Unit 1: Overview of Web Technology

1.1 Introduction to WWW, URL, HTTP, HTTPS

World Wide Web (WWW)

- The World Wide Web (WWW) is a vast information system that is built on hypertext documents that are linked together. It is accessed over the Internet using web browsers (such as Chrome, Firefox, Safari).
- The WWW allows users to access text, images, videos, and other resources via hyperlinks.
- **Example**: When you open a browser and enter a website address like https://www.example.com, you are using the WWW to access information stored on a server.

Uniform Resource Locator (URL)

- A **URL** is a reference to a resource on the web. It's essentially the web address used to find specific information or services. The URL is broken down into the following parts:
  - **Protocol**: This defines how data will be transferred. Common protocols are http and https.
  - **Hostname**: The domain name or IP address where the website is hosted.
  - **Path**: The specific page or resource being requested on the server.
  - **Query**: Optional parameters that can be passed to the server (used in dynamic websites).
  - **Fragment**: A reference to a specific part of the resource.

  Example:

  - URL: https://www.example.com/products?category=shoes#size
  - **Protocol**: https
  - **Hostname**: www.example.com
  - **Path**: /products
  - **Query**: ?category=shoes
  - **Fragment**: #size

Hypertext Transfer Protocol (HTTP)

- **HTTP** is the protocol used to transfer data on the World Wide Web. It defines how messages are formatted and transmitted, and how web servers and browsers should respond to various commands.
- **Request-Response Model**: A user's browser (client) sends an HTTP request to the server, which processes the request and sends back an HTTP response.

  Example:

   - **Request**: The browser sends an HTTP GET request for the page https://www.example.com/products.
   - **Response**: The server responds with an HTTP status code (200 OK) and the HTML content for the webpage.

Hypertext Transfer Protocol Secure (HTTPS)

- **HTTPS** is the secure version of HTTP. It encrypts data transmitted between the client and server using SSL/TLS, ensuring confidentiality and integrity of data.
- Websites with https:// in their URL are secured with HTTPS, and you may see a padlock symbol in the browser address bar indicating the connection is secure.

  Example: Online banking websites, such as https://www.bank.com, use HTTPS to ensure the privacy of sensitive information (like account numbers and passwords).

1.2 The Evolution of Web Technologies

Early Web (Static Pages)

- In the early days of the internet, websites were made up of simple HTML pages. These pages were static, meaning the content did not change unless manually updated by the website owner.
- **Example**: A personal blog from the 1990s or early 2000s would often consist of static HTML pages, where text and images were displayed without interaction.

Web 2.0 (Dynamic Content)

- Web 2.0 introduced dynamic content and interactivity. The use of JavaScript, CSS for styling, and AJAX (Asynchronous JavaScript and XML) allowed websites to be more interactive and real-time. Websites could now update content without requiring a full page reload.

  Example: Social media platforms like Facebook, Twitter, or Gmail, which dynamically update content (such as posts, messages, etc.) without requiring users to refresh the page.

Modern Web (Web 3.0, Progressive Web Apps)

- Web 3.0 focuses on decentralized applications, often using blockchain technology. Web apps are more intelligent, making use of artificial intelligence (AI) and machine learning.
- **Progressive Web Apps (PWA)** provide native-like mobile experiences, allowing users to install websites on their devices and use them offline.

  Example: Decentralized finance (DeFi) applications using blockchain technology, and PWAs like Instagram, which can function both as a web app and a mobile app.

1.3 Website and Web Application

Website

- A website is a collection of web pages that provide static or dynamic content. A website is often informational and can be simple or complex.

  Example: A business website that displays information about services, contact details, and company history is a typical example of a website.

Web Application

- A web application is an interactive software program that runs in a web browser and is designed to perform complex tasks. Web applications allow user input and data manipulation, making them more dynamic than simple websites.

  Example: Gmail (an email service), Google Docs (a word processing tool), and Trello (a task management tool) are all examples of web applications that allow users to interact with the system and modify content.

1.4 Web Client and Web Server

Web Client

- The **web client** is the user's device (typically a browser) that makes requests to the web server and displays the resulting content.
  - Common clients include browsers like Chrome, Firefox, Safari, etc.

  Example: When you open your browser and type https://www.example.com, your browser acts as the client and sends the request to the server hosting the website.

Web Server

- The **web server** is a computer that stores the website's data and responds to client requests. It serves requested resources (web pages, images, etc.) to the client over HTTP/HTTPS.
  - Popular web servers include Apache, Nginx, and Microsoft IIS.

  Example: When a user requests https://www.example.com, the server hosting that site responds with the content of the home page (HTML, CSS, JavaScript).

1.5 HTTP Request and Response

HTTP Request

- An HTTP request is sent by the client to the server to request a resource. It includes:
  - **Method**: The type of request (e.g., GET, POST).
  - **Headers**: Additional information (e.g., browser type, accepted formats).
  - **URL**: The resource being requested.
  - **Body**: Data sent with methods like POST (e.g., form submissions).

  Example:

  - **GET request**: The client requests the home page with a URL like https://www.example.com/.
  - **POST request**: A user submits a login form to a server.

HTTP Response

- The HTTP response is sent by the server back to the client. It contains:
  - **Status Code**: Indicates whether the request was successful (e.g., 200 OK, 404 Not Found).
  - **Headers**: Metadata (e.g., Content-Type, Cache-Control).
  - **Body**: The content being returned (HTML, JSON, etc.).

  Example:

  - **200 OK**: The server successfully returns the requested content.
  - **404 Not Found**: The resource was not found on the server.

1.6 Client-Side and Server-Side Scripting Languages

Client-Side Scripting

- **Client-side scripting** involves writing code that runs on the user's browser. It is used to create interactive features, such as form validation, dynamic content updates, animations, etc.
  - **Example**: JavaScript is the most common client-side scripting language used for interactivity on web pages.

  Example: A contact form validation script that ensures a user has entered an email address in the correct format before submitting it.

Server-Side Scripting

- **Server-side scripting** involves writing code that runs on the web server. It is used for tasks like processing form submissions, interacting with databases, and generating dynamic content.
    - **Languages**: PHP, Python (Flask, Django), Node.js, Ruby, etc.

  Example: A PHP script that processes a user login request by checking the credentials in a database and returning a success or error message.

1.7 DNS and Its Hierarchy

DNS (Domain Name System)

- **DNS** is the system that translates human-readable domain names (e.g., www.example.com) into machine-readable IP addresses (e.g., 192.168.1.1).
- It acts like a phone book for the internet, mapping domain names to servers.

  Example: When you type www.example.com in the browser, DNS translates that domain into the appropriate IP address of the server.

DNS Hierarchy

- **Root DNS Servers**: The highest level of the DNS hierarchy, responsible for directing queries to TLD servers.
- **Top-Level Domains (TLDs)**: Examples include .com, .org, .net, and country codes like .us or .uk.
- **Second-Level Domains**: The part of the domain name directly below the TLD (e.g., example in example.com).
- **Subdomains**: Additional divisions within a domain (e.g., blog.example.com).

1.8 Web Hosting

Types of Web Hosting

- **Shared Hosting**: Multiple websites are hosted on the same server, sharing resources.
- **VPS (Virtual Private Server)**: A virtualized server that provides more control and resources.
- **Dedicated Hosting**: A server is dedicated to hosting a single website.
- **Cloud Hosting**: Scalable hosting provided through cloud computing, allowing resources to be adjusted as needed.

Domain Name Registration

- The process of purchasing a domain name and linking it to your hosting server.
    - Example: Registering www.example.com through a domain registrar (e.g., GoDaddy, Namecheap).

Hosting and Deployment

- **Deployment**: The process of uploading a website or web application to a server and making it accessible to users on the internet.
- **Tools for Deployment**: FTP (File Transfer Protocol), SSH (Secure Shell), and CI/CD pipelines for automated deployment.