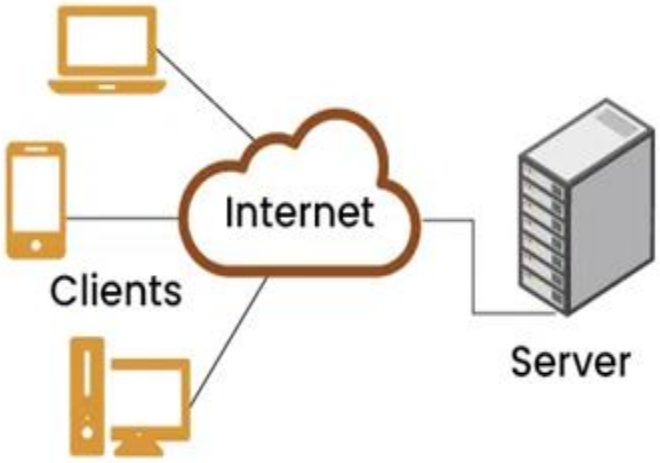
Theory Assignment-1

- Module-1-Overview Of IT Industry

Q-1 Research and create a diagram of how data is transmitte from a client to a server over the internet.

Ans :-

In a client-server model, data transmission over the internet involves a client initiating a request, which is then processed server and a response being sent back to the client.



[Daigram-client architecture]

In detail:-

Step 1: User Enters a URL

The user opens a web browser and types in a website address like www.example.com.

Step 2: DNS Resolution

The browser contacts a DNS (Domain Name System) server to find the IP address of the website. This helps the computer locate the correct server.

Step 3: Establishing a Connection

Once the IP address is known, the client starts a connection with the server using TCP (Transmission Control Protocol), which ensures the connection is stable and reliable.

Step 4: Securing the Connection (if HTTPS)

If the website uses HTTPS, a secure connection is created using TLS (Transport Layer Security). This makes sure the data stays private and protected.

Step 5: Sending a Request

The client sends an HTTP request to the server. This request asks for the content of the webpage (like text, images, etc.).

Step 6: Data Travels Through the Internet

The request travels through various routers and networks until it reaches the correct server.

Step 7: Server Processes the Request

The server receives the request, understands what is needed, and prepares a response (usually a webpage).

Step 8: Server Sends Back Data

The server sends the requested data back to the client in small packets.

Step 9: Browser Displays the Website

The client's browser puts all the packets together and shows the final webpage to the user.

Q-2 Explain the function of the TCP/IP model and its layers.

Ans:-

Function of the TCP/IP Model-

The TCP/IP model (Transmission Control Protocol/Internet Protocol) is a set of rules that allows computers to communicate over the internet. It helps organize how data is sent and received between devices.

The model breaks down communication into layers, where each layer has a specific job. This makes it easier to manage and troubleshoot network problems.

- **Layers of the TCP/IP Model**

The TCP/IP model has 4 main layers:

1. Application Layer

Function: This is where communication starts. It provides services like email, web browsing, file transfer, etc.

Examples: HTTP, FTP, SMTP, DNS

User Interaction: This is the layer closest to the user.

2. Transport Layer

Function: Manages the delivery of data between devices. It breaks data into smaller pieces and ensures it arrives correctly.

Main Protocols:

TCP (Transmission Control Protocol): Reliable and error-checked.

UDP (User Datagram Protocol): Faster but less reliable.

Features: Error checking, data flow control.

3. Internet Layer

Function: Chooses the best path for data to travel and moves it between networks.

Main Protocol: IP (Internet Protocol)

Job: Adds IP addresses and routes the data packets.

4. Network Access Layer (Link Layer)

Function: Handles how data is physically sent over cables or wireless.

Includes: Device drivers, network interface cards (NICs), Ethernet, WiFi

Job: Moves data within the local network.

Q-3 Explain Client Server Communication.

Ans:-

Client-Server Communication is a model in computer networks and software architecture where two parties—**client** and **server**—interact over a network. It is the foundation for most internet and web-based applications.

🔁 How It Works

1. **Client**: A client is a device or software (like a web browser or mobile app) that initiates a **request** for data or service.
2. **Server**: A server is a powerful computer or software that **processes the request** and sends back the **response** (data or result).

Example: When you open a website, your browser (client) requests a webpage from the server. The server processes the request and sends back the webpage data to be displayed.

🧭 Basic Flow of Communication

```
SCSS

Client                →              Server
(Request)                            (Receives)
                                     ↓
                                Process the request
                                     ↓
Client                ←              Server
(Receives)                           (Response)
```

⚙ Steps in Detail

1. **Connection Established**: The client connects to the server (usually over HTTP, HTTPS, or TCP/IP).
2. **Request Sent**: The client sends a request (like GET, POST).
3. **Processing**: The server handles the request using backend logic, databases, etc.
4. **Response Sent**: The server sends data back (HTML, JSON, files).
5. **Client Processes**: The client receives and renders or uses the response.

Q-4 Research different types of internet connections (e.g., broadband, fiber, satellite) and list their pros and cons.

Ans:-

1. Broadband (DSL and Cable)

➢ What is it?

Broadband is a high-speed internet connection. It can be delivered through DSL (Digital Subscriber Line) using telephone lines or through cable using TV cables.

✅ Pros:

- Faster than dial-up
- Always connected (no need to dial)
- Available in most cities and towns
- Affordable for most households

❌ Cons:

- Speed can reduce during peak hours
- Cable broadband is shared, so performance may vary
- DSL may be slower if you are far from the service provider

2. Fiber Optic Internet

➢ What is it?

Fiber internet uses thin glass or plastic fibers to send data at very high speeds using light signals.

✅ Pros:

- Extremely fast (up to 1 Gbps or more)
- Reliable and stable connection
- Great for streaming, gaming, and video calls
- Upload and download speeds are almost equal

❌ Cons:

- Expensive installation
- Not available in all areas (mainly in cities)
- Takes time to expand to rural areas

3. Satellite Internet

➢ What is it?

Satellite internet sends signals from a satellite in space to a dish installed at your home. It’s often used in remote or rural areas.

✅ Pros:

- Available almost anywhere on Earth
- Good for areas where other options don’t exist
- Wireless connection (no cables needed)

❌ Cons:

- Slower compared to fiber or broadband
- High latency (delay), bad for gaming or live calls
- Can be affected by weather (rain, storms)
- Expensive data plans

4. Mobile Internet (4G/5G)

➢ What is it?

Mobile internet uses your mobile phone network (4G or 5G) to connect to the internet. You can use it on your phone or hotspot it to other devices.

✅ Pros:

- Portable and wireless
- Easy to set up
- Fast speeds with 4G and 5G
- Great for travel or temporary use

❌ Cons:

- Data limits can be expensive
- Speeds vary based on location
- Battery usage is high if used as a hotspot

5. Dial-Up Internet (Old Method)

➢ What is it?

Dial-up uses a telephone line to connect to the internet. It's very slow and not common today.

✅ Pros:

- Very cheap
- Can work in areas without modern infrastructure

❌ Cons:

- Extremely slow (not suitable for modern usage)
- Can’t use phone and internet at the same time
- Outdated technology

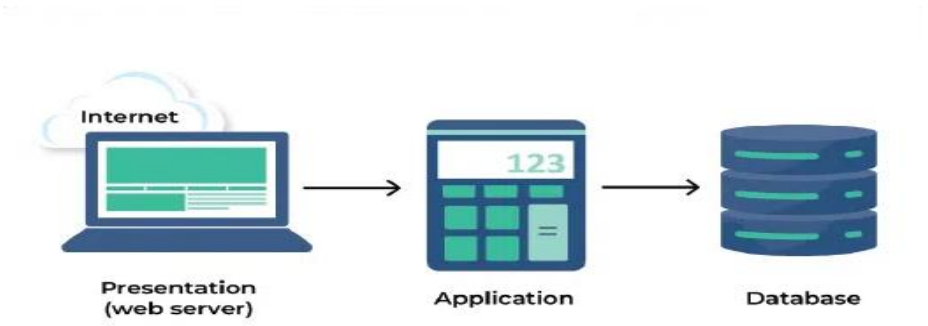
Q-5 Design a basic three-tier software architecture diagram for a web application

Ans:-

In software engineering, a three-tier architecture is a popular design used in modern web applications. It divides the application into three separate layers, each with its own responsibilities. This makes the app easier to manage, update, and scale.

Three Tiers:

1. **Presentation Layer (Client/User Interface)**
2. **Application Layer (Business Logic)**
3. **Data Layer (Database)**



1. Presentation Tier (Client Tier/User Interface):

Function:

This is the user interface (UI) layer that users directly interact with. It handles displaying information to users and collecting their input.

Examples:

Web browsers, mobile apps, or desktop applications.

Technology:

HTML, CSS, JavaScript for web applications; native UI frameworks for mobile or desktop apps.

Interaction:

Sends user requests to the Application Tier and displays the results received from it.

2. Application Tier (Business Logic Tier):

Function: This layer contains the core logic and functionality of the application. It processes user requests, performs calculations, and interacts with the Data Tier.

Examples: Servers running application code (e.g., Java, Python, Node.js).

Technology: Programming languages like Java, Python, Node.js, etc.

Interaction: Receives requests from the Presentation Tier, retrieves data from or stores data in the Data Tier, and sends results back to the Presentation Tier.

3. Data Tier (Database Tier):

Function: This layer manages and stores the application's data. It provides a persistent storage solution for the application.

Examples: Databases (SQL or NoSQL), data warehouses.

Technology: Databases like MySQL, PostgreSQL, MongoDB, etc.

Interaction: Stores, retrieves, and updates data based on requests from the Application Tier.

Diagram Representation:

A simple visual representation would show:

Client (Presentation Tier): Connected to the internet, sends requests.

Web Server (Application Tier): Receives requests from the client.

Application Server (Application Tier): Processes requests and interacts with the database.

Database (Data Tier): Stores and retrieves data.

Q-6 Create a case study on the functionality of the presentation, business logic, and data access layers of a given software system.

Ans:-

◆ Overview

An Online Bookstore is a web-based application where users can browse, search, and purchase books. The system is built using a **three-tier architecture**, which separates the application into:

- **Presentation Layer (PL)**
- **Business Logic Layer (BLL)**
- **Data Access Layer (DAL)**

◆ 1. Presentation Layer (PL)

Function: This is the user interface of the application. It interacts with the user and sends requests to the Business Logic Layer.

✓Responsibilities:

- Display book catalog
- Handle user login & registration
- Capture search and purchase inputs
- Show confirmation messages or error alerts

🛠️ *Technologies Used:*

- HTML, CSS, JavaScript, React (for frontend UI)
- REST APIs (for communication with backend)

📌 *Example Functionality:*

- A user visits the homepage and sees a list of featured books.
- They use a search bar to look for "Data Structures."
- The frontend sends this query to the backend via a REST API.

◆ 2. Business Logic Layer (BLL)

Function: This layer contains the core functionality of the application. It processes user inputs, enforces rules, and coordinates between the UI and the database.

✅ *Responsibilities:*

- Validate user input (e.g., correct login credentials)
- Implement business rules (e.g., discounts for members)
- Process orders (e.g., total price, tax calculation)
- Manage session and cart data

🛠️ *Technologies Used:*

- Node.js, Java, or Python (for backend logic)
- Express.js or Spring Boot (as server framework)

📌 *Example Functionality:*

- Receives the search term "Data Structures"
- Calls the DAL to retrieve matching books
- Filters results based on availability
- Sends book list to Presentation Layer to display

◆ 3. Data Access Layer (DAL)

Function: This layer interacts directly with the database to fetch, insert, update, or delete data.

✓Responsibilities:

- Execute SQL queries
- Handle database connections
- Convert data to and from application objects
- Provide secure data operations

🛠 Technologies Used:

- SQL (MySQL, PostgreSQL) or NoSQL (MongoDB)
- ORM Tools (Hibernate, Sequelize, Mongoose)

📌 Example Functionality:

- Receives a request to fetch books with the title "Data Structures"

Q-7 Explore different types of software environments (development,testing,production).Setup a basic environment in a virtual machine.

Ans:-

🔍 Part 1: Exploring Software Environments

In software development, different **environments** are used to separate stages of the software lifecycle. These ensure the software is built, tested, and deployed in controlled ways.

◆ 1. Development Environment

Used by developers to write and debug code.

- **Purpose**: Code creation, unit testing, debugging.
- **Tools**:
  - IDEs (VS Code, IntelliJ, Eclipse)
  - Local servers (e.g., XAMPP, WAMP, Node.js)
  - Local databases (SQLite, MySQL)
- **Example**: A developer writes a login module on their laptop using Python and Flask.

◆ 2. Testing Environment

A separate environment used for QA and automated tests.

- **Purpose**: Perform integration tests, regression tests, performance tests.
- **Tools**:
  - Jenkins, Selenium, JUnit
  - Mock servers, test databases
- **Example**: After a developer commits code, Jenkins runs tests to verify that it doesn't break existing features.

◆ 3. Production Environment

Live environment accessed by end-users.

- **Purpose**: Run the final version of the application.
- **Characteristics**:
  - High availability
  - Monitoring and logging enabled (e.g., Prometheus, ELK Stack)

- **Example**: The actual website or app (like amazon.in) users interact with.

💻 Part 2: Setting Up a Basic Environment in a Virtual Machine (VM)

We'll set up a **basic development + testing environment** in a VM using **Ubuntu + Apache + MySQL + PHP (LAMP stack)**.

🔧 Steps to Set Up Environment in a VM

👇 You can use **VirtualBox** or **VMware** to create your virtual machine.

❓ Step 1: Install Ubuntu on Virtual Machine

1. Download Ubuntu ISO
2. Create a new VM in VirtualBox:
    - Set memory: 2–4 GB RAM
    - Set storage: 20–30 GB
    - Mount the ISO and install Ubuntu

Q-8 Create a list of software you use regularly and classify them into the Following categories: system, application, and utility software.
Ans:-

🖥️ 1. System Software

Manages hardware and provides a platform for application software.

Software Name	Purpose
Windows 11 / Ubuntu / macOS	Operating systems to manage system resources
Device Drivers (e.g., printer driver)	Enable communication between OS and hardware
BIOS / UEFI	Low-level software to start the system
Android / iOS	Operating systems for mobile devices

📝 2. Application Software

Performs specific tasks for users.

Software Name	Purpose
MS Word / Google Docs	Word processing
Google Chrome / Firefox	Browsing the internet
WhatsApp / Telegram	Messaging and communication
Spotify / VLC Media Player	Playing music and videos
Zoom / Microsoft Teams	Video conferencing
Photoshop / Canva	Image editing
Tally / MS Excel	Accounting and spreadsheets
VS Code / PyCharm	Code editing and development

🛠️ 3. Utility Software

Helps in maintenance and performance of the system.

Software Name	Purpose
Antivirus (e.g., Avast, Windows Defender)	Protects against malware
WinRAR / 7-Zip	File compression and extraction
Disk Cleanup / CCleaner	Removes unnecessary files
Backup Software (e.g., Acronis, Windows Backup)	Data backup and restore
Task Manager	Monitors system performance
Firewall	Controls incoming/outgoing network traffic

Q-9 Follow a GIT tutorial to practice cloning, branching, and merging repositories

Ans:-

🔹 1. What is Git?

Git is a **version control system** that tracks changes in files and coordinates work among multiple people. It's commonly used with GitHub or GitLab for collaborative software projects.

🔹 2. Git Workflow Overview

A typical Git workflow includes:

1. **Cloning** a repository
2. **Creating and switching branches** to isolate changes
3. **Committing** changes locally
4. **Merging** branches back together
5. **Pushing** changes to a remote repository

❖ A. Cloning a Repository

- **What it means**: Cloning creates a **local copy** of a remote repository (e.g., from GitHub).
- **When to use**: You want to start working on an existing project.

Conceptual Steps:

1. Go to a remote repository on GitHub.
2. Use the **clone URL** (e.g., HTTPS).
3. Run the git clone command to copy it to your local machine.

✅ You now have a full working copy, including its history.

❖ B. Creating and Switching Branches

- **What it means**: Branching lets you create a **separate workspace** for changes so you don't affect the main project.
- **When to use**: Working on a new feature, fixing a bug, or testing something.

Conceptual Steps:

1. From the main branch, create a new branch (e.g., feature-1).
2. Switch to that branch to work on changes.

✅ Your changes are isolated and won't impact the main project until merged.

❖ C. Merging Branches

- **What it means**: Merging combines changes from one branch into another (usually into main).

- **When to use**: Once a feature or fix is complete and tested.

Conceptual Steps:

1. Go back to the main branch.
2. Merge the feature branch into main.
3. Resolve any **merge conflicts** if two branches changed the same part of a file.

✅ The main branch now includes all the changes from the feature branch.

🔷 D. Pushing to Remote Repository

- **What it means**: Push sends your local changes back to the remote repository on GitHub.
- **When to use**: After you've merged or committed changes locally.

✅ Your collaborators can now see or pull your changes.

Q-10 How does GIT improve collaboration in a software development team?
Ans-

Git significantly improves collaboration in a software development team by providing a **structured, flexible, and secure way to manage and track changes** in code. Here's how:

🔹 1. Distributed Version Control

- Every team member has a **full copy of the project history**.
- Developers can work **independently** and offline.
- No reliance on a central server to make or track changes.

✅ This allows multiple people to work simultaneously without blocking each other.

🔷 2. Branching and Merging

- Git allows developers to **create branches** for features, bug fixes, or experiments.
- Branches are **isolated** from the main codebase.
- Once changes are ready, branches are **merged** back into the main branch.

✅ This enables **parallel development** without interfering with the main code.

🔷 3. Clear History and Change Tracking

- Git keeps a detailed **commit history** of what was changed, when, and by whom.
- Each commit includes a message that explains the reason for the change.

✅ This helps the team understand the **evolution of the project** and simplifies debugging.

🔷 4. Collaboration Through Pull Requests (PRs)

- On platforms like GitHub or GitLab, team members can open PRs to propose changes.
- Other members can **review, comment, and approve** the changes before they are merged.

✅ Encourages **code review, knowledge sharing, and quality control**.

🔹 5. Conflict Resolution

- When multiple people edit the same file, Git detects **conflicts**.
- Developers are prompted to **manually resolve differences** before merging.

✅ Ensures all changes are acknowledged and agreed upon.

🔹 6. Rollbacks and Recovery

- Git allows you to **revert to previous versions** of files or the entire project.
- Mistakes can be undone without losing the rest of the work.

✅ Increases **confidence and safety** in making changes.

🔹 7. Continuous Integration and Deployment (CI/CD)

- Git integrates with CI/CD tools to:
  - **Automatically test** code after each commit
  - **Deploy** code after approval or merge

✅ Enables faster and safer delivery of software.

Q-11 Write a report on the various types of application software and how they improve productivity.

Ans:-

◆ **Types of Application Software and Their Productivity Benefits**

1. Word Processing Software

Examples: Microsoft Word, Google Docs, LibreOffice Writer
Functions: Create, edit, format, and print text documents.

How It Improves Productivity:

- Templates and auto-correct features save time.
- Collaboration tools allow multiple users to work on the same document.
- Easy formatting enhances the professional quality of documents.

---

2. Spreadsheet Software

Examples: Microsoft Excel, Google Sheets, LibreOffice Calc
Functions: Data organization, analysis, and visualization using rows, columns, and formulas.

How It Improves Productivity:

- Automates complex calculations and data analysis.
- Graphs and charts simplify data interpretation.
- Saves time in budgeting, forecasting, and financial planning.

---

3. Presentation Software

Examples: Microsoft PowerPoint, Google Slides, Prezi
Functions: Create visual aids for communication such as slideshows.

How It Improves Productivity:

- Simplifies idea sharing and reporting.
- Enhances audience engagement through visual content.
- Supports collaboration and feedback.

---

4. Database Management Software

Examples: Microsoft Access, MySQL, Oracle, SQLite
Functions: Store, retrieve, and manage large volumes of data efficiently.

How It Improves Productivity:

- Organizes and secures data for quick access.
- Reduces redundancy through relational databases.
- Useful for managing customers, inventory, and transactions.

---

5. Communication Software

Examples: Zoom, Microsoft Teams, Slack, Skype, Gmail
Functions: Facilitate real-time communication via text, voice, or video.

How It Improves Productivity:

- Speeds up decision-making with instant communication.
- Enables remote work and global collaboration.
- Centralizes team messages and files.

---

6. Graphic Design and Multimedia Software

Examples: Adobe Photoshop, Canva, CorelDRAW, Adobe Premiere Pro
Functions: Create and edit images, videos, and other multimedia content.

How It Improves Productivity:

- Streamlines content creation for marketing and education.
- Templates and presets save time for designers.
- Enhances creativity and brand presentation.

---

7. Web Browsers

Examples: Google Chrome, Mozilla Firefox, Microsoft Edge
Functions: Access and browse websites.

How It Improves Productivity:

- Provides fast access to information and tools.
- Supports extensions for note-taking, grammar checking, etc.
- Facilitates online learning, research, and cloud services.

---

8. Project Management Software

Examples: Trello, Asana, Jira, Monday.com
Functions: Organize tasks, track progress, assign responsibilities.

How It Improves Productivity:

- Visual task boards help prioritize and manage workloads.
- Real-time tracking increases accountability.
- Improves team coordination and goal alignment.

---

9. Accounting and Financial Software

Examples: Tally, QuickBooks, Zoho Books
Functions: Manage income, expenses, payroll, and taxes.

How It Improves Productivity:

- Automates billing, payroll, and financial reporting.

- Reduces manual errors in calculations.
- Speeds up auditing and compliance processes.

Q-12 What is the role of application software in businesses?
Ans:-

🔹 Key Roles of Application Software in Business

1. Automation of Routine Tasks

- Reduces manual work (e.g., invoicing, payroll, data entry).
- Increases accuracy and saves time.
- Examples: Microsoft Excel, QuickBooks, SAP

📌 *Benefit:* Frees up employees to focus on higher-value activities.

---

2. Efficient Communication

- Facilitates real-time messaging, email, video conferencing.
- Enables collaboration across departments and remote teams.
- Examples: Microsoft Teams, Zoom, Gmail

📌 *Benefit:* Enhances teamwork and decision-making speed.

---

3. Data Management and Analysis

- Collects, stores, and analyzes large volumes of data.
- Helps in forecasting, reporting, and strategic planning.
- Examples: Microsoft Access, SQL-based tools, Tableau

📌 *Benefit:* Empowers data-driven decisions and better customer understanding.

---

4. Customer Relationship Management (CRM)

- Tracks customer interactions, sales, and support.
- Helps manage marketing campaigns and follow-ups.
- Examples: Salesforce, Zoho CRM, HubSpot

📌 *Benefit:* Builds stronger customer relationships and boosts sales.

---

5. Financial Management

- Manages budgeting, billing, taxation, and payroll.
- Automates reporting and compliance.
- Examples: Tally, QuickBooks, FreshBooks

📌 *Benefit:* Improves financial accuracy and reduces errors.

---

6. Project and Task Management

- Organizes tasks, deadlines, and team responsibilities.
- Tracks progress and resource allocation.
- Examples: Asana, Trello, Jira

📌 *Benefit:* Increases transparency and project delivery speed.

---

7. Inventory and Supply Chain Management

- Monitors stock levels, orders, and logistics.
- Ensures timely restocking and delivery.
- Examples: SAP, Oracle NetSuite, Zoho Inventory

📌 *Benefit:* Reduces waste and improves customer fulfillment.

---

8. Marketing and Content Creation

- Designs graphics, edits videos, manages campaigns.
- Schedules social media posts and tracks engagement.
- Examples: Adobe Photoshop, Canva, Mailchimp

📌 *Benefit:* Enhances brand visibility and customer engagement.

Q-13 Perform a functional analysis for an online shopping system.

Ans:-

✅ What is Functional Analysis?

Functional analysis means **breaking down what the system does** — all the features and functions it should have for different users.

👥 Who Uses the System? (User Roles)

1. **Customer** – Buys products online
2. **Admin** – Manages the website and products
3. *(Optional)* **Vendor** – Sells products through the platform
4. *(Optional)* **Delivery Staff** – Updates delivery status

🛒 What Can Each User Do? (Functions)

🎁 A. Customer Functions

Function	What It Does
Sign Up / Login	Create an account and log in
Browse Products	Look through product categories
Search Products	Find products using keywords

Function	What It Does
View Product Details	Check images, price, reviews
Add to Cart	Select products to buy later
Checkout & Payment	Enter address, choose payment, place order
Track Order	See delivery status
Write Reviews	Share feedback after buying

🛠️ B. Admin Functions

Function	What It Does
Login	Access the admin dashboard
Manage Products	Add, edit, or remove products
Manage Orders	View and process customer orders
Manage Users	View or block users
View Reports	See sales and performance data

🏪 C. Optional Vendor Functions (If multi-vendor site)

Function	What It Does
Add Products	Upload and manage their products
View Orders	See what customers ordered from them

🔐 Other System Functions (Behind the Scenes)

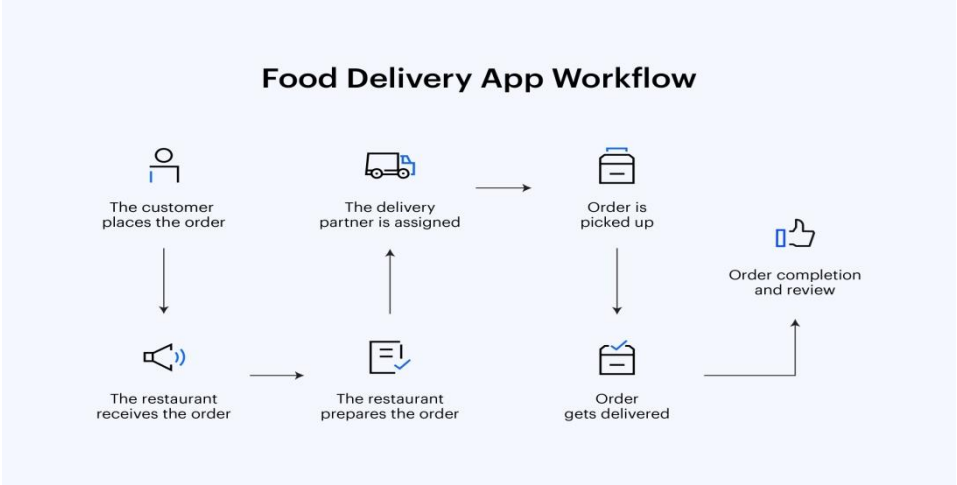
Function	What It Does
Login Security	Keeps user data safe
Email Notifications	Sends updates to users
Mobile Friendly	Works on phone and computer
Online Payment	Connects to payment systems like UPI, cards

🧍 Example Use Case: A Customer Buys a Shirt

1. Logs into their account
2. Searches for “blue shirt”
3. Views details, adds to cart
4. Proceeds to checkout, enters address
5. Pays online
6. Receives confirmation email
7. Later writes a review

Q-14 Design a basic system architecture for a food delivery app

Ans:-



.

A food delivery app connects customers, restaurants, and delivery agents through a structured system. It includes:

- **Frontend (User Apps):** Mobile or web apps for customers, restaurants, and delivery agents.
- **Backend Server:** Handles orders, user accounts, menus, and delivery logic.
- **Database:** Stores user data, orders, menus, and transaction history.
- **External Services:** Includes payment gateways, map APIs for tracking, and notification systems for real-time updates.

Q-15 Document a real-world case where a software application required critical maintenance.

Ans:-

🛠️ Real-World Example: Facebook Outage – Critical Software Maintenance

---

📌 What Happened?

On **October 4, 2021**, **Facebook**, along with its apps **WhatsApp**, **Instagram**, and **Messenger**, suddenly stopped working all over the world for more than **6 hours**.

❓ Why Did It Stop Working?

- Facebook's engineers were doing **routine maintenance** on the system that connects all their data centers (the big computers that store and share data).
- During this process, a **wrong setting (configuration)** was applied to their routers (network devices).

- This caused Facebook's systems to be **disconnected from the internet** — like cutting the wires between Facebook and the outside world.

🌍 How Did It Affect People?

Who Was Affected	What Happened
Users	Couldn't open Facebook, WhatsApp, or Instagram
Businesses	Lost contact with customers and missed sales
Facebook	Lost millions of dollars in ad revenue

🔧 How Was It Fixed?

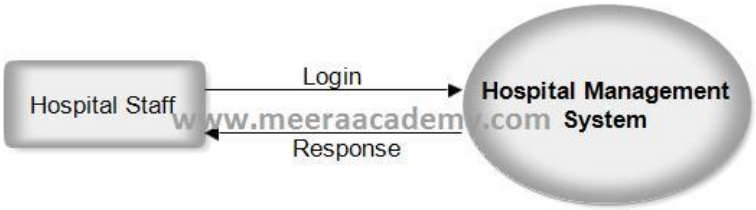
- Engineers had to **go to the actual data centers** physically.
- They manually **fixed the settings** in the routers.
- The system was slowly turned back on to avoid crashing from overload.
- After a few hours, everything came back online.

💡 What Did We Learn?

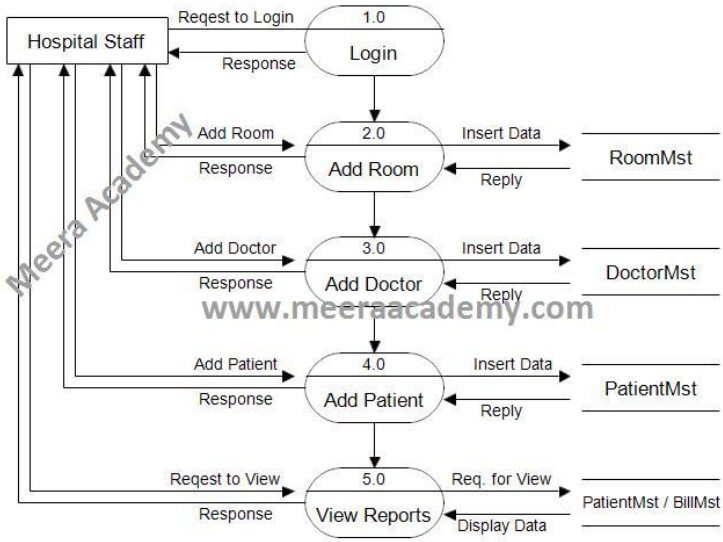
- Even big companies like Facebook can make mistakes.
- Maintenance should always be done **carefully** and with **backups or test systems**.
- It's important to have a **plan to fix things quickly** when something goes wrong.

Q-16 Create a DFD for a hospital management system.
Ans:-

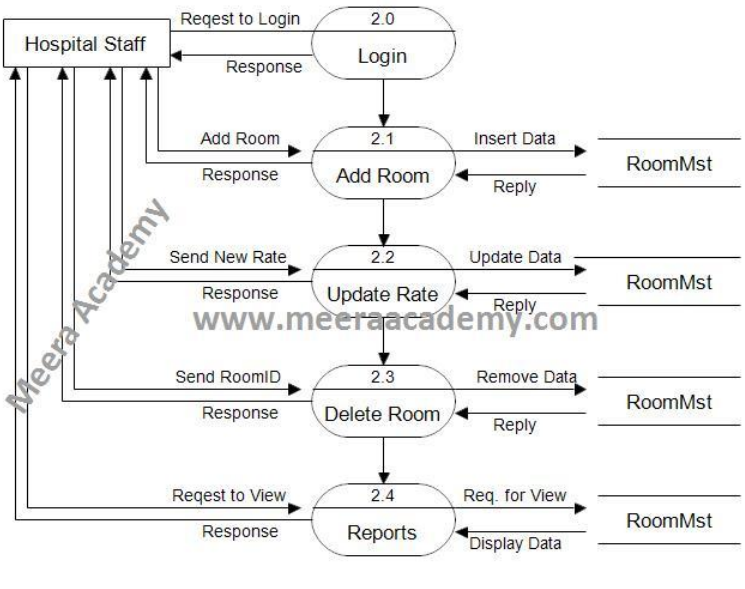
0 Level DFD - Context Level



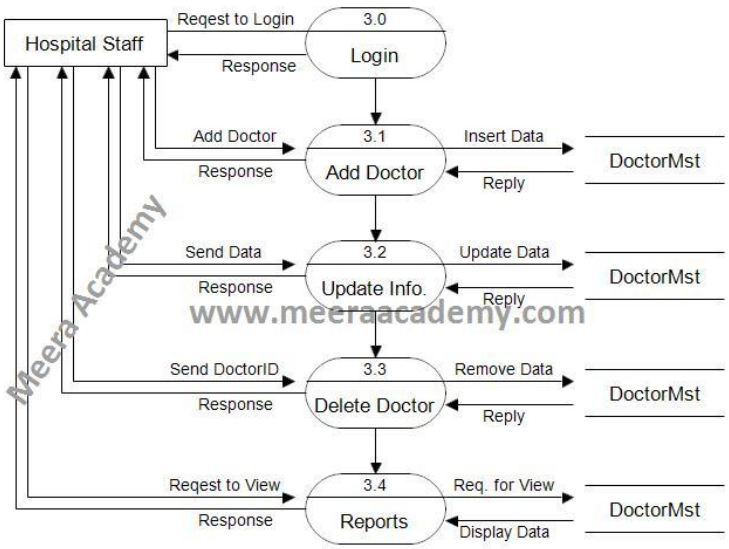
1st Level DFD - Hospital Management System



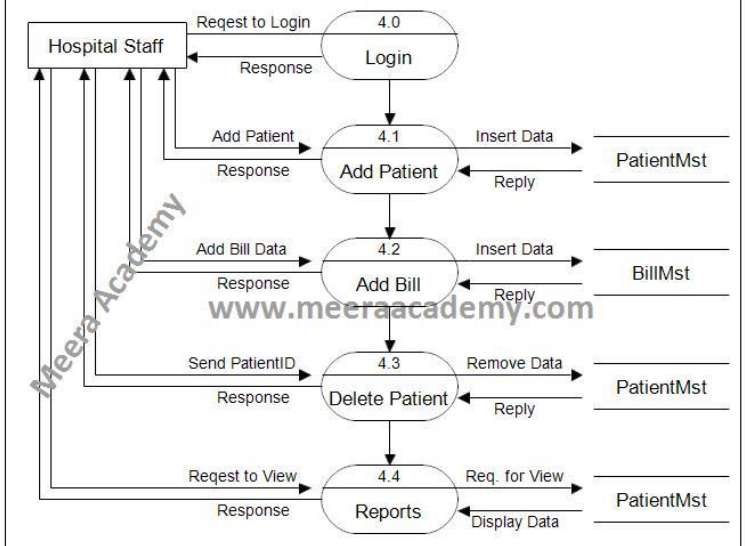
2st Level DFD (2.0) - Hospital Management System

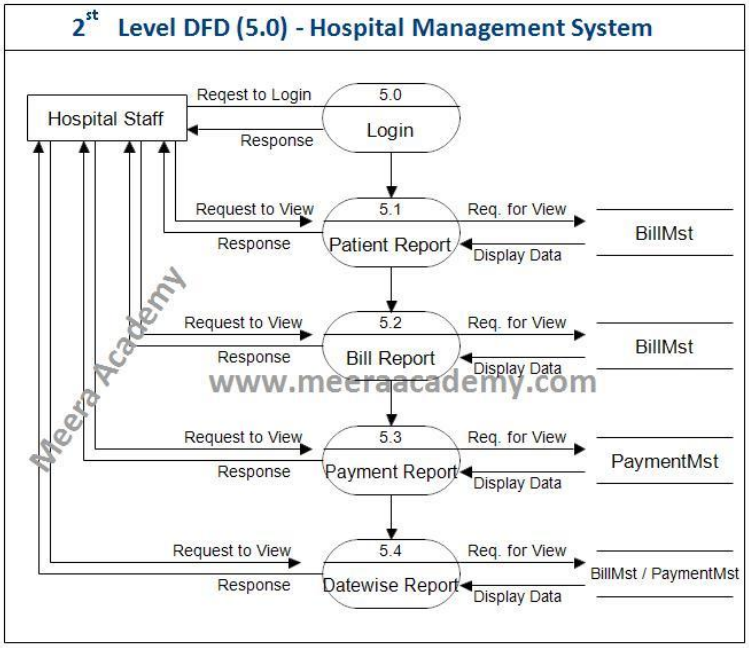


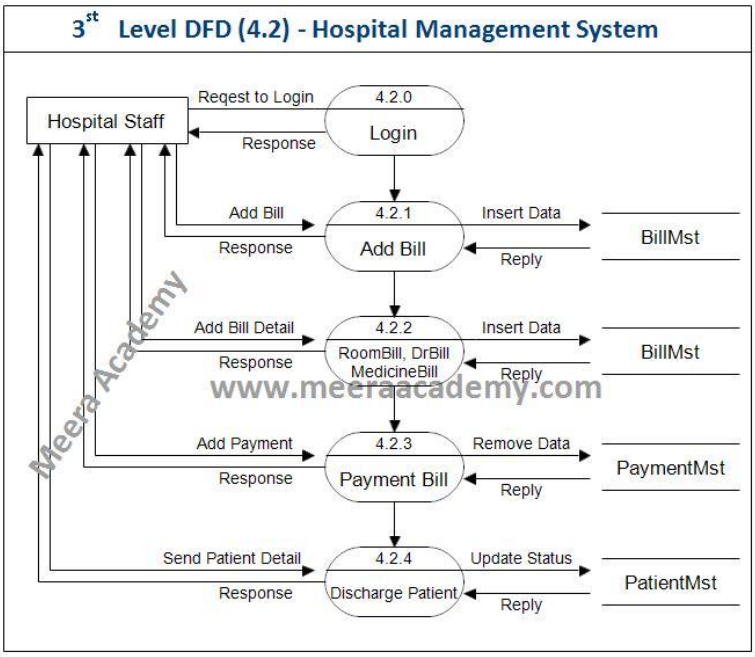
2st Level DFD (3.0) - Hospital Management System



2st Level DFD (4.0) - Hospital Management System

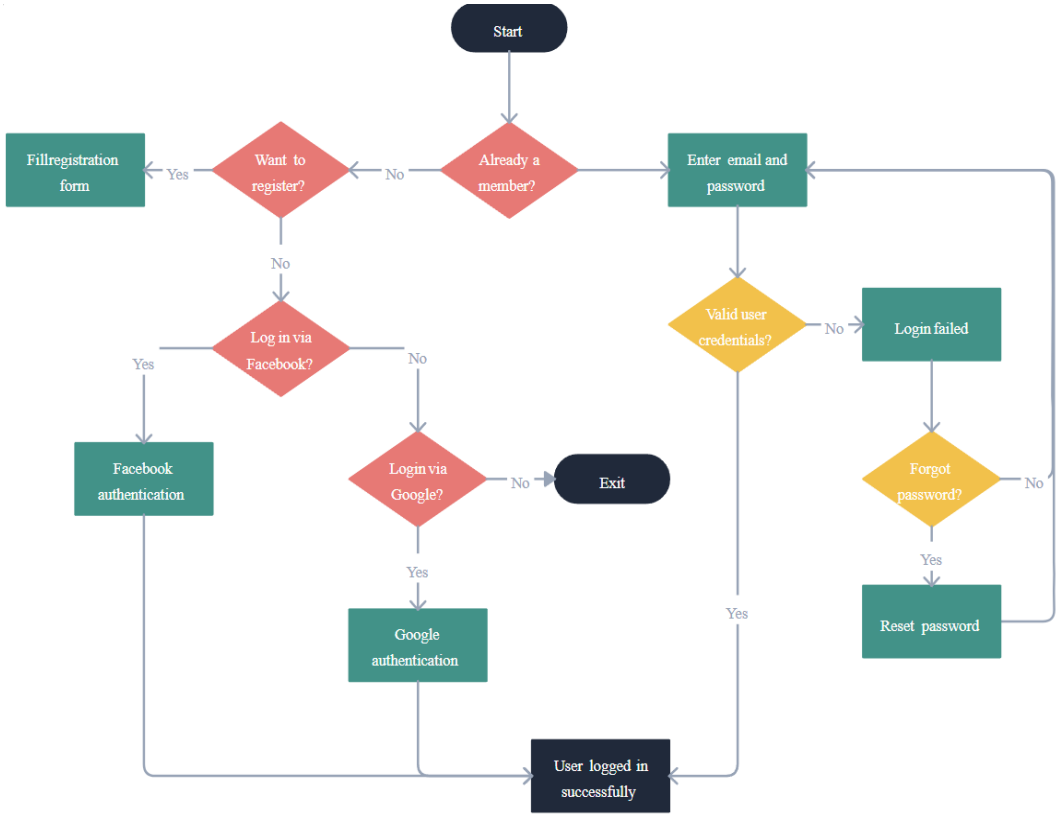






Q-17 Draw a flowchart representing the logic of a basic online registration system.

Ans-



Q-18 What are the main stages of the software development process?

Ans:-

The **main stages of the software development process** are a series of steps that help plan, build, test, and deliver high-quality software. These stages are usually followed in a structured order, often known as the **Software Development Life Cycle (SDLC)**.

Here's a simple breakdown of each stage:

🔹 1. Requirement Gathering and Analysis

Goal: Understand what the software needs to do.

- Meet with clients or users to gather needs.
- Identify what features are required.

- Analyze feasibility (time, cost, resources).

📌 *Example:* A company wants an online food delivery app. The team gathers details about order tracking, menus, payments, etc.

🔷 2. System Design

Goal: Plan how the software will be built.

- Design the software's structure and architecture.
- Create mockups, wireframes, and flow diagrams.
- Decide on technologies, database structure, and interfaces.

📌 *Example:* Designing how the login screen, cart system, and order database will work.

🔷 3. Implementation (Coding)

Goal: Write the actual program code.

- Developers write the code in the chosen programming language.
- Each module or feature is built according to design.
- Code is written in a clean, testable way.

📌 *Example:* Developers create the login system, cart functionality, and payment gateway.

🔷 4. Testing

Goal: Check for errors or bugs.

- Test the software to ensure everything works as expected.

- Perform unit tests, system tests, integration tests, etc.
- Fix any bugs or performance issues found.

📌 *Example:* Test if the cart updates correctly, or if payments go through successfully.

🔷 5. Deployment

Goal: Release the software to users.

- Deploy the software to a live environment (e.g., website or app store).
- May be done in stages (beta, full release).

📌 *Example:* The food delivery app goes live on Play Store and users start ordering.

🔷 6. Maintenance and Updates

Goal: Keep the software running and improve it.

- Fix bugs reported by users.
- Release updates for new features or improvements.
- Monitor performance and security.

📌 *Example:* Adding new payment options or fixing bugs based on user feedback.

Q-19 Write a requirement specification for a simple library management system.

Ans:-

📋 Requirements for a Simple Library Management System

✓ 1. Book Management Requirements

- The system should let the librarian **add new books** (title, author, quantity).
- The librarian should be able to **edit or update book details**.
- The librarian should be able to **delete books** if needed.
- Users should be able to **view a list of available books**.
- There should be an option to **search for books** by name, author, or subject.

✓ 2. Member Management Requirements

- The system should allow the librarian to **add new members** (students or staff).
- The librarian should be able to **update or remove** member details.
- Members should be able to **log in and view** their own book records.

✓ 3. Book Issue and Return Requirements

- The librarian should be able to **issue a book** to a member.
- The system should **not allow issuing a book** that is already borrowed or out of stock.
- The system should **record the issue date and return date**.
- The librarian should be able to **return books** and mark them as available again.
- The system should be able to **calculate a fine** if a book is returned late.

✅ 4. Login and Security Requirements

- The system should have a **login page** for the librarian and members.
- Only librarians should have access to **manage books and members**.
- Members can only **view books and their own records**.

✅ 5. Report Requirements

- The system should generate a report of:
  - All books currently issued
  - Books that are overdue
  - Member borrowing history

✅ 6. Other Requirements

- A member should be able to **borrow up to 5 books** at a time.
- A book can be borrowed for **14 days**.
- The system should charge a **fine of ₹5 per day** for late returns.
- The system should be **easy to use and fast**.

Q-20 What are the key elements of system design?

Ans:-

🔑 Key Elements of System Design

---

1. Architecture Design

This is the **overall structure** of the system.

- Decides how the system is divided into parts (modules or layers).
- Common architectures: **Client-Server**, **Three-tier**, **Microservices**, etc.
- Focuses on how components talk to each other.

📌 *Think of it like a building blueprint.*

---

2. Data Design

How data is **organized, stored, and accessed**.

- Choose how to design **databases** (tables, fields, relationships).
- Decide on **data formats** (e.g., JSON, XML).
- Ensure **data integrity, normalization**, and **security**.

📌 *Example: Designing how book records are stored in a library system.*

---

3. User Interface (UI) Design

How the system looks and interacts with users.

- Layout of screens, buttons, forms, and navigation.
- Should be **simple, consistent, and user-friendly**.
- Focus on **usability and accessibility**.

📌 *Example: The login page or dashboard in an app.*

---

4. Functional Design

What the system **should do**.

- Describes all **features and functions**.
- Includes **inputs, processes, and outputs**.
- Based on user requirements and use cases.

📌 *Example: "Search books by title" or "Issue a book to a student" in a library system.*

---

5. Security Design

How the system is **protected** from threats.

- Includes **user authentication**, **data encryption**, **access control**, etc.
- Ensures data is **safe and private**.
- Must be considered from the beginning.

📌 *Example: Only librarians can delete books from the system.*

---

6. Performance and Scalability

How well the system works **under load**.

- Plan for **speed**, **response time**, and ability to **handle growth**.
- Includes caching, load balancing, database optimization, etc.

📌 *Example: A shopping app that runs fast even during festival sales.*

---

7. Error Handling and Recovery

What happens when things **go wrong**.

- Design how the system handles errors (e.g., wrong input, server failure).

- Should give **clear messages** and allow recovery.

📌 *Example: "Password is incorrect" message during login.*

---

8. Integration and Interfaces

How the system **connects** with other systems.

- APIs, payment gateways, third-party tools, etc.
- Ensure smooth communication between different services.

📌 *Example: Connecting your app to a payment service like Razorpay.*

---

9. Deployment Design

How the system will be **installed and updated**.

- Decide on **servers, hosting**, cloud services, and backups.
- Plan for updates without affecting users.

📌 *Example: Publishing a new version of an app without downtime.*

Q-21 What are the key differences between web and desktop applications?

Ans:-

Web Application vs. Desktop Application

Feature	Desktop Application	Web Application
Platform Dependency	OS-specific (e.g., Windows, macOS)	Platform-independent (browser-based)
Installation	Installed on system	No installation, accessed via URL
Internet Connection	Often works offline	Needs internet connection (mostly)
Deployment & Updates	Manual or auto-updates	Centralized and instant updates
Performance	High, uses full system resources	Depends on browser & internet speed
Security	Local system security	Web security (XSS, CSRF, etc.)
User Interface	Rich, native UI	Browser-limited UI (responsive design)
Access to Resources	Full system access (files, USBs)	Limited browser-level access
Maintenance	Updates per device	Server-side maintenance
Examples	MS Word, Photoshop	Gmail, Google Docs

Q-22 Build a simple desktop calculator application using a GUI library.

Ans:-

- **Tools Used:**
  - Programming Language: Python
  - GUI Library: Tkinter (built-in GUI library in Python)
  - Platform: Windows or any OS with Python installed

- **Project Description:**

The calculator application allows the user to:

- Enter two numbers
- Choose an arithmetic operation
- See the result displayed on the screen
- Use buttons just like a real calculator

➢ **It features:**

Number buttons (0–9)

Operation buttons (+, –, ×, ÷)

Clear (C) and Equal (=) buttons

Display area for input and output

Step-by-Step Process:-

➢ **Step 1: Design the Layout:-**

- Use a simple window
- Add a display area (like a text field or label)
- Arrange buttons in a grid (numbers, operations, etc.)

➢ **Step 2: Create the GUI Window:-**

- Set the window title (e.g., "Simple Calculator")
- Define window size and background color (optional)

➢ **Step 3: Add Buttons and Display:-**

- Add number buttons (0–9)
- Add operator buttons (+, –, ×, ÷)
- Add "=" to calculate the result
- Add "C" to clear the screen

➢ **Step 4: Write Functions:-**

- A function to update the input when a button is clicked
- A function to evaluate the expression when "=" is pressed
- A function to clear the input when "C" is clicked

- **Step 5: Run and Test :-**

- Try different operations (e.g., 5 + 3 = 8)
- Check edge cases like division by zero
- Fix any errors or bugs

Q-23 Design a simple HTTP client-server communication in any Language.

Ans:-

- **Client-Server Model**

- **How Does the Client-Server Model Work?**

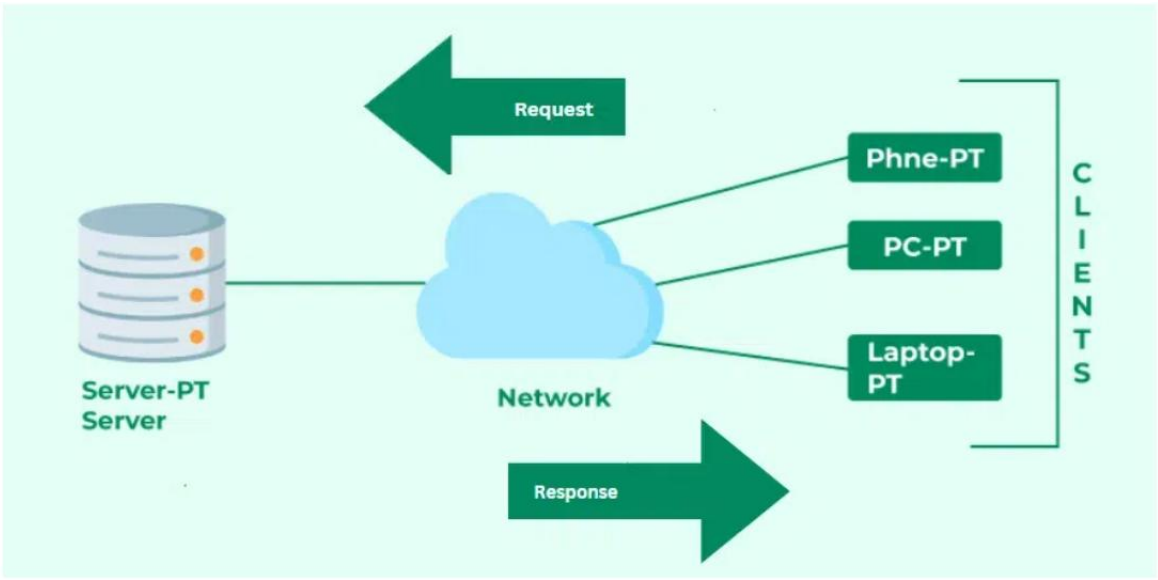
In this article, we are going to take a dive into the **Client-Server** model and have a look at how the **Internet** works via, web browsers. This article will help us have a solid WEB foundation and help us easily work with WEB technologies.

- **Client**

When we talk about a "Client," it refers to a **device** (usually a computer, smartphone, or application) that requests and receives services from a server. The client is the entity that initiates communication, asking for data or resources from the server. For instance, **web browsers** like **Google Chrome**, **Mozilla Firefox**, or **Safari** are common client applications that request data from a server to render web pages.

- **Server**

A **Server**, on the other hand, is a **remote computer or system** that provides data, resources, or services to clients. It listens to incoming client requests, processes them, and sends the required information back. A server can handle multiple client requests simultaneously.
For example, **Web servers** host websites, and **database servers** store and serve databases for applications. **In simple terms**, the client sends a request to the server, and the server serves the request as long as the data or service is available in its system.



Client Server Model

Q-24 Develop test cases for a simple calculator program.

Ans:-

Test Cases for Simple Calculator

✓ Assumptions:

- Calculator takes **two numeric inputs**
- Supports: +, -, *, /
- Handles **division by zero**
- Accepts **decimal numbers**
- Supports both **positive and negative numbers**

Test Case Table

Test Case ID	Input 1	Input 2	Operation	Expected Output	Description
TC01	5	3	+	8	Basic addition

Test Case ID	Input 1	Input 2	Operation	Expected Output	Description
TC02	10	4	-	6	Basic subtraction
TC03	7	2	*	14	Basic multiplication
TC04	8	2	/	4	Basic division
TC05	5	0	/	Error/Infinity	Division by zero
TC06	-5	-3	+	-8	Addition with negative numbers
TC07	-10	4	-	-14	Mixed positive/negative subtraction
TC08	3.5	2.5	+	6.0	Decimal addition
TC09	2	3	^	Error	Unsupported operation
TC10	"a"	5	+	Error	Invalid input (non-numeric)
TC11	100000	200000	+	300000	Large number addition
TC12	0	0	+	0	Adding two zeros

⍰ Edge Cases

Test Case ID	Input 1	Input 2	Operation	Expected Output	Notes
TC13	-0.0	0.0	+	0.0	Test with zero sign handling
TC14	1e10	1e10	*	1e20	Large float multiplication
TC15	1	3	/	0.333...	Precision test for division

✓ Things to Test

- All 4 basic operations
- Negative numbers
- Decimals and large values
- Division by zero
- Invalid or missing inputs
- Unsupported operations (like %, ^)
- Result display and formatting

Practical Assignment

Q-1 Write a simple "Hello World" program in two different programming languages of your choice.

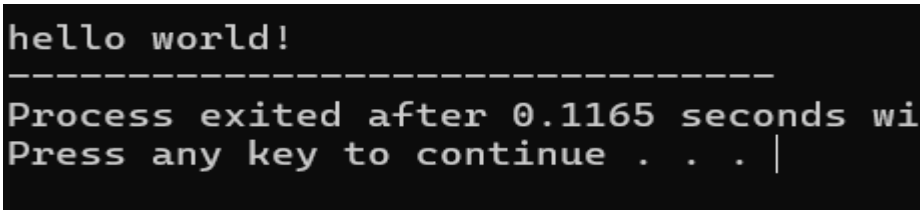
Ans-

C language:

```
#include<stdio.h>
main()

    printf("hello world!");
}
```

OUTPUT:



Python language:

```
print("hello world!")
```

OUTPUT: