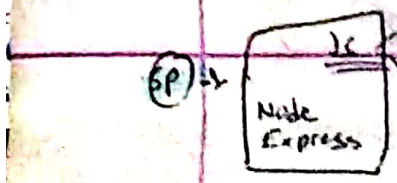


→ combo of hardware & software

web client — web app — web server
(Browser) (runs)

Processing power for searching result — client
(all results) — own device

✓ my result — server (Data Processing on server end)
any

→ diff version, won't affect ~~app~~ app on diff device (updating)

• Generic app vs web app

☐ web application

- Piece of software that runs on a web server
- multiple instance, same version → 2 game on 2 device
- Needs web server, database to operate
  ↓
  app kept on server, not device

☐ web server

- Physical device w/ networking capability
- Serves web content

1. Listen
2. on a port
3. For a request
4. Via some protocol (HTTP, HTTPS)
5. Response
6. Resource (Doc, Image, HTML, JSON)

| web client give req
| web server receive req and send back content.

| web apps separated by port number.

AB2 — webserver
port 5 — web apps
CSE dept — webapp
work — CSE Secretary

Scanned with CamScanner

web client req web server at a particular port address (web app)

at a particular format. AP2 CSE/14/18/19 (address)

54.92.0.0 : 16
ip port

https://www.amazon.com/orders/123 → URL (Uniform Resource Location)

protocol — A set of agreed upon rules betwn client & server)

Domain Name (web app Address) — CSE dept address

resource address uniquely identify each content (ordering id / result)

- Don't get response
  - following protocol

Domain Name to address → Every ISP connected to a DNS server → web address to ipaddress & port no.

web server
ip address
port number

- 1 server can host multiple web app (2^{16} max web apps)

  - Port address 16 digits

- Load distribution — same app, multiple instance, in multiple port

  Each instance one port (amazon)

  If multiple instances give req to same port → load

Default
HTTP – 80 (Port Address)
HTTPS – 443 (Port Address)

when you don't specify port address, go to default)

HTTP req → web server
HTTP response ← web server

Scanned with CamScanner

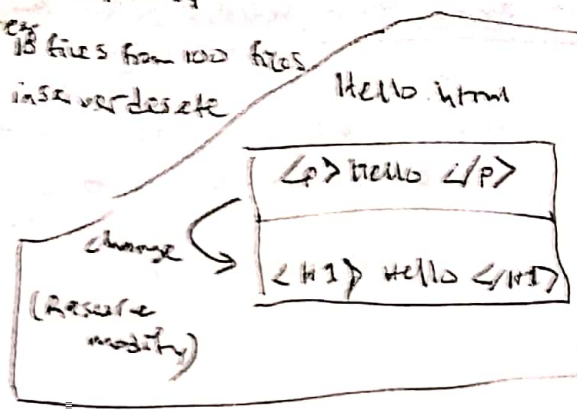
HTTP request

Method
- GET
- POST

Send req to server
get response

- PUT → resource modify
- DELETE → e.g. 13 files from 100 files
insert or delete

Hello.html



- HTTP req format:

POST → Method → resource add 123

GET /users/123

HTTP Header Info

web app host → Host : 54.92.0.0

Port :

device info (client info) → User-agent : - browser → version
- OS → version

Content length : → no body, no content length

connection : keep-alive

new line

Body

only for POST (not GET)
- sensitive info (username, pass)

GET → no body

POST → has body

Response Code (Status Code)

- HTTP response :

HTTP version : 1.1 Response code OK

1xx → Info

2xx → Success

200 → resource exist

201 → resource process successful

3xx → Redirect

4xx → client error

5xx → server down

Phrase — predefined / user defined

eg not valid format

Content length : [ ]

Connection : closed

Resource

client server communication —

connection established

keep alive — keep port open (req)

close — (response) received

- In HTTP protocol, server cannot initiate connection start from client.

Scanned with CamScanner

SP-2

Info transfer thru HTTP protocol

Static vs Dynamic website

- resource not changed
  (data not injected (changed) from server)
- → resource is sent back to client as it is
- → all resources static

x.html

<Title> sample </Title>

<H1> Something </H1>

Dynamic website

- 1/more resources static
- → data injected (modified html file)

*.html

username

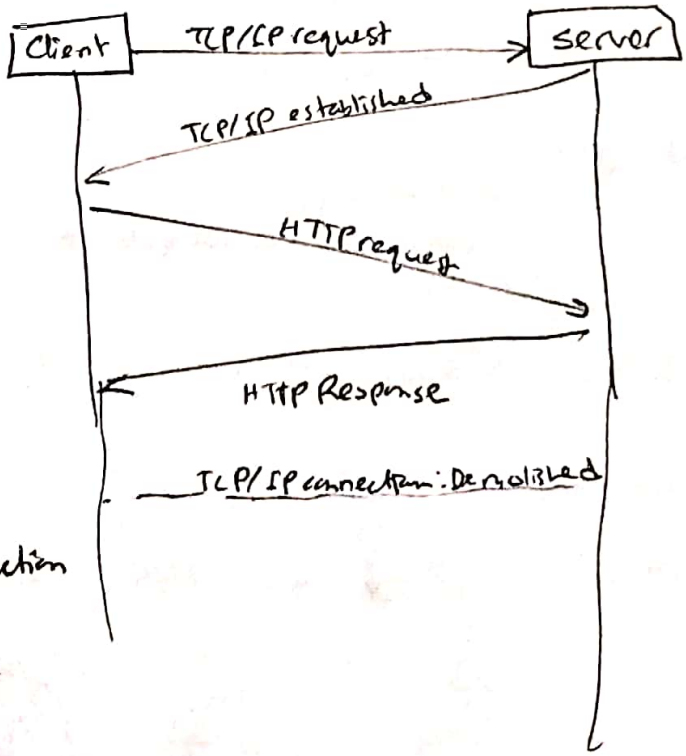
HTTP protocol

- Stateless
- Connectionless
- TCP/IP based
- Application layer protocol
- Media-Independent.
  (Can carry any type of media) in HTTP req

HTTP/1.0 → 1 TCP/IP Time (req, 1 response)

HTTP/1.1
↳ 1 TCP/IP connection:
(many req, many response)

- server decides when to close connection



Scanned with CamScanner

Every request handled by server separately.

Stateful — remember previous states (o req)

Stateless — forgets previous states/info

Connectionless — Renew connection every time (req)

Header

MIME Type = multipurpose Internet mail Extension Type
(file type)

"text/html"

"text/plain"

"image/jpg"

"—— /png

"video/mp4"

RFC = Request for Comment
↳ doc where webrules are written.

TCP/IP

32 bits

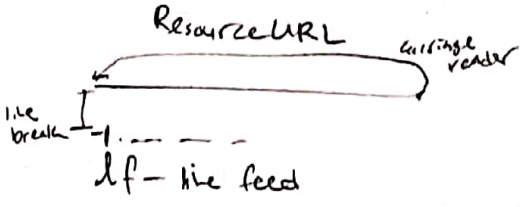
Source Port	Dest port
16 bits	16 bits

max no. of ports = 2^{16}

HTTPreq

GET
POST
PUT
DELETE
OPTIONS
CONNECT
PATCH
HEAD
TRACE

URL — Relative URL
Resource URL



Req line — 1 line
Header — multiline

host?

host: facebook.com → compulsory
agent: - - -
connection: - - - → optional

key value pair in body

GET method

POST method

$key_1 = value_1$ & $key_2 = value_2$

+ special character : %(Hex of ASCII)

username = swe19

Name = SWE 19

username = swe19 & Name = SWE %20 $\boxed{19}$

space (Hex of ASCII)

ASCII = 32

Hex of 32 = 20

Client runs JS code

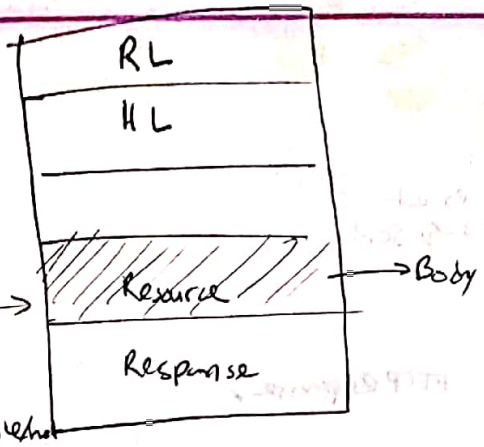
Query data stays in key-value pair.

Query data converted to JSON.

HTTP response

Scanned with CamScanner

GET



Data transfer → GET POST

HEAD →

server info
resource available/not
content length
size
type

GET vs Head

GET →

HEAD →

PUT vs Patch

entire resource change

particular segment of resource change

↓

client end to server — resource change in server (data replace)

has body

Normally, client does not get access to change resources in server. (embed in file)

Cross-side scripting — Js injection prevention

X-XSS Protection: 1

Js inject in response — run in client — if malicious code

OPTIONS — server available options back

CONNECT — create encrypted tunnel between client & server (whole data encrypted) → communication secure → Isp not available

If server has SSL certification (HTTPS) → can use CONNECT

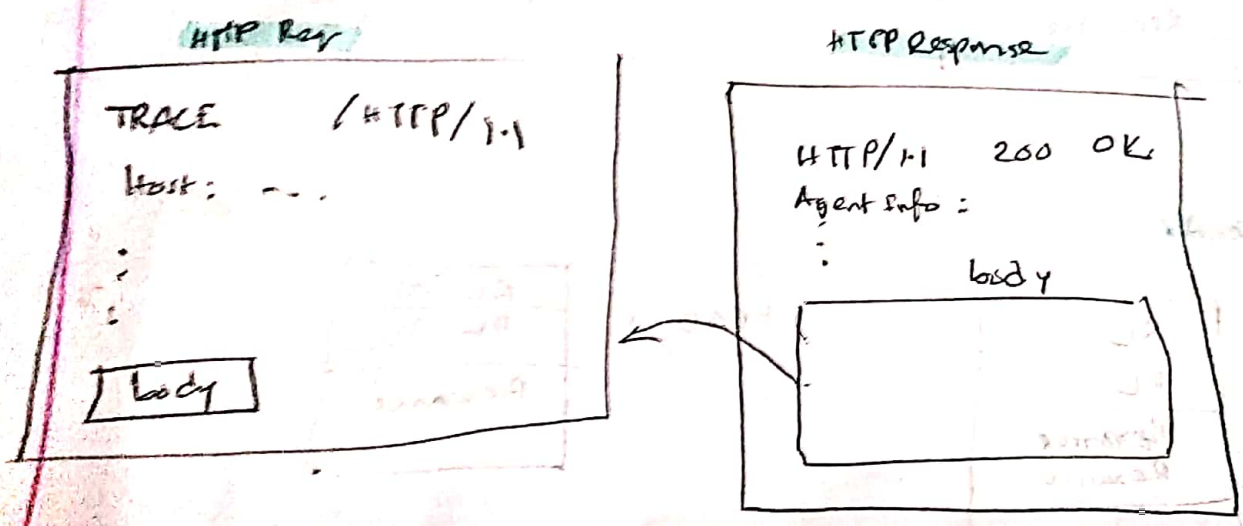
prev sev info not saved (stateless)

SSL certificate → TLS (updated)

Scanned with CamScanner

TRACE → debug in network level

get response (body → exact raw sent)



Scanned with CamScanner

(54)

querystring → ? id = 1 & name = mashiur / Rahman

☐ URL fragments

https://www.swe4537.com/home? theme=dark # projects-section

URL scheme | host (IP+Port) address | resource Path | query string | fragments

☐ What is JavaScript?

- HTML is a markup language that we use to define our web contents
  eg paragraphs, heading, tables etc.

To make component (No processing power)

- CSS is a language of style rules that we use to apply styling to our HTML components
  eg font size, color etc.

- JavaScript is a scripting language that can manipulate both HTML & CSS. which in turn enables us to create dynamically updating content, control multimedia, animate images and so on.
  - client side → javascript, JQuery, AJAX
  - server side → Javascript, PHP, Perl

compiled

Javascript code compilation embed in HTML

HTML

```
<H1> ... </H1>
<P>      div?
id =
<div> → id = projects
</div>
```

Apply condition on any component → dynamic

eg | 24 fps — 24 images per sec
2 sec video (animation)

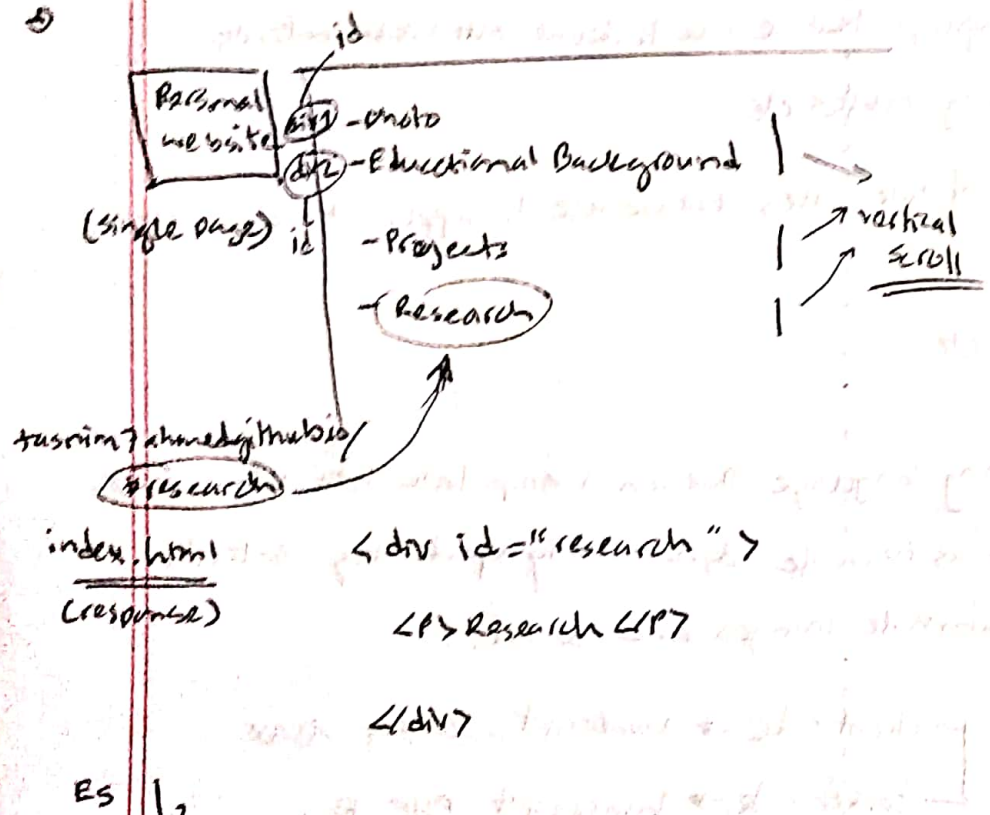
Scanned with CamScanner

client ⇄ (request / response) server

Why is it called Java Script?

- To make websites live/responsive by utilizing client-side processing power.
- Livescript, then, Javascript
- It doesn't make websites dynamic

5)

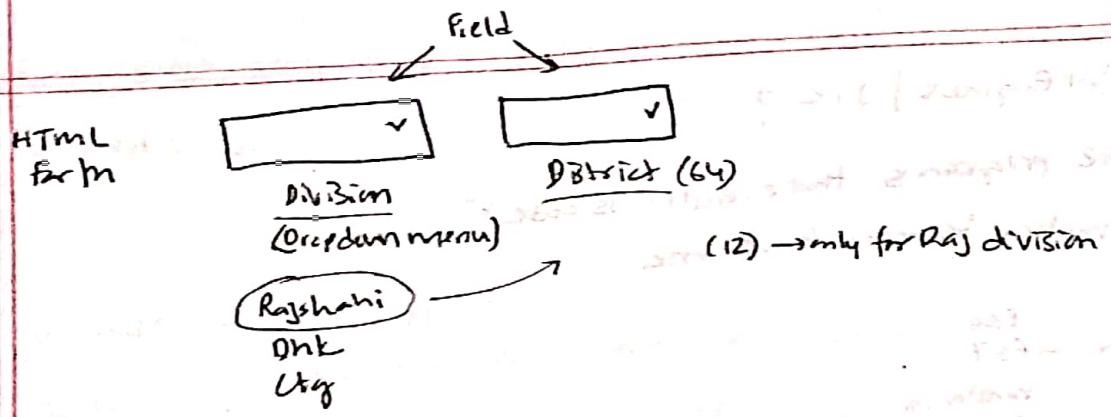


ES Node ? Popup

website —JS→ responsive

static/dynamic depend on server (not JS!)

Scanned with CamScanner



Page refresh / reload everytime you send req to server. (select division = Raj)

JS cannot make a static website to dynamic. It can only make the website responsive.

☐ scripting lang vs programming lang

↓ not compiled, directly execute

↓ all code compiled before execution (.exe file generated)

Prev

Now JIT compiler (Just In Time) – only the portion which is executed will be compiled

vscode → all prog. lang code → CLR compile (prog lang) → .msi format → execute

☐ why website (JS) uses JIT compiler but other programming lang doesn't?

response already given know which functions will be required.

you don't know which functions will be required so need to compile all functi

- Javascript Engines | JIC ?

- software programs that executes JS code
- Every modern browsers have one

- JS versions – ES6
  ES7
  vanilla JS
  ECMA 6, 7

var → let
ES7 → λ function

Lang
- strongly typed : C, C++, C# (variable type fixed)
- Dynamically typed : Python, JS, C# (one variable can hold any data type)

- JS Frontend Framework (uses) — Typescript
  - don't mention
  - mention data type when creating variable
    - no inconsistent state

Frontend communicate w/ serverside.

Scanned with CamScanner

JS syntax

JS — ; / not ✓

comments — // single line

/* */ multiline

prettier — code beautify

variables — var, let

constants — (system specific) const total marks = 100;

(universal) const PI = 3.14;

Data types

- number
- BigInt
- String
- Boolean
- null
- undefined
- Symbol

JS → Primitive — hold 1 value

Non-primitive — hold multiple values → JS object

floating point
Integer
+infinity
~~integer~~
-infinity
NaN (Not a Number) → $\frac{0}{0}$, sk/inf

number + 'n' — 1234n

BigInt ( ) — typecast

JSON — JS object
~~or~~ JS classes

+ → concatenation operation

JS calculation from left.

2 + '5' = '25' → calculation (not math)

'5' + 2 = '52'

4 / 2 = 2

4 / '2' = 2

4 / ' 2 ' = 2

4 / '1 - 2 - 3' = NaN

1 + true (1) = 2

1 + false (0) = 1

Key value | Symbol ×

Null (defined, but no value) → empty value

undefined (not defined) → server give no response (no value)

Scanned with CamScanner

server has list
51 id don't exist
return null

if
elseif
else

String() ✓

toString() ✗

== | value ✓
=== | value + datatype ✓

'1' == 1 → true

'1' === 1 → false

^ → XOR operator

2 ** 3 → 2^3 (exponent)

Generic function
↗ (lambda function)

function sayHello ( )
{
console.log ("Hello");
}

function function_name (parameter_list)
{
}

SayHello ( ); → function call

Scanned with CamScanner

λ function : (Arrow function) :

function stored in variable ↗ ↓ parameter

```
const sayHello = ( ) => {
    console.log ("sayHello");
}
```

$\boxed{\text{sayHello()}}$

sayHello passed as parameter.

DePa

OOP — To make code flexible, reusable, not modified in case of change ....
(modular approach)

object — hold properties of a class

class — Hold data & methods —
(fields)

this

message pass to objects (through method call)

constructor — initialize class
- default constructor
- own constructor

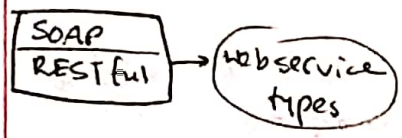
Destructor called automatically (data) — deletes from memory

Access
- Public
- Protected
- Private

Scanned with CamScanner

(SP)

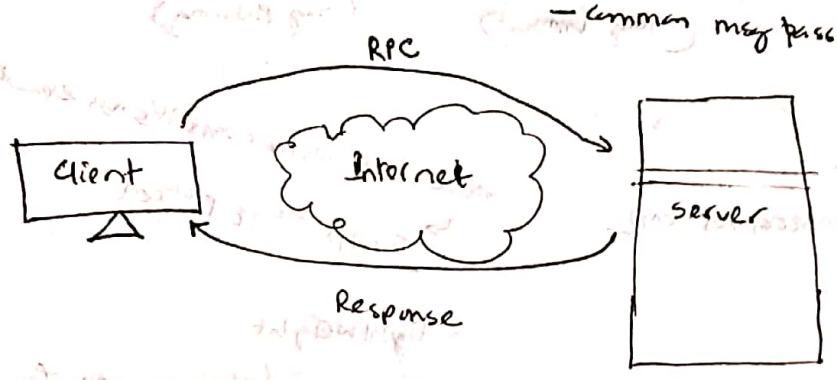
- Web service



Need compatible browser

Interoperability → ~~any~~ (app) seamless connection in any device/OS

comms between diff system (all prog lang understand)
– common msg passing lang → web service



web service is a software system designed to support interoperable machine-machine interaction over a network. Other systems interact with the webservice in a manner prescribed in its description.

- SOAP = Simple Object Access Protocol
  msg passing protocol

RPC = Remote Procedure call

Web Service of Amazon

- product List
- product List + Price
- product List + Category

Apps accessed through network using some particular protocol (HTTP, XML, SMTP) are called web service.
↓
Abstraction layer

Scanned with CamScanner

Web service is an Abstraction layer.

XML Document → SOAP msg.

web services uses

Send XML Doc through HTTP. → SOAP over HTTP

(msg format) (msg passing)

XML – similar to HTML

all prog lang understand XML

since it has restrictions it's not exactly XML

↳ SOAP – msg protocol

- lightweight
- structure follows a specific pattern
- has root element (<Envelope>... </Envelope>)

Authentication | client info, Server info

<Envelope>
<Header> ... </Header>
<Body> ... </Body>
</Envelope>

SOAP msg format

content

content type: text/XML

- content doesn't have any specific format

- except browser, comms w/ server w/ any other app → SOAP
- for browser → HTTP

Scanned with CamScanner

cost function of logistic regression

Likelihood function

Max livelihood — $\max_\theta L_\theta$

$$P(C|x) = \frac{P(x|C)\,P(C)}{P(x)}$$ — Baye's thm

$\log(\text{product}) = \log(a) + \log(b)$
$a \cdot b$

EP

This week — lab task
Next week — viva

SOAD — any type data send (encoded in xml)

ICT centre (webservice)

RPC	StudentInfo
RPC	Result

Web services Description Language (WSDL)

XML Document

SOAP format

- message
  - StudentInfoRequest — SID
  - StudentInfo Response — SID JSON

IRR
- input msg
- output msg

- PortType (in respect to web service)
  - input
  - output

Client $\xrightarrow{\text{SID}}$ Webservice
$\xleftarrow{\text{SID JSON}}$

Scanned with CamScanner

Bindings — Protocol Type : SOAP OVER HTTP
Encoding Type : SOAP

→ XML doc (contains web service documentation)

WSDL Document Format

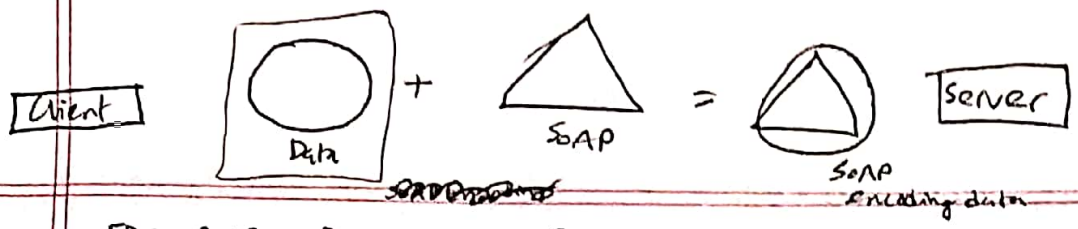
```
<Definition>
    <message>    </message>
    <Port Type>  </Port Type>
    <Bindings>   </Bindings>

</Definition>
```

UDDI = Universal Description, Discovery & Integration
↳ contains list of all webservices (WSDL)

• Advantages of SOAP web services → OS, network, platform independent
(client & server OS may be diff)

- Exposing business functionality | Potential client
- Interoperability (client server comms ✓ no matter what lang)
- Standard Protocol stack (XML → all prog lang understands)
- Reduction in the cost of comms

Scanned with CamScanner



EDI - B2B = Electronic Data Interchange for Business to Business

Protocol stack

1) Service Transport (msg passing)

HTTP, SMTP, FTP

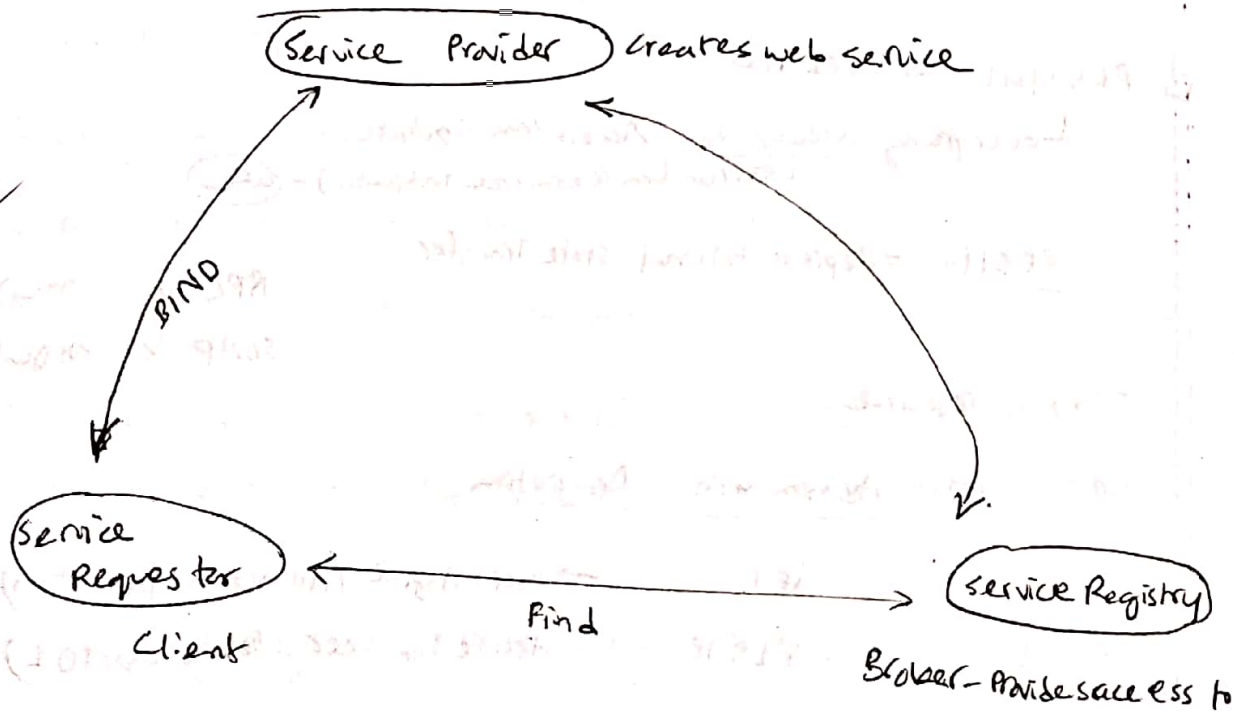
2) XML messaging — Encoded to XML (carry msg)

3) Service Description (WSDL) → webservice

4) Service Discovery (UDDI) → all webservice

} XML

Architecture



Client 1st connects w/ serviceregistry

Scanned with CamScanner

- characteristics

- XML based
- loosely coupled
- synchronous / Asynchronous (RPC call support)

↓ Client 1 req at a time (as long as client comes up server, cannot break connection)

↓ client can req during another req

- Ability to support RPC
- Document exchange

☐ RESTful webservice

└everything is resource: Access/manipulate
URI (uniform Resource Indicator) - URL

RESTful = Representational State Transfer

RPC X XML X UDDI X
SOAP X WSDL X

To hit a resource:

URI	HTTP Action verb	Description
/users	GET	→ user list give (All user information)
/users/1	DELETE	- delete 1 no. user. (Delete user ID 1)

Scanned with CamScanner

- RESTful conventions

1. Resource/Endpoint — Noun
   plural

2. Plural form of resource

3. Response Code

verbs instead of noun in Resource identifier (prob)

/getusers GET ✓

Delete (user) ✗

— User confused!

user ✗
users ✓

☐ SOAP vs RESTful — use SSL

SOAP	RESTful
1. SOAP follows strict pattern	Name
2. SOAP uses XML	XML, JSON, any...
3. RPC call	URL is enough
4. Difficult to implement	✗
5. Requires more bandwidth	✗
6. Easier to implement transactions	✗

↓ Some finish multiple server calls

SOAP
- bank transaction
- user impass

1 database accessed by 3/4 service

1 particular data edit by 3 services

server down → prev state saved — Roll back

connection error

concurrent process — locking (accessibility control)

— RESTful does not do that (faster)

web service vs API

Scanned with CamScanner

(38)

Application Programming Interfaces (API) and web services

network is no issue
over same device (no need network)
lightweight

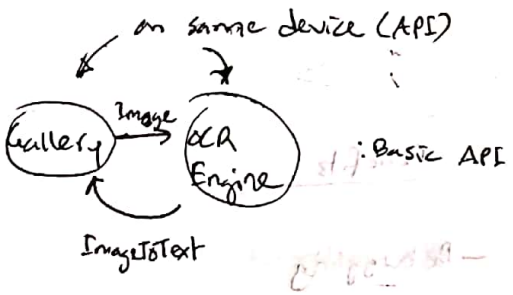
API is a software intermediary that allows two application to talk to each other.

According to world wide web consortium (W3C), web services (over network) provide a standard means of operating between different applications over a network running on variety of platforms and/or frameworks.

internet/organizational
(eg FTP IUT, SIS IUT)

hilo

eg



SOAP msg

```
<Envelope>
  <Header> </Header>
  <Body>
    Hello ! → 6 bytes
  </Body>
</Envelope>
```

web API/
webservice
(same)

All webservices are APIs but all APIs are not web services.

- webAPI
- web API (over the network)
- Basic API (over same device)

Scanned with CamScanner

Criteria	Web Service	API
Need of network	Always	Not always (only needed for web API)
Lightweight Architecture	×	✓
Mode of Communication	SOAP, REST	×
Protocol Restriction	SOAP over HTTP (SOAP | HTTP) or HTTP	×
Hosting	Internet / LocalNet	Device / Internet

meow

Architectural Pattern

- Model View Controller (MVC)
  - Design Pattern

Benefits
- Debugging
- Testing

View – Responsible for all visualization elements (Server data displayed to user)

Model – Database / Storage
↳ Schema

Controller – Set of methods
Business Logic

Scanned with CamScanner

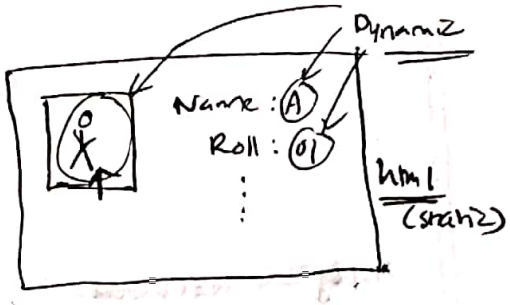
School management system

- Teacher's info
- students info ← show, delete, update

1 website can provide multiple web services.

Server : www.school.com

→ studentID (1) info display →



Route

www.school.com/students/1

HTTP GET request. ⟹ StudentController

↓

display_individual_student

1 controller method bind to each end point.

Students Controller

function call

display_all_student

display_individual_student

delete_individual_student

└ admit_method

methods

Model: retrieve(student-01)

Manipulate view (inject data (P.T))

Return modified view (w/ data TPR)

Model

↓

PostReq

student info

Controller — all changes controller

model: save (studentinfo)

success

↓

view

Benefit

- Stability
- Debugging

unit testing

TDD

(Testdriven Development)

hey

Scanned with CamScanner

SP

Terminologies

MVC

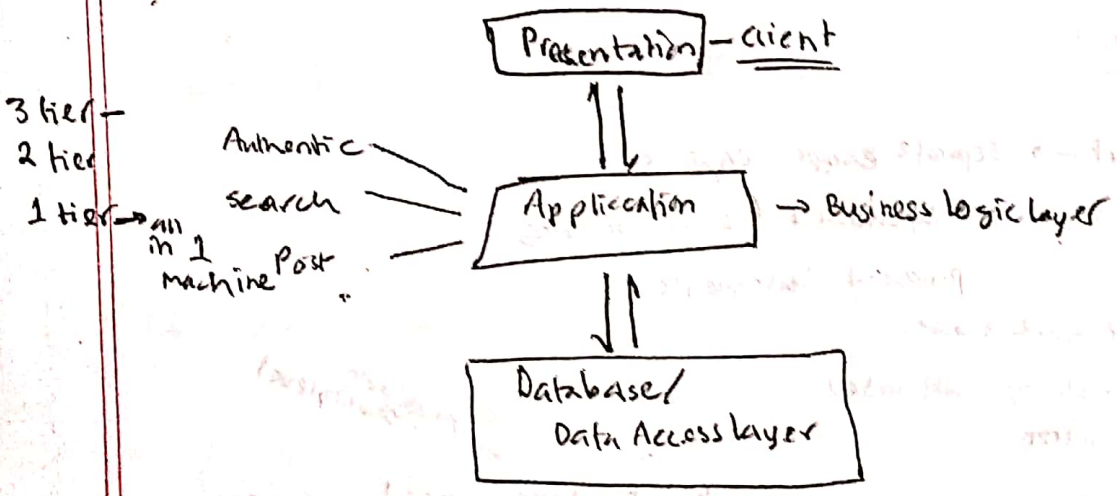
Distributed network

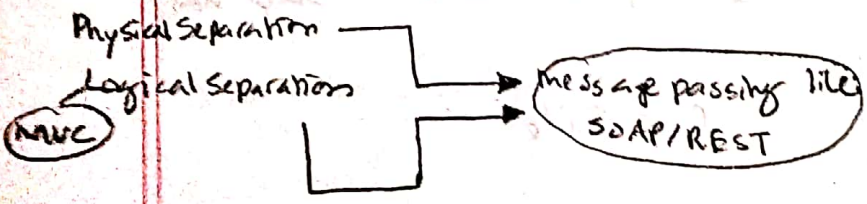
1 computer — multiple servers

- Distributed Network
- Platform → can run any app

□ N-tier Architecture
multi-tier

A program that is distributed among N platforms over a distributed network.





More than 3-tiers

Scanned with CamScanner

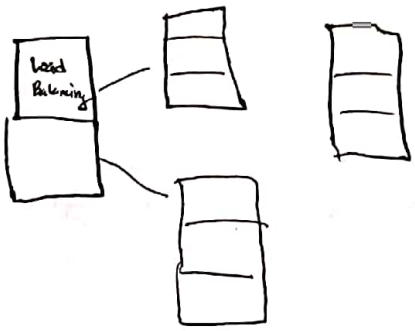
Cons of multitier architecture

- Increase in effort
- Increase in complexity

Every sec 1 million req →
1 server → authentic
1 million req per sec not possible

Pros

- Scalability
  - horizontal scaling → easy is to
  - vertical scaling → case is



- Security
  - ↳ business logic compromised
    but database is not
    but in singletier arch. all are compromise at once.

- Fault tolerance (single point of failure)
  - ↳ search not working
    but other servers working ✓
    Find out which server crashed → which feature under that server

- Maintanibility
- Reusability.

Scanned with CamScanner

(Sp) → System Design

Software Design Architectural Pattern
MVC architecture – server + client
↳ n tier architecture.

↓

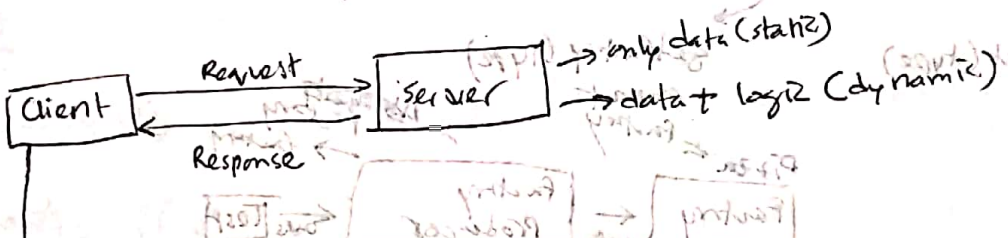
Software Design Architectural Pattern

(1) N-tier / multi-tier / Layer arch. pattern →

Presentation
Business
Database

interconnected but don't know where data comes from

(2) – Basic Client-Server Architecture



- Thick client (more client end processing than server end processing)
  eg PubG, video editor (auth)
- Thin client (less client side processing than server processing)
  eg Netflix → heavy load on server

Server & client both have processing power

Browser/Machine's (client) processing power use → seq. to server less less time. ✓

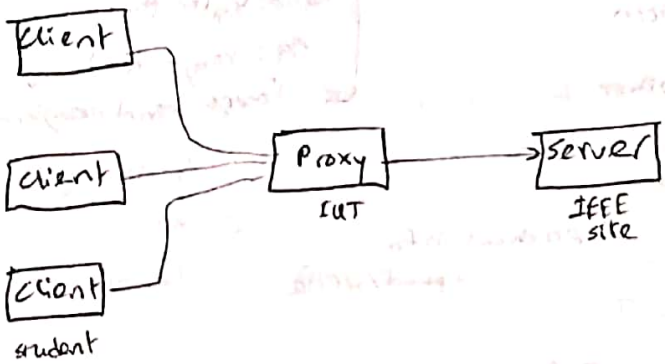
Proxy server → middleman (Intermediary) betwn client & server

Forward Proxy

Reverse Proxy

Scanned with CamScanner

Forward proxy

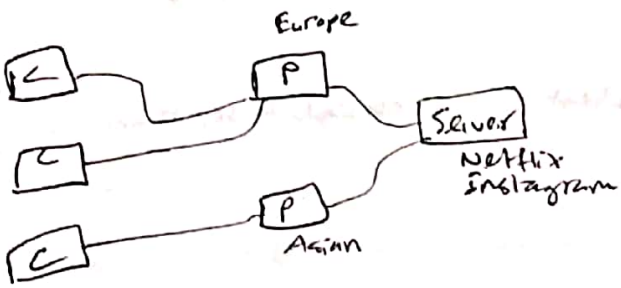


eg IUT is/has proxy server

VPN

Forward Proxy
~~Reverse Proxy~~

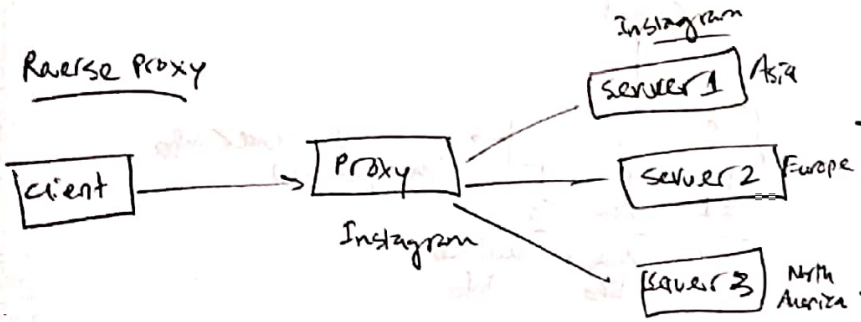
- Anonymity of client
- malicious data blocking
- Subscription (giving access to proxy server means giving access to all clients)
- cached response



Data saved in proxy server. That region pictures saved in Proxy server. (Cache)

main Server gets freed up to do other tasks.

Reverse Proxy



Reverse Proxy

- Cannot attack actual server (can attack proxy server)
- Traffic control / Load balancing (req redirect to server 1/2/3)

S1, S2, S3 → copy of each other (auth all)
only diff is cached data (europe — european celebrities)

Proxy server has no processing power (response generate)
Proxy server → req redirect

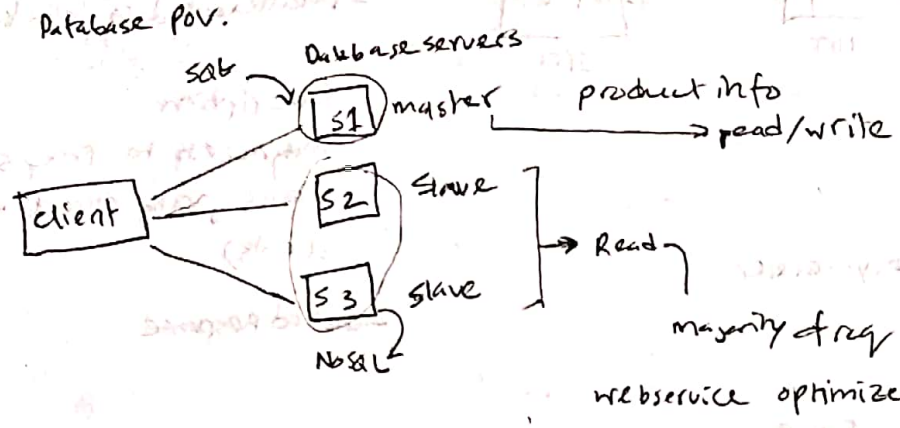
Scanned with CamScanner

③ — Master Slave Arch. Pattern

[Read faster than write.
99% req → read req
(more read req comes)]

multiple servers copy of each other

Database POV.

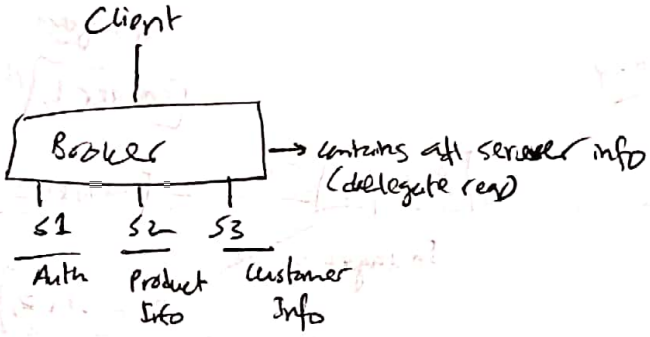


Response time : Time to fetch data for the client

No sql faster than SQL

~~Brokers Arch Pattern~~

④ — Broker Pattern



Scanned with CamScanner

⑤ Peer to Peer

Doesn't distinguish betw^n client and server

e.g. Torrent
→ File sharing
→ centralized server ✗

CONS
- Data decentralized (Data availability depends on a node)
  if node destroyed, data lost
- security

Depends on
- Resource
- Responsibility

→ which arch. to use?

microservice arch. — acc. to need., choose broker/client server/...

Target: Reduce/minimize response time
(optimize — diff arch pattern)

Scanned with CamScanner

(5P)

Express – MVC framework

Node

```
const http = require ('http');

const hostname = "127.0.0.1";
const port = 3000;

const server = http.create server (function (req, res) {
        res.writeHead (200,
  ⋮
```

Express

```
const express = require ('express')

const app = express( )

const port = 3000
  ⋮
```

res.render( ) →
No need res.end( )

Route

/	GET	'Hello world
/home	GET	Home Page
/about	GET	About Page

/admin → admin privilege

(≥1 middleware) | check Auth

/about
/home

app.get ('/', (req, res) => {
  res.send('Hello');
  Home
  About
  Admin
});

→ call other functions

callback function

→ response generate

post
to
delete

tasnim fahmed Express JS Shift 4 5 7 7

(5P) Authentication and Authorization

Auth – who you are

From client POV

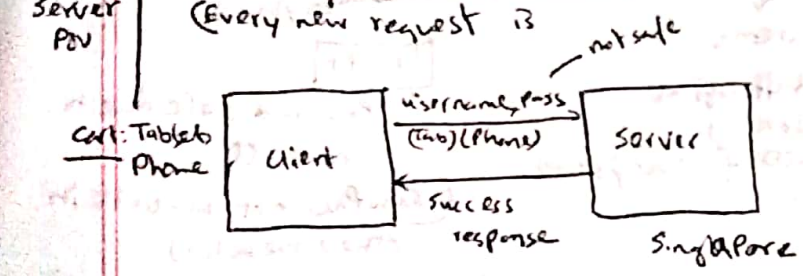
users can be authenticated based on 3 factors

1. Something that you know → Password
2. Something that you have → ATM card
3. Something that you are → Biometrics

1) single factor – use any 1 of the above 3 factors
2) Two factor – use 2 of the above 3 factors
3) multifactor – use 3 of the above 3 factors

From server POV

HTTP is a stateless protocol.
(Every new request is



Eavesdropper
└ encryption? → not full proof
→ computationally expensive ✗

Share time
└→ >1 ISP use the same line

Session-based Authentication ✓

↓
Storage
(Browser/Tab close) – session gone (session lifetime)

Application specific
- until log out
- fixed validity

Scanned with CamScanner

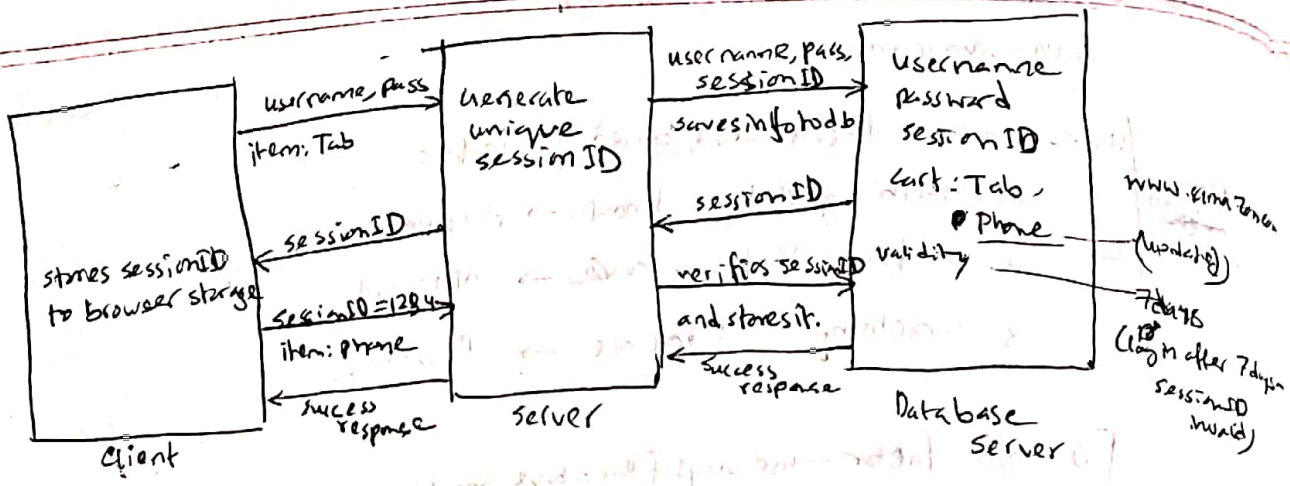


fig: session based Authentication

Limitations

→ Cross-site Request Forgery (Pretend to be sakib while I am Tasnim)

→ Payload

Distributed Systems

10 servers for diff regions (Asia, NA, Australia)
- Need sessionID for every servers

Benefit

↳ Password safe due to SessionID (Same Pass. not vulnerable in other websites)

sessionID compromised → one website vulnerable

password compromised → all websites vulnerable

Browser storage

storage capacity

- local storage : 5MB
- Session storage : 5-10 MB
- cookies : 4KB/website or 4KB/ID

↳ Automatically gets added to HTTP Header

	1	2	3
storage capacity:	5MB	5-10MB	4KB
Accessibility:	Client + server	Client + server	Server ✓

Client — conditional (HTTPonly-true) ↓ no client side code access cookie | Access by any JS code (XSS) → take sessionID

(secure — true) | SSL certificate (HTTPS)

Lifetime:	Infinite (Need to explicitly delete)	Tab/Browser closed	Application specific

Scanned with CamScanner

cookie

- session cookie : lifetime → application specific (normally until logout)
- permanent cookie :
- third party cookie : which type program you are viewing most

Scanned with CamScanner

(SP)

Token Based Authentication

→ Problems w/ session-based auth

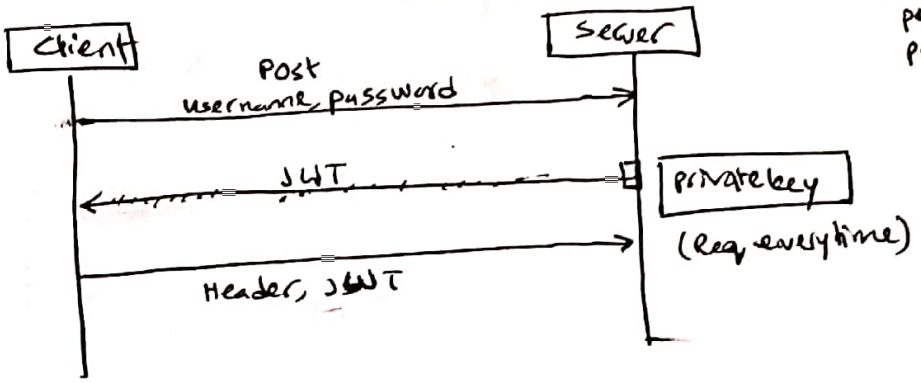
- Distributed systems
- (session ID value compromised means eavesdroppers use them) Session Compromise
- Database Lookup (Time consuming) : prev sessionID

need to store in database

session ID → value

→ use token-based

JWT (JSON web Token) | JSON



public key → data (verify source)
private key → handshaking protocol (client-server)

JWT
- Header
  - Token type → "JWT"
  - Algorithm → (encryption used) "RSA"
- Payload
- Signature (Integrity) → encrypted version of header + payload

eg HMAC ← symmetric algo → 1 key (client, server)

eg RSA, ECDSA ← Asymmetric algo → 1 public key : source verification
1 private key : data integrity check

Scanned with CamScanner

No need for server to store info in database since pass every info in payload from client (No need database lookup)

Payload

↳ user can keep any info

Common info (Basic / default info)

- → issuer (which server issued JWT)
- → subject (authentication, authorization etc)
- → audience (web server address) : receiver address
- → expiration time (validity duration)
- → Not before (validity start date)
- → issuedat (issued date) : JWT generate
- → JWT id

unencrypted

Additional info
- → username
- → role - admin

Change username → cannot understand thru public key

Need private key to verify data change (only client & server know)

Signature

Base64URL(Header) = x

Base64URL(Payload) = y

encoded version

- Base64URL Encoding (Header) → abcdef
  - x string
  - ASCII: 97 98 99 → binary binary binary
  - 0011 01 0011 . . . . . → 01 string (long)
  - 6 digit → integer → ASCII → characters

RSA - private key

HMAC - common key

secret key

$x + '.' + y$ → concat string (z)

→ Algorithm + secret → encrypt → Signature

x . y → z → secret algo → z' Signature

Scanned with CamScanner

JWT

Header
Payload
Signature

- Advantages →

  1. Can be generated anywhere
  2. Authorization
  3. Eavesdropping ✗ more secure ✓

- Disadvantages →

  1. Takes more space / increased bandwidth per request → JWT > session ID (token based / session based)

Private key generated thru handshaking protocol between client & server.
↳ won't be shared in common line / shared

Secret key generation (shared private key) — encrypt header & payload

Tasnim	Sakib
2	2
+	+
secret → 3	4
=	=
5	6

(shared line — HTTP Protocol)

6	5
secret → + 3	+ 4
= 9	= 9

private key

Scanned with CamScanner

- why eavesdropper cannot attack JWT?

issuer
audience
expiration time } eavesdropper change

server knows private key
Decrypt — [Header
Payload } separated

Signature won't match → eavesdropper

[session based — cookie
token based — local storage]

Advantage

- NO DB lookup
- saving on DB space
- More secure
- Faster, flexible
- Automatically prevents CSRF

Mostly used token based

Disadvantage

- Non revocable (List of Revoked JWTs)
- Bottlenecked against one secret key (Private key compromised)
- Data visible to anyone (Payload data)

Scanned with CamScanner

(5P)

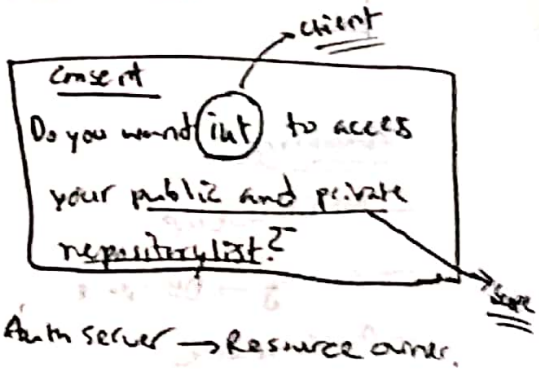
- Open ~~Authentication~~ Authorization 2.0 (OAuth)

- Authentication vs authorization
  - Authentication → verify user
  - Authorization → access to resources

- Client → iut.com
- Resource owner → Bakib (repo owner)
- Authorization server → GitAuth
- Resource server → GitResource

Authorization — from Google POV

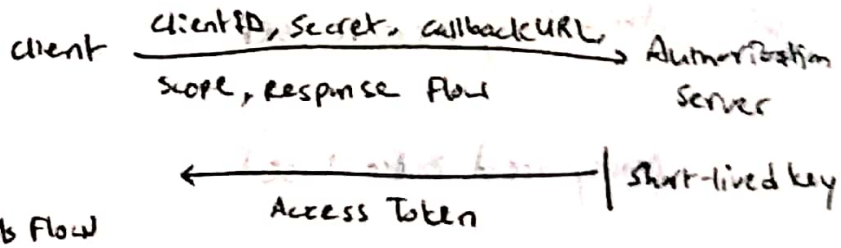
- consent → Client, Scope
- Resource/Callback URL → iut.com
- Scope → public and private repository list

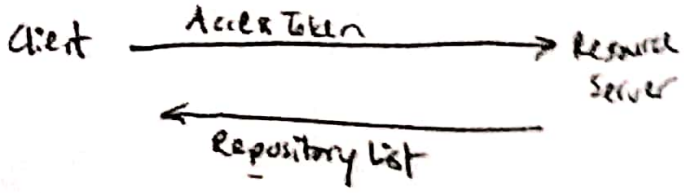


Auth server → Resource owner.

Response Flow

1. Authorization Flow
2. Implicit Flow
3. Resource owner Password Credentials Flow
4. Client Credentials Flow

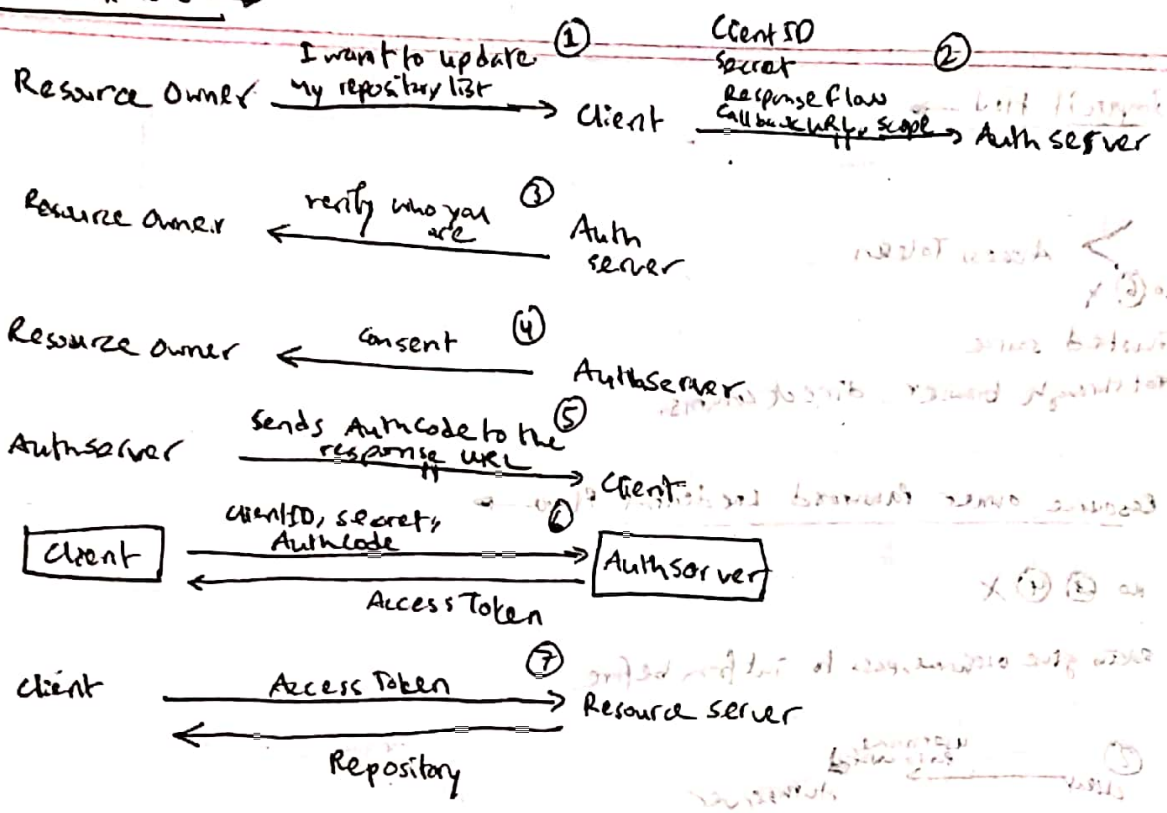




Scanned with CamScanner

Google OAuth API

Authorization Flow →



Phishing Attack : Dumb website made to look real

①② : Client → Resource owner → Authserver
(Direct – no comms between client & authserver)

Access Token vs Auth Code

Scanned with CamScanner

Implicit Flow →

⑤ > Access Token

No ⑥ X

Trusted source

Not through browser, direct comms.

Resource owner Password Credential Flow →

no ③ ④ X

side to give username, pass to int from before

② client ..., username, pass word → Authserver

Client Credentials Flow →

no resource owner

github public repo — public info

No need to login.

no ③ ④ X

no username password X

scope → public repository ✓
private X

OAuth

1.0 vs 2.0

X No backward compatibility → 1.0 stuff won't run on 2.0

X Terminology Difference → user (1.0) — resource owner (2.0)

→ No encryption or signature checking in 2.0

→ SSL Certificate (HTTPS) is a must in 2.0

Scanned with CamScanner

Push notification

- can be triggered locally by an open app
  ↓
  some web app running on browser

- can be pushed from server to client while app is not running
  (Push notifications on the web)

Depends on 2 APIs
- Notification API : responsible for displaying notif on the client side
- Push API : allows a service worker to handle push notifications
  - client side (process)
  - low energy consumption
  - listens for any notif from server

Basic terminologies

1. Notification – A message displayed as a popup to the user
2. Push message – A message sent from server to client

[Push msg & Push notif are not same]

3. Push notification – A notification in response to a push message
4. Push service – Responsible for routing push notif from server to client

Scanned with CamScanner

~~Types of N~~

Notification Types
- Simple Text Display — Browser: Hello!
- Text w/ Image and clicking it will redirect to respective site
- vibration, sound

Server —message→ Push service → Client → Service Worker

Push service → every browser implements their own.

Notif API: display, Interaction

Push API: sleep mode (actively listen)

- Notification API

1. Invocation API | controls how notif is displayed
2. Interaction API | controls how user interacts with it.

Steps
- permissions (check permission in service worker)
- Notif display
- sleep

In a secure channel (https)

server → push service → client

1 browser – 1 Push service
(server that handles
Push notif)

Array – 1000 | product add

~~Client grants permission~~

server requests for permission

↓

client grants permission

↓

Push service sends an object to a client
- end point (unique)
- public key

↓

sends encrypted object
to the server

endpoint: googleAPIs.com/ 64bit UUID – app + client

encrypted by public key

Scanned with CamScanner