1. Introduction to Web Communication

HTTP is the foundation of communication on the World Wide Web.

It follows a client-server model where a client sends a request and the server returns a response.

HTTP works on top of TCP/IP and defines how messages are formatted and transmitted.

It is stateless meaning every request is independent.

Modern applications use HTTP extensively for APIs, microservices and frontend communication.

2. HTTP vs HTTPS

HTTPS is HTTP over TLS encryption.

It protects confidentiality, integrity and authenticity of data.

HTTPS uses certificates issued by Certificate Authorities.

Port 80 is default for HTTP and 443 for HTTPS.

Modern browsers mark HTTP as insecure.

3. URL Anatomy

A URL identifies a resource on the internet.

It contains scheme, host, port, path, query and fragment.

Query parameters pass filters and pagination.

Fragments are used by browsers and not sent to server.

4. HTTP Request Structure

Request contains method, path, version, headers and optional body.

Headers describe metadata about the request.

Body is used in POST, PUT and PATCH methods.

GET requests should not contain body in browsers.

5. HTTP Response Structure

Response contains status code, headers and body.

Headers provide caching, cookies and content information.

Body contains actual data like HTML or JSON.

6. HTTP Methods

GET retrieves data.

POST creates resources.

PUT replaces resources.

PATCH updates partially.

DELETE removes resource.

7. Idempotency

Idempotent methods produce same result on repeated calls.

GET PUT DELETE HEAD OPTIONS are idempotent.

POST is non-idempotent.

8. Status Codes

1xx informational responses.

2xx success responses.

3xx redirection.

4xx client errors.

5xx server errors.

9. Authentication

Authentication verifies identity.

Authorization verifies permissions.

Basic auth sends credentials base64 encoded.

Bearer tokens use JWT.

10. Cookies

Cookies store small data in browser.

They maintain session state.

HttpOnly prevents JavaScript access.

Secure sends only over HTTPS.

11. Sessions vs Tokens

Sessions stored on server.

Tokens stored on client.

JWT allows stateless scalability.

12. CORS

CORS allows cross origin requests.

Browser enforces same origin policy.

Server sends Access-Control-Allow-Origin header.

13. Caching

Cache reduces server load.

Cache-Control defines lifetime.

ETag enables validation.

304 Not Modified avoids full response.

14. REST Architecture

REST uses resources and HTTP verbs.

Stateless communication.

Uniform resource naming.

15. Pagination & Filtering

Query params page limit sort filter.

Improves performance.

16. Rate Limiting

Prevents abuse.

429 returned on excess traffic.

17. HTTP Versions

HTTP/1.1 sequential.

HTTP/2 multiplexed streams.

HTTP/3 UDP QUIC transport.

18. Security

CSRF prevented using SameSite cookies.

XSS prevented by sanitization.

HSTS forces HTTPS.

19. Content Negotiation

Accept header defines response type.

Server responds JSON XML HTML.

20. Streaming & Chunking

Chunked transfer sends data in parts.

Useful for large responses.

21. API Best Practices

Use nouns not verbs.

Use proper status codes.

Version APIs.

22. Frontend Considerations

Retry idempotent requests only.

Handle network failures.

Use caching wisely.