Networking & Web fundamentals

HTTP Methods - Tells server what action the client wants to perform on a resource

* When any service talks to a backend API, it sends a request. HTTP methods defines the intention.

1) GET :- Fetch or read data from the server (Idempotent)
(Loading products, Fetching user profile)
Used in dashboards data loading.

2) POST :- Create new data in the server
Used in signup, placing orders, uploading files.

3) PUT :- Update an entire resource (Idempotent)
→ same result.

4) PATCH : Update partial data (specific fields)
(changing order status

5) DELETE :- Removes resources (Idempotent)

HTTP Status Codes :- 3-digit numbers returned by server to tell the client what happened to the request.

- 1XX - Informational (100 - continue)
- 2XX - success (200 - OK, 201 - Created, 202 - Accepted)
(204 - No content)
- 3XX - Redirection (301 - moved permanently, 302 - found (temp)
304 - not modified)
- 4XX - Client Errors (400 - Bad req. , 401 - Unauthorized
403 - forbidden , 404 - Not found)
429 - Too many req)
- 5XX - Server Errors
(500 - Internal server error, 502 - Bad gateway, 503 - Service Unavail

* What happens when we enter google.com?

1) URL parsing (Browser breaks the URL)

https://www.google.com

- Protocol- HTTPS Domain - google.com Path → / (default)
  (TLS + HTTP)

2) Browser needs the IP address of google.com.

DNS Lookup order

Browser cache → OS cache → Router cache → ISP DNS → Root DNS → TLD → Authoritative DNS

3) Gets the IP

4) TCP connection :- Browser establishes a TCP connection with Google's server.

3-way Handshake - SYN, SYN-Ack, ACK

* Happens before any HTTP data is sent.

5) TLS handshake (HTTPS)

- Browser sends supported encryption methods.
- server sends certificate (pub key)
- Browser verifies → session key generated - Secure channel established.

6) HTTP Req. is sent (Encrypted, sent over TCP through internet)

7) At Google server (Load balancer receives request.
↓
Routes to nearest data center
↓
Web server processes req..
↓
App generates response.

8) Browser rendering

1) Parses HTML → DOM, CSS → CSSOM, Builds render tree
2) Executes JavaScript → loads additional resources.
4. • Cookies sent (auto), caching rules applied, preloading.

DNS

- Translates domain names into machine readable IP add that computers use to locate and connect to websites

Working

1) User enters a domain name : google.com
2) Request to DNS resolver → Resolver queries DNS servers.
3) If IP isn't cached, it asks a series of DNS servers (root, TLD, authoritative) in hierarchy to find IP.
4) IP address returned by authoritative server.
5) Connection made

Nameservers

- Nameservers are DNS servers that know the IP address of a domain.

Types :- 1) Root Nameservers (Total 13 worldwide)
↓
They tell which TLD server to ask.

2) TLD Nameserver : Handles domains like .com, .in, .org.
- Tell which authoritative nameserver handles domain.

3) Authoritative Nameserver (Imp)
- Stores DNS records (ANAME, CNAME)

steps

* Resolver asks Root NS → Root replies → ask .com TLD NS
↓
Authoritative NS returns IP ← ask google.com ← TLD replies
↳ Browser connects to server.

* DNS records :- They are instructions stored in name servers that tell the internet :-
How should traffic for this domain be handled.

1) A Record (Address) :- Maps a domain directly to an IP
→ Points to IPv4, fast lookup, direct mapping.
→ Use : • Backend servers • static IP servers
• APIs • Load balancers with fixed IPs

2) CNAME (Canonical) :- Maps a domain to another domain name.
→ www.exam.com → exam..com
→ Creates an alias & always points to a domain.

Note :- CNAME cannot be used on root domain because it already has NS records.

* Use CNAME when IPs change often.

Propagation :- Time taken for changes to spread everywhere.
- When we change :- Nameservers, CNAME, IP add.;
- That change does not update instantly across whole internet.
- The delay is called DNS propagation.
- To make internet fast and scalable, DNS uses caching. Because of this caching when we change a DNS record, OLd values stay cached until they expire.
- Some users may reach old server some at new server
- TTL (Time to live) → tells DNS to cache a record for certain time.
- If no caching & no TTL, every request will hit authoritative DNS → DNS server will overload, Internet will be slow.