Lecture 01 Week 1 Class 1 9 April, 2021

Web Application

- A web application is a piece of software, which can be accessed from a browser.
- Multiple users, same version of an app.
  (Android/IOS Application e ekekjoner version different thakte pare, ekekjon same application er ekekta instance dekhe.)
- Doesn't need to be installed.
  (pura jinish tai thake web server e)
- For a web app to operate, it needs-
  - Web Server
  - Database

Web Server -> serves web content

1. Listens
2. On a port
3. For a request
4. via some Transport Protocol -> HTTP, HTTPS
5. Response (jar kach theke request paise, take ekta response pathay)
6. Resource -> html, json, pdf, image etc
   (shei response ta kichu resource carry kore)

- Web server wait korte thake kokhon ekta request ashbe, request ashle shei request e ja ja chawa hoise shegula tar storage theke pathay dey.

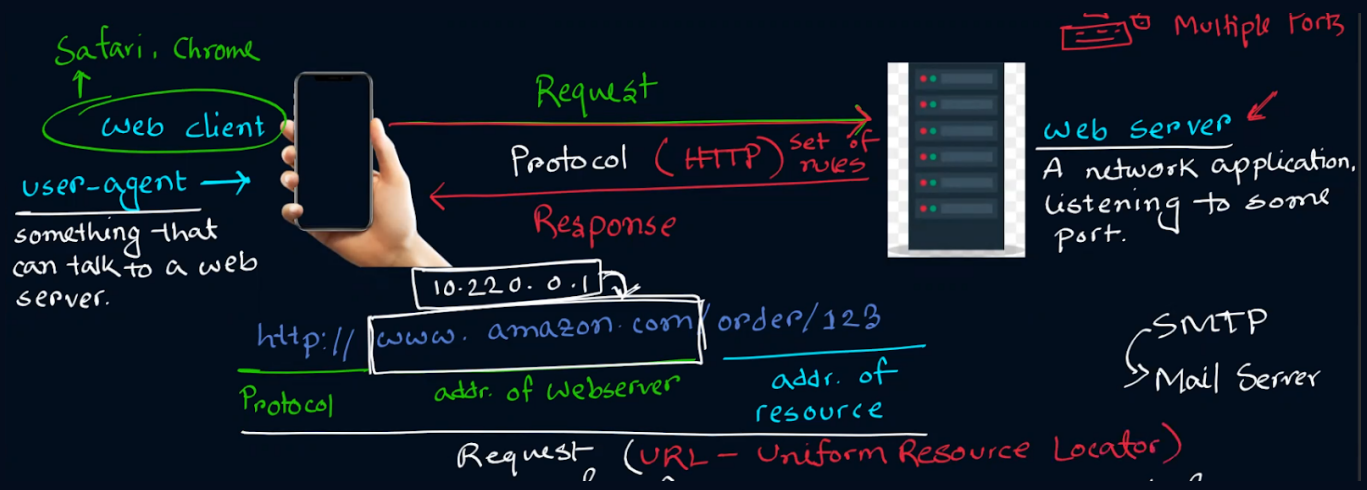
- Any Physical Device with Network Connection can act as a Web Server.
- A web server can host multiple web applications.

Web Client

- Something that can talk to a web server.
- Web server e request ashe web client theke.
- Safari, Chrome,
  Application like Daraz jekhane kono product search diye Daraz er web server e request pathate partesi

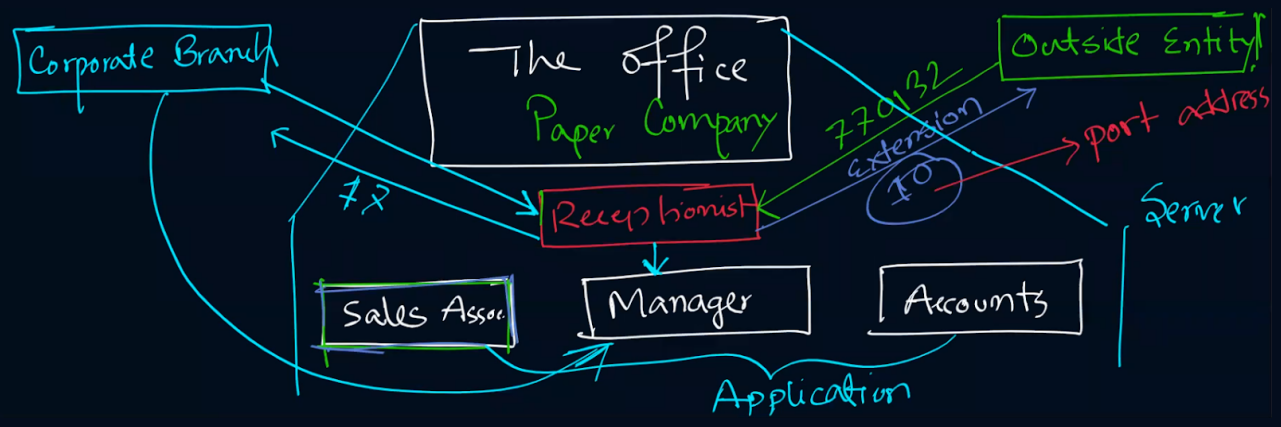
Protocol

- Client & Server er moddhe Set of agreed upon Rules.
- HTTP/HTTPS
  - Application Layer Protocol, bcz Safari/Chrome ei browser der ke HTTP baade onno kono protocol diye request pathaile ora bujhbe na



URL **(Uniform Resource Locator)**

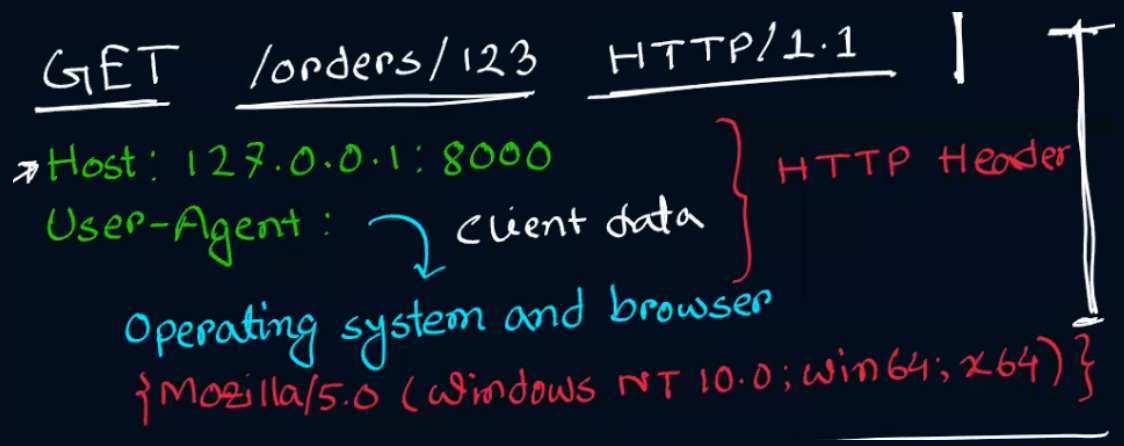
- Whole internet er jekono resource ke uniquely identify korte pare.
- Ekta web server er under e onekgula site, for example, facebook, youtube, amazon thakte pare. Ami kar kache request pathabo sheta define kore dibe Port Address.
- ***How does a web server host multiple web applications?***



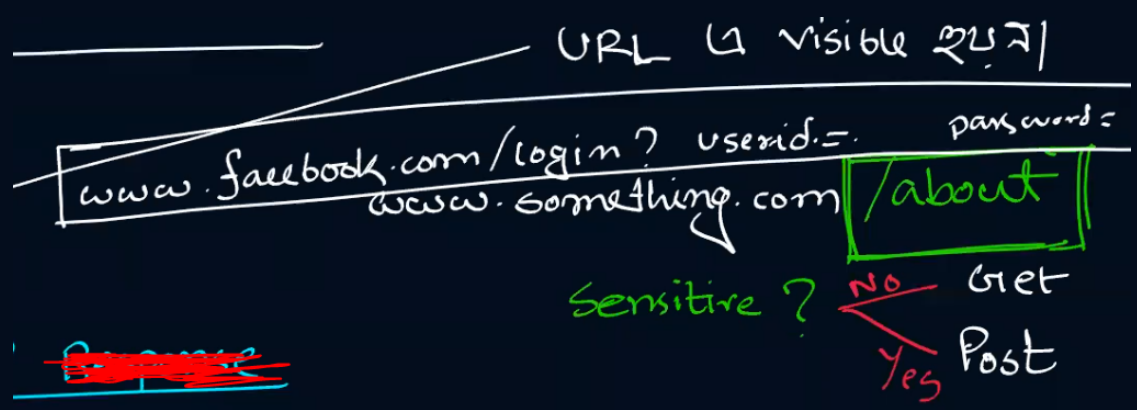
- Default Port
  - HTTP -> 80
  - HTTPS -> 443

HTTP Requests

- GET



- POST



- DELETE

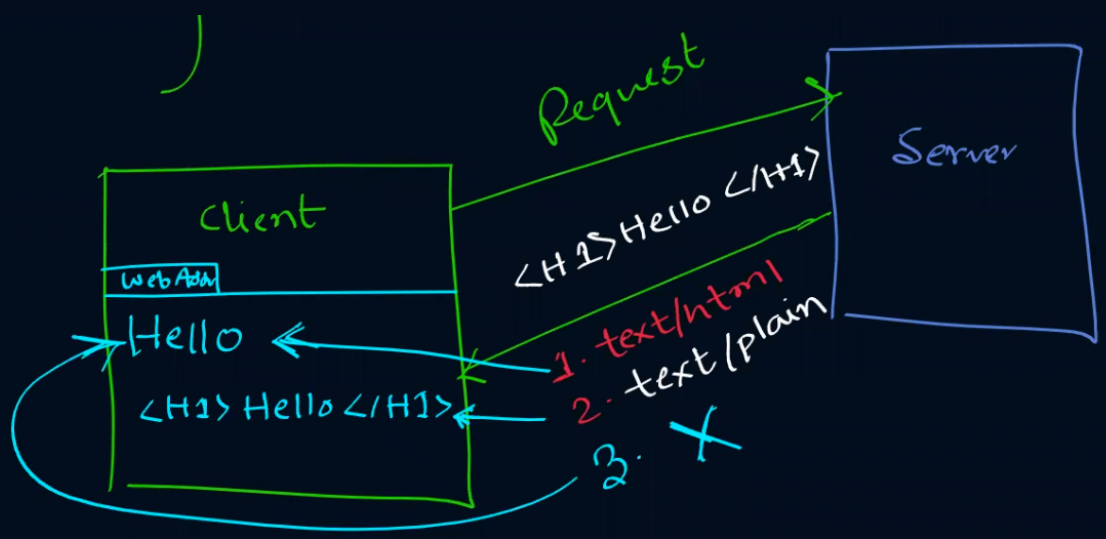
HTTP Response

2XX, 201 } Successful Response.

4XX } Client Error

5XX | 504 } Server Error

➔ HTML vs Plain Text

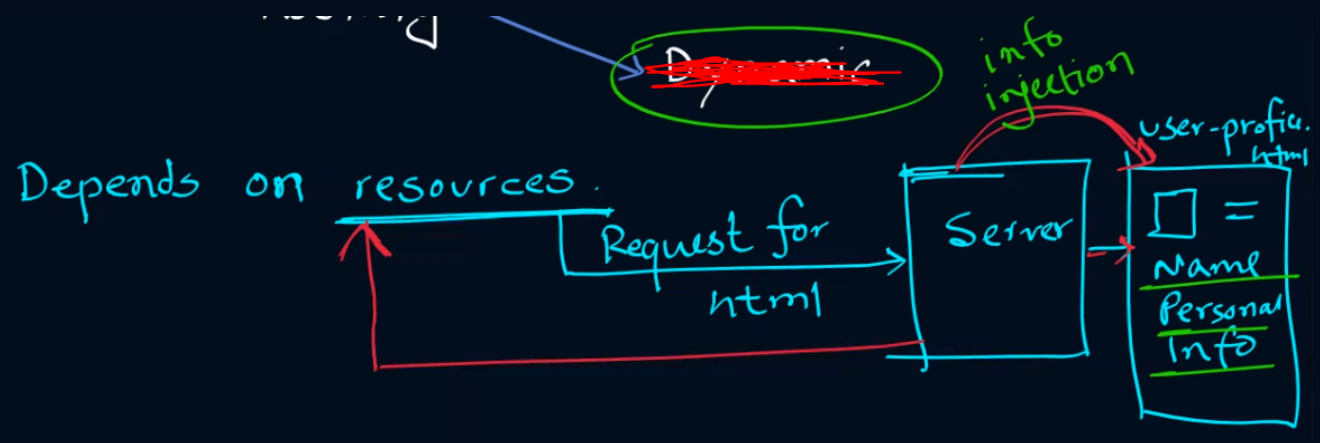


➔ Response write kore end na korleo problem nai. Kintu browser er upore loading circle ta ghurtei thakbe. Bcz end na korle connection close hobe na, client mone korbe response ekhono ashtese.

```
response.write();
response.end();
```

Static vs Dynamic Web Applications

- Depends on resources
- Dynamic
  - Ekta request ashar por resource ta server theke as it is na pathay diye jodi server ke shei resource ta change (data inject/manipulate) kora lage

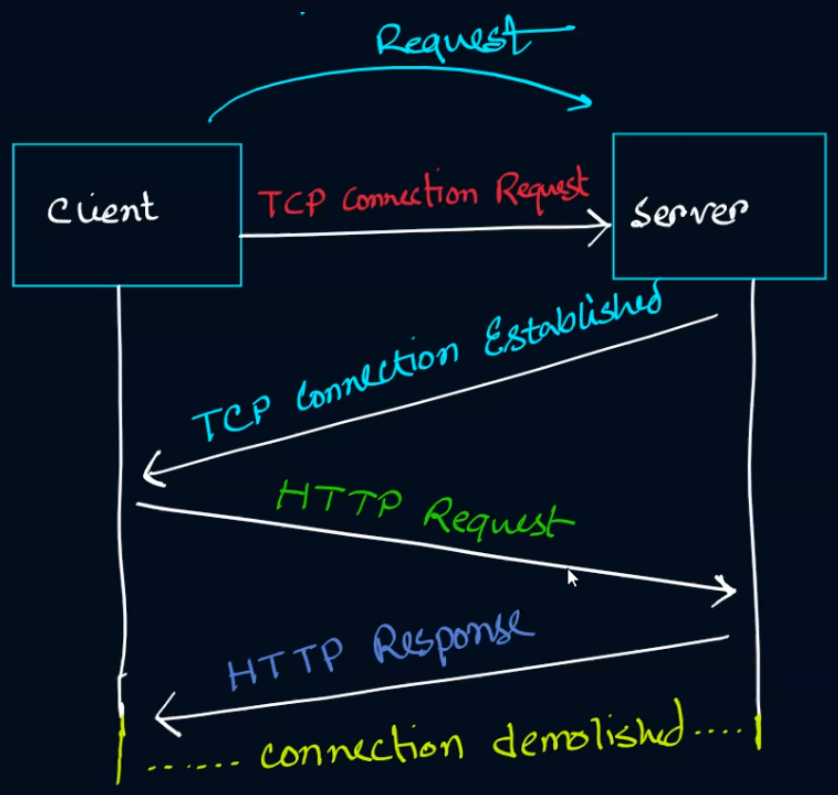


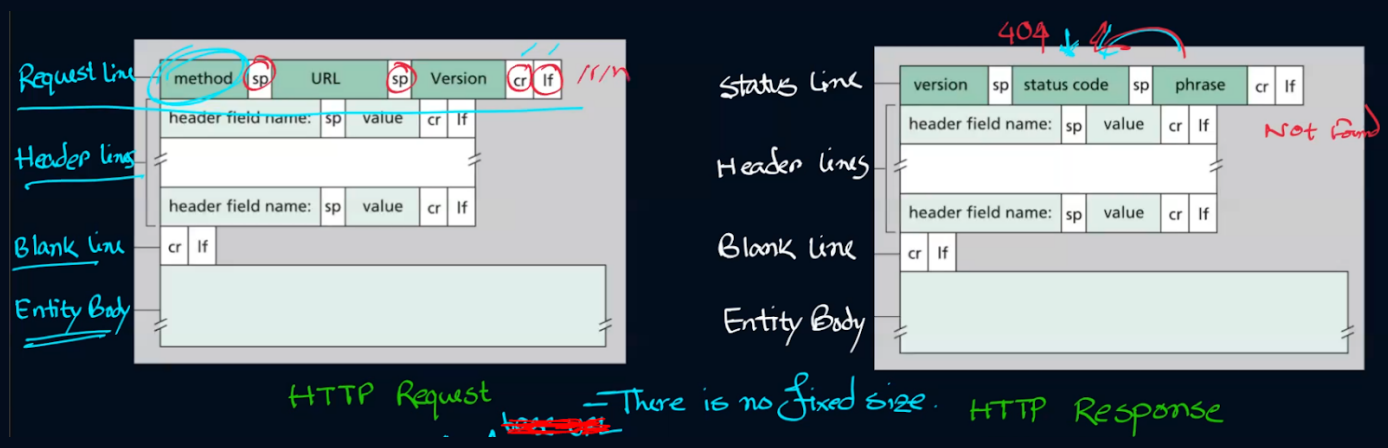
- Static
  - Shob resource jodi static hoy

HTTP

- Connectionless Protocol
  - Client theke server e request pathacchi, server response back kortese. Client theke server e ei request ta jay ekta TCP/IP connection diye. Request jawar somoy connection ta established hoy, ar response chole ashle closed hoye jay. That means server ta client ke chintese shudhu request/response er time tuku. Er baire server client keu kauke chine na.
- Stateless Protocol
  - Ekta client er kach theke joto request ee pai, prottek ta request pawar por oi client ke ekta notun client hishebe handle korbe.
- Application Layer Protocol
  - Jeno browser bujhte pare
- TCP/IP based
- Media independent
  - Response e jekono type media pathano jay. 'Content type' field e just media type ta boshay dite hobe.
- Message Format
  1. Request
  2. Response

➔ *How does connection establish between client & server?*





Cr -> Carriage return; lf -> Line feed!

```
GET / HTTP/1.1
Host: developer.mozilla.org
Accept-Language: fr
```

base URL

connection: keep-alive

blank

Accept encoding: gzip

User-agent:

HTTP Request

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
Content-Length: 55743
Connection: keep-alive
Cache-Control: s-maxage=300, public, max-age=0
Content-Language: en-US
Date: Thu, 06 Dec 2018 17:37:18 GMT
ETag: "2e77ad1dc6ab0b53a2996dfd4653c1c3"
Server: meinheld/0.6.1
Strict-Transport-Security: max-age=63072000
X-Content-Type-Options: nosniff
X-Frame-Options: DENY
X-XSS-Protection: 1; mode=block
Vary: Accept-Encoding,Cookie
Age: 7

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>A simple webpage</title>
</head>
<body>
  <h1>Simple HTML5 webpage</h1>
  <p>Hello, world!</p>
</body>
</html>
```

closed

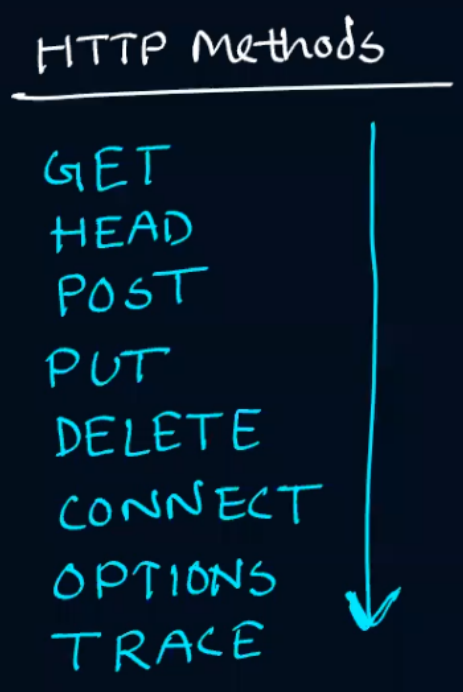
0

Blank line

HTTP Response

- By default, HTTP Request & Response e "Connection: keep-alive" thake. Response pathanor shomoy "Connection: closed" kore dite hoy.
- X-XSS-Protection
  - Cross Site Scripting Attack Protection; server ensure korbe je server theke neya response gula safe. Mane ami je HTML document ta pathacchi, shetar moddhe kono unwanted script onno keu inject kore dite parbe na.

HTTP Methods



- GET
  - Server theke document + related information duitai pabo
- HEAD
  - Same as GET; Syntax same
  - Document related information (status code, status line, header code etc) pabo, actual document pabo na.
- POST
  - Client theke Server e data pathay
  - Ei data URL e visible hobe na; data ta jabe body er moddhe diye
  - Body te data pass hobe key-value pair akare, every key-value pair add hobe '%' diye

```
name=Joe%20User&request=Send%20me%20one%20of%20your%20catalogue
```

  - Every '=' sign indicates key shesh hoise, value start hoise
- PUT
  - Server er kono content ke client side theke data pathaye Replace korbe
  - Full Resource Replace
- DELETE
  - Given URL theke resource delete kore
- CONNECT
  - Client & Server er moddhe ekta Tunnel create kore, shei tunnel er moddhe diye baki conversation Secured/Encrypted way te pass hobe

    - SSL Certificate
        - Duita remote endpoint shudhu request/response gula bujhbe, onno kono server decypher korte parbe na
    - How does CONNECT work?
        - At first, server er kach theke SSL certificate chay. Server er jodi SSL certificate thake, tahole tunnel create hoy.
- OPTIONS
    - Jei server e request pathacchi, oi server er supported method gula response hishabe pathay
- TRACE
    - Client je request pathay, Server oitai return kore; Same request pacche kina check korar jonno
- PATCH
    - Server er kono resource er Particular Part Replace kore dey

Status Codes

1xx - Informational | The request has been received and process is continuing.
2xx - Success | The action was successfully received, understood & completed.
3xx - Redirection | Further action must be taken to complete the request.
4xx - Client Error | The request contains incorrect syntax.
5xx - Server Error | server failed to fulfill an apparently valid request.

URL (Uniform Resource Locator)

http://www.swe4537.com:80/node-for-beginners.html

URL scheme | host | port | resource path

- **Query String**

www.swe4537.com/home?theme=dark#/nodejs

Query string | fragment

➔ Parameter pathate hoy. (Question mark er porer gula parameter bujhay)

➔ Shob gula parameter key-value pair akare server e jay.

passes "key-value" pairs to the server.

key=value& key=value .....

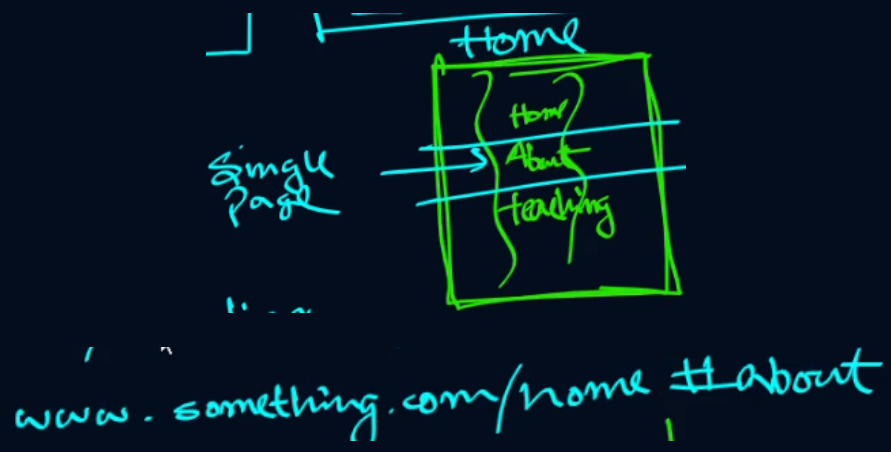
theme = dark & lang = en

key | val | k | v

➔ Each key-value pair is separated using '&'

Fragment

- Kono website load hoye start theke na dekhaye middle kono section theke dekhano shuru korlo. This is controlled by Fragment.
- Fragment e pass kora hoy kono HTML element er id.
- URL e '#' er porer part fragment.



- Fragments are not passed to the web server. It is handled by the browser.

URL Encoding

[space] → %20 or +
↳ Hexa-decimal value of [space]

& → %26
+ → %2B

%20
Sadman+Sadat.

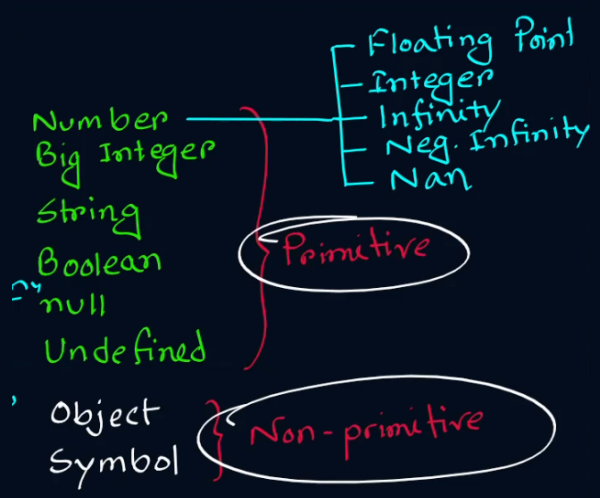
Introduction To Javascript

Coding with Javascript

- Comments -> Same as C/C++
- Variables -> let
- Constants -> const

➔ Semicolon is not mandatory at the end of the line. But it is good to use.

- Naming Convention
  - Universal constant should be all caps.
    - const WHITE = "#FFFFFF"
  - User defined constants or constants received from any API should be normal.
    - const name = "..."
- Data Types
  - Primitive
    - Ekta variable always ekta value carry kore.
  - Non Primitive
    - Ekta variable multiple data carry kore.



JSON : JavaScript Object Notation

- ➔ String is non primitive in Java, but primitive in Javascript.
- ➔ Javascript has a data type named 'Infinity'
- ➔ Some examples-
  - 1/0 Infinity
  - -1/0 -Infinity
  - 1/"abc" NaN
  - 1*"abc" Nan
  - 1/"1" 1 (String ke int e convert kore ney)
  - 1/" 1 " 1
  - 1/" 1 2" NaN
  - 1/"1 2 3" NaN (Number er majhe space thakle convert korte pare na)
  - typeof(1n) 'bigint' (jekono number er pore 'n' add korle big integer e convert hobe)

- ➔ In Addition, if one operand is string, the whole thing will be converted to string.

- 1 + '5' -> 15
- '1' + 5 -> 15
- 1 + 2 + '3' -> 33

➔ But in Division,

- (2+2)/'2' -> 4/'2' **-> 2**

➔ In boolean, true = 1, false = 0

- 1 + true -> 2
- 1 + false -> 1

➔ Variable declare korlam, kintu value dilam na -> undefined

- let a;

➔ For null,

- let a = null;

➔ Suppose, let a = getPosts(); this getPost's value will come from any API call, but the API call is failed. In this case the value will be undefined, not null. Value can only be null if we define it explicitly.

- **JSON** (Javascript Object Notation)
  - Key-value pairs, separated by comma, inside curly braces.

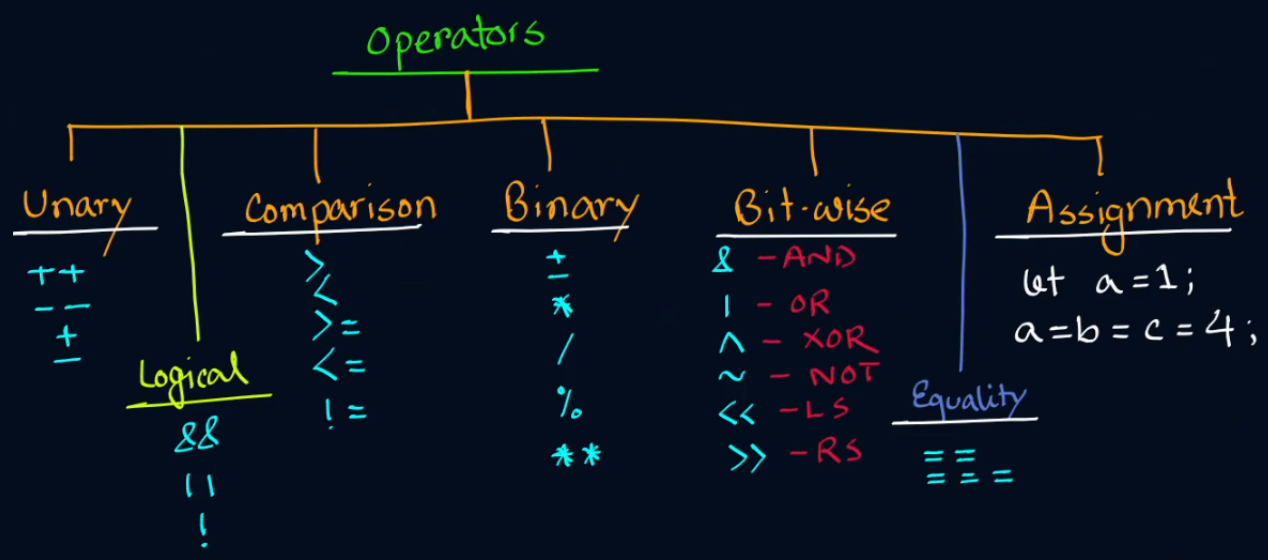
{ key : value ,

}

➔ Key must be string, but the value can be anything.

➔ Ekta JSON er key er value arekta JSON object hoite pare.

Operators



- '+'-> type casting
- a = +'1' -> a er value int e convert korbe
- 2**3 = 8 -> 2^3

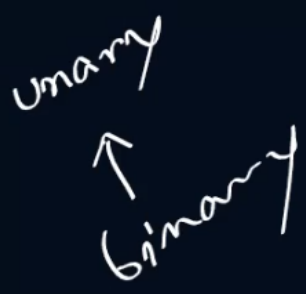
➔ In C, 2 == '2' -> false

But in JS, 2 == '2' -> true

➔ '==' checks only value,

'===' checks value and type.

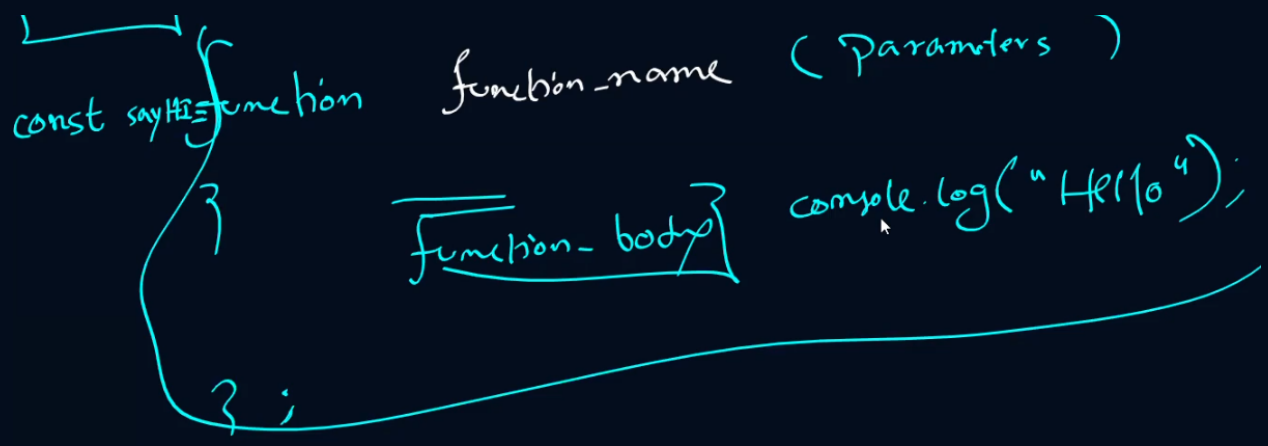
- **Precedence**



Function

➔ In JS, a variable can hold a function.

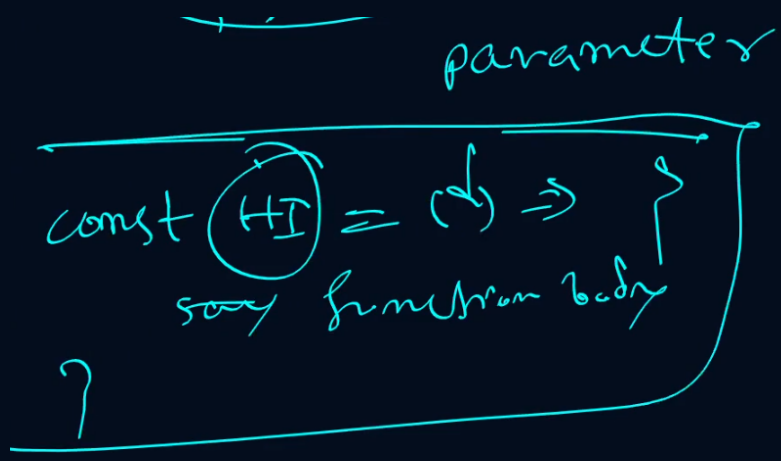
➔



console.log(sayHI) -> will print the whole function body; that means shudhu 'sayHI' variable er value ta print hobe, ar value ta hocche ekta function.

console.log(sayHI()) -> will print Hello; '()' deyar por function ta call hobe

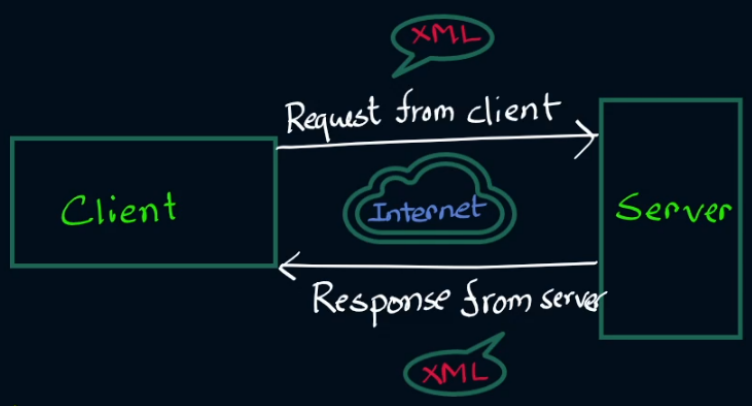
- ***Syntax II***



Lecture 04 Week 2 Class 2 + Week 3 Class 1 03 May, 2021

Web Services

- A standardized medium to propagate communication between client and server applications.
- A software module to perform certain sets of tasks.
- Client linux, server windows, communicate korabe ke? Web service.
- Client java, server javascript, communicate korabe ke? Web service. Bolen thik kina :3



- Udvash admission registration koray deyar jonno jemon temon vabe korbe na, systematic way te ekta form (XML) fillup korte hobe age!

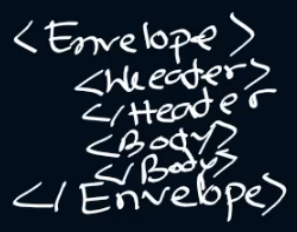
Remote Procedure Calls (RPC)

- Ekta web service er onekgula method thakte pare. Ekek ta method ekek ta functionality provide korbe. Ami client hoye RPC er maddhome ei method gula call korbo.
- Method gula host kore web service. Call korte parbo client side theke.

- Udvash admission registration o koray dey, student admit o kore. Ekektar jonno ekekta form fillup korte hoy.

Simple Object Access Protocol (SOAP)

- Duita application er majhe je data adan prodan hoy, etakei bole SOAP. Message gula normal XML e geleo eder ekta particular protocol follow kora lagbe, ejonno ederke bole SOAP.
- Udvash hocche amar web service, Udvash theke jokhon kono service nite chabo tokhon call dibo RPC ke. Web service er kono ekta method call hobe and amake data ta pathabe. Amar ar udvash er moddhe ei data passing / communication hocche SOAP message er maddhome. Ar ei message gula jacche HTTP protocol diye. So, web service message passing er jonno use kore SOAP over HTTP (/soap/http).
- **SOAP Message**
  - Transport independent messaging protocol
  - Doesn't restrict content
  - But document has to follow a specific pattern
    - Ei SOAP message er ekta root element thaka lagbe, named Envelope



    - Content-type: text/xml

Web Services Description Language (WSDL)

- ➔ Tells us je Web Service ki ki kaj kore
- ➔ Also a XML document, but formatting is not same as SOAP
- ➔ This XML contains 3 parts -
  - ◆ **message**
    - Contains data elements for each operation
  - ◆ **portType**
    - This is the operation to perform
  - ◆ **binding**
    - Protocol

```
<definitions>
   <message name="TutorialRequest">
      <part name="TutorialID" type="xsd:string"/>
   </message>

   <message name="TutorialResponse">
      <part name="TutorialName" type="xsd:string"/>
   </message>

   <portType name="Tutorial_PortType">
      <operation name="Tutorial">
         <input message="tns:TutorialRequest"/>
         <output message="tns:TutorialResponse"/>
      </operation>
   </portType>

   <binding name="Tutorial_Binding" type="tns:Tutorial_PortType">
      <soap:binding style="rpc"
         transport="http://schemas.xmlsoap.org/soap/http"/>
      <operation name="Tutorial">
         <soap:operation soapAction="Tutorial"/>
         <input>
            <soap:body
               encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
               namespace="urn:examples:Tutorialservice"
               use="encoded"/>
         </input>

               <output>
            <soap:body
               encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
               namespace="urn:examples:Tutorialservice"
               use="encoded"/>
         </output>
      </operation>
   </binding>
</definitions>
```

input
output

Universal Description Discovery and Integration (UDDI)

- ➔ Tells the clients je amar web service ki ki service provide kortese
- ➔ Shob client er kase UDDI access thaka
- ➔ UDDI is basically a database which contains all WSDL files

Advantages of Web Services

- ➔ Exposing Business Functionality on the network (ami ekta service provide kortesi, eta internet er maddhome shob potential client der janate partesi)
- ➔ Interoperability amongst applications (Shobai shobar theke service nite partese. Ke JS use korlo, ke PHP use korlo, doesn't matter)
- ➔ Standardized Protocol which everybody understands
- ➔ Reduction in cost of communication

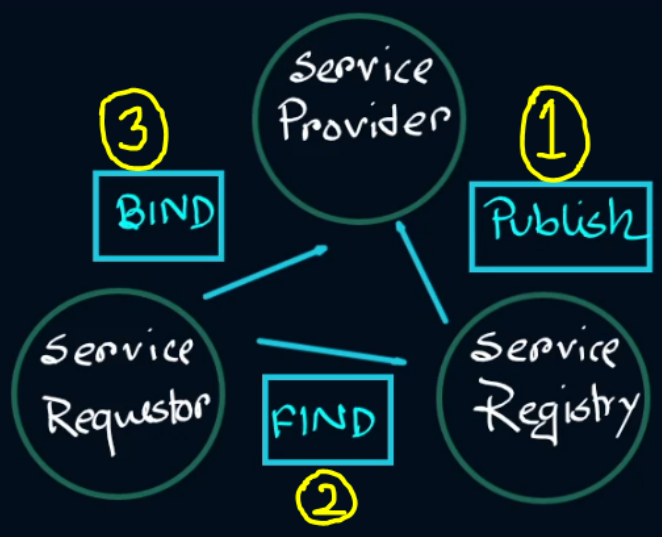
Protocol Stack

- ➔ Web service ke 4 ta layer e divide korte pari. Each layer ke boltesi protocol. And overall jinishta hocche web service er Protocol Stack.

- **Service Transport**
  - responsible for transporting messages
  - Uses HTTP, SMTP, FTP, BEEP protocols
- **XML Messaging**
  - Encodes message in a common XML format

- **Service Description**
  - For describing the public interface (WSDL)
  - Amar ei ei method ase, ei method diye tumi ei ei kaj korte parba. Ami ei method diye kivabe tomake service dicchi sheta tomar jana dorkar na. Tumi shudhu jano tumi ei method e ki input dile ki output paba. Bujho nai bepar ta?
- **Service Discovery**
  - For centralizing services into a common registry and providing easy publish/find functionality (UDDI)
  - Website er joto description ase, shegula ekta centralize jaygay store kore rakhbe, jeno jekono potential client dekhte pare amar required service keu provide kortese kina

Web Services Architecture



- **Service Provider**
  - Creates web service and makes it available to clients.
  - Ekhane service provider hocche Udvash
- **Service Requestor**
  - Client
- **Service Registry/Broker**
  - Application which provides access to UDDI

➔ Service provider first e create kora web service ta publish korbe Service Registry er kase. Service Registry sheta UDDI te add kore dibe. Erpor client er je service dorkar, client sheta Service Registry theke UDDI diye find korbe je tar required service ta kono web service provide kortese kina. Jodi kono web service provide kore, tokhon client oi service provider ke message diye tar sathe bind hoye jabe.

Characteristics of Web Services

➔ XML based

- ◆ uses XML to represent data
- ◆ Eliminates dependency (OS, network, platform dependencies), as it is completely XML based. Bcz XML is one of the most common languages which is understood by all other programming languages.

➔ Loosely coupled

- Web service er jekono ekta entity change hoileo doesn't mean BIND method change hobe. Client er request same thakbe, age jemne communicate korto, ekhono omnei communicate korbe.
- Age Udvash normal bus diye admission candidate niye jaito. Ekhon AC bus e shift hoilo. Kintu ekhono shei candidates der request same ee ase - "aMakE eKtU f0r!kKa d!tE n!yE jAAn bA!yA"

➔ Synchronous and Asynchronous Functionality

- Synchronous
  - Client web service er kache data request korar por parallelly onno kaj korte parbe na.
- Asynchronous
  - Client ekta data request kore nijer onno kaj korte thaklo, in the meantime, web service data ta ready kore client ke diye dibe.

➔ Ability to support RPC

- Ekta web service onekgula method provide kore. Er moddhe jekono ekta method directly RPC call diye call korte partesi. This is a big advantage.

➔ Supports document exchange

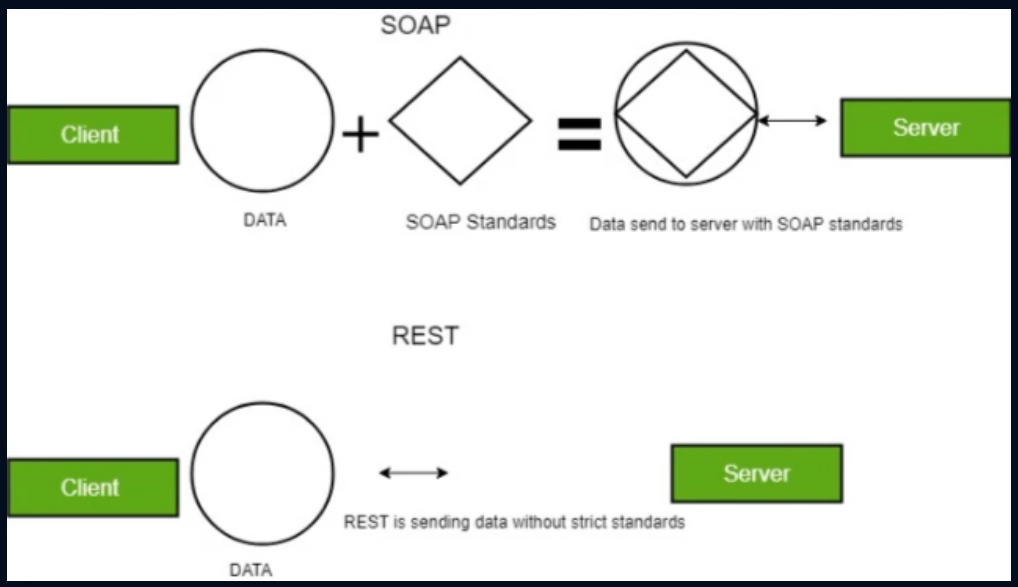
- Simple (e.g. String) theke shuru kore Complex (e.g. Book) data o XML diye access korte parbo

RESTful Web Services

- ➔ REST -> Representational State Transfer
- ➔ RESTful web services are focused on resources, how to access resources, how to manipulate resources etc.
- ➔ It is such an approach that is centred around resources, and things you can do with those resources.
- ➔ There is no RPC or SOAP. Here, we send requests using HTTP protocol. Ei HTTP protocol er kichu action verbs ase such as GET, POST, PUT, DELETE - egula use kore resource gular upor operation chalaite parbo. (But SOAP based web service jei function gula provide korto shegula store kora thakto WSDL e, ami shegula ekta SOAP message/RPC diye invoke kortam. Ei pura jinish ta (request/response) XML format e pathano lagto.)
- ➔ An architecture based web service
- ➔ URL diye resource er upor request pathai. Shudhu ekta URL ee request mean kore ekta RESTful web service er khetre.

- **Service Exposure**
  - RESTful web services use URL to access the components on the hardware devices. Alada kono XML document create kore pathano lagtese na.
- **Bandwidth**
  - SOAP e seperate document lagto request/response er jonno. Ekhane lagtese na, so bandwidth onek kom hobe. It's like 'parar dokandar' jar kono structure nai, je jevabei hok amar registration kore dito, udvash er moto gejaito na :3

- **Data Format**
  - SOAP supported only XML
  - But RESTful supports any data format including XML, JSON plain text etc. But JSON is preferred.



Rules of REST

- ➔ Noun endpoint / Resource (/users, /users/1 etc)
- ➔ Request using HTTP Action Verb (GET, POST etc)
- ➔ Plural of resources (/users, /users/1, not user/1)
- ➔ HTTP Response Code (jeno success/failure client bujhte pare)
- ➔ RESTful web service ta khubi organized way te howa lagbe (e.g. api/users/1 DELETE dekhe bujhtesi user with id 1 ke delete kora lagbe)

URI	HTTP VERB	DESCRIPTION
api/users	GET	Get all users
api/users	POST	Add a user
api/users/1	PUT	Update a user with id = 1
api/users/1	DELETE	Delete a user with id = 1
api/users/1	GET	Get a user with id = 1

Differences between SOAP & REST

- ➔ SOAP is Simple Object Access Protocol.
  REST is Representational State Transfer.
- ➔ SOAP follows strict guideline/pattern. Bcz it's a protocol.
  REST doesn't follow, bcz it's simple, not a protocol.
- ➔ SOAP uses XML.
  REST uses XML, JSON and anything.
- ➔ SOAP web service diye kono service call korte chaile RPC call korte hobe.
  RESTful web service diye kono service nite chaile just URL is enough.
- ➔ SOAP is difficult to implement.
- ➔ SOAP requires more bandwidth.
- ➔ It is easier to perform transaction using SOAP.
- ➔ For security,
  SOAP uses Secure Socket Layer (SSL) and WS - Security.
  REST uses HTTPS and SSL.
- ➔ Bank account, password, card no. -> SOAP is preferred

API vs Web Services

API (Application Programming Interface)

- All Web Services are APIs. But not all APIs are Web Services.
- API is a software intermediary that allows two applications to talk to each other.
- Not necessarily, over the network howa lagbe. Computer er duita software er moddheo communication hote pare. But web Service er khetre obosshoi over the network communication hote hobe.
- Web Services provide a standard means of operating between different software applications over a network (internet/organizational net) running on a variety of platforms and/or frameworks.
- For example, phone er shob ocr related app phone er inbuilt ocr theke help ney. Internet theke kisu lage na.

Differences

- Need of Network
  - API - Not always
  - Web Services - Always
- Lightweight Architecture
  - API - Yes
  - Web Services - No

- Style of Communication
  - API - Any
  - Web Services - SOAP, REST, XML - RPC
- Protocol Restriction
  - API - WebAPI -> HTTP
  - Web Services - HTTP
- Hosting
  - API - Application / Internet
  - Web Services - Internet
- Open Source
  - API - Yes (Mostly)
  - Web Services - Mostly No

Model View Controller

- An Architectural Pattern
- Ekta boro application/software ke 3 ta logical component e divide korte parbo - Model, View, Controller
- Eta use na korle application kaj korbe na emon na. Use korbo jeno application er stability, debugging, reusability baare.

View — Responsible for Visual Elements.

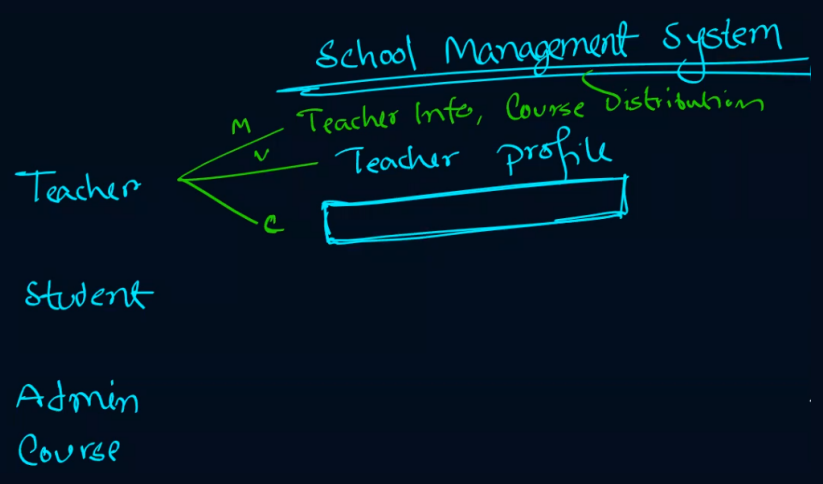
Controller

Model — Database/Storage

How to Design

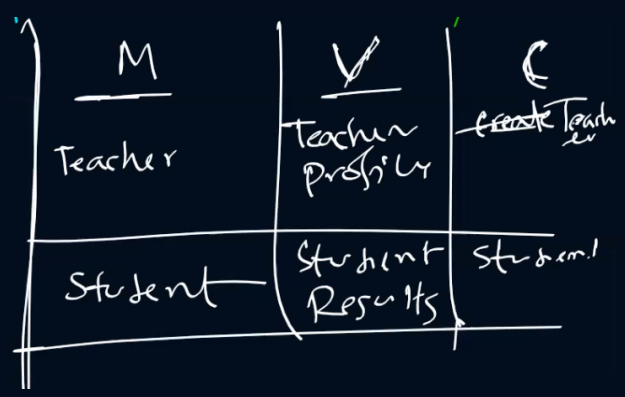
Approach I

- ➔ At first, major entity gula define kore felbo. Then prottek ta entity er jonno model, view and controller define korbo. Evabe ekta ekta kore entity agabo.

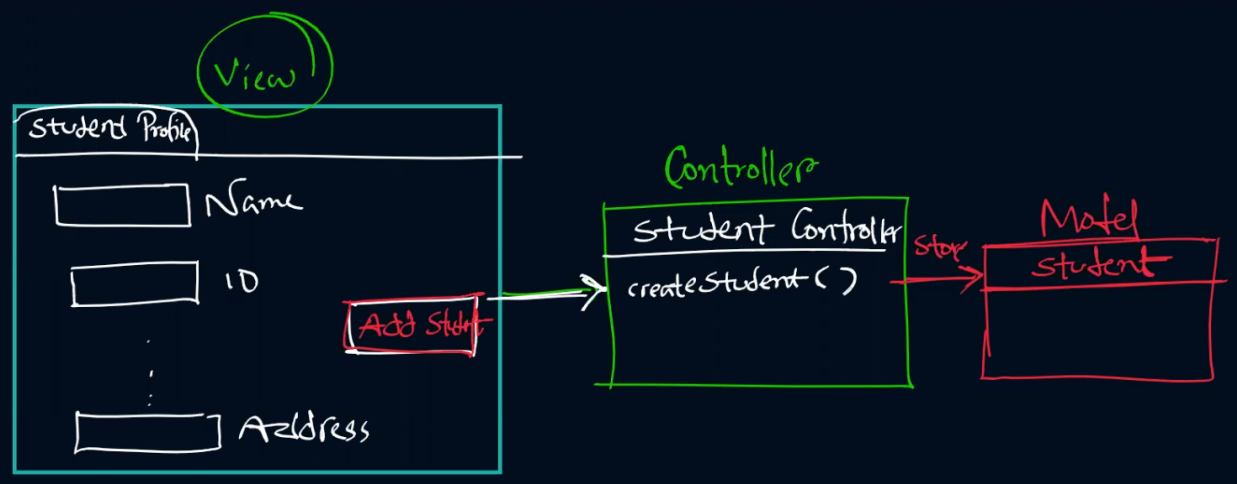


Approach II

- ➔ Full project ta ke first ei model, view, controller e divide kore felbo.



- ***How does Model View Controller communicate with each other?***



View communicate korbe Controller er sathe (actually controller er method er sathe). Pura application er shob business logic jane shudhu controller. View/model er kase kono business logic thake na. Erpor controller model (here, student data table) e view er html ta add kore dey.

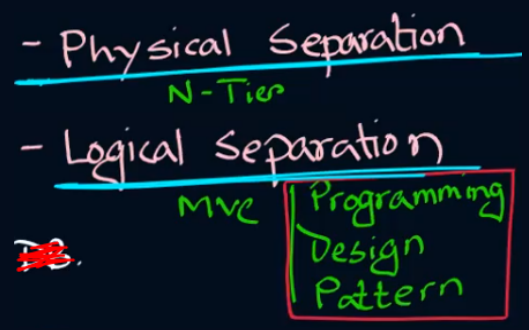
Features of MVC

- ➔ Test Driven Development, Easy debugging
- ➔ Complete control over your HTML and Database (Pura project er control controller er haate)
- ➔ Clear separation logic
- ➔ Code reusability

MVC Explained with Code

(Explained in L06 W4C2 May 20, 1:00:30 - 1:08:55)

N-Tier (Multi Tier) Software Architecture

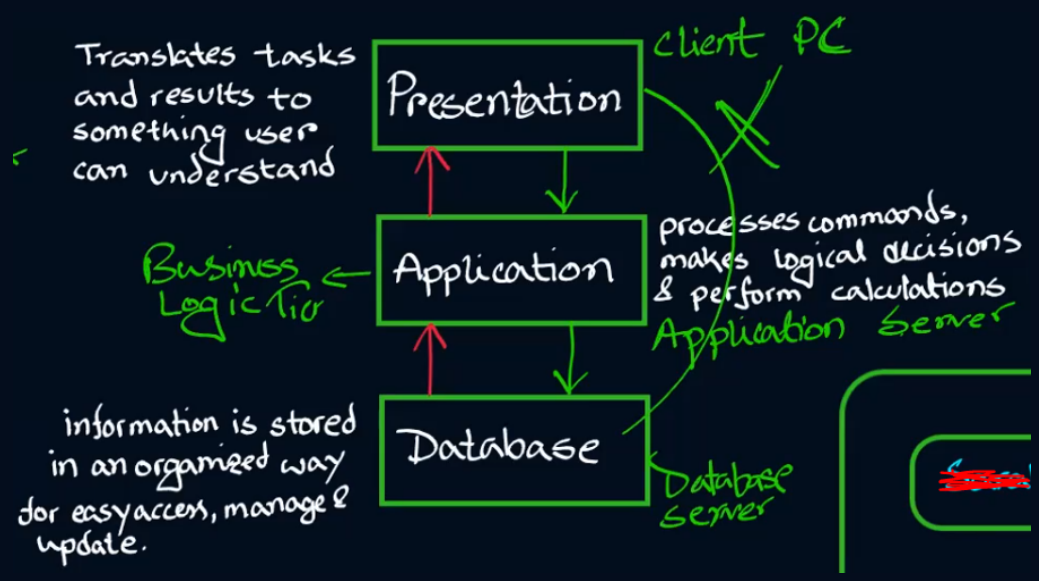


- A program distributed among N separated computers in a distributed network.
  - Distributed network is a network architecture jekhane component gula different different network computer e thake and tara nijeder moddhe communicate kore through message passing.
- 1 ta program N ta separate computers e distributed thakbe.
- ***<u>Difference between Physical Separation and Logical Separation</u>***

  MVC te ekta computer ei shob code chilo (Logical Separation). That was 1-tier. 1-tier e shob code same computer e. 2-tier e ekta computer e kichu code, arekta computer e kichu code rakhbo. 3-tier e ekta computer e kichu code, baki duita computer e baki code. (Physical Separation)

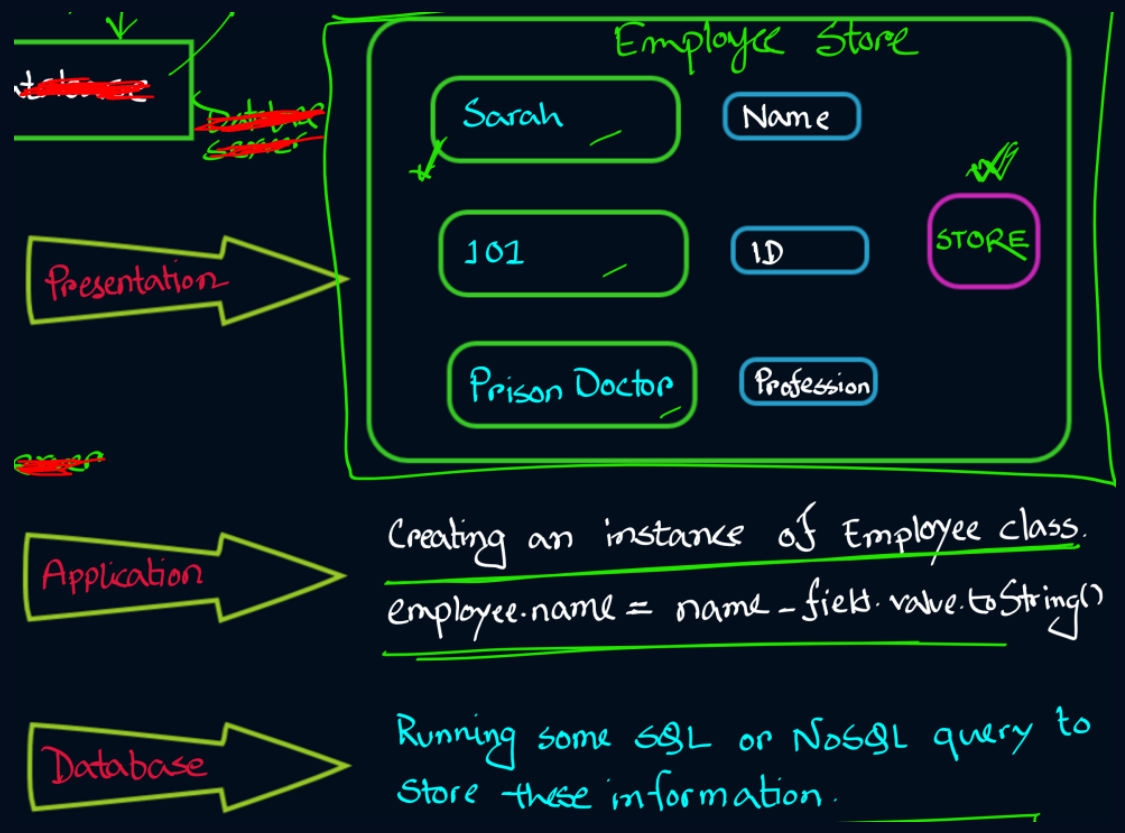
3-Tier Architecture (Most common)

- ➔ 3 ta tier thake, not layer
  - ◆ Layer gula logically separated, but tier gula physically separated.
- ➔ The 3 tiers are -



- ◆ Presentation Tier
  - Ei tier er kaj holo database theke Application tier je data gula retrieve kore, shegula understandable format e convert kore user ke show kora.
  - Presentation tier direct Database er sathe communicate korte pare na.
- ◆ Application Tier (Business Logic Tier)
  - User er kach theke data niye sheta process (logical decisions, calculations etc) kore. Also database e kono data store/retrieve korar dorkar hoile shei request tao pathay.
- ◆ Database Tier
  - Data organized way te save rakhe.

Example:-



2-Tier Architecture

➔ 2 ta tier thake -

- Client / Presentation Tier
  - Business logic ei tier e thake.
  - Client directly database access kore. (Alada kono application nai. Direct API call er maddhome data gula ane.)
- Database Server
  - Kaj hocche shudhu JSON response provide kora.

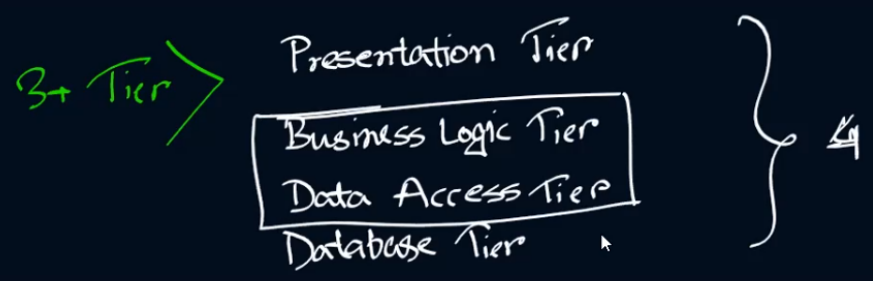
1-Tier Architecture

- ➔ Example - calculator
- ➔ Kono network activity thakte parbe na.
- ➔ Kono message passing thakbe na.

  Standalone Software hobe. Ekta device er moddhei data store shoho shob hoy.

3+ Tier Architecture

- ➔ 3-tier er joto upore jabo, 3 tier oi 3 ta tier gular jekono ta venge multiple tier banabo.
- ➔ Example -



Ekhane Application tier ke venge duita tier kora hoise.

Advantages and Disadvantages

- ➔ **1 Tier & 2 Tier**

  Advantages

  - ◆ Simple; bcz multiple device e message passing handle korte hocche na.
  - ◆ Fast for a lower number of users. Bcz client direct database er sathe communicate kortese.
  - ◆ Low cost for hardware, maintenance, deployment.

➔ **2 Tier**

Disadvantages

- Scalability
  - Client beshi hoile database er load toto beshi porbe. Scalability thakbe na.
- Security
  - Client direct database access kortese without validation. Security nai.
- Fault Tolerance
  - Suppose, amar server ta jekhane shei jaygar electricity nai. Taile to server access korte parbo na. Ekta database server crash korlei pura application down.
  - 2-tier e fault tolerance kom. Database server crash korle pura application down. Backup o rakha jay na, backup rakhte gele tokhon oita ar 2-tier taktese na.
- Data Integrity
  - Data integrity onek kom. Bcz client iccha moto database access kortese, kono proper way maintain kora jay na.
- Reusability
- Maintainability

➔ **Multi Tier**

Advantages

- Scalable
  - Ekta tier venge multiple tier banate pari. Basic Ali, jotogula physical device, totota tier.
  - ***Scalability kemne barabo?***
    - Database clustering kore scale up korte pari.
    - Client direct database access korte partese na, application server er through te access kore. Multiple client application server e access kore, and application server theke ekta connection database e jacche. So database e load kom portese. Application er scalability barlo!
    - Multiple application server, multiple database server er moddhe load ta distribute korte alada ekta separate server rakhbo jeta Load Balancer hishebe kaj krbe. Evabe application ta highly scalable hoilo.
- Fault Tolerance
  - *Ekta server ke easily cluster korte pari for load balancing and fault tolerance.*
  - Fault tolerance beshi. Ekta server crash korle onno server theke request ta niye ashlam.

- Security
  - Business logic & Database e beshi security dorkar. Security baranor jonno prottekta server er jonno alada alada firewall install korte pari. Lagle business logic e more security er jonno extra server ee boshay dilam, jeta 1-tier/2-tier e korte partam na.
- Data Integrity
  - Related to security.
- Reusability
  - Reusable; bcz code ke joto module e vaag korbo, code toto reusable hobe.
- Maintainability
  - Easily maintainable; completely different server/computer howate baki tier gula affect na korei kono tier er code modify korte pari.

<u>Disadvantages</u>

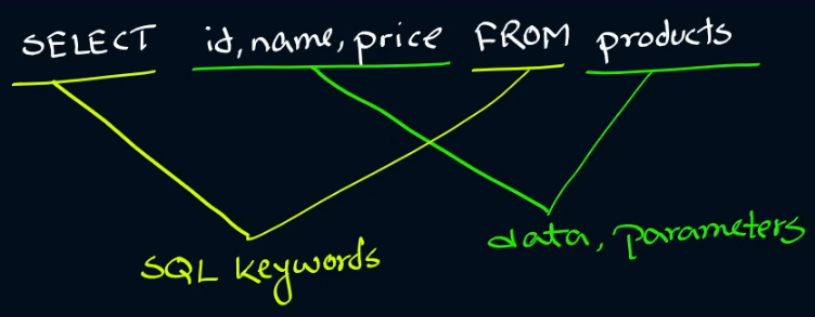
- Higher bandwidth
  - Bandwidth beshi na hoile application slow hoye jabe, bcz prochur message passing.
- Higher cost

SQL vs NoSQL

- SQL -> Relational Database
  NoSQL -> Non-relational Database
- SQL -> Schema, Relation duitai ase
  NoSQL -> Schemaless, Relationless

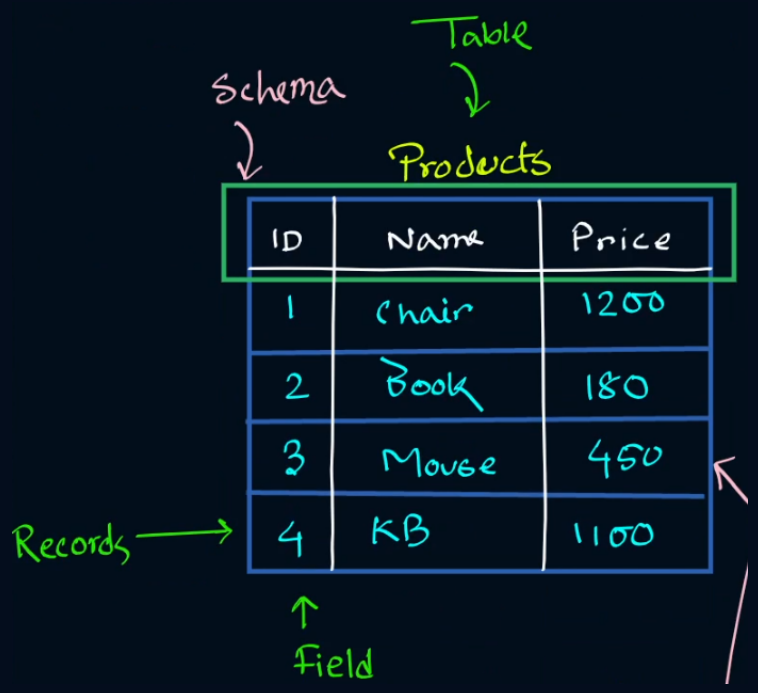
SQL

➔ Structured Query Language



Schema

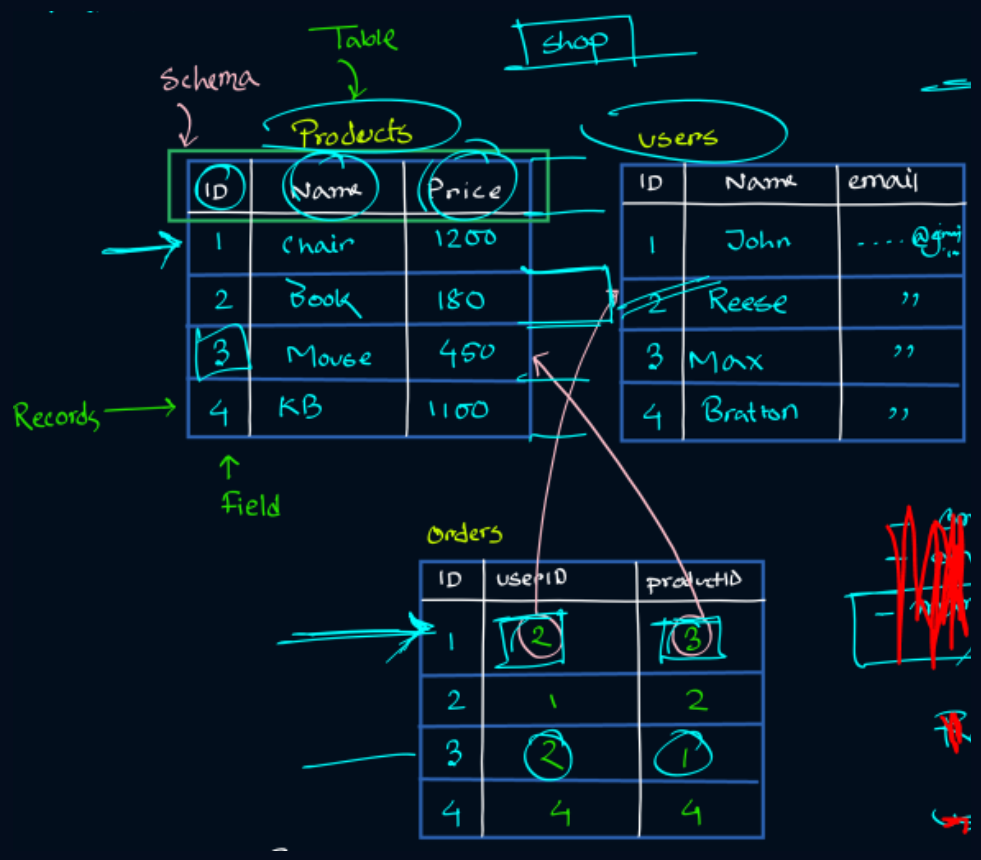
➔ defined by fields



Ekhane jekono ekta row er jonno ekta field baraite parbo na. Baraile shobgula row er jonno baraite hobe. For example, <ID, Name, Price> er jaygay <ID, Price, Name> likhte parbo na, ba shudhu ekta row er jonno <ID, Name, Price, Qty> add korte parbo na. SQL ei strict schema follow kore.

Relations

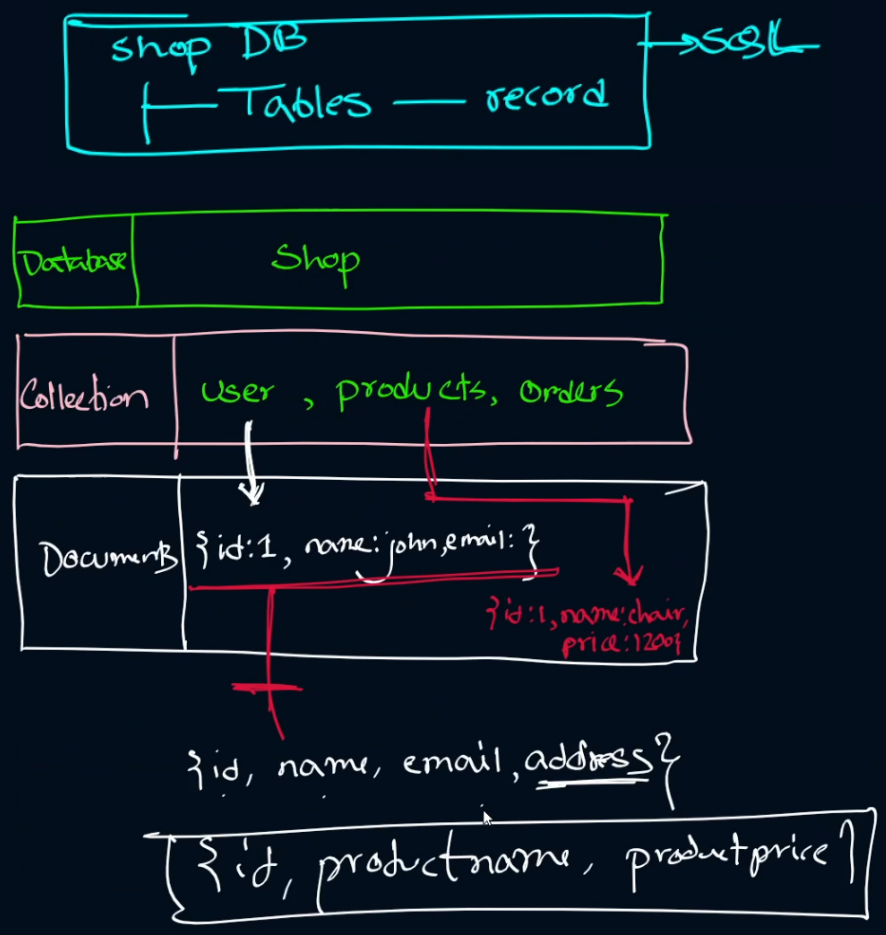
- One to one
  One to many
  Many to many
- Example



NoSQL (Non-Relational) (Schemaless)

- SQL er Table ke NoSQL er Collection er sathe compare korte pari.
- SQL e table er row/info gula ke Record boltam. NoSQL e Documents bolbo. Ar ekhane Documents gula thake JSON akare.
  - Emon kono strict rule / schema nai je prottek document e same information thaka lagbe.

Schemaless



User er moddhe iccha moto Document add kortesi. Oi document er moddhe abar document add kortesi. Strict kono schema nai.

Non-Relational

- ➔ Suppose amar ekta Orders collection ase. Ekhon foreign key matching er jonno UserID User collection e thaka lagbe, ba ProductID Product collection e thaka lagbe. Erokom kono rules nai.
- ➔ As NoSQL e kono relation nai, taile to data security nai. Jekono shomoy data inconsistent state e pore jete

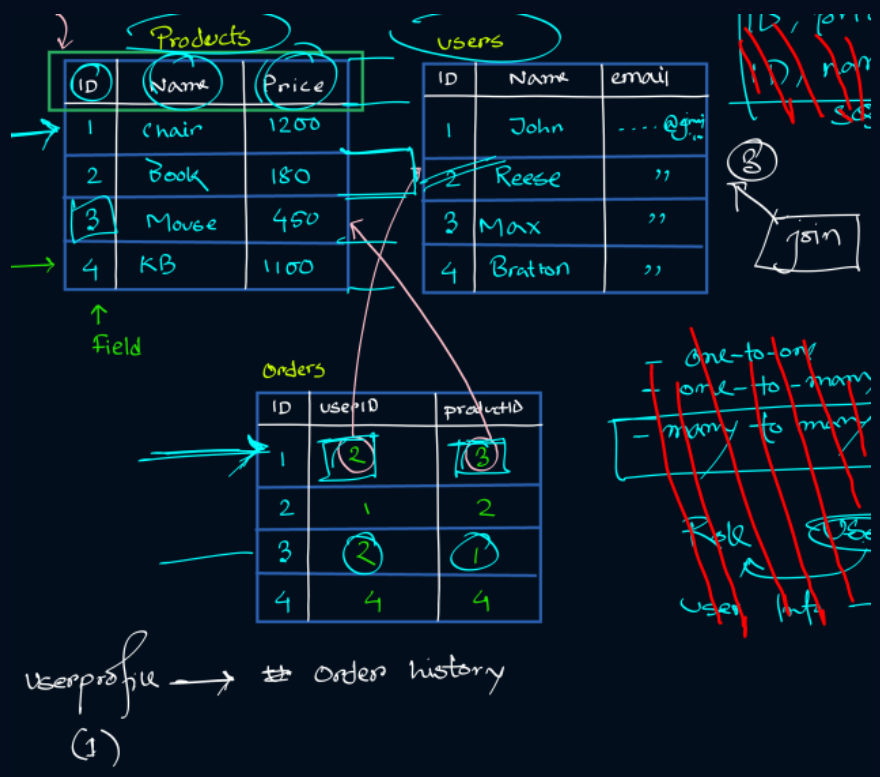
pare. SQL e eta automatic handle kore. But NoSQL e nije theke handle korte hoy. Also, data duplicate hobe, etao handle korte hobe nije.

Flexible - for future changes

- ➔ Future e new kono functionality add korle oita kono user er device e cholte pare, kono user er tay nao cholte pare. SQL hoile pura database er structure change kora lagto. But NoSQL e just oishob user er field e new functionality add korlei hoye jay.

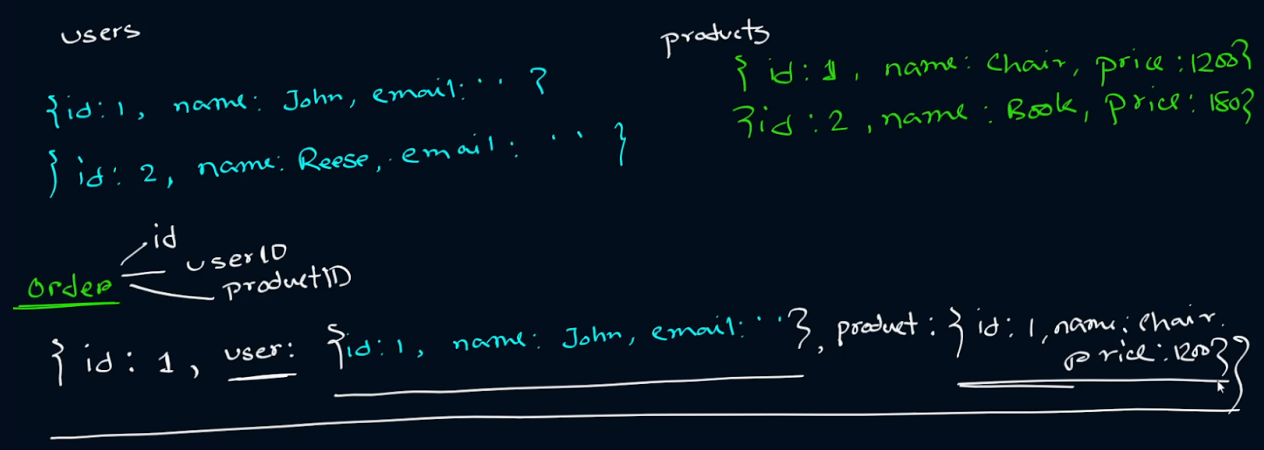
Where NoSQL is needed?

- ➔ **Mass read and write**
  - ◆ NoSQL er main advantage ashe jokhon database e mass reading and writing korbo.
  - ◆ Example -



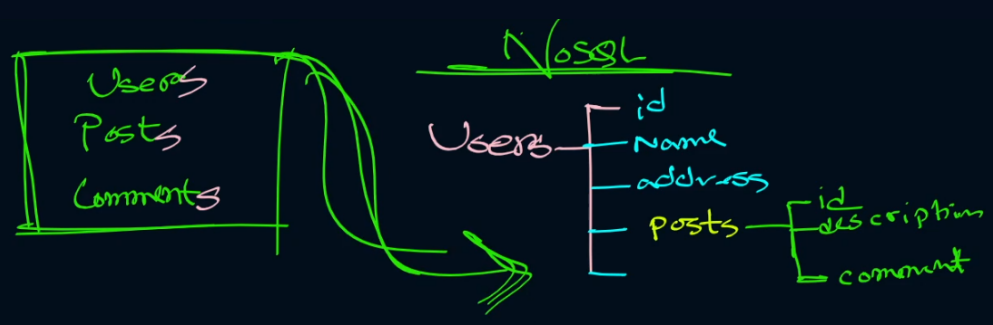
Suppose ekta order history dekhte chacchi. Onek beshi user er khetre tokhon koyekta server theke join query korte gele onek slow hoye jabe.

But in NoSQl -



Suppose amar ei app e beshi write requests ashbe na, khali read requests ashbe prochur. Prochur read requests ashte thakle ekhane just Order collection thekei shob information retrieve korte parbo. Users, Products collection gulay jawar dorkar portese na. Faster process.

- ➔ NoSQL er advantage tokhon pabo jokhon -
  - ◆ **Number of Collections** joto komaite parbo
    - Example -



Differences

SQL	vs	NoSQL
schema		schemaless
Relational		~~Rela~~ non-relational
Data is distributed across multiple tables		merged
Vertical scaling is possible.		✓
Horizontal scaling difficult/Not possible.		✓

- ➔ Vertical Scaling
  - ◆ Server er processing power barano
    - But ekta limit ase. Infinite power deya possible na.
  - ◆ Like building er floors baraye uchu korte thaka. But ekta shomoy por ar baraite parbo na, building dhopasshhh hobe.

- ➔ Horizontal Scaling
  - ◆ Onekgula server e data gula distributed kore rakha
  - ◆ Like boro land er moddhe onekgula building kora

Why Horizontal and Vertical scaling both are possible in NoSQL?

- ➔ Chaile shob gula data ke ekta collection e rakhte pari. Chaile ei collection ke onek gula server e distribute kore rakhte pari.
- ➔ Ekta particular server ke spec up korte partesi -> Vertical Scaling
- ➔ Data ta onek gula server e distribute kore rakhte partesi -> Horizontal Scaling

Why is Horizontal scaling not possible in SQL?

- ➔ SQL e NoSQL er moto easily table gula distribute kore kore multiple server e rakhte parbo na. Shudhu ekta server e spec up korte parbo.

Examples -

- ➔ SQL -> Oracle, MySQL
- ➔ NoSQL -> MongoDB, Firebase

Session Based Authentication

Authentication

- Login form e user password deyar por database er sathe milay dekha thik ase kina

Authorization

- Authentication successful howar por user ke emon kono page e niye jawa jekhane shobar access nai

Factors of Authentication

- Something that you know
  - Password
- Something that you have
  - ATM booth er card
- Something that you are
  - Face, biometric

Ei 3 tar jekono combination niye single factor, two factor, multi factor authentication build kora hoy.

Single Factor Authentication

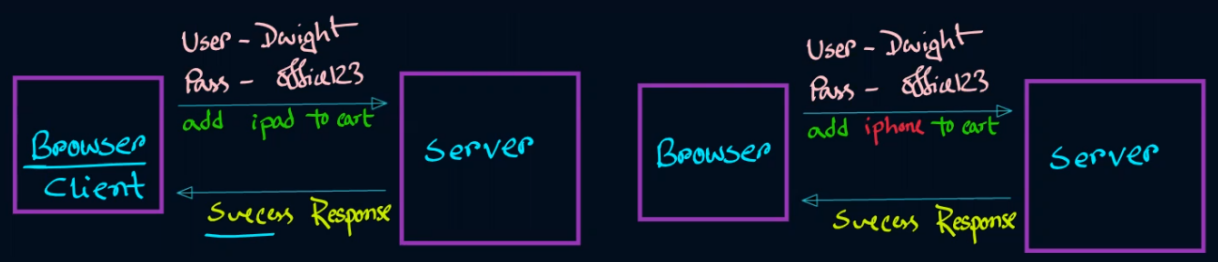
- Just password diye authentication

Two Factor Authentication

- 3 tar moddhe jekono 2 tar combination nite hobe
- Example - Password er sathe biometric/OTP dilam

Multi Factor Authentication

- Naam shuina buijha lou

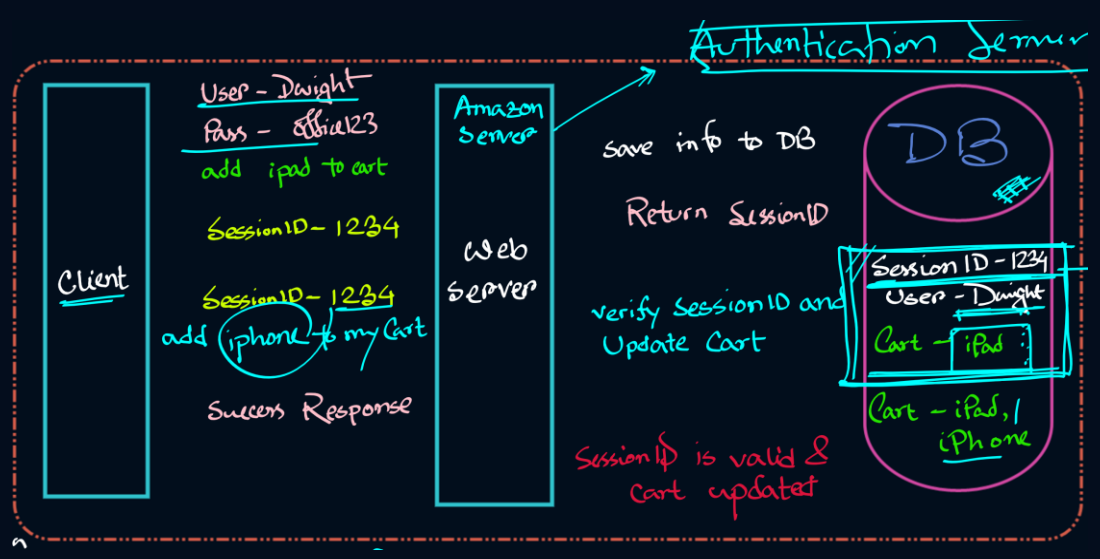


Jehetu HTTP ekta stateless protocol, prottek ta request ke individual hishebe handle korbe. Tai prottek page ei user ke login kore authenticated hoye hoye jaite hobe. Etar solution hocche Session Based Authentication.

How it works

Steps to create a session between client and web server -

- User sends a request to the server containing login credentials and the info it is requesting
- Server authenticates the user, stores all info about user in database and returns a sessionID.
- SessionID is stored in Browser storage. During next session user sends SessionID in HTTP header.
- Server verifies the SessionID and returns the requested information.



CSRF Attack

- ➔ Cross Site Request Forgery
- ➔ Suppose, tomar sessionID ta ami amar HTTP request e inject kore dilam. Ekhon server mone korbe je tomake data pathacche, kintu ashole to amake pathacche hehehe. Etai hocche CSRF attack. Mane tomar kono information use kore server ke bujhacchi ami mane tumi ar tumi mane ami. Don't write it plz don't, amar kase amar cheye bondhu tumi dami, ohnooo...

Browser Storage

- Local Storage
- Session Storage
- Cookie
  - Session Cookie
    - Validity ekta session porjonto
  - Permanent Cookie
    - Validity infinite
  - 3rd Party Cookie
    - Amar ager browsing history er upor base kore ekta refined result show korbe

- ➔ SessionID store kora hoy cookie te, session storage e na.
- ➔ Jokhon same domain theke request ashe, tokhon automatically cookie ta HTTP Header er sathe add hoye jay.

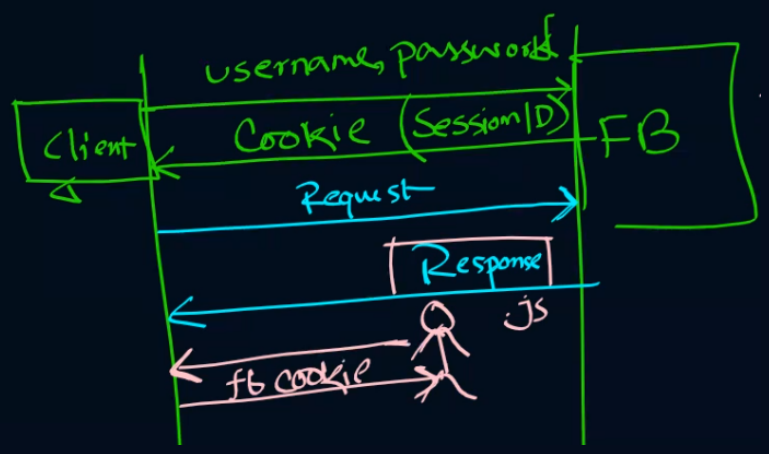
Differences

- Storage Limit
  - Local Storage : 5 MB
  - Session Storage : 5-10 MB
  - Cookie : 4 KB
- Accessibility
  - Local Storage
    - Client + Server dui side thekei accessible
  - Session Storage
    - Client + Server dui side thekei accessible
  - Cookie
    - Server side theke accessible
    - Client side theke not always
- Validity
  - Local Storage
    - Lifetime
  - Session Storage
    - Browser/Tab close

	Local Storage	Session Storage	Cookie
Storage Limit	5MB	5-10 MB	4KB
Accessibility	Client + Server	Client + Server	√ Server — Client
Validity	Lifetime	Browser/Tab Close	

XSS Attack

- Cross Site Scripting



- Jekono ekta jinisher moddhe ekta javascript inject kore dicche. Oi javascript ta client side e run hocche and browser theke cookie / info majhkhaner oi hacker ke diye dicche.

HTTPOnly Flag

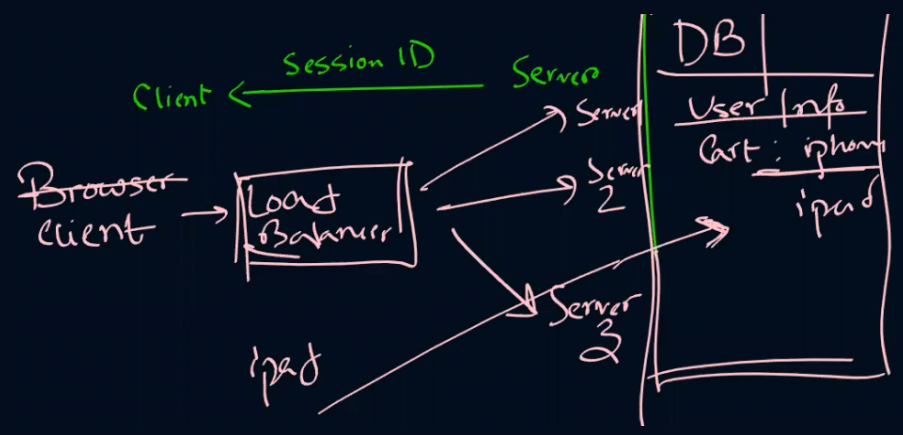
- Facebook oi cookie ta HTTPOnly kore dile cookie ta facebook server baade onno kothao theke access kora jabe na. Also kono javascript code diyeo access kora jabe na, bcz javascript client side e run kore. So majhkhaner oi hacker betar ar pet cholbe na, na khaye morbe.
- HTTPOnly kore dile cookie ta facebook server chara onno kono server jehetu access korte parbe na, tahole amra facebook e onno kichu diye login korte parbo na, like login with google, login with blablah.

Secure Flag

➔ HTTPOnly te same domain na hoile cookie pathaboi na. Kintu ekhane same domain hoileo pathabo na, jodi server er HTTPS ba SSL certificate na thake.

Limitations of Session Based Authentication

- Distributed Systems
  - Distributed System e Session based Authentication e jhamela hoy. Bcz server jokhon client ke sessionID dey, tokhon user er kichu necessary info database e store kore rakhe. Ekhon user notun kichu access korte chaile oita oi same server e asha lagbe, onno server e gele sheta validate korte parbo na, abar login kora lagbe.



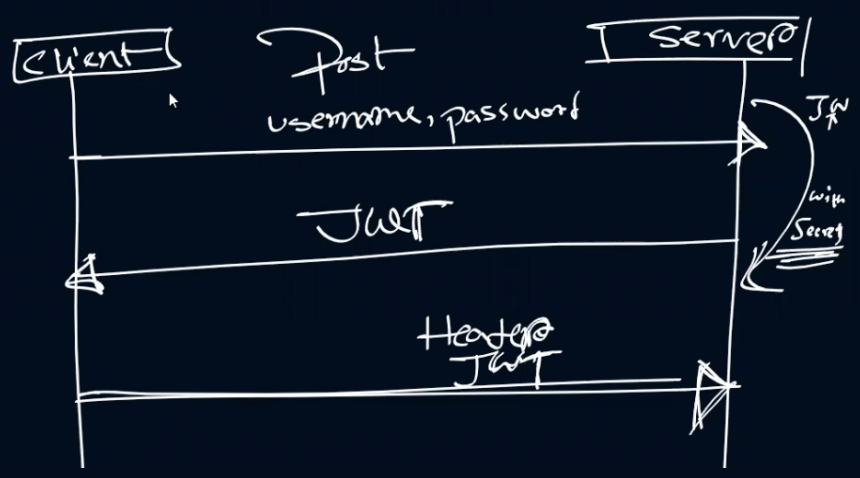
  Validate korte parbe jodi ami sessionID ta shobgula server e store kori. But this is obviously not a good solution.

- Performance Issue
  - Proti ta session er pore database e kisu na kisu store/retrieve korte hocche. Barbar eta kora costly process.

- Cookie Fraud
  - CSRF attack
  - Somehow ami tomar cookie (sessionID) ta paisi. Sheta amar pc theke pathay dilam. Server vabbe tumi request korso, kintu ashole korsi to ami hehe.

Token Based Authentication

- Information gula thake JWT (JSON Web Token) akare.



- Session based e user ke authenticate korar info gula ekta database e store kore rakhto. Kintu ekhane shob info gula JWT te store kore client er kase pathay dey. Erpor jotobar user request pathabe, totobar JWT ta oi server er kase pathabe.

- ***<u>SessionID store kori cookie te, but JWT store kori local storage e keno?</u>***
  SessionID store kortam cookie te, jehetu size choto chilo, 4KB, header er satheo jodi pathai tao pera hobe na temon. Kintu JWT store kori local storage e, bcz eta continuously data store kortei thake. Tai cookie te store kora feasible na. Size limit exceed korte pare.

➔ JWT encrypt korar jonno use kora hoy RSA HMAC-SHA256

➔ Handshaking Protocol er kaj hocche ekta secret generate kora, jei secret ta jodi shared line er moddhe keu intercept o kore tao jeno na jante pare. Ei secret diye JWT ta encrypt kore dei ba signature add kore dei, jeno secret channel er duijon entity chinte pare je tarai data pathaise, onno keu intercept kore data pathay nai.

Limitations of Session Based Authentication

- Scalability and Statelessness
- More secure
- Can be generated anywhere
- Helpful in Authorization
  - Suppose zoom e login korbo kintu 'login with google' diye. Google tokhon zoom e ekta JWT pathay dibe, and zoom authorize korte dibe.

JWT Structure

- Header
  - Header e thake -
    - Algorithm used for signing
    - Type of token

"HMAC SHA-256"

```
{
"alg" : "RSA",  ⟶ algorithm used for signing
"typ" : "JWT"  ⟶ type of token
}
```

- Payload
  - Payload e actual data thake

- Signature
  - Signature e thake hash value, jeta diye verify korte parbo jei data ta pathaisi sheta shared channel diye ashar shomoy konovabe change hoise kina.

➔ Header, Payload, Signature are separated by dots (.)

➔ Header & Payload duitai JSON.

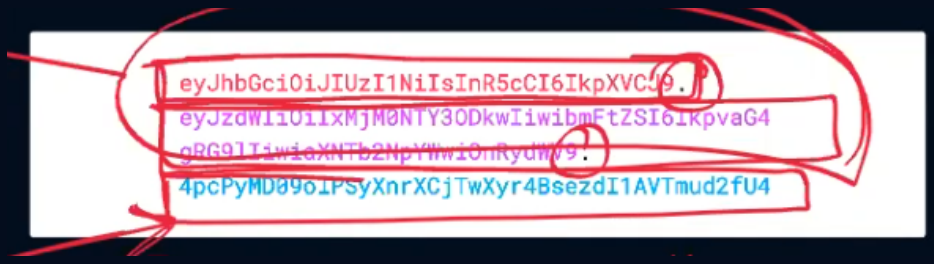
How JWT is Generated

➔ Header ke Base64Url Encoding kore.

➔ Erpor Payload ke Header Base64Url er sathe concatenate kore dibe.

➔ Then (Base64UrlEncode (Header) + "." + Base64UrlEncode (Payload)) ei pura jinish ta oi algorithm er bhitor diye pass korbo, and ekta signature pabo. Then ei signature ta oi header, payload er pore ekta "." diye add kore dibo.

➔ JWT example -



OAuth v2.0

- OpenAUTHorization
- Trusted party theke kisu resource use korar jonno authorize korbe.
- For example, ami je ami eita proof korbe na, but amar github er kisu repository theke information dibe. So OAuth authentication dey na, authorization dey.

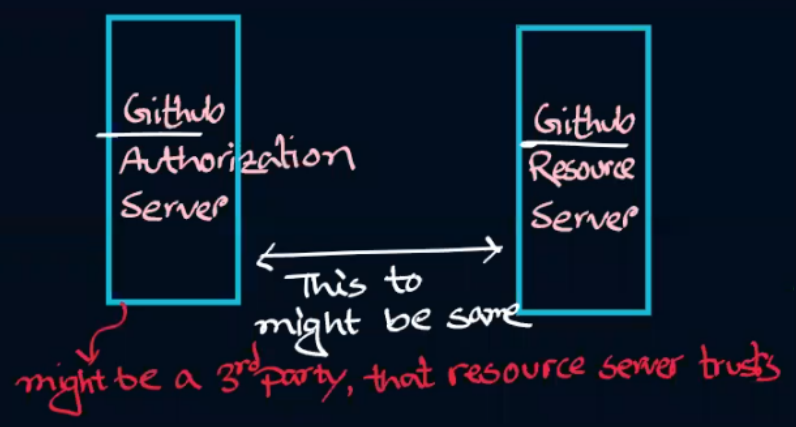
Suppose, ekta app LikeTinder. Ei app ta Nafiz er user profile update korar jonno Github theke data/repository info anbe.

- **Client**
  - Ekhane app hocche client
  - Client e thake -
    - Client ID
    - Client Secret
      - Only Client and Authorization Server knows.
      - Allows to securely share information.

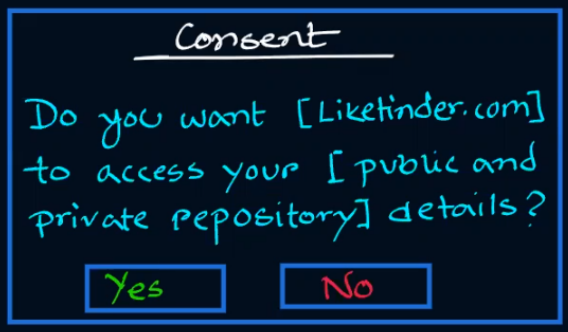
- **Resource Owner**
  - App ta Nafiz er repository info gula use kortese. Ekhane Nafiz hocche resource owner.

- **Authorization Server**
  - Github to jake take info chailei diye dibe na. Authorization server thik korbe kake info dibe, kake dibe na.

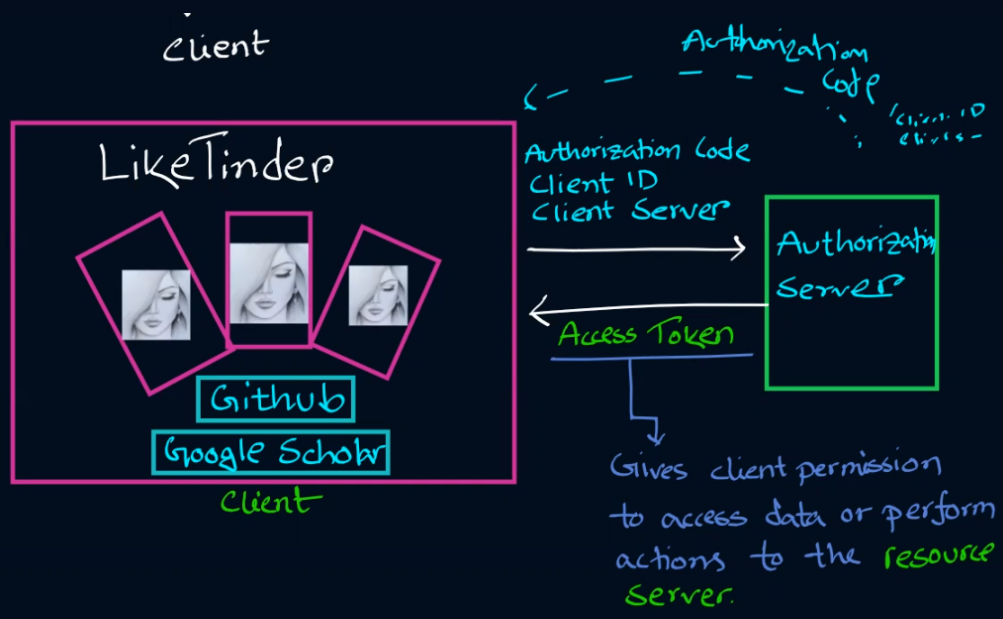
- **Resource Server**
  - Authorization Server jokhon LikeTinder ke authorize korbe, tokhon LikeTinder Resource Server er kache request pathaite parbe resource neyar jonno.



- Response Type
  - Authorization code
- Scopes
  - Resource server theke ki ki nite chai
- Consent
  - Authorization Server last barer moto dekhabe je client wants to access your scopes.



- Redirect / Callback URL
  - Ekhane eta hocche Liketinder.com
- Authorization Code
  - Short validity
  - Authorization Server first ei client ke ekta Authorization Code dey. Then client Client ID, Client Secret abar Authorization Server kei pathay dey. Authorization Server tokhon client ke ekta Access Token dibe. Ei Access Token tokhon client ke Resource Server e access dey. Ei access token e ki ki ase sheta client bujhe na. Client just Access Token ta Resource Server e pathay dey, Resource Server take data ta pathay dey.



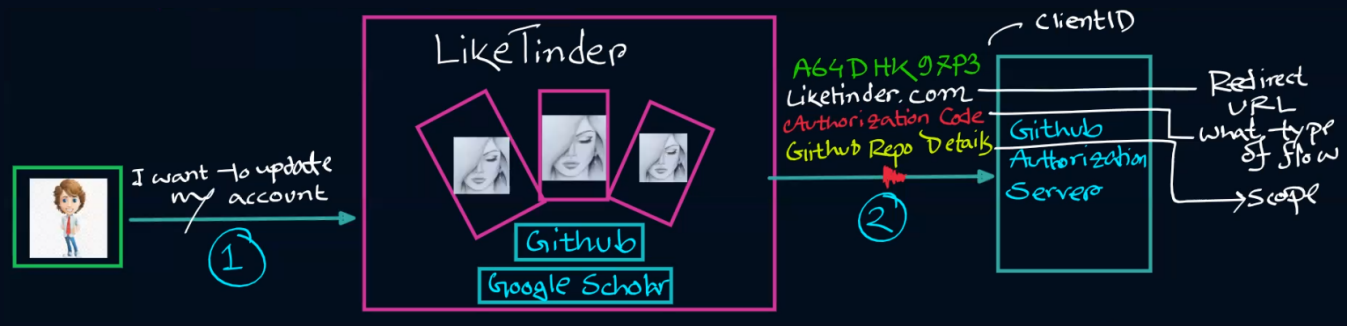
It is the example of most common OAuth Flow -> Authorization Code Flow

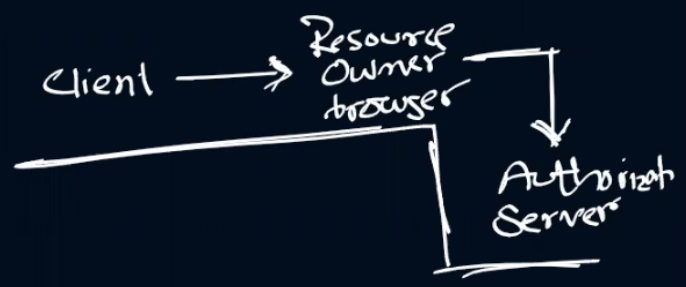
There are 4 OAuth Flows -

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

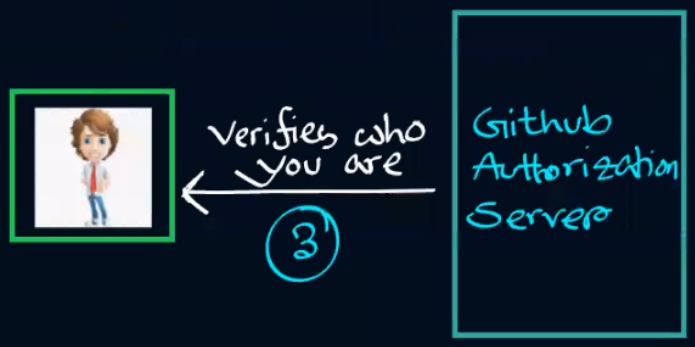
- Response Type
  - Ami above 4 ta theke kon dhoroner flow use kortesi

Authorization Code Flow



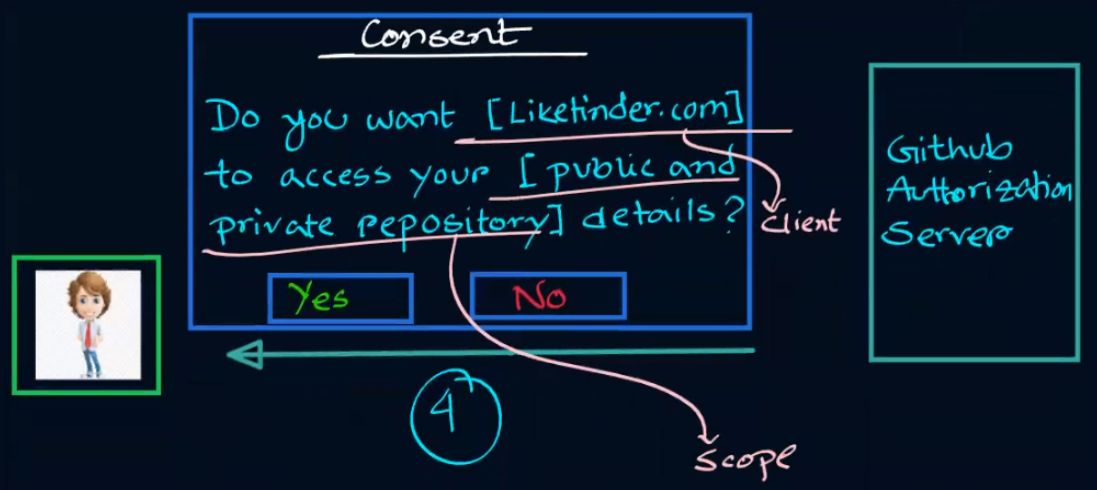


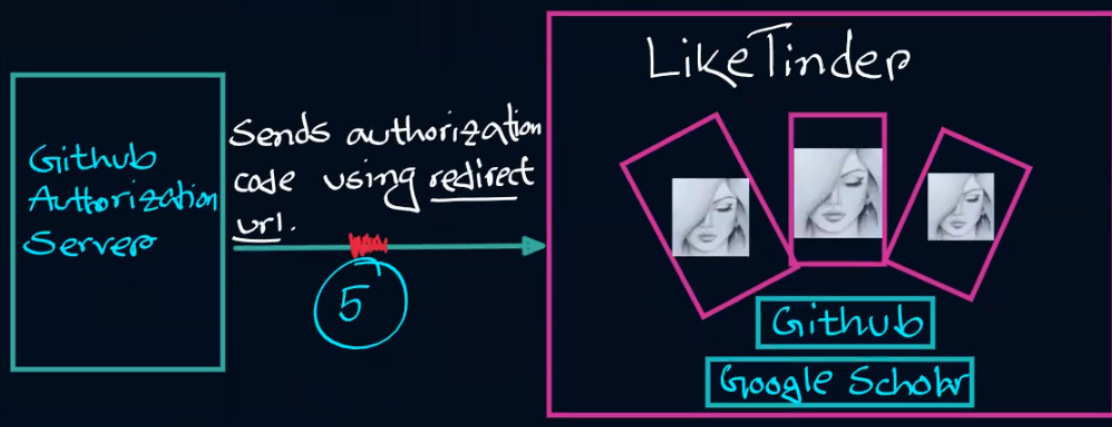
Client ar Authorization Server er moddhe direct communication hoy na. Resource Owner majhkhane intermediary hishebe kaj kore.



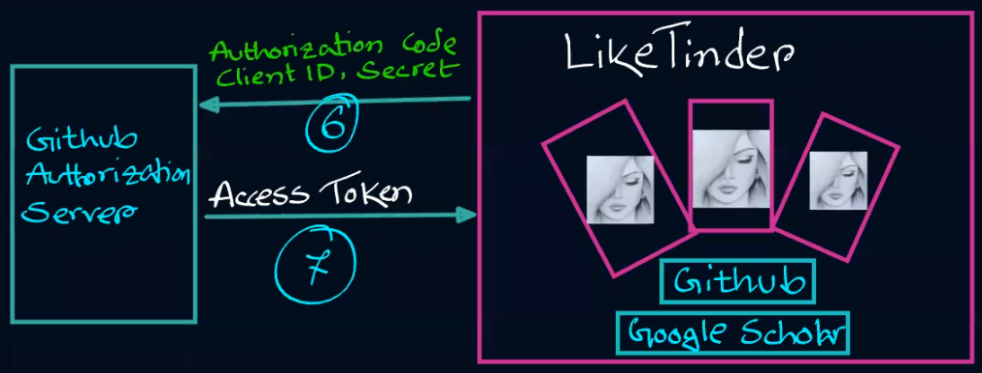
Ei step e Nafiz Authorization Server e kache nijeke authenticate kore. Like github e age theke login thakle direct dhuke jabe, naile login kore dhukte hobe.

Ekhon porjonto kono step e client Resource Owner ke authenticate kortese na. That's why we call it OpenAuthorization, not OpenAuthentication.

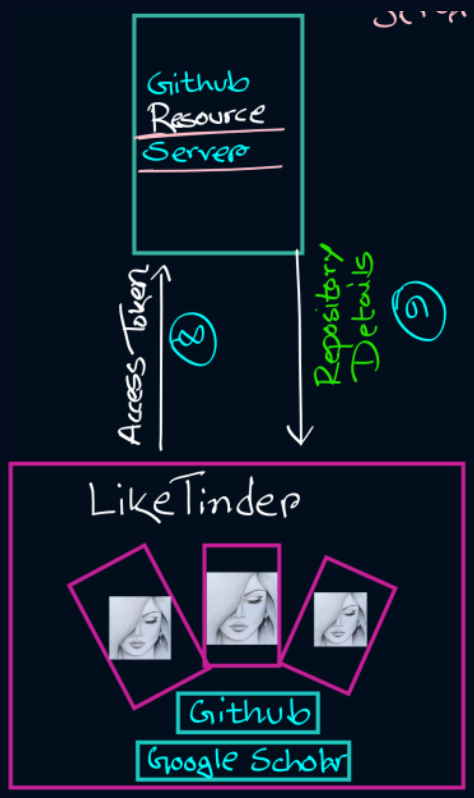




Ekhane Github Authorization Server authorization code kake pathabe, eta bujhbe redirect url diye jeta step 2 tei github authorization server e pathay deya hoisilo.



Ei access token diye ekhon LikeTinder er right ase Nafiz er github er public/private repository access korar.
Github ar LikeTinder er moddhe shared secret ta always same thake.



Ei access token e ki ase sheta LikeTinder er janar dorkar nai, o access token ta just github resource server e pathay dibe, and then resource server Nafiz er repository details LikeTinder ke pathay dibe. Ekhon Nafiz shukhe shanti te LikeTinder use korte thakbe uWu

Implicit Flow

Ekhane Secret ta step 2 tei clientID, redirect url etc ogular shathe diye dibe.

So step 5 e server Authorization Code na diye direct Access Token tai diye dibe.

- ➔ Khubi trusted customers thakle implicit flow use korbo. Naile authorization flow use korbo. Bcz je keu step 5 e browser e script inject kore diye hack korte pare.

Resource Owner Password Credentials Flow

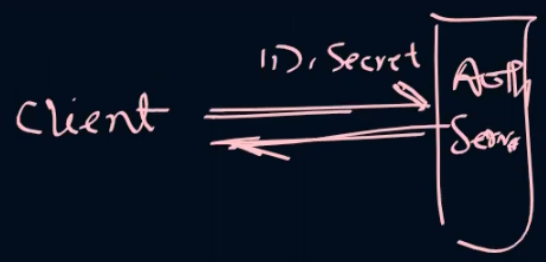
Step 3 te verify kora lagtesilo je oita Nafiz.

Ei flow te client Nafiz er credentials jane. Jokhon she github ke request pathabe, tokhon resource owner Nafiz er credentials o pathay dibe. Authorization server erpor resource owner er credentials (clientID, secret) and client er credentials match korbe, and access token pathay dibe.

- ➔ Khubi trusted hoile use korbo, as client ke user pass diye rakhtesi.

Client Credentials Flow

Ekhane client direct Auth Server er sathe communicate kore. Auth server client ke authenticate kore clientID, secret diye. Erpor client direct authorization server theke data access kore.



➔ ***Why do we need OAuth?***

- To prevent phishing attack
- OAuth jokhon chilo na, tokhon LikeTinder ei Github er moto ekta portal khulto, oikhane client kei amar user pass diye ditam.

  Disadvantages

  - Client user pass use korte pare.
  - Suppose client er server ta kono karone compromise hoye gelo. Tahole user entity shob gulai compromise hoye jabe.
- But OAuth e kono sensitive info, credentials karo sathe share kortese na. So we use it to share resources without sharing credentials.

OAuth 1.0 vs OAuth 2.0

- Different terminology
  - For example OAuth 2.0 te resource server, authorization server OAuth 1.0 te Service Provider. 2.0 te client 1.0 te user.
- Backward compatibility nai.
  - OAuth 1.0 te ekta app banaile oita pore OAuth 2.0 te replace korte parbo na and vice versa, eder working mechanism etotai different.
- Security
  - OAuth 1.0 beshi secured. Client server er moddhe communication shobi ekta shared signature diye encrypt kora thakto. But prochur message er ei encrypting, decryption, signature checking overall performance ke komay dito.
  - OAuth 2.0 te kono security nai. Client server er moddhe message gula plain text e pathay. Kintu shorto hocche, TLS / SSL Certificate na thakle 2.0 use korte parbo na.
    - That means, OAuth 2.0 te jotogula message passing dekhe ashlam, ektao use korte parbo na jodi HTTP protocol use kori. Only HTTPS hoile parbo.