Introduction to Web Applications & Architecture

1. What is a Web Application?

A **web application** is a **software system accessed via a web browser** over a network (usually the Internet) that performs **dynamic operations**, processes user input, interacts with databases, and returns customized responses.

Key Characteristics

- Runs on a **client–server model**
- Uses **HTTP/HTTPS** as communication protocol
- Supports **dynamic content generation**
- Often handles **authentication, sessions, and data storage**

Examples

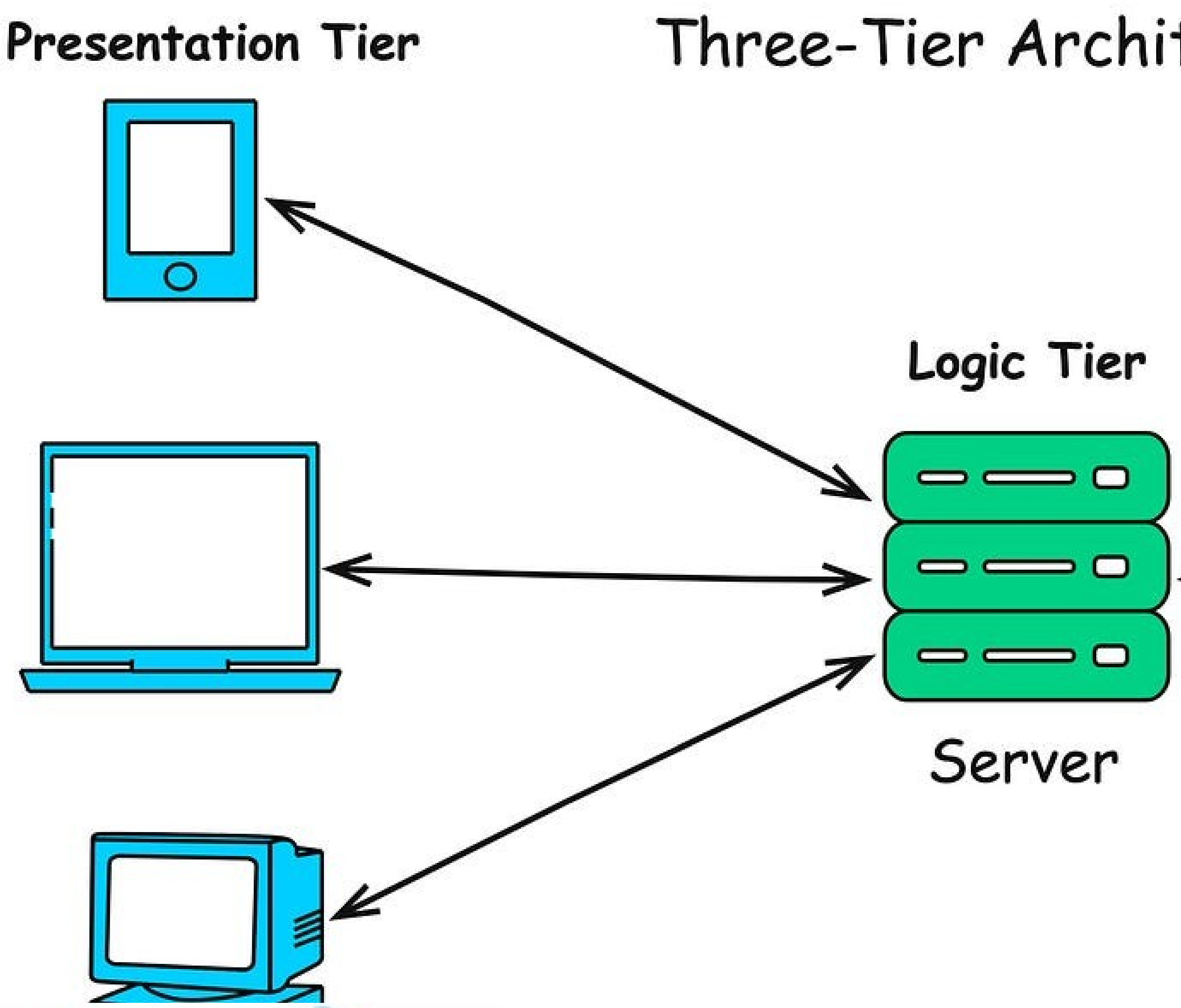
- Online banking portals
- E-commerce websites
- Learning Management Systems (LMS)
- Cloud dashboards (AWS Console, Azure Portal)

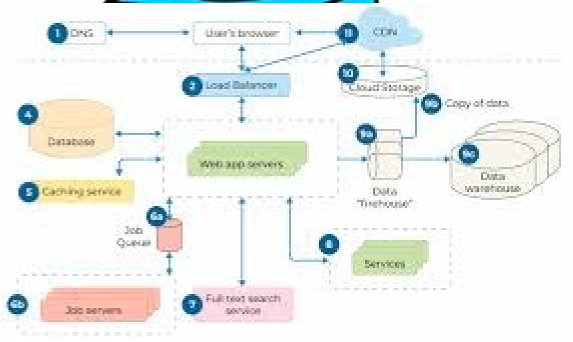
---

2. Difference Between Website and Web Application

Feature	Website	Web Application
Nature	Informational	Interactive
User Input	Minimal	Extensive
Backend Processing	Limited	Heavy
Database	Optional	Mandatory
Security Risks	Lower	High
Example	Blog, News Site	Banking App, E-commerce

3. Basic Web Application Architecture (3-Tier Model)





Tier 1: Client (Presentation Layer)

- Runs on **user's browser**
- Technologies:
  - HTML (structure)
  - CSS (styling)
  - JavaScript (logic)

- Sends **HTTP requests** and receives **responses**

📌 **Security relevance:**
XSS, CSRF, clickjacking, client-side validation bypass

Tier 2: Application Server (Business Logic Layer)

- Processes client requests
- Enforces **authentication & authorization**
- Applies **business rules**
- Communicates with database

Common technologies:

- PHP, Java, Python, Node.js
- Frameworks: Spring, Django, Express, Laravel

📌 **Security relevance:**
SQL Injection, Broken Authentication, SSRF, RCE

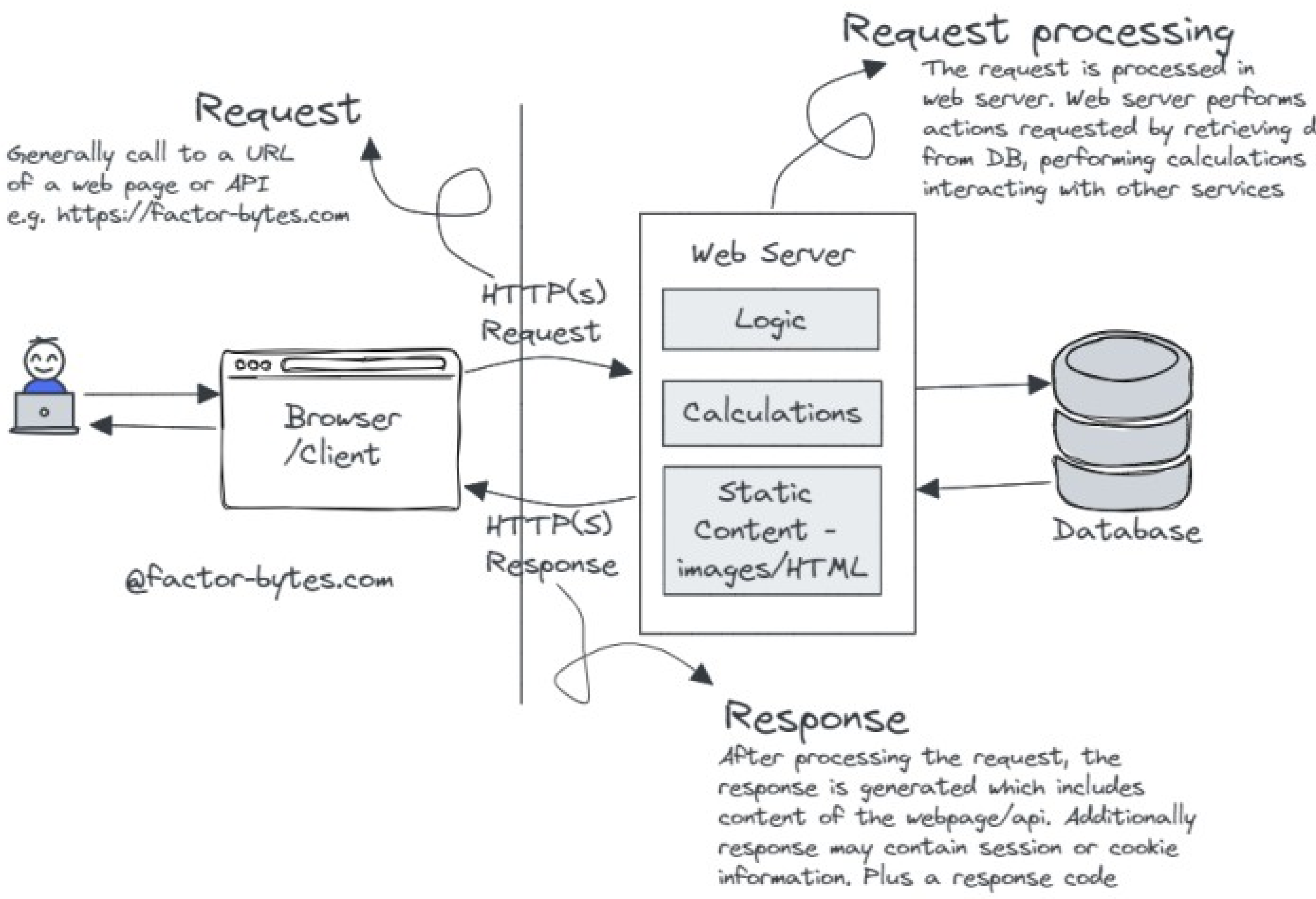
Tier 3: Database Server (Data Layer)

- Stores:
  - User credentials
  - Application data
  - Logs and transactions
- Examples:
  - MySQL, PostgreSQL
  - MongoDB (NoSQL)

📌 **Security relevance:**
Data leakage, SQL/NoSQL Injection, privilege escalation

---

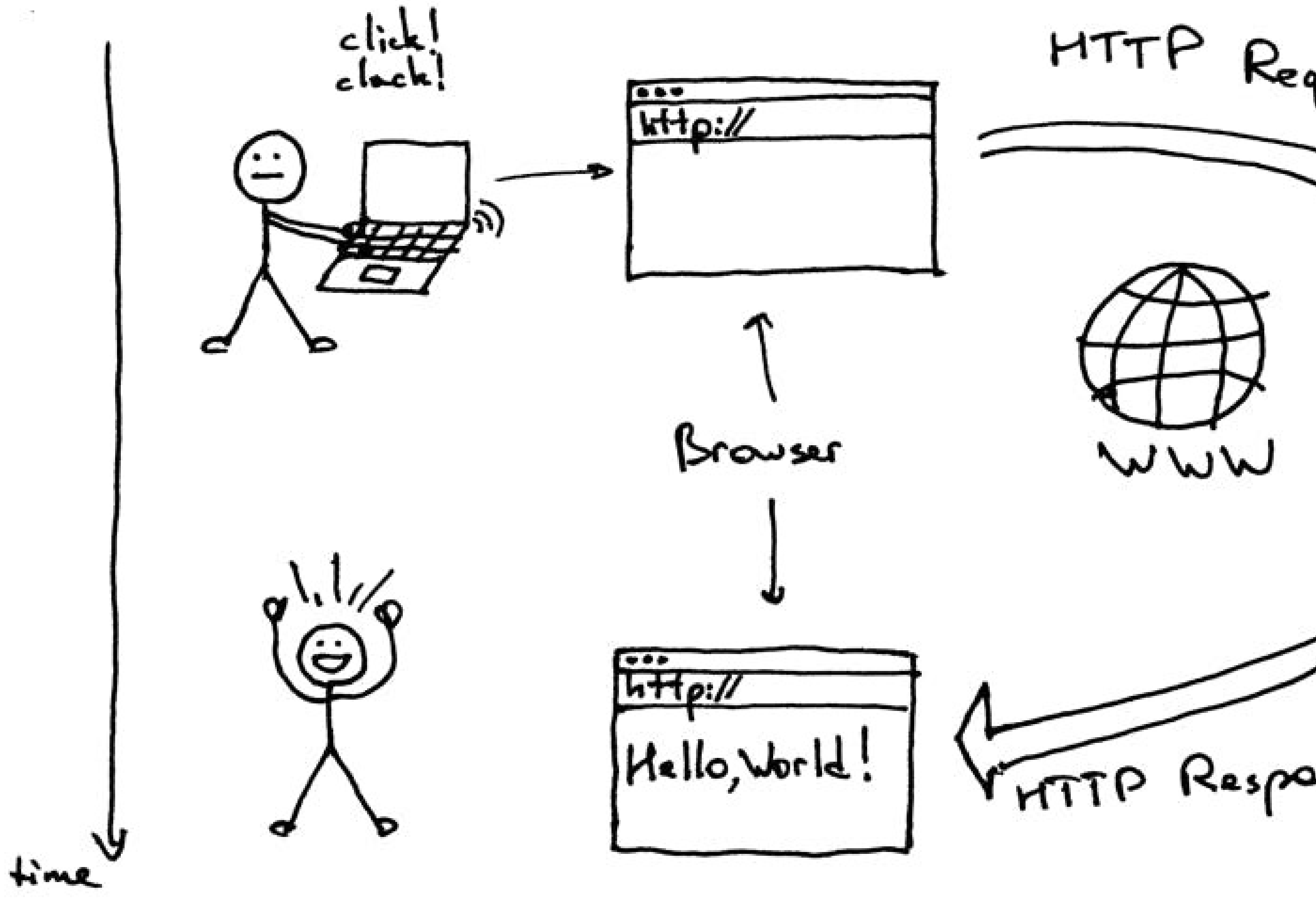
4. Detailed Request–Response Flow



Generally call to a URL of a web page or API e.g. https://factor-bytes.com

The request is processed in web server. Web server performs actions requested by retrieving d from DB, performing calculations interacting with other services

After processing the request, the response is generated which includes content of the webpage/api. Additionally response may contain session or cookie information. Plus a response code



Step-by-Step Flow

1. User enters URL in browser
2. Browser sends **HTTP request**
3. Web server forwards request to application logic
4. Application queries database (if required)
5. Server generates response (HTML/JSON)
6. Browser renders output

---

5. Core Components of Web Application Architecture

5.1 Web Server

Handles incoming HTTP requests.

Examples:

- Apache
- Nginx

- IIS

Functions:

- Routing requests
- Serving static content
- Forwarding dynamic requests

5.2 Application Logic

- Implements rules and workflows
- Handles:
  - Login validation
  - Form processing
  - Role-based access control

5.3 Database

Stores persistent data.

Types:

- Relational (SQL)
- Non-relational (NoSQL)

5.4 Session Management

- Maintains user state
- Uses:
  - Cookies
  - Session IDs
  - Tokens (JWT)

📌 **Security relevance:**
Session hijacking, fixation, replay attacks

---

6. Authentication vs Authorization

Authentication	Authorization
Who are you?	What can you do?

Authentication	Authorization
Login process	Access control
Username/password	Roles & permissions

7. Types of Web Application Architectures

7.1 Monolithic Architecture

- All components in one codebase
- Simple but hard to scale

📌 Security risk:
Single vulnerability → full compromise

7.2 Microservices Architecture

- Application split into services
- Each service runs independently

📌 Security risk:
API abuse, service-to-service attacks

7.3 Serverless Architecture

- Backend logic runs as functions
- Example: AWS Lambda

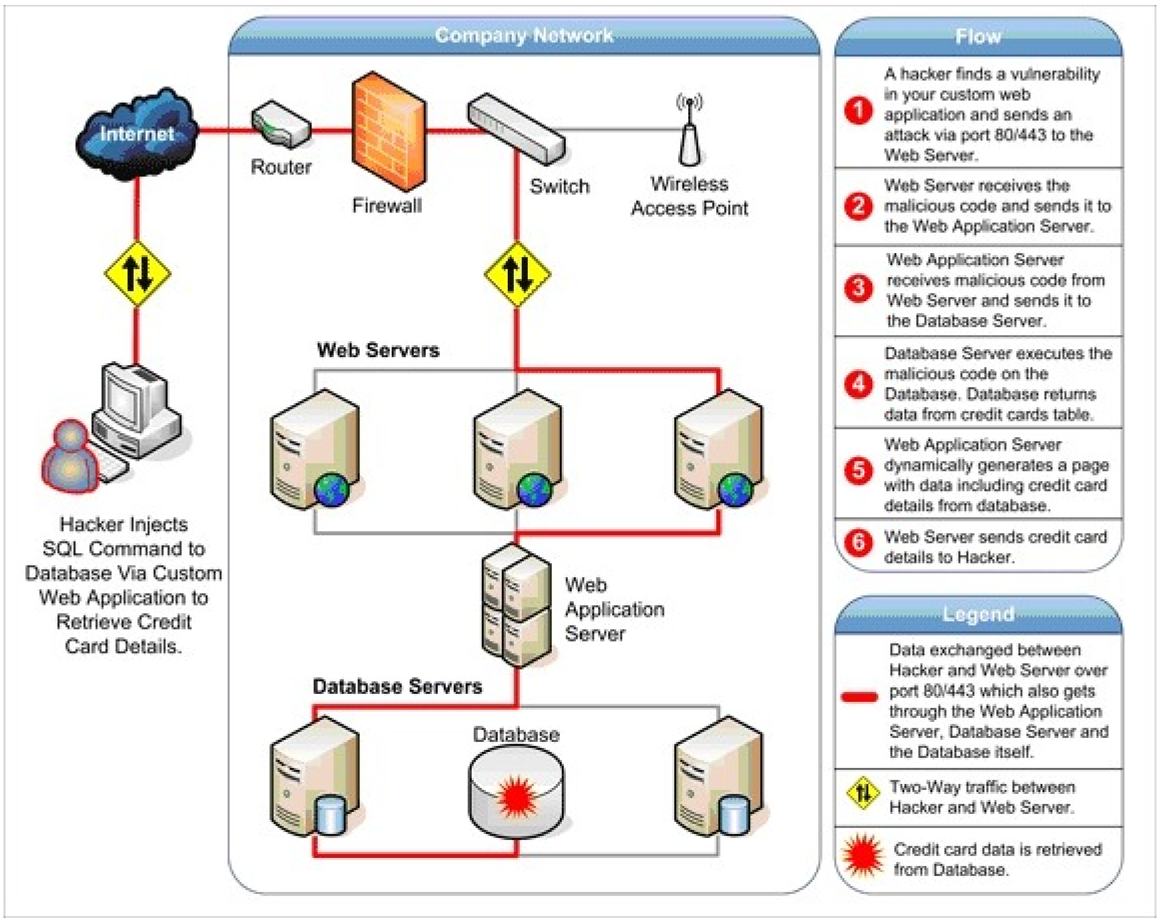
📌 Security risk:
Misconfigured permissions, event injection

---

8. Common Web Application Attack Surface

V2: Page Creation Method (PCM)

V3: Degree of Distribution (DOD)


Company Network
Flow
Internet
Router
Firewall
Switch
Wireless Access Point
Web Servers
Web Application Server
Database Servers
Database
Hacker Injects SQL Command to Database Via Custom Web Application to Retrieve Credit Card Details.
1 A hacker finds a vulnerability in your custom web application and sends an attack via port 80/443 to the Web Server.
2 Web Server receives the malicious code and sends it to the Web Application Server.
3 Web Application Server receives malicious code from Web Server and sends it to the Database Server.
4 Database Server executes the malicious code on the Database. Database returns data from credit cards table.
5 Web Application Server dynamically generates a page with data including credit card details from database.
6 Web Server sends credit card details to Hacker.
Legend
Data exchanged between Hacker and Web Server over port 80/443 which also gets through the Web Application Server, Database Server and the Database itself.
Two-Way traffic between Hacker and Web Server.
Credit card data is retrieved from Database.

Layer	**Common Attacks**
Client	XSS, CSRF
Server	SQLi, RCE
Database	Data leakage
Network	MITM
Authentication	Brute force

9. Why Web Applications are High-Risk Targets

- Internet-facing
- Handle sensitive data
- Complex codebases
- Frequent third-party dependencies
- Poor input validation

---

10. Security by Design – Key Takeaways

✔️ Validate input at server side
✔️ Use prepared statements
✔️ Secure session management

✔️ Enforce least privilege
✔️ Apply HTTPS everywhere