ASSIGNMENT 1

1. Difference Between Frontend, Backend, and Full-stack Development

- **Frontend Development**:
  - Refers to the part of the web that users interact with. It is the visual aspect and the user interface of the website or application.
  - Technologies used: HTML, CSS, JavaScript, React, Angular, Vue.js.
  - **Example**: The layout of a website, buttons, forms, and navigation bars you see on a webpage.
- **Backend Development**:
  - Refers to the server side of the application, where data is processed, and requests from the frontend are handled.
  - Technologies used: Node.js, Python, Ruby, PHP, Java, C#, databases like MySQL, PostgreSQL.
  - **Example**: When you log into a website, the backend handles the verification of your credentials and retrieves your data.
- **Full-stack Development**:
  - A full-stack developer works on both the frontend and the backend of an application, handling everything from the user interface to the database and server-side logic.
  - **Example**: Building a website like Facebook where the frontend (UI) and backend (data processing, authentication) are both managed by one developer or team.

2. Client-Server Model Diagram

The client-server model involves the client (typically a browser) sending requests to a server, which processes the requests and sends back a response.

The Client-Server Model is the foundational concept of the web:

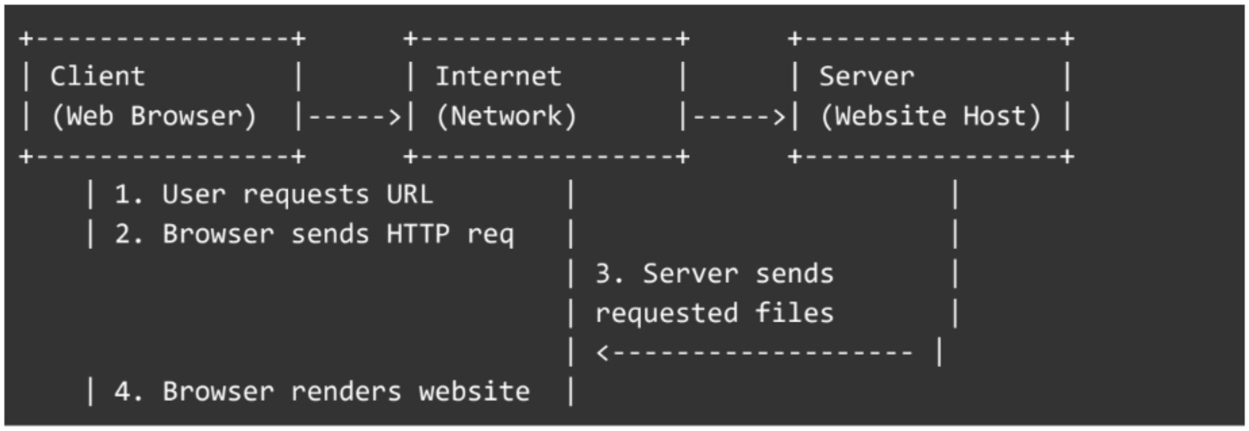
- **Client:** Usually a web browser that sends requests for data or content.

● **Server:** Responds to client requests by sending requested data (HTML pages, JSON,

files).

● **Internet/Network:** Acts as the communication channel between client and server.

Diagram Example:



```
+---------------+      +---------------+      +---------------+
| Client        |      | Internet      |      | Server        |
| (Web Browser) |----->| (Network)     |----->| (Website Host) |
+---------------+      +---------------+      +---------------+
    | 1. User requests URL       |                        |
    | 2. Browser sends HTTP req  |                        |
                                 | 3. Server sends        |
                                 | requested files        |
                                 | <------------------- |
    | 4. Browser renders website |
```

3. Browser Requesting and Displaying a Web Page

- **Step 1**: The browser sends a request to the server via the URL entered in the address bar (HTTP request).
- **Step 2**: The web server processes the request and sends the required HTML, CSS, and JavaScript files back to the browser.
- **Step 3**: The browser renders the page. It first parses the HTML, applies the CSS for styling, and then executes the JavaScript to add interactivity.
- **Step 4**: The page is displayed to the user.

4. Tools Required to Set Up a Web Development Environment

- **Text Editor (e.g., VS Code)**: To write and edit code.
- **Web Browser (e.g., Chrome, Firefox)**: For testing and rendering websites.
- **Local Web Server (e.g., XAMPP, WAMP, or Live Server extension in VS Code)**: To run server-side code locally.

- **Version Control (e.g., Git, GitHub)**: To manage and track code changes.
- **Command Line (e.g., Terminal on Mac or Command Prompt on Windows)**: For running scripts and managing projects.
- **Package Manager (e.g., npm, yarn)**: To manage JavaScript libraries and dependencies.

5. Web Server and Examples

- **Web Server**: A web server is software that serves web pages to users by accepting HTTP requests and sending back responses.
- **Examples of Web Servers**:
  - **Apache HTTP Server**: Open-source, commonly used in Linux environments.
  - **NGINX**: Known for high performance and scalability, used as a reverse proxy.
  - **Microsoft IIS**: A web server for hosting web applications in Windows environments.
  - **Node.js (Express.js)**: JavaScript runtime that can act as a web server.

6. Roles of Frontend Developer, Backend Developer, and Database Administrator

- **Frontend Developer**:
  - Responsible for creating the visual aspects and user interfaces of a website.
  - Works with HTML, CSS, JavaScript, and UI/UX design tools.
- **Backend Developer**:
  - Handles the server-side logic, APIs, databases, and application logic.
  - Works with server-side programming languages like Node.js, PHP, Python, Ruby, and frameworks like Django, Express.
- **Database Administrator (DBA)**:

    - Manages the database, ensuring data integrity, security, and performance.
    - Works with SQL or NoSQL databases and performs tasks like database design, backups, and query optimization.

7. Install VS Code and Configure it for HTML, CSS, and JavaScript Development

- **Install VS Code**: Download from here.
- **Configure for HTML, CSS, and JavaScript**:
    - Install the "Live Server" extension for real-time preview of HTML files.
    - Install the "Prettier" extension for automatic code formatting.
    - Install "ESLint" for JavaScript linting.
- **Screenshot**: You can take a screenshot after opening VS Code with a project setup containing `index.html`, `styles.css`, and `app.js`.

8. Difference Between Static and Dynamic Websites

- **Static Websites**:
    - Consist of fixed content that doesn't change unless manually edited.
    - Technologies: HTML, CSS, JavaScript.
    - **Example**: A personal portfolio website with fixed information.
- **Dynamic Websites**:
    - Content is generated on the fly, based on user interactions or server-side logic.
    - Technologies: PHP, Node.js, Python (Flask, Django).
    - **Example**: A blog where posts are dynamically loaded from a database.

9. Research and List Five Web Browsers

- **Google Chrome**
- **Mozilla Firefox**
- **Safari**
- **Microsoft Edge**
- **Opera**

Rendering Engines:

- **Google Chrome**: Uses Blink rendering engine.
- **Mozilla Firefox**: Uses Gecko rendering engine.
- **Safari**: Uses WebKit rendering engine.
- **Microsoft Edge**: Uses Blink rendering engine (after switching from EdgeHTML).
- **Opera**: Also uses Blink (same as Chrome).

10. Basic Web Architecture Flow Diagram

- **Diagram Example:**

```
+------------+       Request (HTTP)       +------------+
|   Client   | -------------------------> |   Server   |
| (Browser)  | <------------------------- | (Web App)  |
+------------+    Response (HTML, CSS)    +------------+
          |                                    |
      +---+---+                            +--+---+
      | DBMS  | <--- API Calls (Data) ------> | Server|
      +-------+                            +------+
```