**FULL STACK WEB DEVELOPMENT**

**What is front-end?**  
A Front-End Developer is someone who creates websites and web applications.

The difference between Front-End and Back-End is that Front-End refers to how a web page looks, while back-end refers to how it works. Front-End as **client-side** and Back-End as **server-side**.

The basic languages for Front-End Development are HTML, CSS, and JavaScript.

Client Software (Front-End)

The basic languages of Front-End Development are:

How the html,css and javascript works:  
**1. HTML (Structure):**  
• HTML is like the skeleton of a webpage. It creates the basic structure, like headings, paragraphs, images, and links.  
• Example: It places text or images on the page.  
**2. CSS (Style):**  
• CSS makes the webpage look good. It adds colors, fonts, spacing, and controls the layout (how things are arranged).  
• Example: It changes the color of text, makes buttons look nicer, or arranges things in rows or columns.  
**3. JavaScript (Interactivity):**  
• JavaScript makes the webpage interactive. It allows actions like clicking buttons, showing messages, or updating information without reloading the page.  
• Example: When you click a button, JavaScript can show a message or change something on the page.  
How this three works :  
• HTML builds the page structure.  
• CSS makes it look pretty.  
• JavaScript adds interaction (things that happen when you click or type).

**client** In computing, a client is a device or software that interacts with a server to access services or resources. It’s essentially the "user" side of a client-server model, where the client makes requests to a server, and the server provides the requested data or service. Client-Server Model In the client-server model, the client makes requests, and the server processes them and responds. For example: Client (You): The person who uses the web browser to visit a website. Server: The website’s server, which stores and serves the web pages. Real-Life Example:  
Imagine a restaurant: The Client: You (the customer) ordering food. The Server: The waiter, who takes your order to the kitchen and brings the food back to you. The Kitchen: The system (server) preparing the food based on the order.

**Backend Development**

Backend Development involves the logic, database, and other operations that are built behind the scenes to run the web servers efficiently. Backend Development refers to the server-side development of the web application. It is the part of the application where the server and database reside and the logics is build to perform operations. It includes the main features and functionalities of the application on the server. Programming languages for the backend are mainly Node. JS (for JavaScript), Django (for Python), Spring (Java), etc.

**Difference between the frontend and backend**

*Front-end works on UI and is heavily based on designing to make your website as attractive as possible while the back-end deals with all the complicated and messy stuff that actually makes your website run (like database operations, user authentication, application logic, etc)*  
**What is the Cloud?** The cloud is like a big storage and computing space you can access online. Instead of storing files, running programs, or processing data on your personal device, you use the internet to access powerful systems that do this for you.

The most used cloud services are

Example’s: • Amazon Web Services (AWS)  
• Microsoft Azure  
• Google cloud  
• Dropbox

**API’s**  
An API (Application Programming Interface) is a set of rules and tools that allow two software applications to communicate and exchange data with each other. It acts like a bridge, enabling one application to interact with another without needing to know the internal workings of the other app. Real-Life Examples of APIs  
Google maps Api’s Payment api’s Social media api’s Weather api’s Messaging api’s

**IP address**

An **IP address** (Internet Protocol address) is a unique numerical identifier assigned to each device connected to a network that uses the Internet Protocol for communication. It allows devices to locate and communicate with each other over the internet or a local network..

**Examples of IP Address Usage:**

1. When you enter a URL (e.g., www.google.com), your device translates the domain name into an IP address using DNS (Domain Name System) to locate the server.
2. Your router assigns private IP addresses to devices in your local network while using a public IP address to connect to the internet.

**browser**

 A browser (short for web browser) is a software application that allows you to access and interact with websites on the internet. It acts as a gateway between you and the web, displaying content like text, images, videos, and more, by interpreting website code (HTML, CSS, JavaScript, etc.).  
Example’s: • Google Chrome: Fast, popular, and widely used.  
• Mozilla Firefox: Privacy-focused with open-source origins.  
• Microsoft Edge: Integrated with Windows and optimized for performance.  
• Safari: Designed for Apple devices.  
• Opera: Features built-in ad blockers and free VPN.

**Server**  
A server is a powerful computer that stores, processes, and delivers information to other devices (like your phone or laptop).  
Example’s:  
• HTTP Server  
• Local Server  
  
**local storage:** Local storage refers to a way of storing data on a device or within a browser, so that the data can persist even after the user closes the app or web page. It's used to save information that needs to be available locally (on the user's device) without constantly needing to retrieve it from a server..  
 **Static and Dynamic web pages?**  
**Static web-page:** A static web page is a simple page where the content remains the same every time it is loaded, meaning it doesn’t change or respond to user interactions. It’s ideal for sites that don’t require frequent updates or customization, like informational websites or blogs.  
**Dynamic web-page:** A dynamic web page is one where the content is generated or updated in real-time based on user input, preferences, or other variables. It often involves interaction with a server or database, making it suitable for websites like social media, online stores, or news sites, where content frequently changes or needs to be personalized.  
**Why we only store the database in cloud?** We store databases in the cloud because it offers scalability, flexibility, remote access, automatic backups, and reduced infrastructure costs, allowing businesses to focus on core activities while ensuring data is secure and highly available.  
**Why we can’t store the data in server?** We don't store data directly on the server or client because of issues:  
1. Server Issues: The server might run out of space or become slow if it stores too much data. If the server crashes, data could be lost.  
2. Client Issues: Devices (like phones or computers) don't have enough space to store lots of data. It's harder to keep data safe and secure on a device. Data on one device can't easily be shared with others.  
 **Uses of a Database?** Databases store data in a safe, organized place, and are easier to manage.  
• They allow quick access to data, even if there’s a lot of it. Databases also make it easier to back up and protect data.  
**Simple Analogy:** Server storage is like a messy drawer that can get full. Client storage is like putting everything in your personal bag, which can be lost or damaged. Databases are like an organized filing system where everything is safe and easy to find. Databases are the best way to store data because they keep it organized, secure, and easy to access.

**Domain**

Software is like a vast universe with different galaxies, each offering unique experiences and challenges. Here are some of the most prominent domains in software:

* **Game development**
* **Web development**
* **Block chain**
* **Data science**
* **Artificial Intelligence and Machine Learning**
* **Mobile app development**
* **Cybersecurity**
* **Cloud computing**

CRUD:

CRUD stands for Create, Read, Update, and Delete. It's a set of operations used to manage data in databases. CRUD operations are a fundamental part of computer programming, database design, and application development.

**Create :**

Inserting new data into a database

Using the "POST" method to create a new record

**Read:**

Retrieving data from a database

Using the "SELECT" function in SQL or the "GET" HTTP method in RESTful APIs

Using search criteria to retrieve specific records

**Update:**

Modifying existing data in a database

Using the "PUT" or "PATCH" method to update a record

Changing data based on criteria, such as changing a customer's address or a product's price

**Delete:**

Removing data from a database

Using the "DELETE" method to remove a record

Performing a hard delete to remove a record completely, or a soft delete to flag a record but leave it in place

CRUD operations are essential for building efficient, secure, and scalable applications.

**Web applications**

A web application is a software program that runs on a web server and is accessed through a web browser over the internet or an intranet. Unlike traditional desktop applications, web applications do not need to be installed on a user's device; instead, they can be used from any device with a compatible browser and internet connection.

**Examples of Web Applications:**

* **E-commerce platforms**: Amazon, Flipkart.
* **Social Media**: Facebook, Twitter, Instagram.
* **Email Services**: Gmail, Outlook.
* **Online Banking**: Banking portals like ICICI or SBI's net banking.
* **Collaboration Tools**: Google Docs, Slack.

**Standalone applications**

Standalone applications are software programs that operate independently without requiring external components, network connectivity, or integration with other systems to function. They are self-contained and typically include all the necessary components for execution.

**Characteristics of Standalone Applications:**

1. **Independence**: Does not rely on external systems or the internet.
2. **Platform-Specific**: Designed for a specific operating system or platform (e.g., Windows, macOS, Linux).
3. **Installation**: Typically installed on a local machine and runs directly from the device.
4. **Performance**: Often optimized for the specific hardware and software environment.
5. **No Network Dependency**: Can function entirely without internet or external servers.

**Examples of Standalone Applications:**

* Media players (e.g., VLC Media Player)
* Text editors (e.g., Notepad, Sublime Text)
* Games (e.g., single-player PC games)
* Utility software (e.g., calculators, compression tools like WinRAR)

**Advantages:**

1. **Offline Functionality**: Can work without an internet connection.
2. **Faster Response**: Not limited by network latency or bandwidth.
3. **Security**: Reduced risk of online threats since it doesn't require network access.

**Disadvantages:**

1. **Limited Collaboration**: Cannot share data in real-time without external integration.
2. **Updates**: Manual updating is often required unless a feature is added to connect to update servers.
3. **Resource Intensive**: May require significant local resources (CPU, memory, storage).