Q1. Explain the difference between frontend, backend, and full-stack development with suitable real-world examples.

Ans.

1. Frontend Development (Client-Side)

Definition:
Frontend development focuses on what users see and interact with directly on a website or application. It deals with the user interface (UI) and user experience (UX).

Technologies Used:

- Languages: HTML, CSS, JavaScript
- Frameworks/Libraries: React, Angular, Vue.js, Bootstrap

Example (Real-World):
Imagine you open Amazon — the homepage you see (buttons, product images, search bar, categories, etc.) is the frontend.
When you click "Add to Cart," that button's design, color, and animation are part of the frontend.

2. Backend Development (Server-Side)

Definition:
Backend development is about what happens **behind the scenes** — it powers the logic, database interactions, and server operations.

Technologies Used:

- **Languages:** Node.js, Java, Python, PHP, C#, Ruby
- **Databases:** MySQL, MongoDB, PostgreSQL
- **Frameworks:** Express.js, Spring Boot, Django, Laravel

Example (Real-World):
When you log into **Instagram**, the backend checks your username and password against its **database**, retrieves your data, and sends it to the frontend.
You don't see this process — it happens on the server.

3. Full-Stack Development

Definition:
A **Full-Stack Developer** works on **both frontend and backend** parts of an application — they can handle the entire system end-to-end.

Technologies Used:

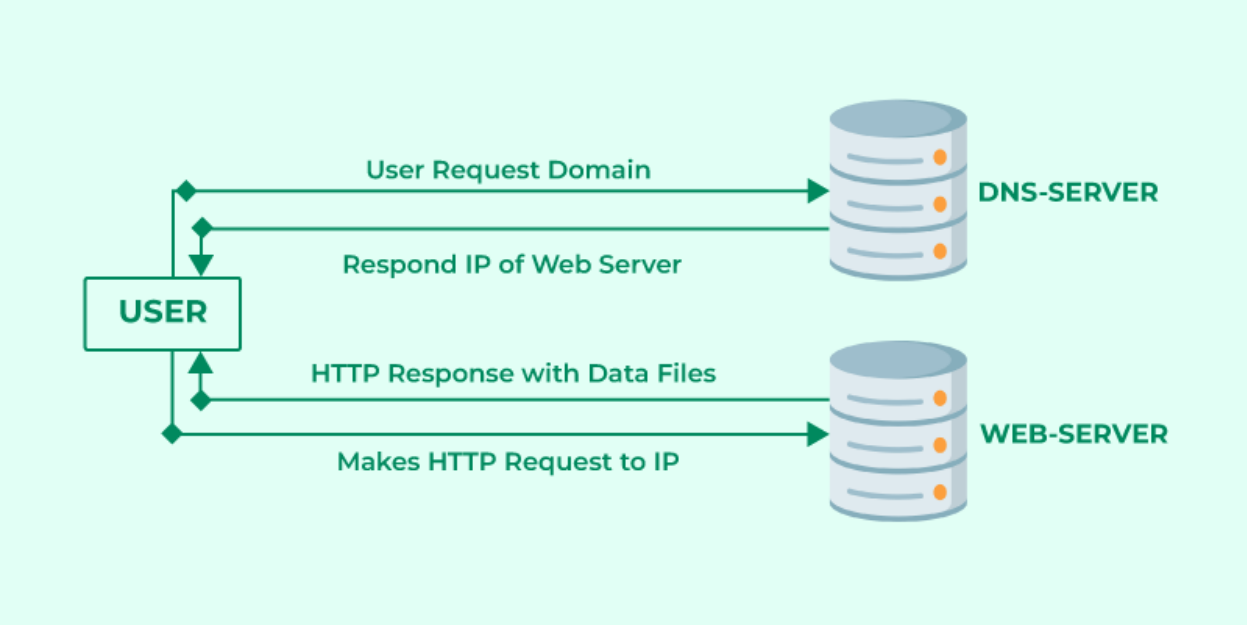
- **Frontend:** HTML, CSS, React, etc.
- **Backend:** Node.js, Express.js, MongoDB, etc.
- **Common Stacks:**
    - **MERN Stack:** MongoDB, Express.js, React, Node.js
    - **MEAN Stack:** MongoDB, Express.js, Angular, Node.js
    - **LAMP Stack:** Linux, Apache, MySQL, PHP

Example (Real-World):
A developer builds a **blog website**:

- **Frontend:** Creates the post layout, comments section, and navigation.
- **Backend:** Stores blog posts and user data in a database.
- **Full-Stack Developer:** Does both.

Q2. Create a simple diagram showing how the client-server model works in web architecture.

Ans.



1. **User Request Domain:**
   The user types a website name (like www.google.com) in the browser.
2. **DNS Server:**
   The DNS server converts the domain name into the IP address of the web server.
3. **User Makes HTTP Request:**
   Using that IP, the browser sends an HTTP request to the web server to get the website data.
4. **Web Server Responds:**
   The web server sends back an HTTP response containing the website's files (HTML, CSS, images, etc.).

Q3. Describe how a browser requests and displays a web page from a web server.

Ans.

1. User Enters a URL

When you type a web address (like www.example.com) into your browser and press Enter, the process begins.

2. DNS Lookup

- The browser asks a **DNS (Domain Name System)** server to find the **IP address** of the web server that hosts the website.
- Example: www.example.com → 192.168.1.1

3. Browser Sends an HTTP/HTTPS Request

- Using the IP address, the browser connects to the **web server**.
- It sends an **HTTP (or HTTPS)** request asking for specific resources (like index.html).

4. Web Server Processes the Request

- The **web server** receives the request.
- It fetches the required files (HTML, CSS, JavaScript, images, etc.) or generates them dynamically (using backend code).

5. Server Sends Response

- The server sends back an **HTTP response** containing:
    - Status code (e.g., 200 OK)
    - The requested files (HTML, CSS, JS, etc.)

6. Browser Renders the Page

- The browser receives the response and starts displaying it:
    1. **Reads HTML** → builds the structure (DOM)

2. **Loads CSS** → applies styles
3. **Executes JavaScript** → adds interactivity
4. **Loads media files** → shows images, videos, etc.

Q4. Identify and list the tools required to set up a web development environment. Explain the purpose of each.

Ans.

1. Code Editor / IDE

Examples: Visual Studio Code, Sublime Text, Atom
Purpose:

- Used to write and edit code efficiently.
- Provides features like syntax highlighting, auto-completion, debugging, and extensions.
  Example: VS Code helps developers write HTML, CSS, JavaScript, and backend code all in one place.

2. Web Browser

Examples: Google Chrome, Mozilla Firefox, Microsoft Edge
Purpose:

- Used to **test and view** web pages.
- Developer Tools (Inspect Element, Console, Network tab) help debug layout and performance issues.
  Example: Chrome DevTools let you check CSS styles, inspect HTML structure, and monitor network requests.

3. Version Control System

Examples: Git, GitHub, GitLab, Bitbucket
Purpose:

- Tracks **changes in code** over time.
- Allows collaboration among multiple developers.
  Example: GitHub stores your project code online and manages versions through commits and branches.

4. Web Server / Local Server

Examples: XAMPP, WAMP, Node.js, Apache, Nginx
Purpose:

- Used to **run and test** websites locally before deployment.
- Handles HTTP requests and responses.
  Example: Node.js helps run a JavaScript backend server locally for testing APIs.

5. Database Management System (DBMS)

Examples: MySQL, MongoDB, PostgreSQL, SQLite
Purpose:

- Stores and manages **website data** (like users, products, posts).
  Example: MongoDB stores data in a NoSQL format for backend apps using Node.js.

6. Package Manager

Examples: npm (Node Package Manager), Yarn, pip
Purpose:

- Installs and manages **external libraries or dependencies** needed for the project.
  Example: npm installs frameworks like Express.js or React easily via terminal.

Q5. Explain what a web server is and give examples of commonly used servers.

Ans.

A **web server** is a **software or hardware** system that **stores, processes, and delivers web pages** to users over the Internet using the **HTTP or HTTPS protocol**.

When you enter a website URL in your browser (like www.google.com), the web server receives that request, finds the right web page or resource, and sends it back to your browser for display.

Functions of a Web Server

1. **Store Website Files:**
   It holds all the files (HTML, CSS, JavaScript, images, etc.) required for a website.
2. **Process Client Requests:**
   It listens for incoming requests from browsers (clients) and responds with the requested data.
3. **Serve Dynamic Content:**
   It can work with backend languages (like PHP, Node.js, Python, etc.) to generate content dynamically.
4. **Handle Security & Access:**
   It manages secure connections (HTTPS), authentication, and permissions.

How It Works (Simplified):

1. User enters a website URL.
2. Browser sends an **HTTP request** to the web server.

3. The web server processes the request.
4. It sends back an **HTTP response** containing the requested webpage.
5. Browser displays the webpage to the user.

Q6. Define the roles of a frontend developer, backend developer, and database administrator in a project.

Ans.

1. Frontend Developer (Client-Side Developer)

Role:
Responsible for designing and building the part of the website or application that users see and interact with directly.

Key Responsibilities:

- Create user interfaces (UI) using HTML, CSS, and JavaScript.
- Ensure the website is responsive (works on all screen sizes).
- Improve user experience (UX) through smooth navigation, layout, and animations.
- Integrate frontend with backend APIs to display dynamic data.
- Test and debug design and interaction issues in browsers.

Example:
Designing the login form, navigation bar, or product page layout of an e-commerce website.

2. Backend Developer (Server-Side Developer)

Role:
Handles the logic, database interactions, and server-side operations that power the frontend.

Key Responsibilities:

- Develop and maintain server-side logic (authentication, data processing, etc.).
- Build APIs to connect frontend and backend.
- Manage database operations like storing and retrieving user data.
- Handle security, authorization, and data validation.
- Optimize server performance and scalability.

Example:
When a user logs in, the backend verifies the credentials from the database and sends a success or failure response.

3. Database Administrator (DBA)

Role:
Responsible for **designing, managing, and maintaining the database system** to ensure data integrity, security, and performance.

Key Responsibilities:

- Design the **database schema** (tables, relationships, etc.).
- Manage **data backups and recovery**.
- Ensure **database security** and user access control.
- Monitor and optimize **database performance**.
- Support developers by writing or optimizing **SQL queries**.

Example:
Managing the **user accounts table**, ensuring fast query performance, and setting up backups for data recovery.

Q7. Install VS Code and configure it for HTML, CSS, and JavaScript development. Take a screenshot of the setup.

Ans.

Step 1: Download and Install VS Code

1. Go to the official website:
   https://code.visualstudio.com/
2. Click Download for Windows (or Mac/Linux depending on your system).
3. Run the downloaded installer (VSCodeSetup.exe) and follow the installation steps:
   - Check “Add to PATH” during installation.
   - Select “Open with Code” options.

Step 2: Open VS Code

After installation, launch **VS Code**.
You’ll see a welcome screen with a sidebar (Explorer, Search, Source Control, etc.).

Step 3: Install Required Extensions

Step 4: Create Your Project Folder

1. Open a new folder in VS Code (e.g., MyWebProject).

2. Inside it, create:
    - index.html
    - style.css
    - script.js

Step 5: Write Basic Code

Step 6: Run the Project

1. Right-click on index.html
2. Select **"Open with Live Server"**
3. Your default browser will open the page with auto-reload support.

Step 7: Take a Screenshot

Make sure your screenshot shows:

- The **VS Code window** open with index.html, style.css, and script.js files.
- The **Live Server** running in the browser.
- The **Extensions panel** (optional).

Q8. Explain the difference between static and dynamic websites. Provide an example of each.

Ans.

1. Static Website

Definition:
A **static website** contains **fixed content** — each page is coded in HTML and looks the same for every user.
The content doesn't change unless a developer **manually edits the code**.

Key Features:

- Built using **HTML, CSS, and JavaScript** only.
- No database or server-side processing.
- **Faster loading** because content is pre-built.
- Best for **small or informational** websites.

Example:
A **personal portfolio website** or a **company's brochure site**.
Example:
index.html, about.html, contact.html — all static files stored on the server.

Real Example:
A simple website like
https://www.example.com/about.html

2. Dynamic Website

Definition:
A **dynamic website** displays **different content** and can **change automatically** based on user interaction or data from a database.

Key Features:

- Built using both **frontend and backend technologies** (like Node.js, PHP, Python, etc.).
- Connected to a **database** (like MySQL or MongoDB).
- Can **personalize** content for each user.
- Common for **blogs, e-commerce, or social media** sites.

Example:
An **e-commerce website** like **Amazon** or a **social media site** like **Facebook**.
The product listings or user feeds change dynamically based on the database.

Q9. Research and list five web browsers. Explain how rendering engines differ between them.

Ans.

Web Browsers and Their Rendering Engines

A web browser is a software application used to access and display web pages on the internet. Each browser uses a rendering engine to interpret HTML, CSS, and JavaScript code and then display the formatted web page to the user.
Different browsers use different rendering engines, which can cause slight variations in how websites appear or perform.

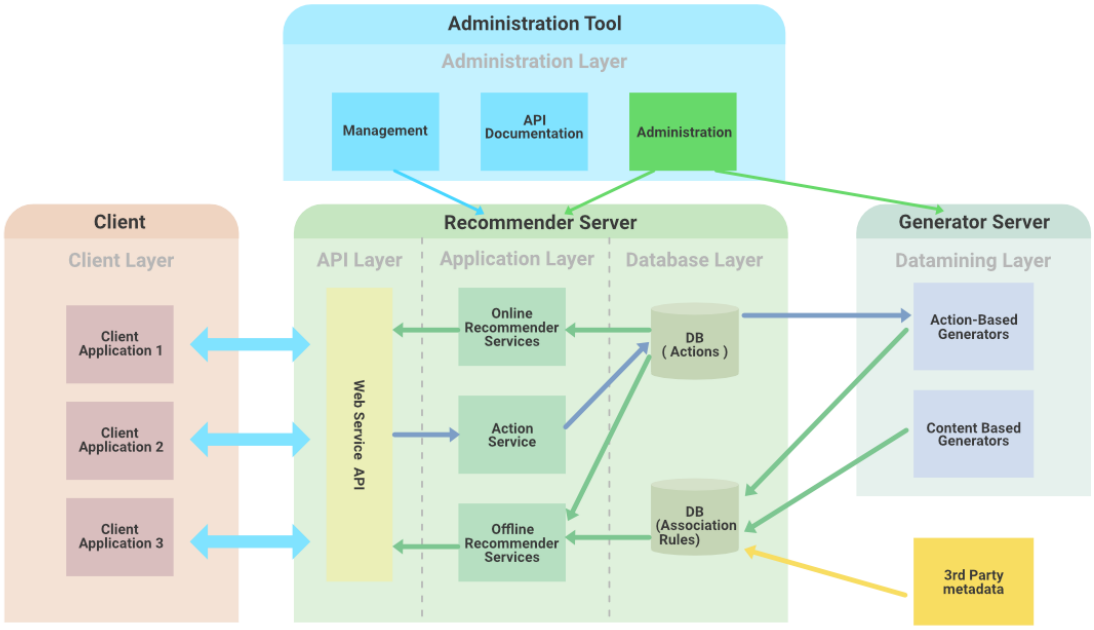
Web Browser	Rendering Engine	Developer / Organization
Google Chrome	Blink	Google
Mozilla Firefox	Gecko	Mozilla Foundation
Apple Safari	WebKit	Apple Inc.
Microsoft Edge	Blink	Microsoft (Chromium-based)
Internet Explorer	Trident	Microsoft

Differences Between Rendering Engines

1. **Blink (used in Chrome, Edge, Opera):**
   - Developed by Google.
   - Very fast and optimized for modern web standards.
   - Provides excellent performance for JavaScript-heavy web applications.
2. **Gecko (used in Firefox):**
   - Developed by Mozilla.
   - Strong focus on open standards and web compatibility.
   - Provides advanced support for new CSS and HTML features.
3. **WebKit (used in Safari):**
   - Developed by Apple.
   - Highly optimized for macOS and iOS devices.
   - Ensures smooth graphics and energy efficiency on Apple hardware.
4. **Trident (used in Internet Explorer):**
   - Developed by Microsoft.
   - Older engine, now outdated and replaced by newer technologies.
   - Limited support for modern web standards.
5. **EdgeHTML (used in older Microsoft Edge):**
   - Designed as a modern replacement for Trident.
   - Provided faster performance and better standards support.
   - Later replaced by Blink when Edge became Chromium-based.

Q10. Draw a labeled diagram showing the basic web architecture flow — client, server, database, and APIs.

Ans.


Administration Tool
Administration Layer
Management
API Documentation
Administration
Client
Client Layer
Client Application 1
Client Application 2
Client Application 3
Recommender Server
API Layer
Application Layer
Database Layer
Web Service API
Online Recommender Services
Action Service
Offline Recommender Services
DB (Actions)
DB (Association Rules)
Generator Server
Datamining Layer
Action-Based Generators
Content Based Generators
3rd Party metadata