Article 2 – Client-side vs Server-side

Modern-day web applications work on the basis of two complementary environments to deliver functionality: the client side, which runs on the user’s browser, and the server-side, which runs on a remote device.

**Client-side (Front-end)**

Runs in the user’s browser by rendering HTML/CSS/JS files to display the User Interface. Performs immediate validations (such as ensuring a field isn’t empty in a form) and acts as the middleman between the user and the server.

It yields fast feedback and reduces server load. However, it lacks device resources and is less secure.

**Server-side (Back-end)**

Runs on remote servers such as cloud instances using languages like Node.js, Python (Flask, Django), Ruby (Rails), Java (Spring). Stores and retrieves data, business logic (domain logic), authentication, and authorization. It also generates HTML/JSON responses to the client-side.

The communication between the client and server can be explained using the HTTP request-response cycle.

**HTTP Request-Response Cycle**

1) DNS Lookup

When the client searches for a URL, it is first converted from human-readable text to an IP Address via the DNS.

2) TCP (and TLS)

TCP 3-way Handshake (for HTTP 1.1 and HTTP 2):

SYN 🡪 SYN-ACK 🡪 ACK

TLS Encryption (for HTTPS): Encrypts parameters and establishes session keys.

3) Client sends HTTP Request

Headers: Metadata (content type, cookies)

Body: Payload (for methods like POST, PUT, PATCH)

4) Server Processes the Request

The server authenticates/authorizes, runs business logic, applies routing rules, and prepares response data.

5) Server Sends HTTP Response

Headers: Metadata (server, content type, caching)

Body: The actual resource or data

6) Connection Handling

If both sides use “Connection: Keep-Alive”, then the TCP/TLS sessions is kept connected for longer. Otherwise, the connection is closed after the response.