Module-01

Computer-Network Essential for Web Dev

How Internet Works

- First your device (like a phone or laptop) connects to the internet through your Internet Service Provider (ISP).
- When you type a website name, like www.google.com, your device doesn't understand names, it understands numbers.
- That's where the Domain Name System (DNS) comes in: It converts the website name into it's IP address.
- Once the IP address is known, your request is prepaired to be sent. But instead of sending it as a big piece of data, it is broken into small packets.
- Each packet carries three things: part of the data, the destination address, and a sequence number (to arrange it in order later).
- These packets then travel through the network with the help of routers.
- A router is like a traffic manager - it looks at each packet & decide the best path for it to travel.
- Because of it/this, packets belonging to the same request may take different routes, but all head towards the same destination.

Date / /

- The server (like Google's server) receives these packets, understand your request, and then send back the response again in the form of packets.

- On the return journey, the packets are also guided by routers until they reach your device.

- Finally, your browser reassembles the packets in the correct order using the sequence number, and if any are missing, it asks again.

- When everything is complete, the full webpage, or image or video appears on your screen & this entire process happens in just milliseconds.

What is IP Address

- An IP address (Internet protocol Address) is like the unique identification number of a device on a network.

- Just like your home has a postal address so letters can reach you, your device has an IP address so data can reach you.

- It is a set of numbers that tells two things:
  - Who the device is (it's identity)
  - Where it is located on the network (it's location)

Date / /

- For example: 192.168.1.1 is a common IP address format.

Types of IP Addresses:

1. IPV4 – Uses 32 bits, written as four numbers (eg 192.168.a
2. IPV6 - Newer version, uses 128 bits, looks longer e.g: (2001:0db8:85a3::8a2e:0370:7334).

IP Address and Internet Connection.

- An IP Address is required for any device to communicate over a network.
- If you are connected to the internet, your device is assigned an IP address (public or private) which works as it's identity for sending and receiving data.
- If you are not connected to any network (neither internet nor local WiFi), your device will not have an active IP address.
- However if your device is only connected to a local network (LAN WiFi) it may still have a private IP address assigned by the router - but this will not be usable for global internet communication. (without internet LAN WIFI)

Date / /

Key Differences b/w Public & Private IP address.

Features	Public IP	Private IP
Visibility	Visible on the Internet	Only visible inside local network
Assigned by	Internet Service provider	Router or network admin
Uniqueness	Must be unique worldwide	Can be reused in different local n/w
Example	103.45.67.89	192.168.0.1, 10.0.0.2

Key Differences b/w IPV4 and IPV6

Features	IPV4	IPV6
Address size	32 bit address	128 bit address
Address format	Written in decimal saperated by dots (eg: 192.168.1.1)	Written in hexadecimal saperated by colons (eg., 2001:0db8:8593::842e:0370:7334)
Address Space	About 43 billion unique addresses	About 340 undecillions unique addresses (almost unlimited)

Date / /

Features	IPV4	IPV6
Security	Security depends on applications (IP Sec optional).	IP Sec is mandatory, so it's more secure by default.
NAT (Network Address Translation)	Uses NAT because of limited addresses	No need for NAT (enough addresses for every device).
Speed & Efficiency	Slower in packet processing compared to IPV6	Faster routing and packet processing
Deployment	old, still widely used	New, gradually replacing IPV4.

What is MAC address.

- MAC address stands for Media Access Control Address.
- It is a unique hardware identification number given to a device's network interface card (NIC) [like a wifi card or Ethernet card].
- Unlike an IP address (which can change), A MAC address is permanent and comes built into the device by the manufacturer.

Date / /

What is Client and Server

Client:

- A client is a device or software that requests services or resources from another system (the server).

- Example
  - A web browser (like Chrome, firefox) is a client that requests web pages.
  - Your phone app (like whatsapp) acts as a client to connect to whatsApp servers.

- Clients typically send requests & then wait for responses.

Server:

- A server is a device or software that provides services or resources to clients.

- Example
  - A web server (like Apache, Nginx) stores and serves web pages.
  - A mail server handles sending & recieving emails.

- Servers usually listen for incoming client requests and respond with the requested data.

Date / /

The Evolution of the Web

The internet hasn't always looked the way it does today. Over the years, it has grown in stages - each one more interactive & smarter than the last. Here's how the web has evolved.

1. Web 1.0 - The Static Web (1990s - early 2000s)
This was the very first version of web, often called the "read only web". Websites back then were mostly static pages filled with text & few images. You couldn't really interact with them - you just visit a site, read what was there, and moved on.
   - Example: Early Yahoo pages or personal blogs where you could only read but not comment or share.

2. Web 2.0 - The Social Web (2000s - today)
Next came Web 2.0, also known as the "read-write web". This is when the internet became interactive. Users could create content, share opinions, upload photos, and collaborate online. Social media platforms & interactive websites exploded in popularity.
   - Example: Facebook, Instagram, Youtube, Wikipedia.

3. Web 3.0 - The Decentralized & Smart Web (happening now)
Web 3.0 is a current shift we're seeing. It's often described as the "read-write-own web". Here users are taking back control of their data through technologies like blockchain, while artificial intelligence is making the web smarter & more personalized.

Date / /

Examples:

- Decentralized apps (dApps): Etherum-based apps where no single company controls everything.

- AI-powered assistants: Search engine or chatbots that understand meaning, not just keywords.

Web 4.0 - The Future Web (still emerging)

Looking ahead, Web 4.0 is expected to be the "intelligent and fully connected web". It's where AI, virtual reality (VR), augmented reality (AR), & the Internet of Things (IoT) came together to create seamless interaction between the digital and physical worlds.

Examples (future vision)

- A smart fridge that orders groceries automatically.
- Virtual reality classrooms where students & teachers interact as if they were in the same room.
- AI powered assistants that anticipate your needs before you even ask.

1.
2.
3.
4.
5.

Date / /

Client - Server Architecture

Introduction

- Client - Server Architecture is a model used in computer networks

- In this model, tasks are divided between two main parts:
  - Client → The requester (user's device)
  - Server → The provider (powerful computer that gives the service)

- They work together like a team: client requests & server responds.

How Does It Work (Step by step)

1. Client sends a request - e.g., you type www.google.com in your browser.
2. Server receives the request - Google's server gets your query.
3. Server processes the request - find the homepage data
4. Server sends a response - sends back to webpage in packets.
5. Client displays the response - your browser shows the google homepage.

Date / /

Examples in Daily Life

- Web Browsing: Browser (client) ↔ website server.
- Email Service: Gmail app (client) ↔ Gmail server.
- Banking Apps: App (client) ↔ Bank's server.
- Streaming Services: Netflix app (client) ↔ Netflix server.

Advantages of Client-Server Architecture

- Centralized Data Management: All data stored on servers, easy to update.
- Multiple Client Supported: Many users can connect at the same time.
- Better Security: Data can be monitored & secured at the server.
- Scalability: Servers can be upgraded to handle more clients.

Disadvantages of Client-Server Architecture

- Single Point of Failure: If the server crashes, client cannot access services.
- Server Overload: Too many client requests can slow down the server.

Date / /

- Costly setup: High performance servers are expensive to maintain.

Conclusion

- Client-Server Architecture is the backbone of the Internet & most applications we use today.

- It provides a structured way for communication: The client always requests, & the server always responds.

- Although it has some drawbacks, it's efficiency & simplicity makes it the most widely used architecture in computer networking.

HTTP

- HTTP (HyperText Transfer Protocol) is a communication protocol used on the web

- It defines how a client (like a browser) & a server (like a website's server) exchange information.

① HTTP Request -> Response -> Render process.

1. HTTP Request :-

- When you type a website address (like www.google.com) in your browser:

- The browser (client) sends an HTTP request to the web server.

- This request usually asks for resources such as the

Date / /

HTML file, CSS, Javascript, or images.

- Example of a request:

```
GET /index.html HTTP/1.1
Host: www.google.com
```

2. HTTP Response
   - The server recieves the request, processes it, and sends back an HTTP response.
   - The response contains:
     - Status codes (e.g., 200, 404, etc)
     - Headers (extra information like content type, length)
     - Body (The actual data, like the HTML of the webpage).
   - Example of a response:

```
HTTP/1.1 200 OK
Content-Type: text/html
<html>... webpage content ...</html>
```

3. Rendering (Display)
   - The browser receives the response body (HTML code)
   - Then it:
     - Parses the HTML (understands the structure)
     - load extra files (CSS for design, JS for interactivity, imgs, etc)
     - Build the DOM (Document object model) a tree structure of the pg
     - Applies styles & executes scripts.
     - Finally renders (paints) the webpage on your screen.

Date / /

Frontend

- The frontend is the part of a website or app that the user sees and interact with.
- Example: Buttons, text, images, forms, menus, etc.
- It is built using HTML, CSS, Javascript and frontend frameworks & libraries like React, Angular, etc.

Backend

- The backend is the behind the scene part that the user cannot see.
- It handles data storage, server logic, authentication, and processing.
- Example: When you log in, the backend checks your username & password in the database.
- It uses programming languages like Node.js, Python, PHP, Java, C++, etc.

Hosting

- Hosting means putting your website or app on a server connected to the internet, so anyone can access it.
- Example: When you deploy your project on Github pages, Netlify, or AWS, it becomes live for the world.

Date / /

What is TCP

- The internet is all about communication between computers (or devices)
- For this communication to work, we need rules & system so that data can travel safely.
- That's where TCP (Transmission control Protocol) comes in.
- TCP is used almost everywhere - web browsing, email, file sharing, video calls, etc.

Role of TCP with IP

- IP (Internet protocol): Finds the address of the computer (like the house address)
- TCP (Transmission Control Protocol): Makes sure the message is delivered properly.
- Together, TCP/IP is the backbone of the internet.

How TCP Works - a connection establishment (3 way handshake)

- Before sending data, TCP sets up a reliable connection:

1. Client -> SYN: "Hi server, I want to connect."

Date / /

2. Server -> SYN-ACK: "okay, I'm ready. Are you ready too?"

3. Client -> ACK: "Yes, let's start sending data."

- After this handshake, both sides agree on how they'll communicate.

-> The data they share is in the form of packets.

Flow Control and Reliability

- TCP prevents overwhelming the reciever by controlling the speed of data flow.

- If the network is busy (congestion), TCP slows down to avoid packet loss.

Connection Termination

- When communication is done, both sides close the connection properly:

- Client says -> "I'm done sending."
- Server replies -> "okay I'm also done."

- Connection closes.

- This ensures no leftover packets are lost in between.

Why TCP is Important

- Without TCP, data could:
- Get lost
- Arrive in the wrong order.

Date / /

- Be corrupted or incomplete.

* TCP makes the internet reliable & ensures you see a complete webpage, recieve full emails, or download entire files.

What is UDP

- UDP (user Datagram protocol) is another important communication protocol used on the internet.
- Unlike TCP, UDP is connectionless & lightweight.
- It does not gaurentee delivery, order, or error connection
- Because of this, UDP is faster than TCP but less reliable.

☆ The sender just starts sending packets without asking, "Are you ready?"

Data Sent as Datagrams

- UDP sends data in datagram (like packets).
- Each datagram has:
  - Source port (who sent it)
  - Destination port (who should recieve it)
  - length (size of the data)
  - Checksum (for simple error detection, but no correction)

☆ If a datagram is lost, UDP does not resend it.

Date / /

No Reliability or Ordering

- UDP does not track pack/datagram order.
- If packets/datagram arrive late or out of order then the application must handle it.
- If some packets/datagrams never arrive → they are simply gone.

Fast Transmission

- Since there's no extra work like error-checking, retransmission, or handshake, UDP is very fast.
- It's ideal for real time communication where speed matters more than accuracy.

Where UDP is Used

- Online games (missing one packet/datagram won't stop the game).
- Video streaming (a missing frame is okay, but delay is not).
- Voice call (VoIP) Voice over Internet Protocol (you prefer a little glitch over waiting).
- Live broadcast (better to see something in real time than perfectly later).

Date / /

Key Differences Between TCP & UDP

Features	TCP	UDP
Connection	Connection-oriented (need handshake before data transfer)	Connectionless (no handshake, just sends)
Reliability	Reliable - ensures delivery of all packets	Unreliable - packets may be lost, no gaurentee.
Order of Data	Maintains order - rearrange packets if needed.	No order - packets may arrive out of sequence.
Error handling	Error detection & correction.	Only simple error detection (no correction)
Speed	Slower (extra work for reliability)	Faster (no extra checks)
Data size	larger overhead (extra info. in headers)	Smaller overhead (lightweight)
Use-Cases	Web browsing (HTTP/ HTTPS), Email, FTP, remote access (SSH)	Online gaming, Video Audio streaming, Live broadcasts, Voice calls, etc.

Date / /

Introduction to HTTP

- HTTP (HyperText Transfer Protocol) is a foundation of the World wide web.
- It's a protocol (a set of rules) that defines how clients (like browsers) and servers (like website hosts) communicate.
- In other words, it's the language of the web - browser and servers "talk" to each other using HTTP.

* HTTP is based on Client - Server Model.

Characterstics of HTTP

1. Stateless:
   - Each request is independent.
   - Server does not remember previous requests unless cookies / sessions are used.

2. Text - based & Human - readable:
   - Easy to read (GET, POST, 200 OK).

3. Extensible:
   - Headers allow adding new features (security, caching, compression).

4. Flexible:
   - Can transfer any type of data: text, images, video, JSON, etc

Date / /

Versions of HTTP

1. HTTP /0.9 (1991) - The Beginning

- The very first version, extremely simple.
- Could only make GET request (to fetch data).
- Only supported plain HTML (no images, CSS, or multimedia)
- No headers, no status codes.

Example: Ask for a page, server sends back HTML, that's it.

2. HTTP/1.0 (1996) - More Features

- Added headers (extra information like content type).
- Introduced status codes (200 OK, 404 NOT found).
- Allowed sending more than just HTML (like images, audio, video)

Example: To load a webpage with 5 images the browser had to open 6 seperate connections.

Limitation: Each request required a new TCP connection which makes it slower.

3. HTTP/1.1 (1997) - The Standard for Years.

- Became the most widely used version (still used today)
- Introduced persistent connections -> multiple requests in one connection (much faster)
- Added pipelining (send multiple requests without waiting).

Date / /

- Added caching, chunked transfer, better error handling.

- Faster than 1.0, but still not great with many resources (like modern websites with hundreds of images, CSS, JS).

4. HTTP/2 (2015) - Speed Boost

- Designed to solve performance issues of 1.1.

- Key Features:
  - Multiplexing: Sends multiple requests in parallel over a single connection.
  - Header compression: Saves bandwidth.
  - Binary protocol: Faster to process than text.
  - Server push: Server can send resources before the client even asks.

Result: Pages load much faster, especially modern complex websites.

5. HTTP/3 (2022) - The Modern Standard

- Built on QUIC protocol (based on UDP) instead of TCP.

- Advantages:

  - Faster connection setup (no TCP handshake delay).
  - Built-in encryption (always secure, like HTTPS)
  - Handles packet loss better (no need to wait for transmission like TCP).

- Especially good for mobile devices & poor network conditions.

Date / /

- Think of it as HTTP designed for today's internet: video streaming, online gaming, real-time apps.

Why HTTP is Important

- Without HTTP, your browser wouldn't know how to request or display websites.
- All major activities on the internet use HTTP (or HTTPS):
  - Browsing websites
  - Watching Youtube videos.
  - Shopping online.
  - Reading web-based emails.

☆ HTTP is literally the bridge between users & web content.

HTTP status codes

- When a client (like your browser) makes a request to a server, the server replies with a status code along with the response.
- These are 3-digit numbers that tells whether the request was successful, redirect, failed, or had an error.
- They are grouped into categories based on the first digit.

Date / /

1. Information Responses (100-199)

- Used to indicate that the request is still being processed.
- Example:
  - 100 Continue -> Server says "Okay, keep sending the rest of your request."

2. Success Responses (200-299)

- Means the request was successfully received and understood.
- Examples:
  - 200 OK -> Request worked perfectly, and you got the data.
  - 201 Created -> Something new was created (like when you add a new record in a database).
  - 204 No Content -> Request successful but nothing to send back.

3. Redirecting Messages (300-399)

- Tells the client it needs to take some extra steps to complete the request (usually move to another URL).
- Examples:
  - 301 Moved Permanently: Resources has been moved to a new URL permanently.
  - 302 Found (or Temporary Redirect) -> Resources temporarily at another place.
  - 304 Not Modified -> Tells browser to use cached version (saves bandwidth).

Date / /

4. Client Error Responses (400-499)

- The problem is on the client-side (like wrong request or missing page).
- Examples:
  - 400 Bad Request -> The server didn't understand your request (maybe malformed)
  - 401 Unauthorized -> You need to log in first.
  - 403 Forbidden -> You're not allowed to access this resource.
  - 404 Not found -> Page/resource doesn't exist.
  - 429 Too Many Requests -> You've hit the rate limit.

5. Server Error Responses (500-599)

- The request was valid, but the server failed to handle it.
- Example:
  - 500 Internal Server Error -> Something went wrong on the server.
  - 502 Bad Gateway -> Server got an invalid response from another server.
  - 503 Service Unavailable -> Server is down or overloaded.
  - 504 Gateway Timeout -> Server took too long to respond.

Why Status Codes Matter in HTTP

- They help browsers know what to do (reload, redirect, show cached pag. etc)

Date / /

- They help developers debug problems (client error vs server error)

- They are part of the HTTP standard & exist in every single response (even when you don't see them directly).

Introduction to HTTPS

- HTTPS = HTTP over TLS.
- It makes web traffic encrypted, integrity-protected, and authenticated, so data between your browser & a website stays private & you can trust the server's identity.

Goals of HTTPS (why we use it)

- Confidentiality: prevent eavesdroppers from reading data (encryption).

- Integrity: prevent undetected modification of data in transit.

- Authentication: confirm the server really is who it claims to be (certificates / CA)

Major building blocks (the part you must know)

TLS (Transport layer security) - the protocol that does encryption & key exchange.

Public/Private key Cryptography - used to authenticate & establish secure keys.

Date / /

- Symmetric Encryption - used to encrypt the actual HTTP payload once keys are agreed.
- Certificate Authorities (CA) - trusted entities that sign certificates & create the "chain of trust."

What is HTTPS?

- HTTPS stands for HyperText Transfer Protocol Secure.
- It is the secure version of HTTP, the protocol used for communication between a web browser (client) and a web browser.
- The extra "S" at the end means that all communication is encrypted to protect data from hackers or unauthorized access.

Why Do We Need HTTPS?

- On normal HTTP, data is transferred as plain text. This means if someone intercepts the data, they can read it (like passwords, bank details, messages)
- With HTTPS, the data is encrypted before sending, so even if intercepted, it will look like scrambled, unreadable text.

Date / /

- This makes HTTPS essential for:
  - Online banking.
  - E-commerce transactions.
  - Login systems
  - Any website handling sensitive information.

How HTTPS Work (Basic Idea)

- Encryption -> Uses SSL / TLS protocol to encrypt data
- Authentication -> The browser checks the server's SSL/TLS certificate to ensure the website is real is not fake
- Data Integrity -> Ensures the data sent & received is not modified on the way.

Key Features of HTTPS

1. Secure Communication -> Encrypts data (protects from hackers)
2. Trust & Authentication -> Website with HTTPS show padlock icon 🔒 in the browser.
3. Better SEO Ranking -> Google prefers HTTPS websi over HTTP.
4. Prevent Man-in-the-Middle Attack -> Stops attackers from secretly altering communication.

HTTP Vs HTTPS

Date / /

Features	HTTP	HTTPS
Full form	Hyper-Text Transfer Protocol	Hyper-Text Transfer Protocol Secure
Security	Data is sent in plain text -> not secure	Data is encrypted using SSL/TLS -> secure
Port used	Uses Port 80 by default	Uses Port 443 by default
Data Protection	Can be intercepted & read easily	Encrypted, prevents eavesdropping and tampering.
Authentication	No authentication of websites	Uses SSL/TLS certificate to verify authenticity
Speed	Slightly faster (no encryption overhead)	Slightly slower (due to encryption) but modern HTTPS is highly optimized
Browser Indicator	No padlock, sometimes marked as "Not Secure"	Show padlock icon in browser
Use cases	Non-sensitive sites (blogs, public info)	Sensitive sites (banking, e-commerce, login sys
SEO Ranking	lower Ranking on search engines	Preferred by search engines (better SEO

Date / /

SSL and TLS

What are SSL and TLS

- SSL (Secure Sockets Layer) and TLS (Transport layer Security) are security protocols used to protect communication over the internet.

- They are the backbone of HTTPS, ensuring that data exchanged between a client (like a browser) and a server (like a website) is safe.

- TLS is the newer & more secure version of SSL. Today, even though people still say "SSL," in practice, websites mostly use TLS.

Why Do We Need SSL/TLS

- Without SSL/TLS (just HTTP), data is sent as plain text → anyone can read or steal it.

- With SSL/TLS is encrypted → even if hackers intercept it, they only see scrambled text.

- It provides:
  - Encryption: Keeps data private.
  - Authentication: Verify the website is genuine.
  - Integrity: Ensures data is not changes in transit.

How SSL/TLS Work (High-level Idea)

- When you connect to a secure website, the browser & server perform a handshake.

Date / /

- The server shows a digital certificate to prove it's identity.
- Both agreess on an encryption method and securely exchange keys.
- From then on, all data is encrypted and safe.

SSL / TLS Handshake Process

1. Initial Contact
   - The client (browser) says "Hello" to the server.
   - It shares:
     - Supported encryption methods (algorithms).
     - A random number (for key generation).
2. Server Responds
   - Chosen encryption method.
   - It's own random number.
   - It's digital certificate (contains the server's public key and identity)
3. Authentic (Public / Private Keys)
   - The browser checks the certificate with a Certificate Authority (CA) to ensure the server is genuine.
   - Using the server's public key (from the certificate), the client encrypts a special value (called Pre-Master Secret) & sends it to the server.

Date / /

- The server uses it's private key to decrypt the Pre-Master Secret.

⊙ At this point, both sides have the same secret value

4. Creating a Shared Session Key

- Using the random numbers + the Pre-Master Secret, both the client & the server generate the same session key.

- This session key is now used for symmetric encryption (same key for both sides)

5. Secure Communication Begins

- From now on:
  - Data is encrypted & decrypted using the shared session key.
  - This is much faster than using public/private keys for everything.

Proxy and Reverse Proxy

What is Proxy

- A proxy is like a middleman between a user (client) and the internet.

- Instead of connecting directly to websites, your request first goes to the proxy, and then the proxy forwards it to the internet

Date / /

- When the response come back, the proxy sends it to you.

Key Points about Proxy

1. Hides Your Identity -> The website sees the proxy's IP address, not yours.
2. Filtering & Blocking -> Organizations can use proxies to block certain websites.
3. Caching -> Frequently visited web pages can be stored in the proxy to load faster.
4. Security -> Can protect internal networks by hiding user details.

What is Reverse Proxy

- A reverse proxy also sits in the middle, but it works in the opposite direction.
- It stands in front of one or more servers, & clients (users) connect to the reverse proxy, not directly to the server.
- The reverse proxy then decides which server should handle the request & passes it along.

Date / /

Key Points about Reverse Proxy

1. Load Balancing: Distributes traffic across multiple servers to avoid overload.
2. Security: Hides the actual servers, so attackers can't directly reach them.
3. Caching: Stores responses so future requests can be served faster.
4. SSL Termination: Handles encryption (HTTPS) so backend servers don't have to.

What is VPN

- A VPN (Virtual Private Network) is a technology that creates a secure and private connection between your device and the internet.
- Instead of sending data directly to websites, your data goes through a VPN server, which hides your real identity & encrypts everything you send or receive.

A VPN acts like a secure tunnel between you and the internet.

Date / /

Why do we need a VPN

- Normally, when you browse the internet:
  - Your ISP & sometimes hackers can see your activity.
  - Websites can track you through IP address
- With a VPN
  - Your IP address is hidden
  - Your data is encrypted
  - Your online activity stays private.

How Does VPN Work.

1. Connection Setup
   - You connect to a VPN application (on phone, laptop, etc)
   - It creates a secure tunnel b/w your device and the VPN server.
2. Encryption
   - All the internet traffic is encrypted (scrambled)
   - Even if ISP or hackers intercept it, they can't read it.
3. IP Address Change
   - The VPN server replaces your real IP address with it's own.

Date / /

- Website's only see the VPN server's IP, not yours.

4. Data Transfer

- The VPN forwards your request to the internet.
- The website responds back to the VPN server → which then sends it to you securely.

Benefits of using a VPN

1. Privacy Protection: Hides browsing activities from ISP, hackers, and governments.
2. Data Security: Encrypts sensitive information (like banking, passwords).
3. Bypass Restrictions: Lets you access blocked websites (example: using a VPN to watch region-locked show on Netflix).
4. Secure Public WiFi: Protects you when using free WiFi in cafes, airports, etc.
5. Remote Work Access: Employees can securely access company networks from home.

Proxy Vs VPN

Date / /

Features	Proxy	VPN
Main_Purpose	Hides the user's IP address	Hides IP & encrypts all internet traffic.
Security level	low → only makes IP, doesn't encrypt data.	High → Encrypt all data (secure tunnel)
Data Encryption	No encryption (data can be read if intercepted)	Strong encryption (SSL/TLS)
Scope	Works at the application lvl (e.g. browser proxy affects only browser traffic)	Works on the system level (encrypts traffic from all apps)
Privacy	Provides anonymity but limited security.	Provide anonymity + strong privacy + security
Use Cases	Bypass website restriction.	Enable remote work.