article and map it to the number which is inturn the id. It also enhances Readability.

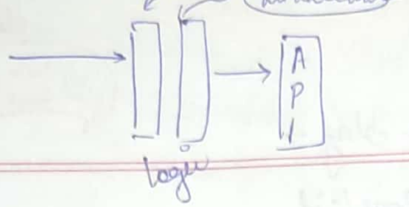
Ex /blogs/2

↳ /blogs/segment-tree

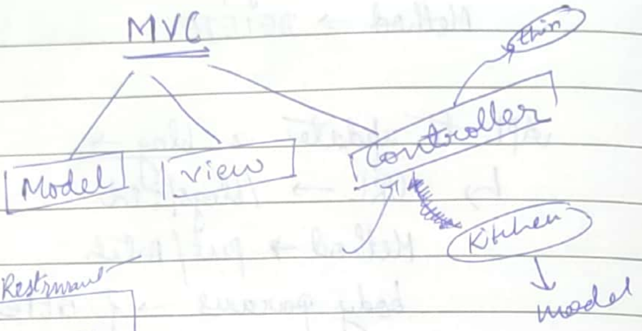
* We can't have space in the URL that's the reason we see %20 sort of a thing in the URL.

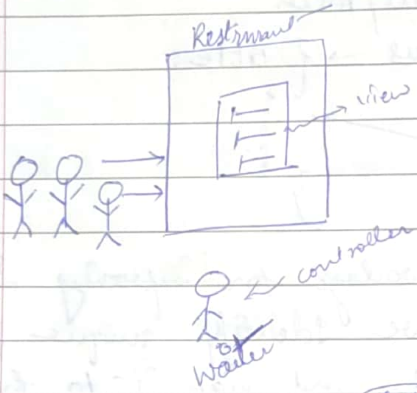
res.status(200)

↳ It returns the same response object by setting the response code as 200.

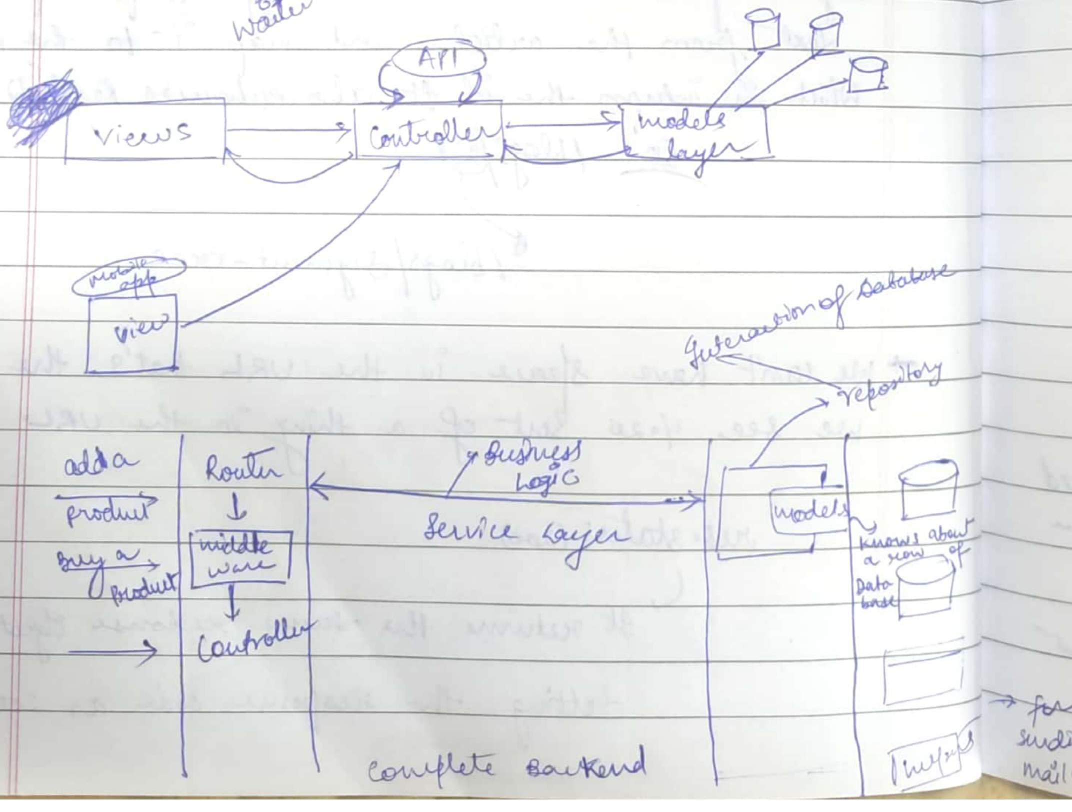


MVC





API



complete Backend

we
with an example

class

}

* function

1) Task based

Task Based

Features Bas

we can understand reepository and model with an example of oops

class product { → model

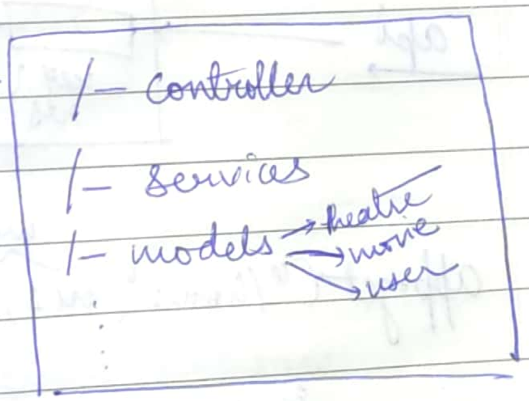
}

✱ function to make product will be inside repository

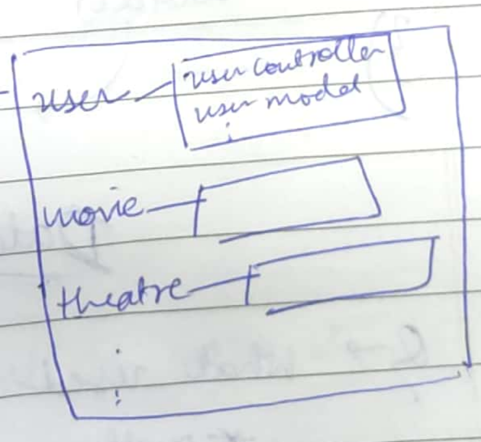
How to setup the Folders

1) Task based 2) Features Based

Task Based ——



Features Based ——

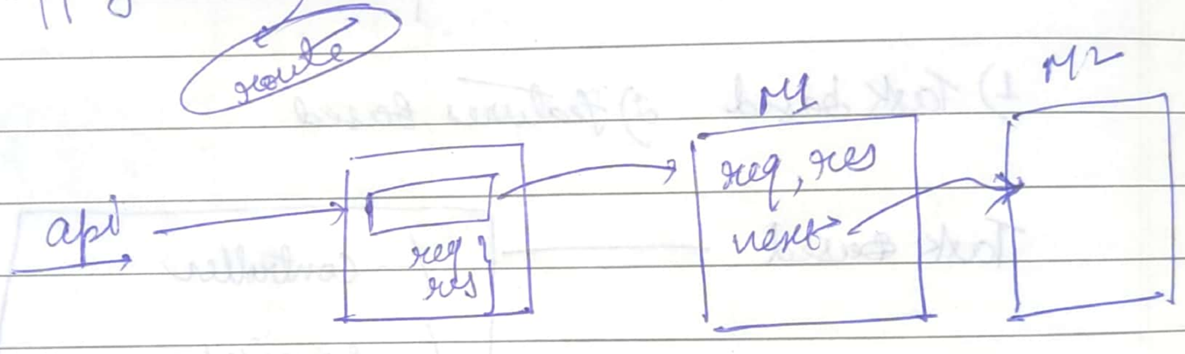


→ for sending mails and or accessing extra services

Middlewares

Middleware Functions are Functions that have access to the request object (req), the response object (res), and the next middleware Function in the application's request-response cycle. The next middleware Func^n is commonly denoted by a variable named next.

app.get ("/home")



middleware controller

app.get ("/home", m1, m2, (req, res) => {

Service()

})