CDAC MUMBAI

ECGC Training : Day 1

Web Fundamentals & Basics

Dr Kiran Waghmare
&
Vipul Tembulwar

Start Slide

Agenda

- Web Application basics
- Static vs. Dynamic Web Applications
- HTTP Protocol, HTTP methods (GET/POST)
- Request, Response, URL, Port mapping
- Hot Deploy Mode
- HTML Basics
- CSS Basics
- JavaScript and jQuery Basics

Overview

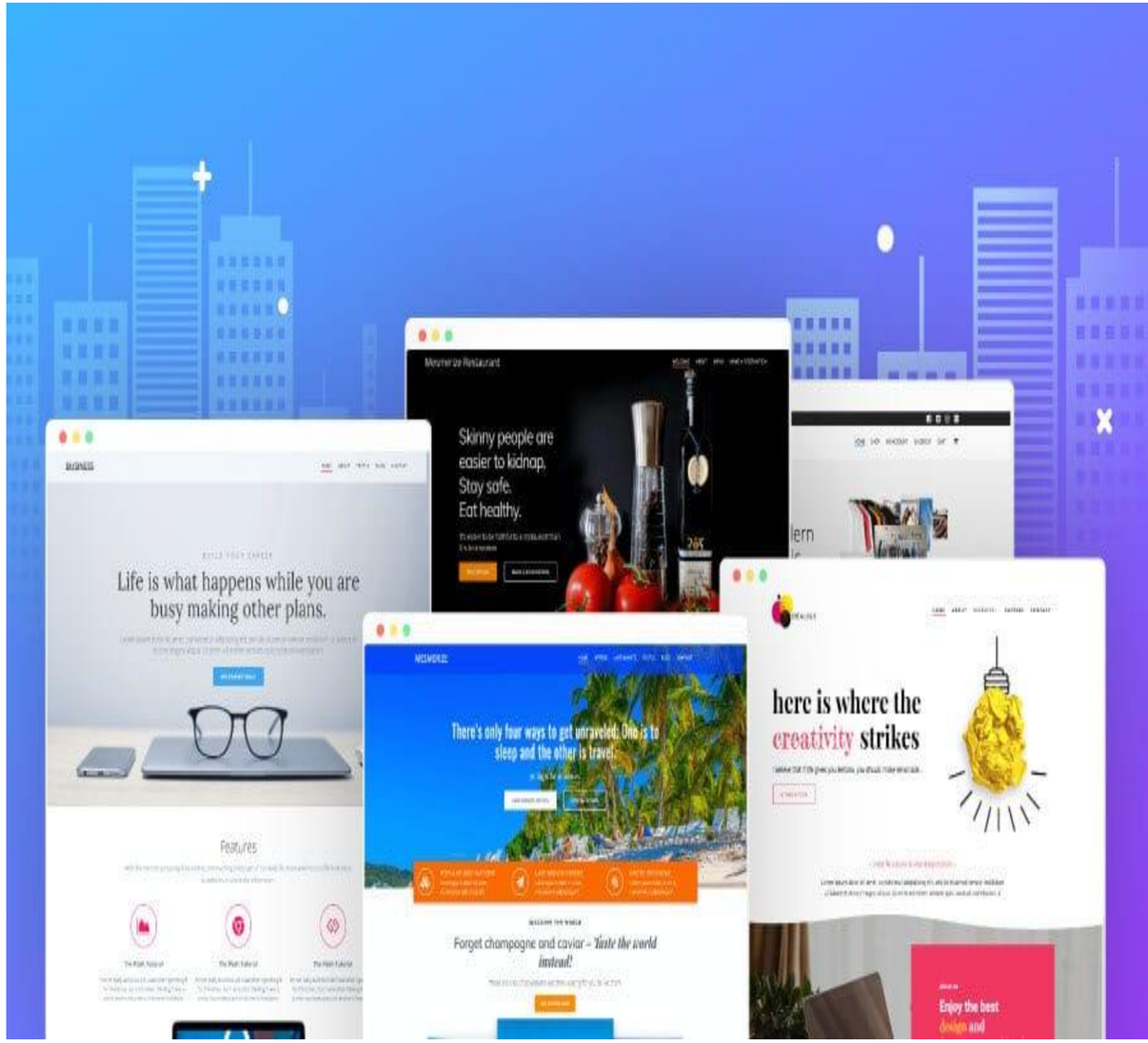
- **What is Web Development?**
  - Definition, key areas, and types of web development.
- **Frontend Development**
  - Overview of technologies like HTML, CSS, JavaScript.
  - The role of UI/UX in web development.
- **Backend Development**
  - What happens on the server-side: databases, server-side programming, and APIs.
- **Full Stack Development**
  - Combining frontend and backend skills to build comprehensive solutions.
- **Web Design & User Experience (UX)**
  - The importance of design in development: aesthetics, usability, and performance.
- **Web Hosting & Deployment**
  - Making your website available to users globally through web hosting and cloud services.
- **Web Development Tools & Frameworks**
  - Introduction to tools that speed up development (e.g., code editors, version control, frameworks).

Web Development

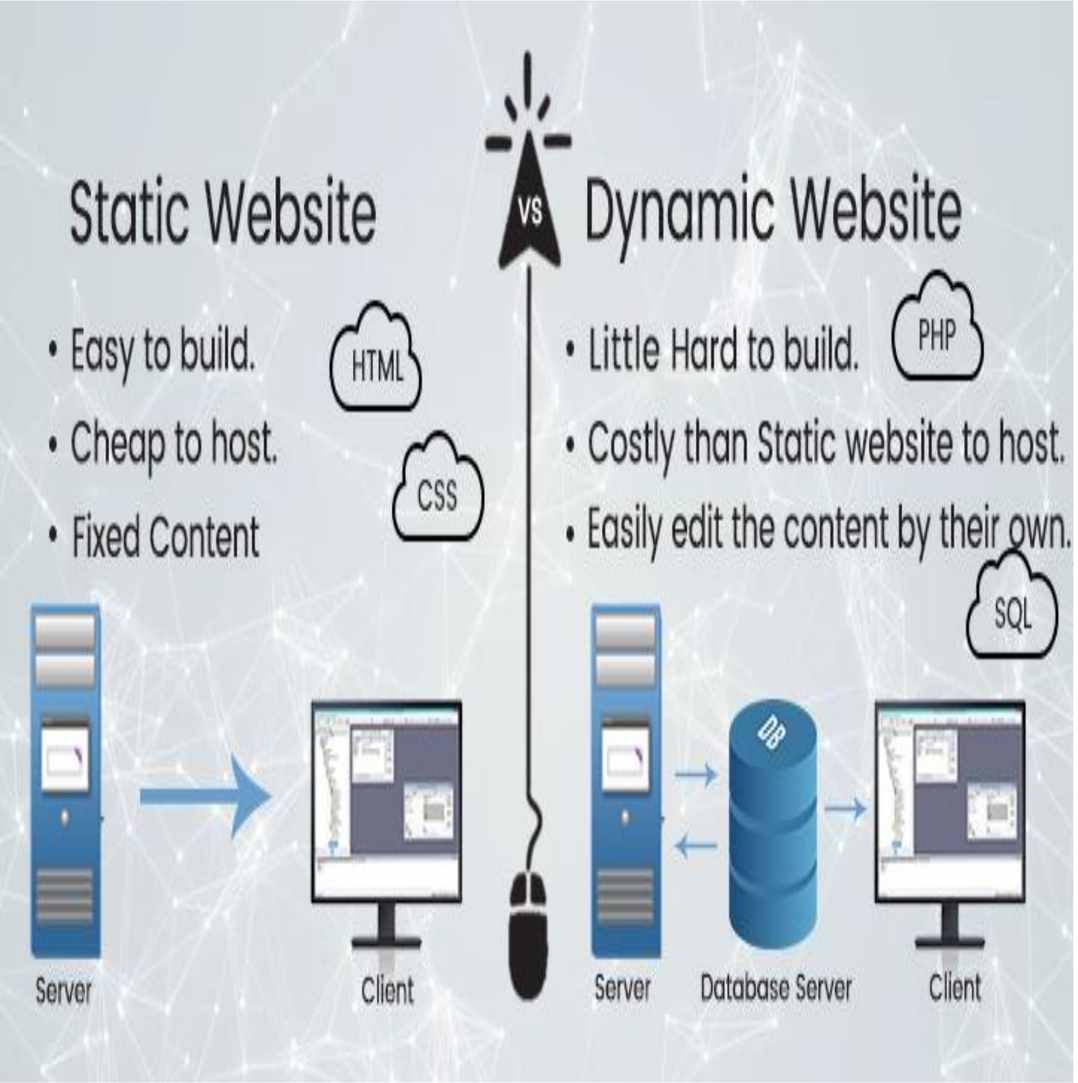
- Web development is the process of creating, building, and maintaining websites or web applications.
- Involves platforms accessible via the internet or a private intranet.
- Covers both backend functionality (server-side logic, databases) and frontend interface (design, user experience).
- Ensures the website or app is functional, visually appealing, and user-friendly.
- Combines technical coding skills with design principles for an optimal user experience.

What is a Website?

- **Definition: A website is a set of related web pages, multimedia content, and other resources identified by a common domain name (e.g., www.example.com).**
- **Components:**
  - **Web Pages**: Individual pages, often built with HTML, CSS, and JavaScript, that are displayed to the user.
  - **Web Server**: The server where the website files (HTML, images, scripts, etc.) are stored and served to users upon request.
  - **URL (Uniform Resource Locator)**: The address used to access a website (e.g., www.example.com).



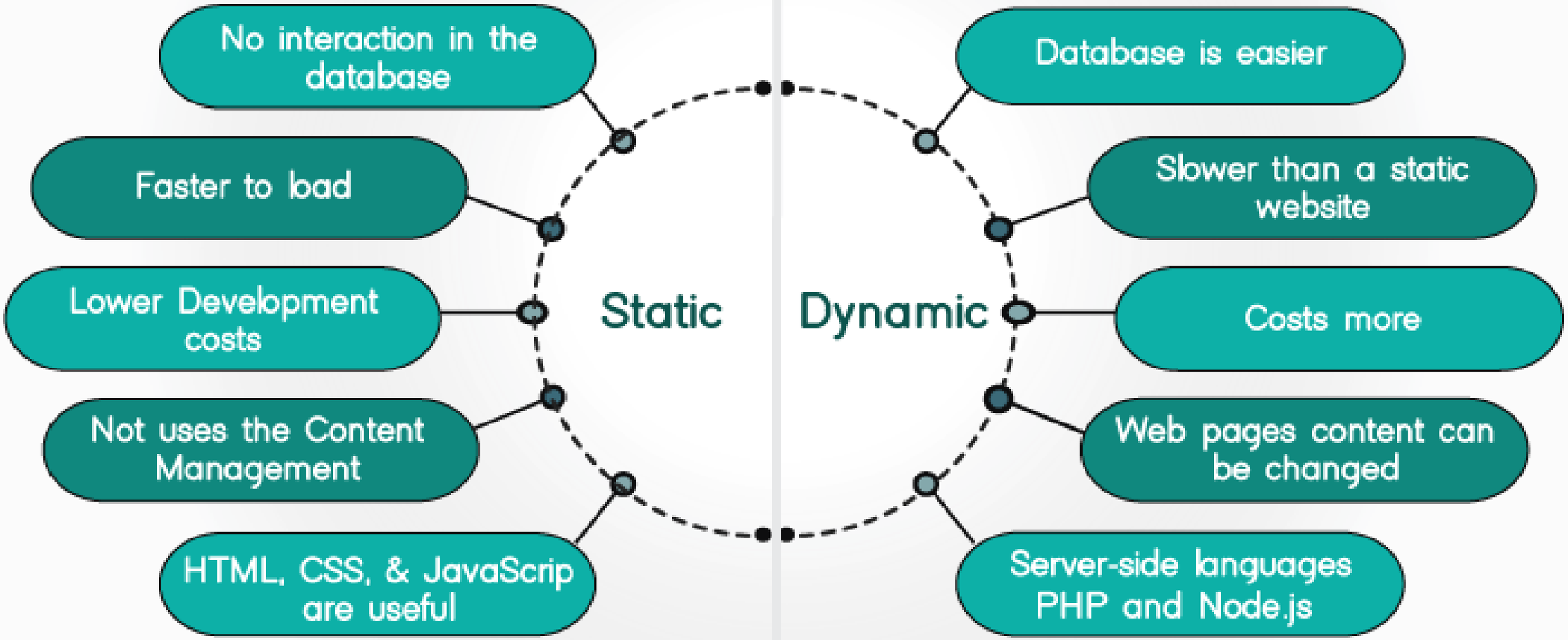
Types of Websites



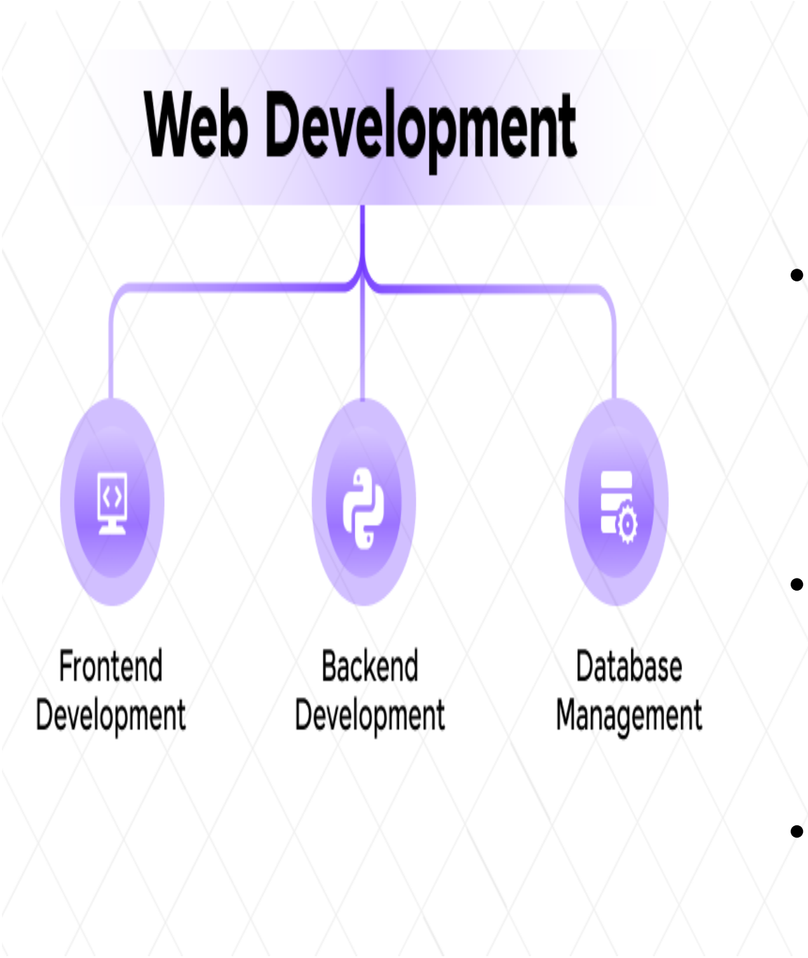
- **Static Websites: These are basic websites where the content doesn't change dynamically. It's like a digital brochure.**
  - Typically, only **HTML** and **CSS** are used.
  - Easy to build and host.

- **Dynamic Websites: These websites can change content dynamically based on user interaction or other data.**
  - Built with server-side programming languages like **PHP**, **Python**, **Node.js**, or frameworks like **Spring**, **Django**, etc.
  - They interact with databases (e.g., **MySQL**, **MongoDB**) to deliver personalized content.

Comparison of The Static & Dynamic Website



Key Areas of Web Development



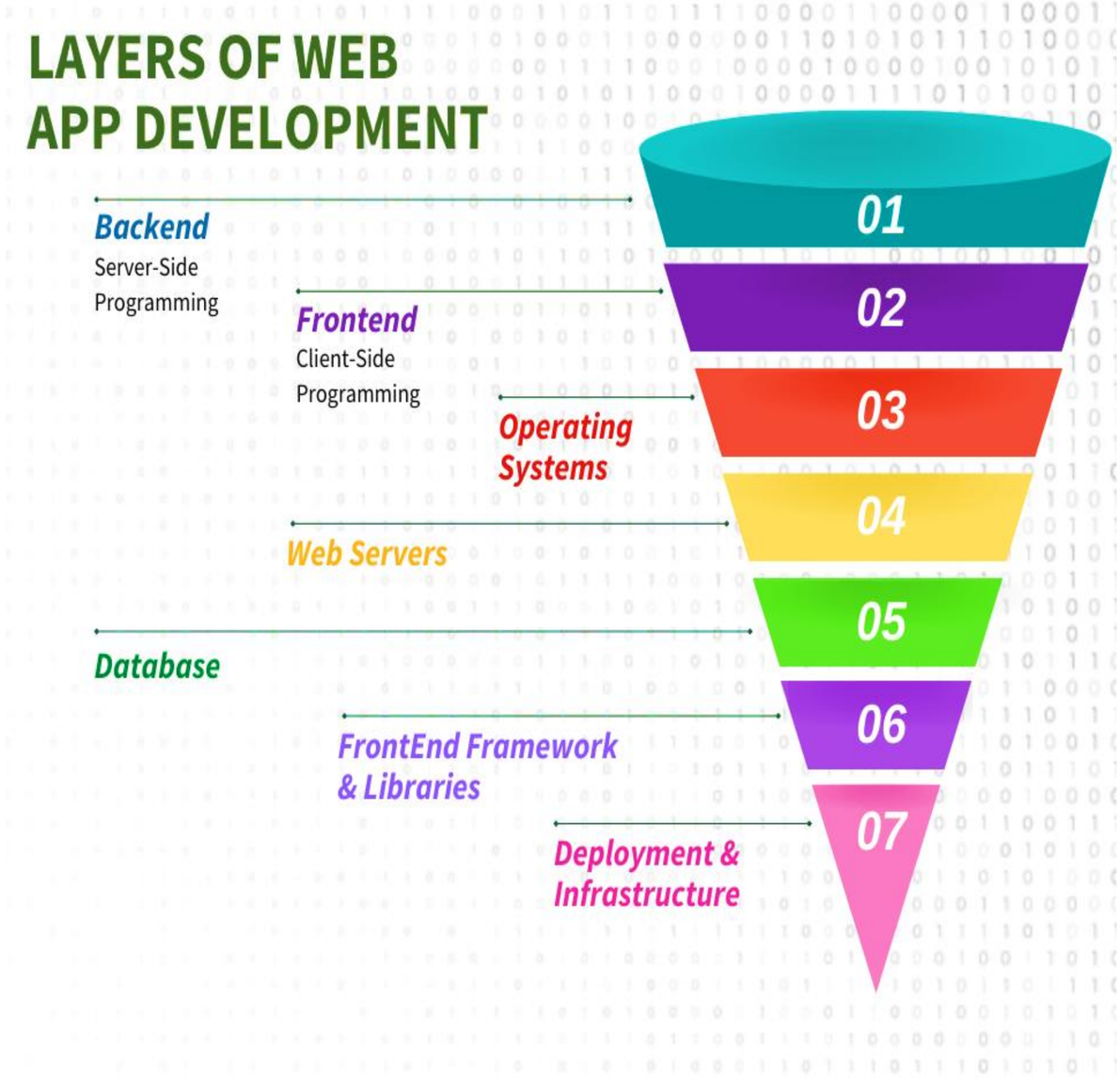
- **Frontend Development**
  - Focuses on the *client-side* (what users see and interact with).
  - Uses technologies like **HTML, CSS, JavaScript** and frameworks such as **React, Angular, Vue.js**.
  - Involves UI/UX design principles, responsive layouts, and accessibility.
- **Backend Development**
  - Handles the *server-side* operations, business logic, and database management.
  - Uses programming languages and frameworks like **Node.js, Python/Django, PHP/Laravel, Ruby on Rails**.
  - Ensures data processing, authentication, and API integrations work smoothly.
- **Database Management**
  - Organizes, stores, and retrieves application data.
  - Uses relational databases (**MySQL, PostgreSQL**) or NoSQL databases (**MongoDB, Cassandra**).
- **DevOps & Deployment**
  - Manages hosting, server configuration, continuous integration (CI), and deployment (CD).
  - Involves cloud platforms (**AWS, Azure, Google Cloud**) and automation tools.
- **Web Security**
  - Protects websites from threats like hacking, malware, and data breaches.
  - Involves HTTPS, encryption, authentication protocols, and regular security audits.

Best Practices

- **Plan & Define Requirements**
  - Target audience, objectives, core features, avoid scope creep.
- **Choose the Right Technology Stack**
  - Match to project goals, ensure scalability, security, and team expertise.
- **Focus on User Experience**
  - Conduct research, usability testing, responsive design.
- **Adopt Agile Methodology**
  - Iterative progress, feedback loops (Scrum, Kanban).
- **Ensure Robust Security**
  - Encryption, authentication, updates, audits, penetration testing.
- **Optimize Performance**
  - Reduce load time, use CDNs, browser caching, image optimization.
- **Implement Thorough Testing**
  - Unit, integration, and end-to-end testing; automate where possible.
- **Maintain Documentation**
  - Clear code, API, and process documentation for scalability & maintenance.

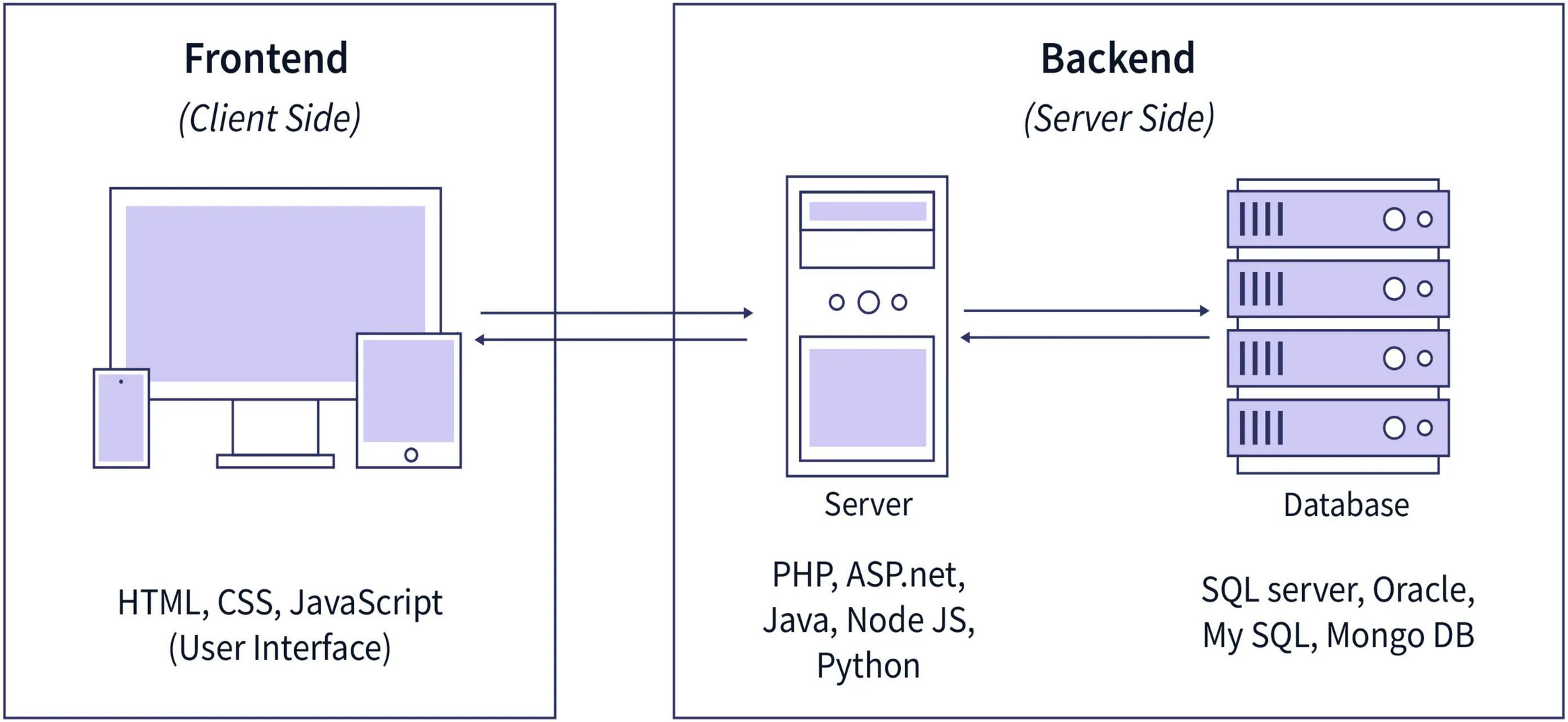
Technology Stack (Tech Stack)

- **Definition:**
  - A set of technologies, programming languages, frameworks, and tools used together to develop and run a software application.
  - Often includes frontend (client-side), backend (server-side), databases, and infrastructure tools.
  - Can be tailored to specific needs
    - e.g., web apps, mobile apps, AI solutions.
- **Common examples:**
  - **MERN Stack:** MongoDB, Express.js, React.js, Node.js.
  - **LAMP Stack:** Linux, Apache, MySQL, PHP.
  - **MEAN Stack:** MongoDB, Express.js, Angular, Node.js.
- **Choosing the right tech stack depends on project requirements, budget, team expertise, and scalability goals.**



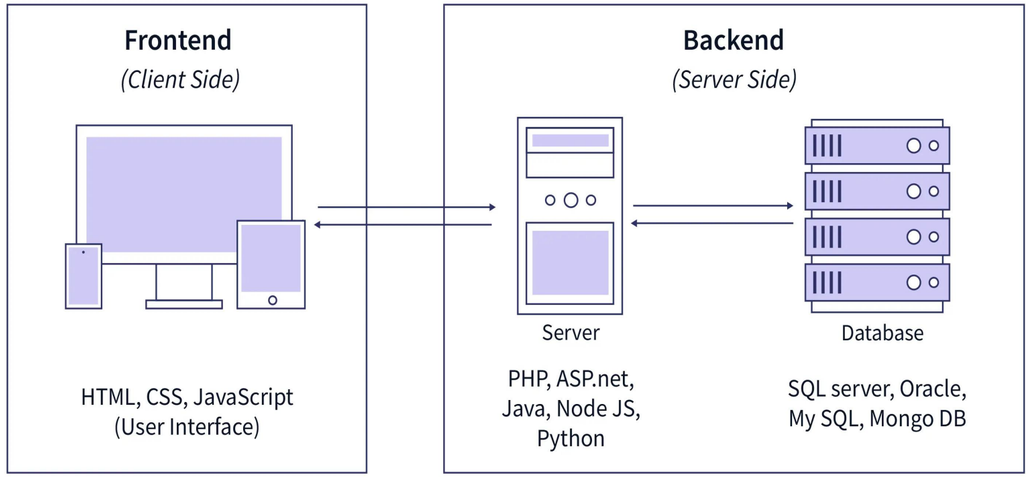
Web Development Stack

Full Stack Development



Client–Server Architecture

- **Definition: A network design model where tasks are divided between clients (requesters) and servers (providers).**
- **Client:**
  - End-user device (e.g., browser, mobile app) that sends requests.
  - Handles **user interface** and presentation logic.
  - Examples: Web browser requesting a webpage, mobile app fetching data.
- **Server:**
  - Central system that processes client requests and sends back responses.
  - Handles **business logic**, database operations, and data storage.
  - Examples: Web server, database server, application server.



- **Communication:**
  - Clients and servers communicate over a network (usually the Internet) using protocols like **HTTP/HTTPS**.
- **Request–Response Cycle:**
  - Client sends a request to the server.
  - Server processes the request.
  - Server sends back the response (data, files, HTML, JSON, etc.).
- **Advantages:**
  - Centralized control and management.
  - Easy to update and maintain.
  - Scalable (can add more servers for load handling).
- **Examples in Real Life:**
  - Gmail (browser = client, Google's servers = server). Online banking apps, E-commerce websites.

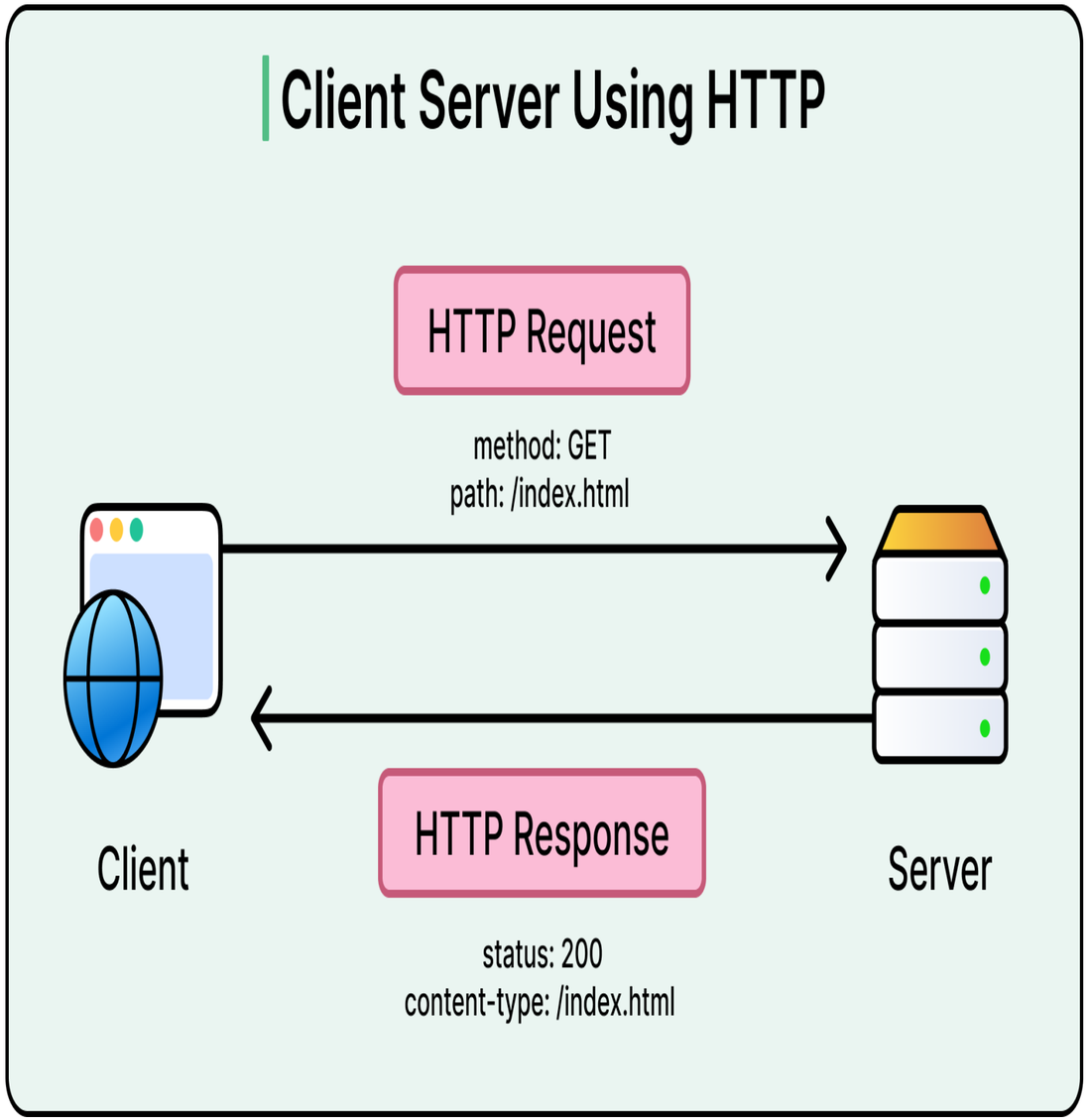
Approximately 1.8 billion active users, representing about **22% of the global population**.

As of **2024**, SBI had **500 million (50 crore)** customers.

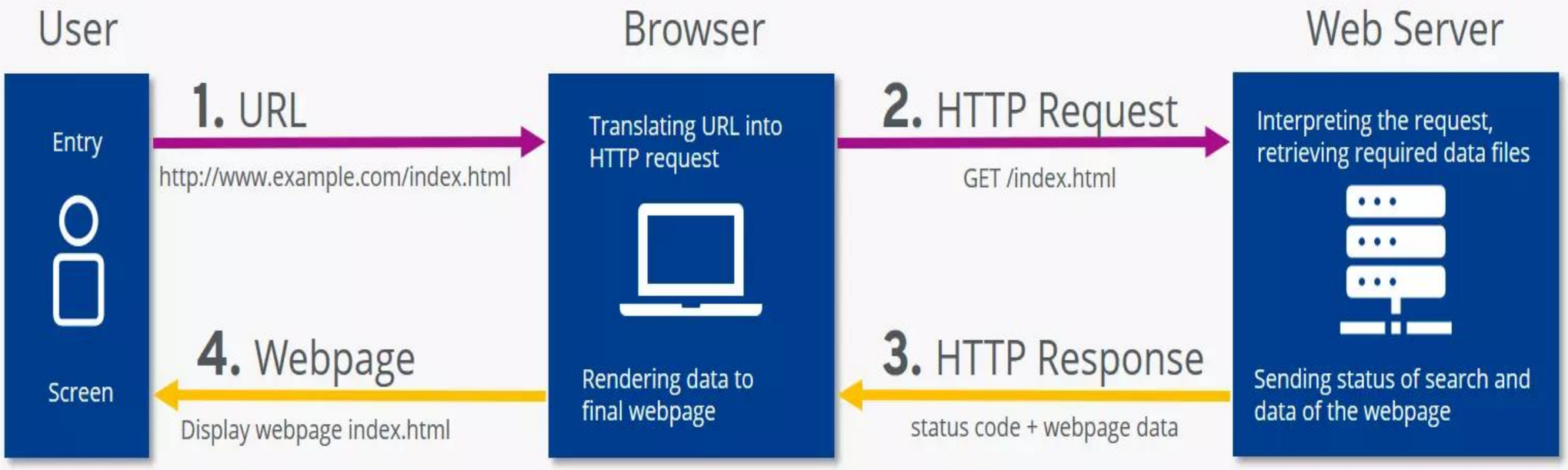
Amazon Active Users (Customers)
Over 310 million active users globally

HTTP (HyperText Transfer Protocol)

- **Definition:**
  HTTP is the protocol used for transferring hypertext (web pages) between a client (like a web browser) and a server over the internet.
- **Purpose:**
  It defines how messages are formatted, sent, and received so that web browsers and servers can communicate.
- **How it works (basic flow):**
  - The **client** sends an HTTP request to the server (e.g., asking for a webpage).
  - The **server** processes the request and sends back an HTTP response (e.g., HTML, images, data).



Communication process according to HTTP



HTTP (HyperText Transfer Protocol)

- **Key Characteristics:**
  - **Stateless:** Each request is independent; the server does not retain information between requests (unless cookies or sessions are used).
  - **Text-based:** Easy for humans to read and debug.
  - **Uses TCP/IP:** Usually operates over port **80** (HTTP) or **443** (HTTPS).
- **HTTPS = HTTP + SSL/TLS encryption for secure communication.**
- **Versions:**
  - **HTTP/1.1** – Most widely used, supports persistent connections.
  - **HTTP/2** – Faster with multiplexing (multiple requests at once).
  - **HTTP/3** – Uses QUIC protocol for better performance and lower latency.
  - QUIC stands for **Quick UDP Internet Connections.**
  - It's a transport layer network protocol developed by Google and later standardized by the IETF.
- **HTTPS:**
  - Secure version of HTTP using **SSL/TLS encryption** to protect data during transfer.
- **Common HTTP Methods:**
  - **GET:** Retrieve data.
  - **POST:** Send data to the server.
  - **PUT:** Update existing data.
  - **DELETE:** Remove data.

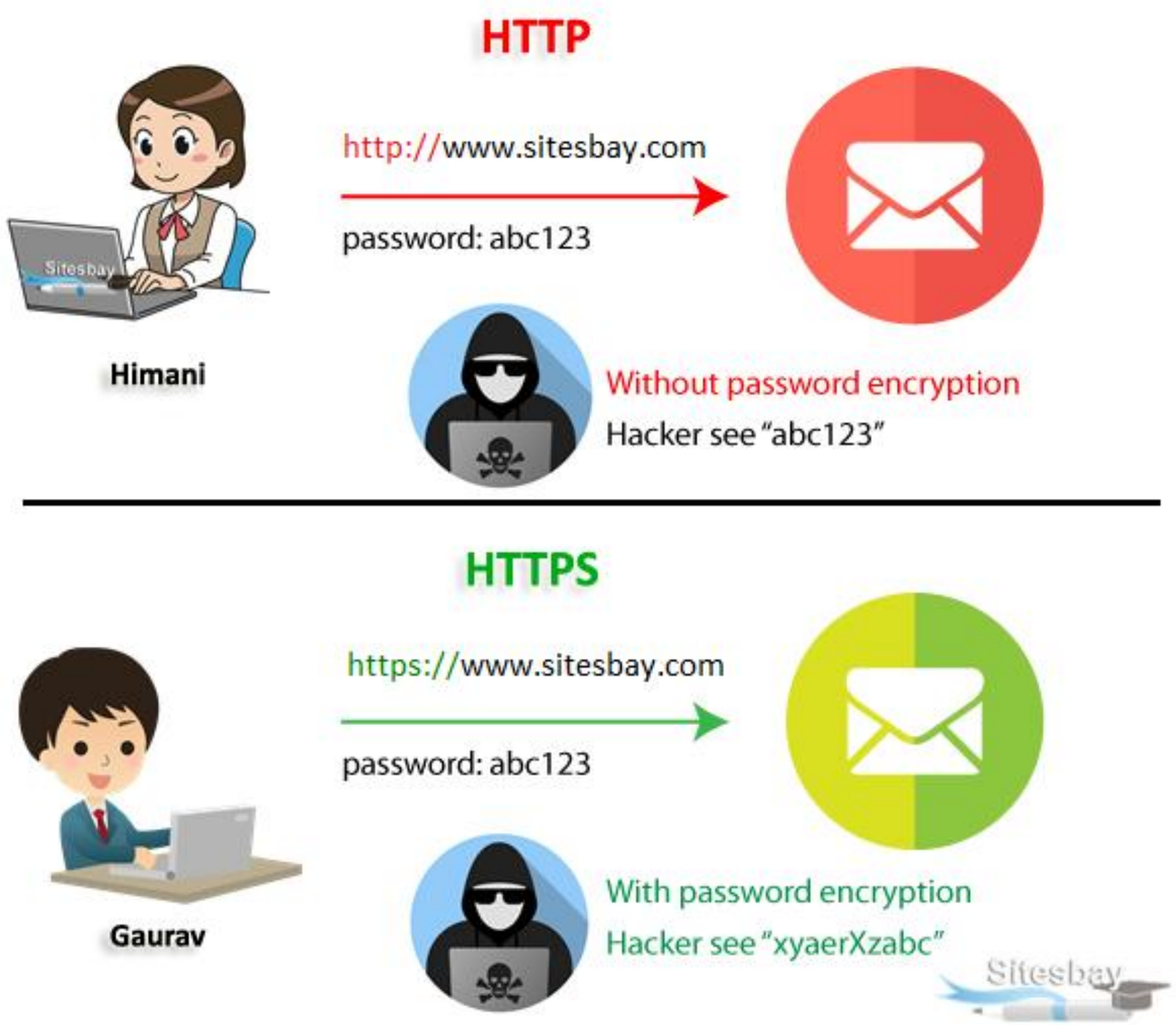


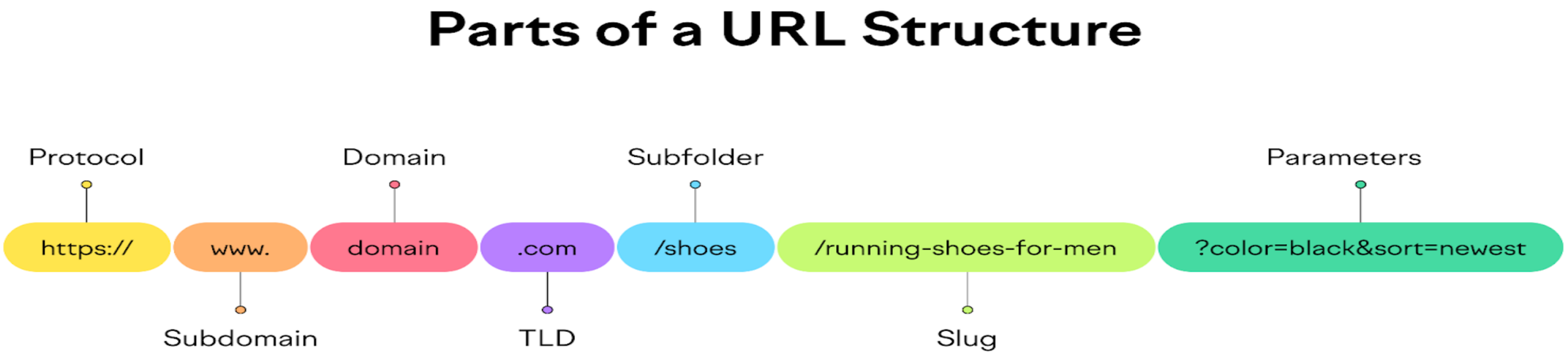
Fig: Difference Between http and https

HTTP Request Methods

GET	POST	PUT	DELETE	PATCH	HEAD
retrieve data from server	add data to an existining file or resource	update(replace) an existing file or resource in server	delete data from server	update a resource partially (modify)	retrieve the resource's headers

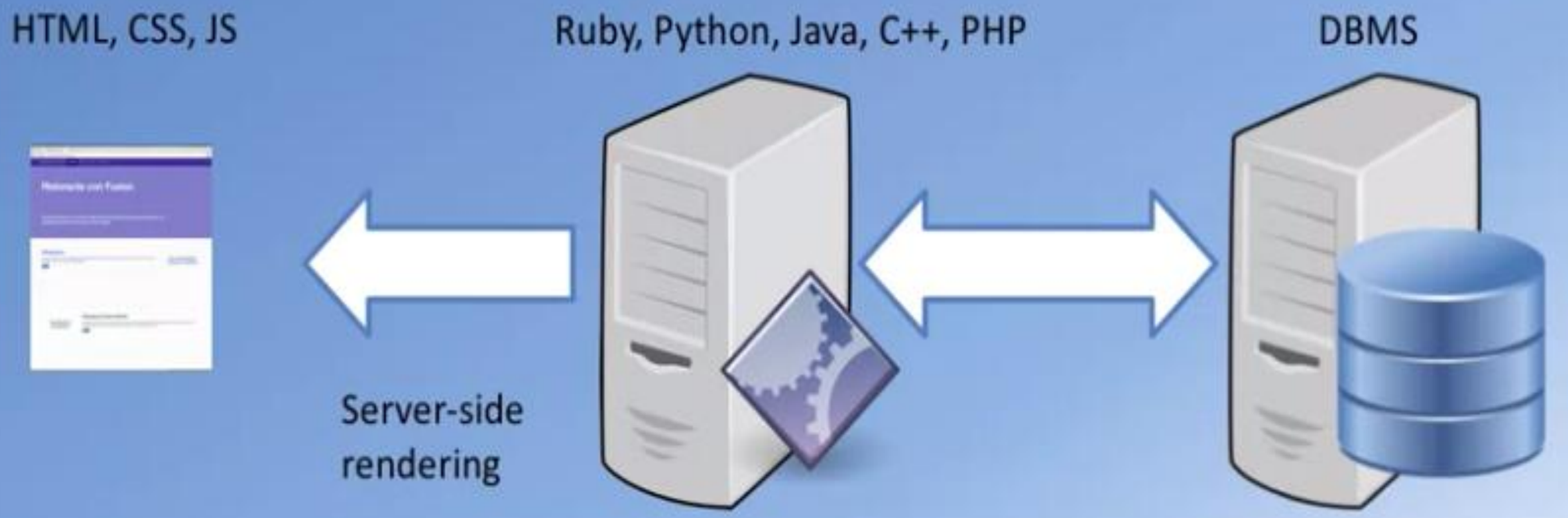
- **CONNECT** is used to open a two-way socket connection to the remote server;
- **OPTIONS** is used to describe the communication options for specified resource;
- **TRACE** is designed for diagnostic purposes during the development.
- **HEAD** retrieves the resource's headers, without the resource itself.

URL (Uniform Resource Locator)



- **Identifies the location of a resource on the internet.**
- **Structure:**
  - protocol://domain:port/path?query#fragment
  - Example: https://example.com:443/products?id=10#details
  - Protocol: http/https
  - Domain: example.com
  - Port: 80/443 (default) or custom
  - Path: /products
  - Query: ?id=10
  - Fragment: #details

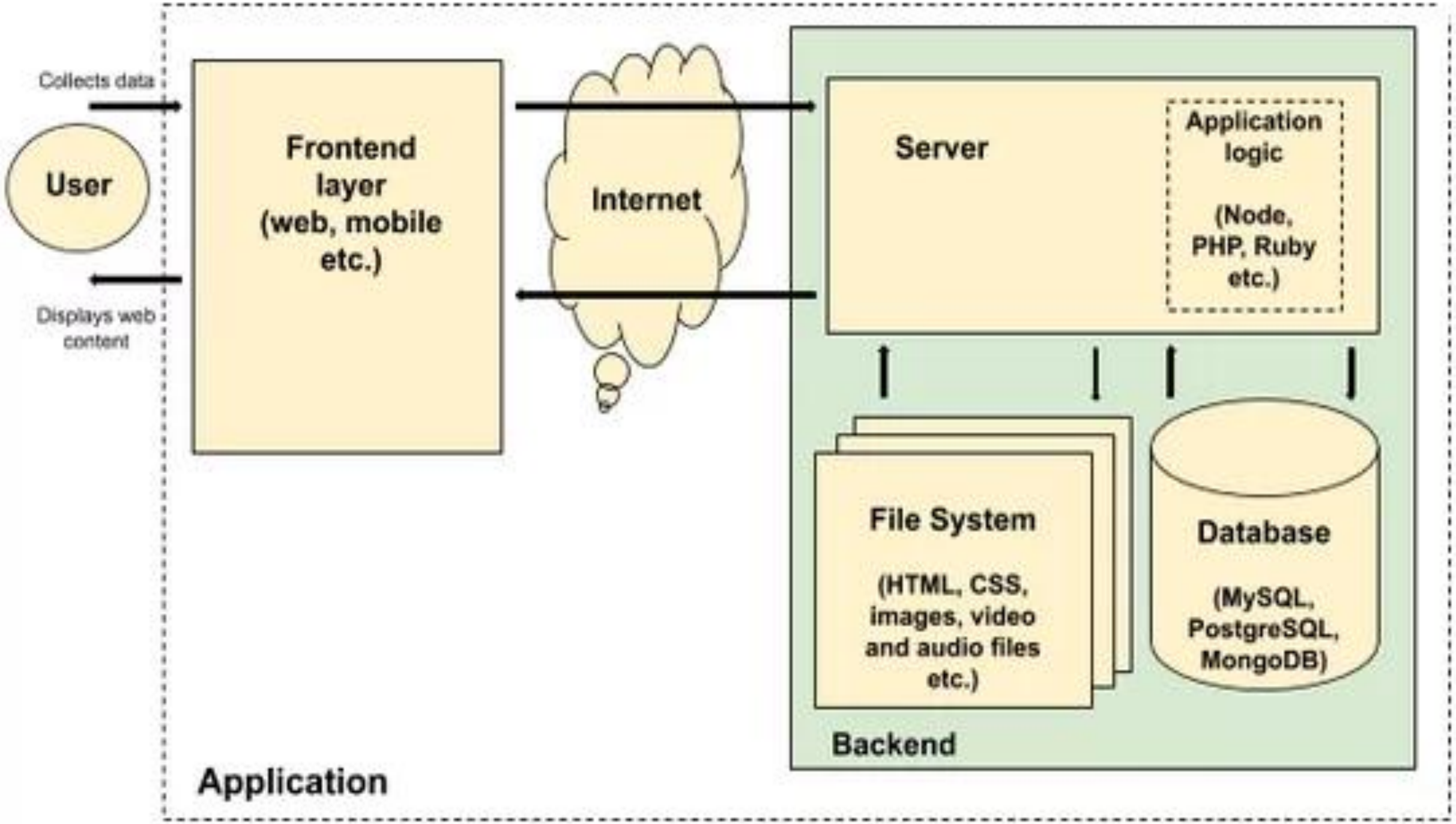
Traditional Web Development



Presentation layer

Business Logic layer

Data Access layer


Collects data
User
Displays web content
Frontend layer (web, mobile etc.)
Internet
Server
Application logic
(Node, PHP, Ruby etc.)
File System
(HTML, CSS, images, video and audio files etc.)
Database
(MySQL, PostgreSQL, MongoDB)
Backend
Application

Hands-on Practice

HTML
Hypertext Markup Language

Create the structure

- Controls the layout of the content
- Provides structure for the web page design
- The fundamental building block of any web page

CSS
Cascading Style Sheet

Stylize the website

- Applies style to the web page elements
- Targets various screen sizes to make web pages responsive
- Primarily handles the "look and feel" of a web page

Javascript

Increase interactivity

- Adds interactivity to a web page
- Handles complex functions and features
- Programmatic code which enhances functionality

Thank You

FOR ALL YOUR ATTENTION