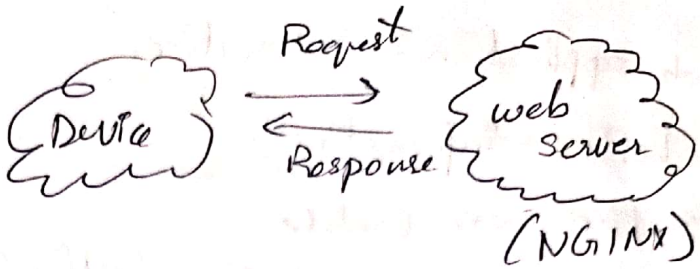
https://———.html

URL – Uniform Resource locater

protocal | Domain name | file path.

http/s



→ web hosting:
1) Shared hosting
2) VPS virtual private service
3) dedicated hosting
4) cloud hosting

http request response protocal

GET – Retrieve
POST – send
PUT – Update
Delete – Remove
Patch – partially update

Status message
100 – 599
200 – OK.
404 – Not found

informational 100-199 (interim response ex: 100 continue)
Successful 200-299 (operation done ex: 200 OK)
Redirection 300-399 (moved ex: 301 moved permanently)
client error 400-499 { 400 - bad data, 401 - User login 403 - web server body
404 - not found.
Server error 500-599 (500 - Internal Server error).

https: Encryption used (Secret Code).

Dynamic host Configuration Protocal - assign your Computer an IP address,

DNS → check DNS server with the domain name & then return the correct IP address.

IMAP, SMTP, POP, FTP →
IMAP (Internet message Access Protocal) – download email
SMTP (Simple mail Transfer protocal) – Send email
POP (Post office protocal) – Store in mail box
FTP (File Transfer protocal) – local device to send files to Server

SSH – Secure Shell protocal
→ It is used to interact with the Computer (Server) remotely with Security

SFTP (Secure file Transfer Protocal)
→ Security is added using SSH

web application – interactive

website – informative.

libraries – re-usable

framework – structure developers build with.

API - advanced functionality with simple code

API

- Browser / web API
  - → Adding some
  - → DOM API (html document to tree of node)
  - → Geo location API
  - → Fetch API
  - → Canvas API
  - → history API
  - → web storage API
- REST (Representation State Transfer) API
  - → set of principles used to build highly efficient APIs
  - → Send & receiving data from db.
- Sensor Based API.
  - → IOT based
  - →

IDE : Integrated development Env.

<!DOCTYPE html> — Bhtml document

<html>

<head <title> li— </title>

</head>

<body>

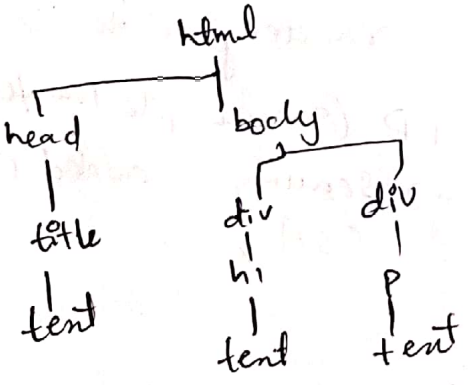
</body>

</html>.

→ DOM : (Update & delete Element) (whenever click radio it is delete)

Document object module/model

→ like, comment functionality

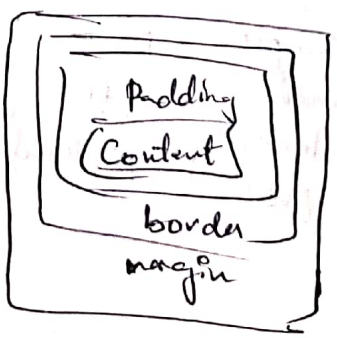


DOM & (javascript can be change DOM)

ARIA :

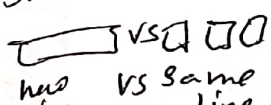
Accessible Rich Internet Application

Box Model :



Scanned with OKEN Scanner

Block vs Inline



two line vs same line

bootstrap library:

link tag.

bootstrap also provide js library:

package manager:

→ download dependencies

because some use old version of html, CS, JS libraries so that package manager is used.

→ It take care of dependency tree

→ Ex: Node Package Manager (NPM).

→ Bundle up (Bundling tools)

→ Gulp.

→ webpack

→ gather all your dependencies and combine into a single file.

Modifiers: Change the visual style of Component.

Primary
Secondary
Success
Info
Warning
Danger
Light
Dark.

Bootstrap Component:

Responsive Design

- Content change acc to device

→ CSS media queries

→ fluid images

→ fixed & fluid grid.

Bootstrap is often described as a way to "build fast, responsive sites". feature-packed, powerful and extensible frontend toolkit.

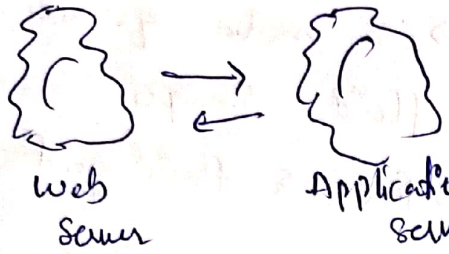
breakpoint: how layout changes (infix)

breakpoint	Class infix	Dimension
Extra Small		<576 px
Small	sm	≥ 576 px
medium	md	≥ 768 px
large	lg	≥ 992 px
extra large	xl	≥ 1200 px
extra extra large	xxl	≥ 1400 px

infix

Static vs dynamic :-

dynamic is made when http request made.



Caching:
A saved copy of dynamic content readily available upon request.

Application Server

→ Run application logic
→ Communicate with dbs.
→ Check permissions.

Single page Application:
→ feel faster
→ Bundling
→ Lazy loading.

React : library
less code

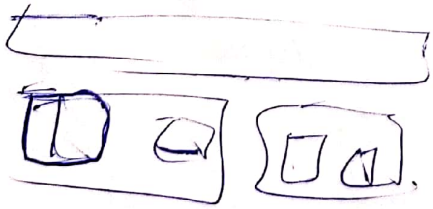
React Component:
A Small piece of a User interface.
→ Isolated development
→ " Testing
→ Reusing Template

MVC - JavaScript model view Controller.

render - method.
⇒ It returns lightweight description of what the DOM should look

Virtual DOM → browser DOM
update

Component hierarchy :-



<img src = " " alt = " " >

alt → alternative mean whenever image can't be load such situation shows the text instead image.

href : hyperlink refrence.

Specificity :

1. Inline
2. id
3. class
4. type.

Responsive Design :

→ view port
→ grid
→ flex box
→ bootstrap library lg/sm/xl.

Code ⌥ vscode.

Sass :-

→ additional feature
→ have variable.
→ different extension Scss
→ avoid repeatation.
→ Compilation is needed

Cmd : Sass variables.scss : variable.css → enter.

Sass to css

To avoid Complaction again & again

Cmd : Sass --watch variables.scss : variables.css

```
div {
   p {
   }
}
```

} Provide nesting properties & more enhance Syntax.

→ It provide Inheritance.

```
% message {
}
.warning {
}
.Success {
   @extend %message;
   background-color : value;
}
```

Git :- Version Control.

→ Keep track of changes to Code.
→ Synchorizes code b/w different people.
→ Revert back to old version of code. → Github is use to Store repo.

Commands :

→ git clone <url>
→ git add .(all) — all file with message
→ git commit -m "message" — am
→ git status.
→ git push.
→ git pull.
→ git log.
→ git reset.
→ git branch.
→ git checkout -b <name>.
→ git merge <name>.

merge Conflicts :

your changes
<<<< head
{ b=2
= = = = =
remote change
{ b=0
>>>> 834459

Github

→ fork.
→ Pull request
→ merge.

→ Github pages.
→ github.github.io.

Python:

```
print(" ")
input("Name?")
print(f"Hello {name}")   # formatted string.

if :
    print( )
elif :
    print( )
else:
    print( ).
```

loops:

```
for i in [0,1,2,3,4,5]:  # range(6)
    print(i).
```

dictionaries: eg

```
houses = {"name": "Harry", "house": "Gryffindor"}
print(houses["Harry"])
```

fn:

```
def Square(x):
    return x * x.
```

```
from functions import Square   # name of file ; method in that file.
```

decorators (Functional programming)

```
def announce(f):
    def wrapper():
        print("About to run the fn")
        f()
        print("Done with the fn")
    return wrapper.
```

```
@announce
def hello():
    print("Hello")
```

Sequences

```
name = "Harry"
print(name[0]).
```

List (mutable)

```
names = ["a", "a", "a"]
```

tuples: .0 values together

⇒ Coordinate = (10.0, 20.0)

(non-mutable / immutable values)

set - Collection of Unique values

dict - Collection of Key-value pairs

list:

```
.append(" ")
.sort( ).
```

Sets: (no repetition)

```
s = set()
s.add(1)
s.remove(1)
len(s),
```

OOP's in python.

```
class Point():
    def __init__(self, x, y):
        self.x = x
        self.y = y
p = Point(2,8)
print(p.x)
print(p.y)
```

lambda.py

```
people = [
    {"name": "Harry", "house": "Gryffindor"},
    {"name": "Cho", "house": "Ravenclaw"},
    {"name": "Draco", "house": "Slytherin"}
]
people.sort(key=f)
print(people)
def f(person):
    return person["name"]
```

// people.sort(key = lamda person: person["name"])

house

Exceptions: handling.

```
import sys
try:
    result = x/y
except ZeroDivisionError:
    print("Error: Cannot divide by 0")
    sys.exit(1)
print(f"{x} / {y} = {result}")
try:
    x = int()
    y = int()
except ValueError:
    print("Error: Invalid input")
    sys.exit(1)
```

Scanned with OKEN Scanner

Django: python web framework :- dynamic html pages.

Get / http/1.1

host: www.example.com.

Response

http/1.1 200 ok

Content-Type: text/html.

pip - python install manager.

pip3 install Django

django-admin startproject PROJECT-NAME

python manage.py startapp App_Name.

→ setting.py → Installed app → App_Name

view.py

from django.http import HttpResponse.

def index(request):

return HttpResponse("Hello world").

→ In App folder

urls.py

from django.urls import path

from . import views

urlpatterns = [

path("", views.index, name="index")

Routes

views to update

urls update.

Generic

views.py

def greet(request, name):

return HttpResponse(f"hello, {name}").

url.py

path("<str:name>", views.greet, name="greet"),

{{ }} → To insert name

{% %} → To insert logic.

→ {% if %}

<h1>

{% else %}

<h1>

{% endif %}

→ link

→ {% for task in tasks %}

{{ task }}

{% endfor %}

virtual env

→ python3 -m venv .venv.

→ source .venv/bin/activate.

Scanned with OKEN Scanner

Templates: html pages

```
def index(request):
    return render(request, "hello/index.html")
```

template/hello/index.html.

load static: styles & css

```
{% load static %}
<link href="{% static 'newyear/styles.css' %}" rel="stylesheet">
```

tasks.

for loop.

forms: (Inheritance).

```
<html>
<head>
</head>
<body>
    {% block body %}
    {% endblock %}
</body>
</html>.
```

=> {% extends "tasks/layout.html" %}.

```
{% block body %}
{% endblock %},
```

(Add a New Task)

Switch between pages -

```
<form action = "{% url 'tasks:add' %}" method="post">
    {% csrf_token %}
```

403
CSRF
Cross Site Request forgery
Verification failed.
tokens: r

inbuild form in Django:-

```
class NewTaskForm(forms.Form):
    task = forms.CharField(label
        = "New Task")

def add
    return render(request,
    "tasks/add.html", {"form"
    : NewTaskForm() })
priority = forms.IntegerField(label =
"Priority", min_value=1, max_value=10
```

Session:

Suppose you are making task manager then both two user will see the same page because of global variable now

Session will store information about you.

```
def index(r):
    if "tasks" not in request.session:
        request.session["tasks"] = []
    return render(request, "tasks/index.html", {
        "tasks": request.session["tasks"]
    })
```

Cmd: python manage.py migrate.

SQL:- (RDBMS) languages to interact with the db.

model: python class & object

migration: technique to update db & Underlined model.

Data: → MySQL
→ Postgre SQL
→ SQLite

SQLite Types:-
- → Text
- → Numeric
- → Integer
- → Real
- → Blob. - Binary large data.

MySQL Types:
- Char(size)
- Varchar(size)
- Small Int
- Int
- BIGINT
- Float
- Double

Step1

```
Create TABLE flights(
    id INTEGER PRIMARY KEY
    AUTOINCREMENT,
    Origin TEXT. NOT NULL,
    destination TEXT NOT NULL,
    duration INTEGER NOT NULL
    );
```

Constraints:
- → Check (range) rating 0-9
- → Default
- → Not NULL
- → Primary Key
- → Unique.

Scanned with OKEN Scanner

Step 2

Insert Into flights
(origin, destination, duration)
VALUES ("New York", "London", 415);

step3

select * FROM flights;
select Origin, destination FROM flights;
Select * FROM flights WHERE id = 3;
Select * " " " Origin = "New York";

touch flight.sql : file name: (SQLite)

.tables – Show all tables.

→ Sqlite3 flight.sql
→ Select * FROM flights WHERE duration > 500;
→ Select * FROM flights WHERE " " " AND/OR destination
= "PARIS";
→ " " " " " Origin IN ("New York", "Lima");
→ " " " " " origin LIKE "%a%";

functions:
→ Average
→ Count
→ MAX
→ MIN
→ SUM

UPDATE flights
SET duration = 430
WHERE origin = "New York"
AND destination = "London";

DELETE

DELETE FROM flights WHERE destination = "Tokyo";

Other clauses:
→ LIMIT → ORDER BY → GROUP BY → HAVING.

Scanned with OKEN Scanner

Foriegn key: (normalization it is used).

↳ airport data store seperate rather than involving in the main table flights.

JOIN:

SELECT first, origin, destination FROM flights JOIN passengers ON passengers.flight_id = flights.id;

→ JOIN / Inner Join
→ left outer join
→ right Outer join
→ full Outer join.

CREATE INDEX:

Create Index name_index On passengers (last);

SQL Injection attack:- this will ignore rest of the part of the line

Select * from users
where username = "hacker" -- " AND password = " ";

→ Don't directly interact with the SQL, Use Interface.

→ Regular steps in Django
then
models file

Cmd: python manage.py make migrations.
" " migrate
" " shell.

models.py:

```
class Airport(models.Model):
    code = models.CharField(max_length=3)
    city = models.CharField(max_length=64)
    def __str__(self):
        return f"{self.city} ({self.code})"
```

class Flight(models.Model):

origin = models.ForeignKey(Airport, on_delete = models.CASCADE, related_name = "departures").

destination = " " (Airpod, " ", " ", " " = "arrivals")

duration = models.IntegerField()

def __str__(self):

return f"{self.id}: {self.origin} to {self.destination}"

cmd: repeat cmd: ③Step

cmd

from flights.models import *

jfk = Airport(code = "JFK", city = "New York") } 4 Airport.

jfk.save().

f = Flight(origin=jfk, destination=lhr, duration=415)

→ f.save().

→ f → f.origin.code

→ f.origin → lhr.arrivals.all().

index.html

{% extends "flights/layout.html" %}

{% block body %}

<h1> Flights </h1>

{% for flight in flights %}

 Flight {{flight.id}}: {{flight.origin}} to {{flight.destination}}

{% end for %}

{% endblock %}

Shell

Airport.objects.filter(city="New York").first()

.get(")

So little task to do again & again use shell to update

thinks that why use admin-django.

cmd:

Python manage.py createSuperuser.

→ Username
→ email
→ password: OP ....

Instead of using shell, admin provide GUI interface to update the details.

→ Admin.
→ Create User login page using in build fn authetication System.

Java Script :-

→ Client-side-code (inside the client computer) no need for server processing
→ can manipulate DOM directly.

```
<script>
    alert('Hello, world');
</script>
```

Events : anything User does on the html page
→ like User click, User Scroll, User select dropdown list are ex.

```
function hello()
{
    alert("  ")
}
```

```
<button onclick="hello()"> click me </button>
```

Variable :

1) let

Query Selector : select the element from the html & manipulate it.

ex: function hello()
{
    let heading = document.querySelector('h1')
    document.querySelector('h1').innerHTML = 'goodbye';
}
heading.

2) Const - if value is not changing.

DOM manipulation :

1 - back tick . instead of f" " in JS.

```
if (counter % 10 === 0) {
    alert(`Count is now ${counter}`);
}
```

plugin

```
document.addEventListener('DOMContentLoaded', function() {
});
```

} Anonymous fn.

Scanned with OKEN Scanner

Query Selector

- document.querySelector('tag')
- document.querySelector('#id')
- document.querySelector('.class').

→ data-color Attribute.

new notation arrow.

→ document.querySelectorAll('button').forEach(button => {

```
    button.onclick = function() {
document.querySelector('#hello').style.color = button.dataset.color;
    }
});
```

→ document.querySelector('select').onchange = function() {

```
document.querySelector('#hello').style.color = this.value;
    }
```

Events:

→ Onclick
→ On mouse Over
→ onkeydown
→ On Keyup
→ Onload
→ onblur.

Intervals : time

```
Set Interval(count, 1000);
```

local Storage :-

→ local Storage.getItem(key)
→ local Storage.setItem(key, value).

```
if(!localStorage.getItem('counter')) {
    local Storage.setItem('counter', 0);
}.
```

APIs (Application programming Interfaces)

→ well defined Services can Communicate with each other.

→ Interact with map, amazon, wheather.

→ JSON. (JS Object Notation).

```
{
    "Origin": "New York",
    "destination": "London",
    "duration": "415
}
```

JSON representation.

```
<script>
<DomLoaded>
fetch('https ').then(response => {
    return response.json()
})
```

```
{ "rates": {
    "EUN": 0.907,
    "JPY": 109.716,
    "GBP": 0.766,
    "AUD": 1.479
  },
  "base": "USD"
}
```

```
fetch('
.then(response => respons.json())
.then(data => {
    const currency = document.querySelector
        ('#currency').value.toUpperCase();
    const rate = data.rates[currency];
    if (rate !== undefined) {
        d.qs('#result).innerHTML= " ";
    } else d.qs( ).  "  = 'Invalid'
    }
})
.catch(error => {
    Console.log('error', evor);
});
return false;
};
```

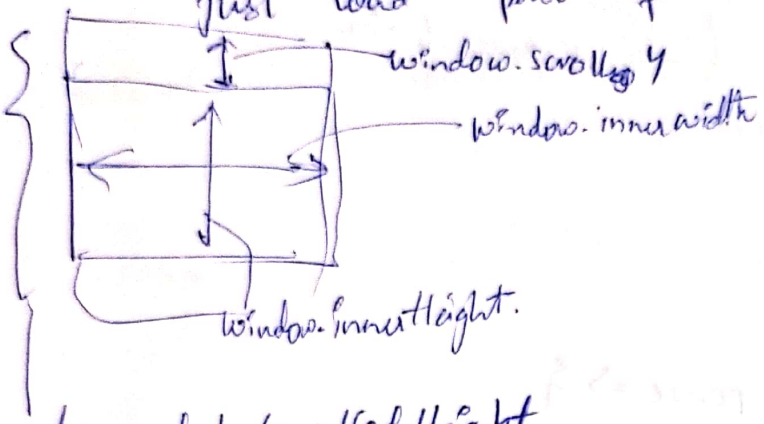
User Interfaces :-

→ Single-page Application.

(Django uses routes to change the html page (multi page app)

→ JS changes the DOM which is single page.

just load part of the page.



→ to store history of the page JS is used.

⇒ changing content of the page without url is changing JS is used.

document.body.offsetHeight

```
window.onscroll = () => {
    if (window.innerHeight + window.scrollY >= document.body.offsetHeight) {
        document.querySelector('body').style.background = 'green';
    } else {
        "  "  "  "  "  = 'white';
    }
}
```

⇒ It is use for infinite scroll. (Ex: news app).

⇒ Animation - ability to move things around.

- To change size of something.

(a) Keyframes grow { (grow is name)

```
from {
    font-size: 20px;
}
to {
    font-size: 100px;
}
}
```

```
h1 {
    animation-name: grow; / move  position: relative
    animation-duration: 2s;
    animation-fill-mode: forwards;
}
```

(a) Keyframe move {

```
0%. { left: 0%.;
}
50%. { left: 50%.;
}
100%. { left: 100%;
}.
```

opacity:
animation: padding 0 → 50 → 75 → 100.
margin

as web pages lot of code of JS.

→ more interactive & dynamic so on & so for.

React: JavaScript library (framework).

→ Design UI which is very interactive.

Declarative Programming :-

view
<h1>0</h1>

It is imperative Programming :

logic

```
let num = parseInt(document.querySelector('h1').innerHTML);
num += 1
document.querySelector('h1').innerHTML = num;
```

Too much code for simple update.

Scanned with OKEN Scanner

declarative programming :-

View :

<h1> {num} </h1>

Logic

num += 1 ;

} with the help of react.

react divides our application into whole bundle of Component

→ Component based on the Underline State. Can be manipulate the State.

→ Package js for react :

1) React (define component & how they behave)
2) ReactDOM (take react Component & insert into DOM
3) Babel. (translate code into one language to another). jsx → js file for browser

jsx (extention of Javascript to write react).

Script :

src = "https://unpkg.com/react@17.0.2/umd/react.production.min.js"

<script src = " " /react-dom@17.0.2/umd/react-dom.production.min.js </sc

<Script src = " " /babel-standalone@6.26.0/babel.min.js .

```
<body>
  <div id = "app" > </div>
  <script type = "text/babel" >
    function App() {
      return (<div> Hello ! </ /div >
      ); }
```

Scanned with OKEN Scanner

ReactDOM.render(<App />, document.querySelector("#app"));

</script>

</body>

React.useState() → React hook.

Counter.html

```
<div id="app"></div>
<script type="text/babel">
  fn App() {
    Const [count, setCount] = React.useState(0);
    function updatecount() {
        Set count(count+1);
      }
    return (
      <div>
      <div>{count}</div>
      <button onclick={updatcount}>count</button>
      </div>
    );
}
ReactDOM.render(<App />, document.getElementById('app'));
</script>
```

Testing :-

for python : assert command : Something should be true.

Assation Error will be Occur. (Square is example)

→ Script is used to check the test cases :

(Sh) shell sciph .sh file is made

test.sh

python3 -c "from test0 import test_prime; test_prime(1, False

" " " " " " (2, True)

" " " " " " (8, False)

- Compile

  Sh filename.sh.

Unit Test :

```
import Unittest
from prime import is_prime
class Tests(Unittest.TestCase):
    def test_1(self):
        """check that 1 is not prime."""
        self.assertFalse(is_prime(1)) /2/e/1,/25/28
if __name__ == "__main__":
    Unittest.main().
```

your test should be good Covered of test.

django testing : flight to ~~avoid~~ valid if origin & destination should be different & duration should be greater then zero.

→ Django provide test.py } does not affect the actual database.
to check

Cmd: python manage.py test.

→ browser testing Unit Test mean, (what is we get VS, what should be)

→ Selenium is Used to test browser.

```
from tests import *
uri = file_uri("counter.html")
driver.get(uri)
driver.title
driver.find_element_by_id("increase")
increase = "  "  "
increase.click().
for i in range(25):
    increase.click()
```

decrease = " " "
" "
" "
"

Unit test methods:

→ assertEqual
→ assertNotEqual
→ assertTrue
→ assertFalse
→ assertIn
→ assertNotIn.

Scanned with OKEN Scanner

CI/CD:

→ Continuous Integration
  1) frequent merges to main branch
  2) Automated Unit testing.

→ Continuous Delivery
  1) Short release schedules.

Git Action: create work flow
  → To automate the test

YAML: language is used. (yml file extension).

```
key1: value1
key2: value2
key3:
  - item1
  - Item 2
  - Item3.
```

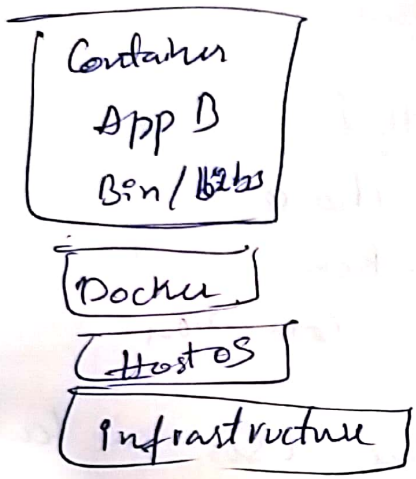
ci.yml

```
name: Testing
on: push

jobs:
  test-projects:
  runs-on: ubuntu latest
  steps:
  -uses: actions/checkout@v2
  -name: Run Django Unit tests
  run: pip3 install --user django
       python3 manage.py test.
```

Scanned with OKEN Scanner

Docker :

-> diffrence b/w your machine to Sever.
-> To Solve this issue docker is
-> It coork's on my machine problem.
-> run In an Container.



Dockerfile : (no extension) Name is imp.

```
FROM  python :3
COPY .  /usr/svc/app
WORKDIR  /usr/src/app
RUN  pip install -r requirement.txt
CMD = ["python 3", "manage.py",
   "run server", "0.0.0.0:8000"]
```

docker-compose.yml

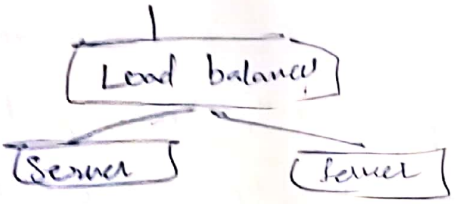
```
// docker version
version : '3'
services :
    db :
        image : postgres

    web :
        build : .
        volumes :
            - . :/usr/src/app
        ports :
            - "8000 : 8000"
              host  Container
```

docker-compose UP

Servers:
→ on cloud
→ On premise

→ Benchmarking. } check how many User can handle at a particular time.
tools: Apache bench.

1) Vertical Scaling. [S] → [Server]
2) horizontal Scaling → [S] [S]



L.B method
1) Random choice
2) Round-Robin
3) Fewest Connection

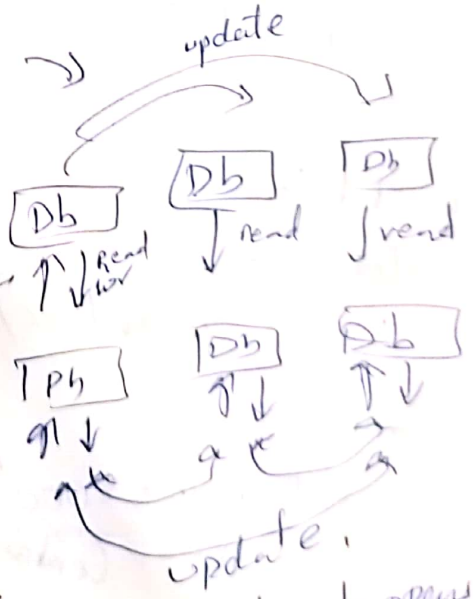
Session-Aware Load Balancer:
1) Sticky Sessions.
2) Session in Database
3) Client-side Sessions.

Some time User are low
Some time User are high
Ex: new website
Sol": Autoscaling.
Scale up → Increases Se
Scale down → decrease Se

Scaling Database:
1) Database partitioning.
flights
flights_domestic
flights_international.

2) Database Replication
i) Single-Primary
2) Multi-Primary

Single point of failure →



Conflict happens.

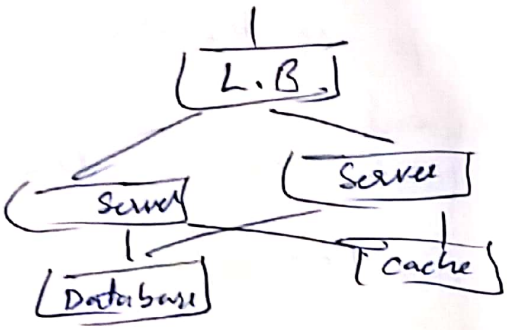
Caching:
Storing Save version of some info.
no need of continuous request from Server.

Client - Side Caching :-

Cache - Control : max - age = 86400

ETag : "48xxxxxxx" — resource — load new version of resource / change according to the update.

Server - Side Caching :-



Django Cache Framework :

1) Per-view Caching
2) Template Fragment ~~Chan~~ caching
3) low-level Cache API

Security :

Git - Open Source Software. → If credential was expose It can be backtreat tract.

html → phishing attack → www.goggle.com → take you phising website.

http - information can be extract from the middle like b/w router

Secret-key Cryptography :

key plain text → ciphertext

Ciphertext key → plain text.

Public-key Cryptography :-

1) public key 2) private key.

plaintext public key → ciphertext

Ciphertext private key → plain text.

https

SQL Security:

User table.

id | username | password. } this info should be hide.

hash function → use to hide password info of the user.

→ 486284CB.

that why bank doesn't know your password because of which they will say that change your password there only know about hash function. (update table is easy) to say.

email to reset password:

leaking information of User have account (or) not.

SQL Injection:

username

"hacker" -- " } — ignore rest of the password & login.

API keys:

Rate limiting - no user can request so many request can lead to crash

Route Authentication — API keys allocate to User.

Cross Site Scripting: Uses: env variable.

→ javascript vulnerability

→ on your web App javascript of another person code will run.

→ someone else can change javascript.

ex. 127.0.0.1:8000/<script>alert('hi');</script>.

→ lead DOM, API request This is another person changes wrote the JS.

Scanned with OKEN Scanner

Cross site Request forgery :-

→ fake a request to a website.

click me

through img :

<img src = "https://yourbank.com/transfer?to=brian&amt=2000">

through Post request :

<form action = "https://yourbank.com/transfer" method = "post">
<input type = "hidden" name = "to" value = "brian">
<input type = "hidden" name = "amt" value = "2500">
<input type = "Submit" value = "Click me">
</form>
<body onload = "document.forms[0].submit()"> : starting named of class

{% csrf_token %} is use to safe guard.

Other frameworks web :

- Server side
  - Express - js
  - Ruby on Rails
- client - side
  - Agular JS
  - React
  - Vue.js .

Scanned with OKEN Scanner

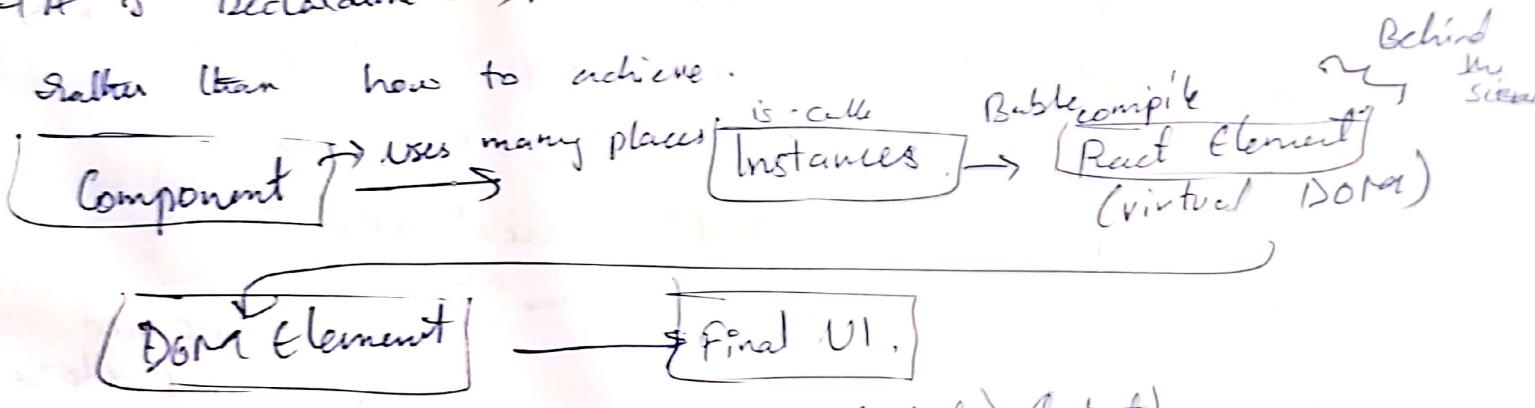
React - 3 project. Library for to create awesome frontent. (UI).

React is all about Component.

→ Don't repeat your self.

→ It all about making Common code
nav bar is Common for all page
nav link, header, and Card.

→ It is Declarative. It describes what the UI should look like
Rather than how to achieve.



npm → yarn → pnpm → bun (latest) (fastest).
(no one use).

Project Structure :-

→ node_structure : It Contain all necessary libraries & dependencies by React
→ ignore folder.

→ public : It Contain all Static files like images, video, font etc

→ Src : It Contain all Source file (Component, JS file, CSS)
→ App.jsx is main Component.
→ main.jsx is an entry point
→ eslintrc.cjs It make sure code is prefect or not.
→ we always ignore node_structure → git ignore

→ Index.html → React App is loaded serving the entry point for web browser.

Scanned with OKEN Scanner

→ package.json : It contain all detail & configuration of the project.

Naming Convention:

Camel Case: Variables, function, properties inside obj, file names etc

Pascal Case: Component names, class names, types etc.

Snake_case: In python, Seperated by '_'

Kebab-case: file name, css class, id, Separated by '-'.

Component (.jsx): html code which return jsx file.

Two ways: 1) class based Components
2) functional Components (✓) Imp.

Hot Module Replacement (HMR) help to save changes live

React Fragment: (Component can't return multiple element without a wrapper div) prob solved.

→ Array of Elements with keys.

import { fragment } from "react"; <></> // </>

return <fragment> // </Fragment>

Dynamic Values in jsx:

{ } → Insert the values. (To write JS in HTML)

Conditionals in JSX return {age >= 18 ? "Adult" : "Minor"} .

→ ternary Operator is used

Scanned with OKEN Scanner

Portfolio website :-

1 rem = 16 pixel

0 rem = 32 pixel

Scanned with OKEN Scanner