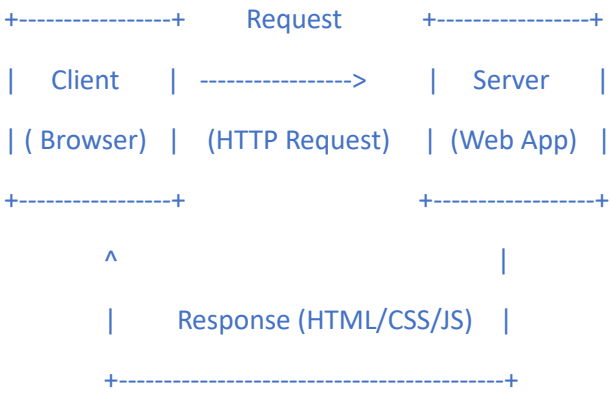
1. Explain the difference between frontend, backend, and full-stack development with Suitable real-world examples.

Difference between Frontend, Backend, and Full Stack Development

Type	Description	Real-World Example
Frontend	The visible part of a website that users interact with — built using HTML, CSS, JavaScript.	When you open Amazon, the layout, product images, buttons, and search bar are frontend.
Backend	The server-side part that processes data, runs logic, and interacts with the database.	When you click "Add to Cart", backend (Node.js/Python/Java) stores the data in a database.
Full Stack	Developer who works on both frontend and backend sides.	A developer who designs the webpage (frontend) and writes APIs to handle login/register (backend).

2. Create a simple diagram showing how the client-server model works in web architecture.



3. Describe how a browser requests and displays a web page from a web server.

Step-by-Step Process:

1. User enters a URL (e.g. www.example.com).
2. Browser sends a **DNS request** to find the server's IP address.
3. Browser sends an **HTTP/HTTPS request** to the server.
4. Server receives request → finds the requested HTML file.
5. Server sends **HTML + CSS + JS** back to browser.
6. Browser **renders** HTML (structure), **CSS** (style), and **JS** (behavior).

7. User sees the web page displayed.

4. Identify and list the tools required to set up a web development environment. Explain the purpose of each.

Tool	Purpose
VS Code / Sublime Text	Code editor for writing HTML, CSS, JS
Web Browser (Chrome/Edge/Firefox)	To run and test your website
Live Server Extension	Instantly preview HTML changes
Node.js	Backend JS runtime environment
Git & GitHub	Version control and code collaboration
XAMPP / WAMP / Localhost	Local web server setup for backend testing

5. Explain what a web server is and give examples of commonly used servers.

A **web server** is software or hardware that delivers web pages to clients (browsers) over HTTP or HTTPS.

Examples:

- **Apache HTTP Server** → Open-source, widely used
- **Nginx** → Fast, handles high traffic
- **Microsoft IIS** → Windows-based web server
- **Node.js HTTP Server** → Used for JavaScript-based apps

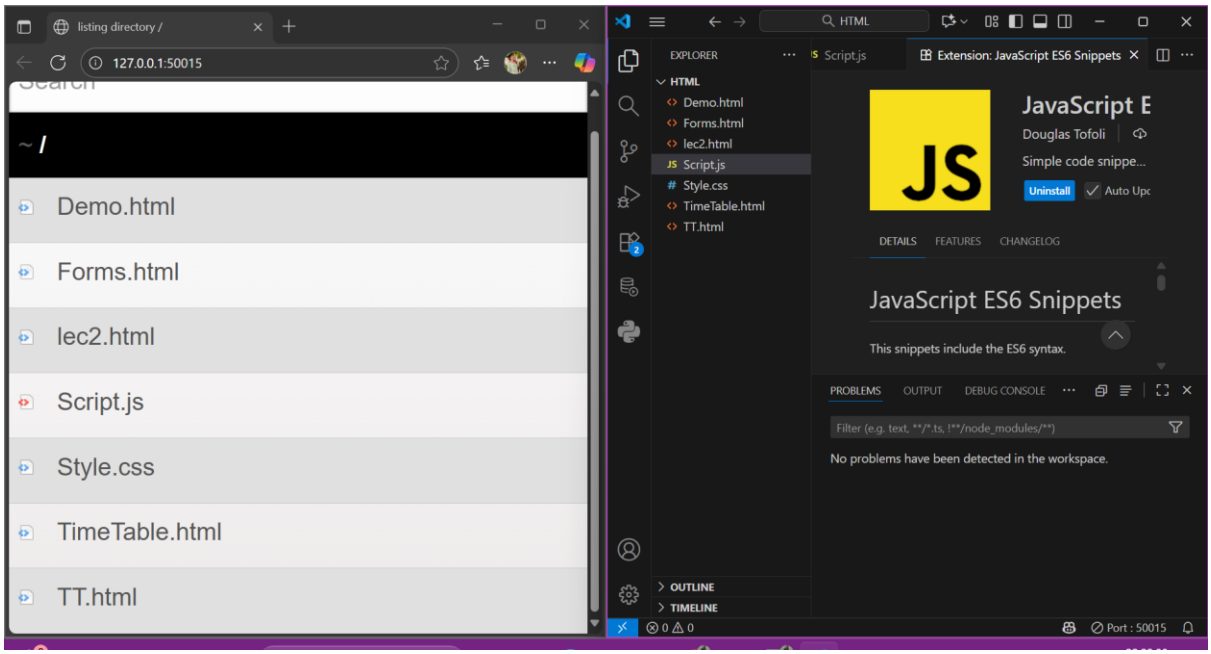
Real Example:
When you visit YouTube, your browser sends a request to Google's servers → the web server returns the page.

6. Define the roles of a frontend developer, backend developer, and database administrator in a project.

Role Responsibilities

Frontend Developer	Builds user interface (HTML, CSS, JS, React)
Backend Developer	Manages logic, APIs, databases, authentication
Database Administrator (DBA)	Designs and maintains database (MySQL, MongoDB, etc.), ensures data security and backups

7. Install VS Code and configure it for HTML, CSS, and JavaScript development. Take a screenshot of the setup.



8. Explain the difference between static and dynamic websites. Provide an example of each.

Static vs Dynamic Websites

Type	Description	Example
Static Website	Content remains same for all users. No server-side processing. Built with HTML/CSS only.	A personal portfolio or company info page.
Dynamic Website	Content changes based on user interaction or database data. Built using backend (Node.js, PHP, etc.)	Facebook, Amazon, Instagram — content changes dynamically.

9. Research and list five web browsers. Explain how rendering engines differ between them.

Five Web Browsers and Their Rendering Engines

Browser	Rendering Engine
Google Chrome	Blink
Mozilla Firefox	Gecko
Safari (Apple)	WebKit
Microsoft Edge	Blink (formerly EdgeHTML)
Opera	Blink

Difference: Rendering engines interpret HTML, CSS, JS differently — e.g., **Blink** (Chrome/Edge) is faster and supports latest standards, while **WebKit** (Safari) has its own optimizations for macOS/iOS.

10. Draw a labeled diagram showing the basic web architecture flow — client, server, database, and APIs.

[CLIENT (Browser)]

|

| HTTP Request

v

[SERVER (Backend Logic)]

|

| Query

v

[DATABASE (Stores Data)]

^

| Response (via APIs)

|

[API Layer (connects Server & DB)]

Explanation:

- Client sends request → Server processes it → API fetches data from Database → sends back to client.