# Assignment 1: Introduction to Web Development

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

Frontend development deals with the part of a website that users interact with directly. It includes everything that users experience visually in their browser — layout, colors, fonts, buttons, etc. Technologies used include HTML, CSS, and JavaScript. For example, the design of Amazon’s product page is frontend.  
  
Backend development focuses on the server-side logic and database management. It handles data storage, user authentication, and communication between the server and frontend. Technologies used include Node.js, Python, Java, and databases like MySQL or MongoDB. For example, when you log in to Instagram, the process that checks your credentials runs on the backend.  
  
Full-stack development combines both frontend and backend. A full-stack developer can build both the client and server parts of a web application. For example, a full-stack developer could build a complete web app like a blogging site using React (frontend) and Node.js with MongoDB (backend).

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

Diagram Description:  
  
Client (Web Browser) → sends request → Web Server → processes request → Database → sends data → Web Server → sends response → Client (Displays Web Page).

+-------------------+ +-------------------+

| CLIENT | | SERVER |

| (Web Browser) | | (Web Server, DB) |

+-------------------+ +-------------------+

| ^

| Request (e.g., URL) | Response (e.g., HTML, Data)

|--------------------------->|

| |

|<---------------------------|

| |

| Process & Display | Process Request & Retrieve Data

| (Render Webpage) | (from Database if needed)

| |

+-------------------+ +-------------------+

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

When a user enters a website URL, the browser sends an HTTP request to the web server hosting that site. The server processes the request, retrieves the appropriate HTML, CSS, and JavaScript files, and sends them back as a response. The browser then interprets and renders these files into a visual webpage for the user to view and interact with.

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

1. Code Editor (e.g., VS Code): Used to write and edit code efficiently.  
2. Web Browser (e.g., Chrome, Firefox): Used for testing and viewing web pages.  
3. Version Control (e.g., Git, GitHub): Tracks code changes and allows collaboration.  
4. Package Manager (e.g., npm): Manages libraries and dependencies.  
5. Local Server (e.g., XAMPP, Node.js): Runs and tests web applications locally.

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

A web server is software that delivers web content to clients (browsers) via HTTP or HTTPS. It processes incoming requests, retrieves resources (like HTML or data), and sends responses back. Examples: Apache, Nginx, Microsoft IIS, and LiteSpeed.

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

• Frontend Developer: Designs and implements the user interface using HTML, CSS, and JavaScript.  
• Backend Developer: Handles server-side logic, APIs, and database communication.  
• Database Administrator (DBA): Manages database design, security, backup, and optimization.

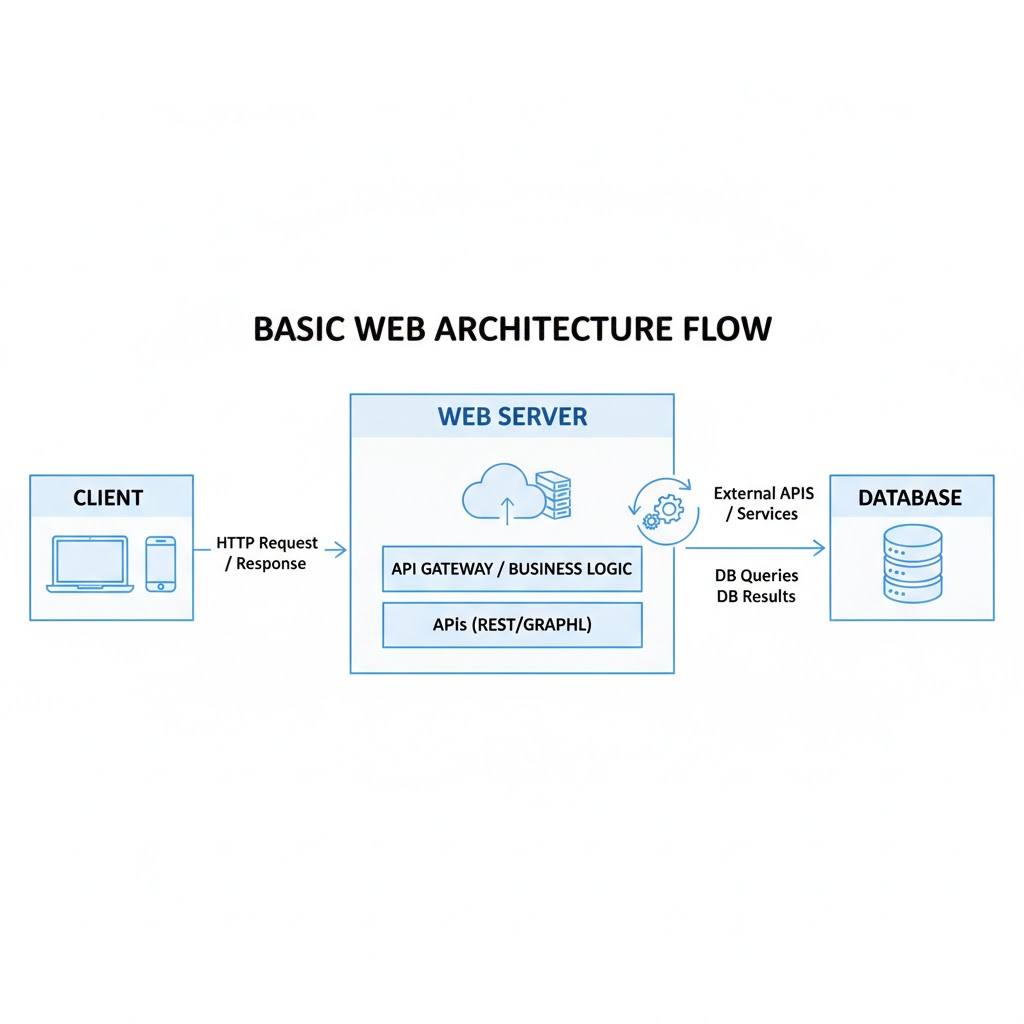
## 7. Install VS Code and configure it for HTML, CSS, and JavaScript development. Take a screenshot of the setup.

Steps:  
1. Download and install Visual Studio Code.  
2. Install extensions: Live Server, Prettier, and JavaScript (ES6) snippets.  
3. Create a folder and open it in VS Code.  
4. Create index.html, style.css, and script.js files.  
5. Use 'Live Server' to preview your project in the browser.  
  
Note: Insert your screenshot in the report before submission.

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

Static websites have fixed content coded directly in HTML, CSS, and JS. They display the same content to all users. Example: A company’s information site.  
  
Dynamic websites generate content dynamically using server-side languages like PHP, Node.js, or Python. They interact with databases and display user-specific data. Example: Facebook or Amazon.

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

1. Google Chrome – Uses Blink engine.  
2. Mozilla Firefox – Uses Gecko engine.  
3. Microsoft Edge – Uses Blink (previously EdgeHTML).  
4. Safari – Uses WebKit engine.  
5. Opera – Uses Blink engine.  
  
Rendering engines interpret HTML, CSS, and JavaScript differently, affecting page layout, performance, and compatibility.

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

Diagram Description:  
  
Client (Browser) ↔ Server ↔ API ↔ Database  
  
1. The client sends a request to the server.  
2. The server processes it and may interact with APIs or the database.  
3. The database returns the requested data.  
4. The server sends a response back to the client for display.