Unit I: Introduction

1.1. Web Basics

Overview:

- Internet
- Intranet
- WWW
- Static and Dynamic Web Pages
- Web Clients
- Web Servers

1️⃣ Internet vs. Intranet vs. WWW

Term	Definition	Example
Internet	A global network connecting millions of computers and devices worldwide.	Accessing websites like google.com, amazon.com
Intranet	A private network used within an organization. Not accessible to the public.	Company internal portal accessible to employees only
WWW	A service on the Internet. A collection of web pages and resources identified by URLs.	Websites like Facebook, Wikipedia, YouTube

---

2️⃣ Static vs. Dynamic Web Pages

Feature	Static Web Page	Dynamic Web Page
Content Type	Fixed, same for every visitor	Changes based on user interaction or server data
Technologies	HTML, CSS	HTML, CSS + JavaScript, PHP, ASP.NET, etc.
Example	A simple About Us page	Facebook news feed, E-commerce product pages
Content Update	Manual (edit HTML file)	Auto-generated from databases
Speed	Faster to load (no server processing needed)	Slower (requires server-side processing)

Examples:

1. **Static Web Page Example:**

```
<!DOCTYPE html>
<html>
```

```
  <head><title>About Us</title></head>
  <body>
    <h1>Welcome to XYZ Company</h1>
    <p>This is our static web page.</p>
  </body>
</html>
```

1. **Dynamic Web Page Example (PHP):**

```
<?php
  echo "Hello " . $_GET['name'];
?>
```

Accessing: `http://example.com/page.php?name=Alice` will output:

`Hello Alice`

3️⃣ Web Clients and Web Servers

Component	Definition	Examples
Web Client	A device or software that requests web pages/services from servers.	Browsers (Chrome, Firefox), Mobile apps
Web Server	A system/software that hosts websites, handles HTTP requests, and serves content.	Apache, Nginx, IIS

Simple Client-Server Interaction Diagram:

```
[Web Client: Browser]  <——HTTP Request——  [Web Server: Apache/Nginx]
[Web Client: Browser]  ——HTTP Response——> [Web Server: Apache/Nginx]
```

Practice Problems:

1. Define **Intranet** and give one practical example of its use.
2. Explain the difference between **Static** and **Dynamic Web Pages** with one example each.
3. What is the role of a **Web Server**? Name two popular web servers.
4. Identify whether the following is Static or Dynamic:
   - An online shopping cart page
   - A company's contact page

Quick Recap Table:

Concept	Key Point
Internet	Global public network
Intranet	Private internal organizational network
WWW	Collection of web pages, part of the Internet
Static Web Page	Fixed content, faster, simple HTML
Dynamic Web Page	Content changes based on data/user, server-side processing
Web Client	Requests data (Browser, App)
Web Server	Serves web content (Apache, Nginx, IIS)

1.2 Client-Server Architecture

🌐 What is Client-Server Architecture?

Client-Server Architecture is a model used in network communication where **clients request services/resources**, and **servers provide them**.

Think of it like a restaurant:

- **Client = Customer (makes orders)**
- **Server = Waiter/Chef (prepares & serves food)**

Key Components:

Component	Description
Client	Device/software (e.g., web browser) requesting data
Server	Device/software providing data/resources
Network	The medium (internet/intranet) connecting them

Types of Architectures:

Architecture Type	Explanation	Example
Single-Tier	Client & Server on **same machine**	Local database app
Two-Tier	Client & Server on **separate machines**, directly connected	Web Browser & Web Server
Multi-Tier (N-Tier)	Middle layers (like Application Servers, Database Servers) between client and server	E-commerce websites (browser → web server → app server → database)

Diagram Overview:

```
Single-Tier:
 [ Client + Server (same) ]

Two-Tier:
 [ Client ]  ←——→  [ Server ]

Multi-Tier:
 [ Client ]  ↔ [ Web Server ] ↔ [ App Server ] ↔ [ Database Server ]
```

Advantages of Client-Server Model:

Advantage	Explanation
Centralized Control	Server manages resources securely
Scalability	Servers can be upgraded separately
Maintenance	Easier to maintain (server-side updates reflect for all clients)
Data Integrity	Data stored centrally is consistent

Example:

When you visit **www.facebook.com**:

- Your browser (Client) sends a request.
- Facebook's servers (Server) process the request and send back your personalized feed.

Recap Table:

Topic	Key Point
Client	Requests services
Server	Provides services
Single-Tier	Client & Server same
Two-Tier	Client ↔ Server directly
Multi-Tier	Client ↔ Web Server ↔ App Server ↔ Database
Main Advantages	Centralization, Scalability, Easy Maintenance, Integrity

Practice Problems:

1. **Question:**

   Which architecture is used when a browser communicates directly with a database server on the same machine?

2. **Question:**

   List **two advantages** of using a Multi-Tier Architecture.

3. **Question:**

   Identify if the following scenario is Single-Tier, Two-Tier, or Multi-Tier:

> "A banking website where the client interacts with a web server, which communicates with an application server, and then the database server."

1.3 HTTP (HyperText Transfer Protocol): HTTP Request and Response, URL

🌐 What is HTTP?

HTTP stands for **HyperText Transfer Protocol**. It is the **communication protocol** used by web browsers (clients) and web servers to exchange data over the web.

Analogy:

Think of HTTP like a postal service:

- **Client = You (sending a letter)**
- **HTTP = Postal system (delivers the letter)**
- **Server = Receiver (reads & responds)**

1️⃣ HTTP Request:

A **client (browser)** sends an HTTP request to the server asking for data (like a web page, image, etc.).

Main Components of HTTP Request:

Part	Example	Description
Request Line	`GET /index.html HTTP/1.1`	Method, Resource, Version
Headers	`Host: www.example.comUser-Agent: Chrome`	Extra info (browser, language, cookies)
Body	Data sent by client (optional; mainly in POST requests)	

Common HTTP Request Methods:

Method	Purpose
GET	Retrieve data (like web pages)
POST	Send data to server (like forms)
PUT	Update data on server
DELETE	Delete data on server

2️⃣ HTTP Response:

The **server processes the request** and sends back an **HTTP Response**.

Main Components of HTTP Response:

Part	Example	Description
Status Line	`HTTP/1.1 200 OK`	Protocol version + Status code
Headers	`Content-Type: text/html`	Info about data type, caching, cookies
Body	Actual content (HTML, JSON, etc.)	The web page/data being sent back

---

Common HTTP Status Codes:

Code	Meaning
200	OK (Successful)
404	Not Found
500	Internal Server Error
301	Moved Permanently (Redirect)
403	Forbidden

🔗 URL (Uniform Resource Locator)

URL is the address used to access resources on the web.

Structure of a URL:

```
https://www.example.com:443/path/page.html?query=value#section
```

Part	Example	Description
Protocol	`https`	Communication protocol (HTTP, HTTPS)
Host/Domain	`www.example.com`	Server name (DNS)
Port	`443` (optional)	Communication port (default: 80 HTTP, 443 HTTPS)
Path	`/path/page.html`	Resource location on server
Query String	`?query=value`	Key-value pairs sent to server
Fragment	`#section`	Points to a section within the resource

📝 Recap Table:

Concept	Key Point
HTTP	Protocol used for web communication
HTTP Request	Sent by client; contains method, headers, optional body

HTTP Methods	GET, POST, PUT, DELETE, etc.
HTTP Response	Sent by server; contains status code, headers, body
Common Status Codes	200 (OK), 404 (Not Found), 500 (Error)
URL	Address to locate resources
URL Components	Protocol, Host, Port, Path, Query String, Fragment

---

🧠 Practice Problems:

1. **Identify which HTTP method you'd use to submit a login form.**
2. **What is the meaning of HTTP status code `404`?**
3. **In the URL `https://shop.example.com/products/item?id=45#reviews`, name each part (protocol, host, path, query, fragment).**
4. **Which HTTP request method is used to update an existing resource?**

1.4 Client Side Scripting vs Server Side Scripting

🌐 What is Scripting?

Scripting refers to writing small programs (scripts) that automate tasks or control the behavior of web pages/web servers.

🖥️ Client Side Scripting

🔷 What is it?

- Scripts that **run on the user's browser (client machine)**.
- Mainly used for **UI behavior**, **validation**, **dynamic content changes**, etc.

🔷 Examples of Client Side Scripting Languages:

Language	Purpose
JavaScript	Manipulate HTML/CSS, validate forms, events
HTML (markup)	Defines structure (not logic but often grouped here)
CSS (styling)	Controls design/layout (not scripting, but affects client display)

🔷 How it works:

Browser downloads the script from the server → Executes it locally.

Example:

```
<!DOCTYPE html>
<html>
<body>

<h2>Client Side Example</h2>

<p id="demo">Click the button:</p>

<button onclick="document.getElementById('demo').innerHTML = 'Hello, World!'">Click Me</butt
on>

</body>
</html>
```

Here, JavaScript changes the text dynamically **without contacting the server again.**

🗄️ Server Side Scripting

🔷 What is it?

- Scripts that **run on the web server** before sending the final web page to the client's browser.
- Used for **database interaction**, **user authentication**, **file processing**, etc.

🔷 Examples of Server Side Scripting Languages:

Language	Purpose
PHP	Dynamic page generation, database
Python (Django, Flask)	Server-side logic
Node.js	JavaScript on the server
Java (JSP)	Server pages

🔷 How it works:

Client sends request → Server runs the script → Server sends back final HTML content.

Example (Python Flask Server Side Script):

```
from flask import Flask
app = Flask(__name__)

@app.route("/")
def hello():
    return "<h1>Hello, World from Server!</h1>"
```

```
app.run()
```

Here, the **server** generates the HTML and sends it to the client.

📊 Difference Between Client Side & Server Side Scripting:

Feature	Client Side Scripting	Server Side Scripting
Where it runs	Browser (client machine)	Web server
Execution Speed	Fast (no need to communicate with server)	Slower (server processing needed)
Security	Less secure (viewable by user)	More secure (runs on server)
Access to files/database	No direct access	Direct access possible
Common languages	JavaScript, HTML, CSS	PHP, Python, Node.js, Java
Use cases	Form validation, animations	Login systems, database queries

---

🧠 Recap Table:

Concept	**Important Points**
Client Side Scripting	Runs on browser, fast, used for UI interaction
Server Side Scripting	Runs on server, used for database interaction & authentication
Client Side Languages	JavaScript, HTML, CSS
Server Side Languages	PHP, Python, Node.js, Java
Security	Server side is more secure
Access	Server side can access database, client side cannot

📝 Practice Questions:

1. Which scripting (client or server) is better for user login systems? Why?
2. Name two common client side scripting languages.
3. Why is server side scripting considered more secure?
4. In which scenario would you use client side scripting:
   a) Validating user input in a form
   b) Fetching data from a database

📌 1.5 Web 1.0 and Web 2.0

🔥 Overview:

The terms **Web 1.0** and **Web 2.0** refer to different phases in the evolution of the World Wide Web.

Let’s break them down clearly:

🌐 Web 1.0: The Static Web (1990s to early 2000s)

Characteristics:

Feature	Description
Nature	Static (Read-Only)
Content Generation	By developers/administrators only
User Interaction	Minimal to none (No user-generated content)
Technology Stack	HTML, basic CSS, limited use of JavaScript
Example Sites	Early Yahoo!, early personal websites
Data Flow	One-way: Server → Client
Customization	Very limited; same content for every user

Example:

A personal blog from the 1990s, where the webmaster posts articles and users can only read them.

No comments, no sharing, no interactivity.

🌐 Web 2.0: The Social & Dynamic Web (2004 onwards)

Characteristics:

Feature	Description
Nature	Dynamic (Read-Write)
Content Generation	Users contribute content (UGC - User Generated Content)
User Interaction	High (comments, likes, shares, social networking)
Technology Stack	HTML, CSS, JavaScript, AJAX, XML, JSON, PHP, MySQL, APIs
Example Sites	Facebook, YouTube, Wikipedia, Twitter
Data Flow	Two-way: Server ↔ Client
Customization	Highly personalized (feeds, recommendations)

Example:

Facebook — users can post, comment, like, share, interact dynamically with content.

🆚 Difference Table: Web 1.0 vs Web 2.0

Criteria	**Web 1.0**	**Web 2.0**
Nature	Static (Read-only)	Dynamic (Read & Write)
User Role	Passive consumer	Active participant

Interactivity	Minimal	High (comments, uploads, interactions)
Content Creation	By developers/admins only	By users and admins
Examples	Early Yahoo!, Static blogs	Facebook, YouTube, Wikipedia
Technologies Used	HTML, CSS	HTML, CSS, JavaScript, AJAX, PHP, APIs
Customization	Same content for all users	Personalized content

---

💡 Key Technologies Shift:

> **Web 1.0** → Simple websites

> **Web 2.0** → Rich, interactive, community-driven platforms powered by **JavaScript, AJAX, APIs, Databases**.

📝 Quick Recap Table:

Point	**Web 1.0**	**Web 2.0**
Content	Static	Dynamic & User-Generated
User Interaction	Very limited	High interactivity
Technologies	HTML, basic CSS	HTML, CSS, JS, AJAX, APIs
Examples	Early websites	Social media, wikis, blogs

📚 Practice Problems:

Q1. Name two examples of Web 1.0 websites.

Q2. What is the key difference in user interaction between Web 1.0 and Web 2.0?

Q3. Which key technologies enabled the transition from Web 1.0 to Web 2.0?
