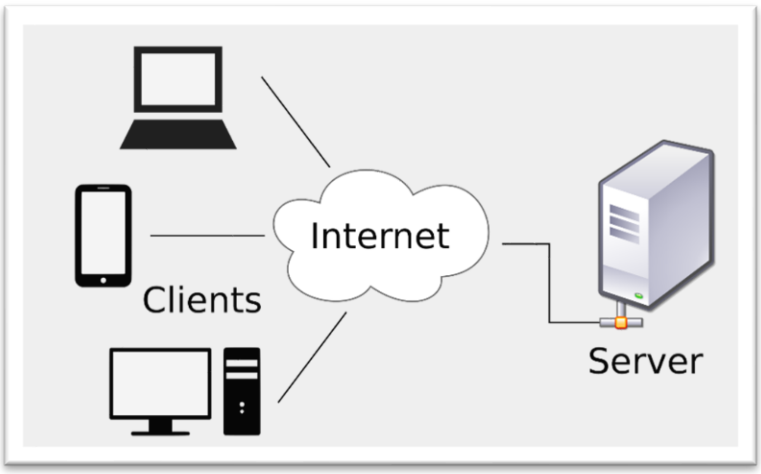
ASSIGNMENT: INTRO TO WEB DEV

1. **Explain the difference between frontend, backend, and full-stack development with suitable real-world examples.**
   - *Frontend Development*
     - ➢ The frontend is the part of a website or app that users directly interact with - what you see on the screen.
     - ➢ Languages/Frameworks: HTML, CSS, JavaScript, React, Angular, etc.
     - ➢ Example: When you visit Netflix, the homepage layout, movie thumbnails, buttons, and animations are all frontend.
   - *Backend Development*
     - ➢ The backend is the behind-the-scenes part that handles data, logic, and communication with the database or server.
     - ➢ Languages/Frameworks: Python (Django, Flask), Java (Spring Boot), Node.js, MySQL, MongoDB, etc.
     - ➢ Example: When you log into Netflix, the backend checks your username and password, fetches your watchlist, and streams content from the server.
   - *Full-Stack Development*
     - ➢ Full-stack developers work on both frontend and backend - they handle the entire application from user interface to database.
     - ➢ Example: A food delivery app developer who designs the app's interface (frontend) and writes the code that manages orders and user accounts (backend) is a full-stack developer.

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.**

When you enter a URL into your browser, it initiates a series of steps to retrieve and display the web page:

i. The browser sends a **Hypertext Transfer Protocol (HTTP)** request to the web server that hosts the website, asking for the web page (usually an HTML file).
ii. The web server receives & process the request and sends back an HTTP response back to the browser. This response contains the requested files, including the main **HTML** file, as well as links to other resources like **CSS** stylesheets, **JavaScript** files, and images.
iii. Once the browser has received all the files, it renders the HTML, CSS, and JavaScript. The HTML provides the structure, the CSS dictates the visual styling, and the JavaScript adds interactivity. The browser then assembles these elements and displays the complete, interactive web page to the user.

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

Tools Required for a Web Development Environment:

1) **Code Editor / IDE** – It is used to write and edit code efficiently. A good editor provides features like syntax highlighting, auto-completion, and extensions to streamline development. (Example: VS Code, Sublime Text).
2) **Web Browser with Developer Tools**: You need a browser (like Chrome, Firefox, or Edge) to view your work and interact with it as a user would. Browsers come with built-in **Developer Tools**, which are essential for debugging, inspecting page elements (HTML and CSS), and monitoring network requests.
3) **Version Control System** – Tracks code changes and manages collaboration and revert to previous versions if needed. (e.g., Git, GitHub).
4) **Terminal / Command Line** – Used to run commands, install packages, manage projects and interact with other development tools and frameworks.
5) **Local Web Server**: For some projects, especially with backend development, you'll need to run a local server to test your application. Tools like **Live Server** (VS Code extension) or frameworks like Node.js can handle this.

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

A **web server** is computer program that stores, processes, and delivers web pages to users when requested through a browser using **HTTP/HTTPS** protocols. It handles client requests and sends back the required files like HTML, CSS, JavaScript, or images.

Commonly used web servers include:

- **Apache HTTP Server**: One of the oldest and most popular open-source web servers, known for its flexibility and wide range of modules.
- **Nginx**: A high-performance web server often used for serving static content and as a reverse proxy or load balancer.
- **Microsoft Internet Information Services (IIS)**: A web server software included with Microsoft Windows that's widely used in corporate environments.
- **Node.js**: While technically a runtime environment, it has a built-in HTTP module that allows it to function as a web server, making it popular for building modern web applications.

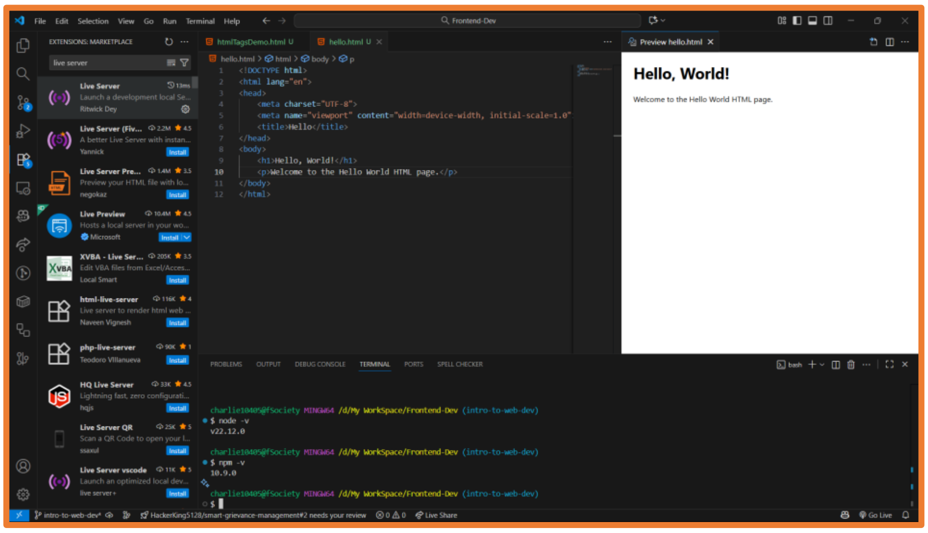
6. **Define the roles of a frontend developer, backend developer, and database administrator in a project.**

   i. Frontend Developer: Designs & builds the user interface(UI) of a website or app. They work with HTML, CSS, and JavaScript to ensure the site looks good, is responsive, and provides a smooth user experience.

   ii. Backend Developer: Handles the server-side logic of the application. They work with databases, APIs, and server frameworks (like Django, Node.js) to process data and connect the frontend with the database.

   iii. Database Administrator: Manages and maintains the database system. Their job includes creating, organizing, securing, and backing up data, ensuring that the database runs efficiently and reliably.

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



```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Hello</title>
</head>
<body>
    <h1>Hello, World!</h1>
    <p>Welcome to the Hello World HTML page.</p>
</body>
</html>
```

Hello, World!

Welcome to the Hello World HTML page.

```
charlie10405@fSociety MINGW64 /d/My WorkSpace/Frontend-Dev (intro-to-web-dev)
$ node -v
v22.12.0

charlie10405@fSociety MINGW64 /d/My WorkSpace/Frontend-Dev (intro-to-web-dev)
$ npm -v
10.9.0

charlie10405@fSociety MINGW64 /d/My WorkSpace/Frontend-Dev (intro-to-web-dev)
$
```

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

A **static website** consists of fixed, pre-built files (HTML, CSS, images) that are sent to the user exactly as they are stored on the server. The content doesn't change based on user input or other factors.

- **Example**: A simple personal portfolio or a small business landing page with unchanging content.

A **dynamic website** generates content "on the fly" using server-side processing and databases. The content can change based on user interactions, time, or other conditions, providing a personalized and interactive experience.

- **Example**: An e-commerce sites like **Amazon**, , where product listings, prices, and user data change dynamically.

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

   i. **Google Chrome:** Uses Blink; fast, modern, supports most web standards.

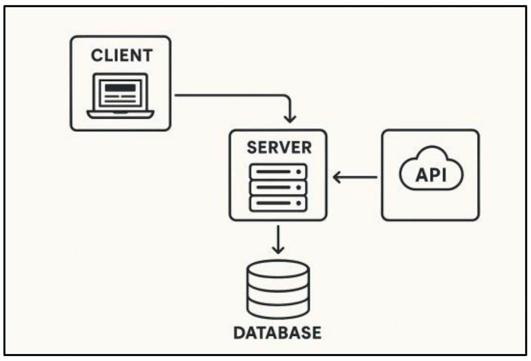
   ii. **Mozilla Firefox:** Uses Gecko; independent engine, emphasizes privacy and open standards.

   iii. **Microsoft Edge:** Uses Blink (Chromium-based); compatible with Chrome extensions.

   iv. **Safari:** Uses WebKit; optimized for Apple devices, energy-efficient, sometimes slower with heavy JS.

   v. **Opera:** Uses Blink; similar to Chrome but includes extra features like built-in VPN.

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


CLIENT
SERVER
API
DATABASE