1. What is Program?

-> A program is a set of instructions that a computer or other device follows to perform a specific task.

2. Write a simple “Hello World” program in two different programming languages of your choice. Compare the structure and syntax.

-> C:

```
#include<stdio.h>
int main() {
printf(“Hello World”);
}
```

Python:

```
print(“Hello World”)
```

Feature	C	Python
Program Structure	Requires full structure (headers, main function)	Very simple, no mandatory structure
Execution Start	Begins at main() function	Top-to-bottom execution
Printing	Uses printf() from stdio.h	Uses built-in print()
Semicolon	Required after each statement	Not required
Compilation	Must be compiled (e.g., gcc file.c)	Interpreted directly by Python
Type System	Statically typed	Dynamically typed
Length	Longer, more formal	Shorter, easier for beginners

3. Explain in your own words what a program is and how it functions.

-> A set of instructions written by a programmer to tell the computer what to do. These instructions are written in a programming language (like C) and then compiled into machine code that the computer can execute.

4. What is Programming?

-> **Programming** is the process of writing instructions (code) that tell a computer what to do. It involves creating programs to solve problems, perform tasks, or control how software works.

5. What are the key steps involved in the programming process?

-> a. Understand the Problem

b. Plan the Solution

c. Write the Code

d. Compile/Run the Program

e. Test the Program

f. Debug

g. Maintain/Improve

6. Types of Programming Language.

-> C,C++,HTML,CSS,JS,Java,Python.

7. What are the main differences between high-level and low-level programming languages?

-> **High-Level Languages**

Examples: **C, Python, Java**

- Easy to read and write
- Close to human language
- Less control over hardware
- Portable across different machines
- Slower compared to low-level (because they need translation)

Low-Level Languages

Examples: **Assembly, Machine language**

- Harder to read and write
- Close to computer hardware
- Gives full control over memory and CPU
- Not portable (depends on the machine)
- Very fast and efficient

8. World Wide Web & How the Internet Works.

-> The **World Wide Web (WWW)** is a large collection of web pages and websites that you can view using a browser. It uses the internet to send and receive information.

The internet itself is a massive global network that connects millions of computers and devices. When you open a website, your device sends a request through your router and internet provider to the website's server, and the server sends the page back to your browser so you can see it. The Web is just one part of the internet.

9. Reasearch and create a diagram of how data is transmitted from a client to server over the internet.

->

When a client (like a computer or smartphone) sends data to a server, the information doesn't travel as a single block. Instead, it is broken into smaller units and sent across the network, where it is reassembled at the server.

1. Client-Side Preparation

- **Data Segmentation:** The client divides the information into small chunks called **packets**.
- **Adding Headers:** Each packet is given a header containing details like:
    - Source and destination IP addresses
    - Packet number for order tracking

2. Packet Transmission

- **Travel Across the Network:** Packets are sent from the client into the internet. They pass through multiple devices such as routers and switches.
- **Flexible Routing:** The internet selects the best path for each packet. Some packets may take different routes depending on traffic or network conditions, ensuring faster and reliable delivery.

3. Server-Side Handling

- **Receiving Packets:** The server collects all incoming packets, which may arrive in any order.
- **Reassembling Data:** Using the sequence numbers in the packet headers, the server reconstructs the original data.

10. Describe the roles of the client and server in web communication.

-> **1. Client**

The client is the device or application that requests services or resources from a server. Examples include web browsers, mobile apps, or email clients.

Key roles:

- **Initiates Requests:** Sends requests to the server for data, files, or services.
- **Processes Responses:** Receives data from the server and presents it to the user (e.g., renders a webpage).
- **User Interaction:** Acts as the interface between the user and the server.

Example: When you type www.example.com in a browser, your browser (client) requests the webpage from the server.

2. Server

The server is a system that provides resources or services in response to client requests. Examples include web servers, database servers, and email servers.

Key roles:

- **Receives Requests:** Listens for incoming requests from clients.
- **Processes Requests:** Performs necessary actions (retrieving data, running applications, etc.).
- **Sends Responses:** Sends data or confirmation back to the client.

Example: When a web server receives a request for www.example.com, it sends the HTML, CSS, and images for the webpage back to the client.

11. Network Layers on Client and Server.

-> **Network Layers on Client and Server**

Both client and server use the **TCP/IP protocol stack**, which has four layers. Their roles at each layer are slightly different:

Layer	Client Role	Server Role
Application	Generates requests (HTTP, FTP, SMTP)	Receives requests and prepares responses (HTTP, FTP, SMTP)
Transport	Breaks data into packets and ensures delivery (TCP/UDP)	Reassembles packets and ensures reliable delivery (TCP/UDP)
Network	Adds IP addresses to packets and routes them	Reads destination IP to receive packets and route replies
Data Link & Physical	Sends bits over physical media (Ethernet, Wi-Fi)	Receives bits and converts to frames for upper layers

12 .Design a simple HTTP client-server communication in any language.

-> Server:

```
import http.server
import socketserver

PORT = 8000

Handler = http.server.SimpleHTTPRequestHandler

with socketserver.TCPServer(("", PORT), Handler) as httpd:
    print("Serving at port", PORT)
    httpd.serve_forever()
```

Client:

```
import http.client

HOST = 'localhost'
PORT = 8000
FILE_PATH = '/index.html'  # Requesting a file named index.html

try:
    conn = http.client.HTTPConnection(HOST, PORT)
    conn.request("GET", FILE_PATH)
    response = conn.getresponse()

    print(f"Status: {response.status}")
    print(f"Reason: {response.reason}")

    data = response.read()
    print(f"Response Body:\n{data.decode('utf-8')}")

except ConnectionRefusedError:
    print(f"Error: Connection refused. Is the server running on {
except Exception as e:
    print(f"An error occurred: {e}")
finally:
    if 'conn' in locals() and conn:
        conn.close()
```

13. Explain the function of the TCP/IP model and its layers. Client and Servers.

-> **Function of the TCP/IP Model**

The **TCP/IP model** is a set of communication protocols used to interconnect network devices on the internet. Its main function is to provide a standardized framework that enables devices to send, receive, and interpret data reliably across different networks.

Layers of the TCP/IP Model:

Application Layer

Transport Layer

Internet Layer

Network Access Layer

Client

- **Initiates Requests:** Sends requests to servers for data or services.
- **Uses Application Layer Protocols:** e.g., a web browser using HTTP or HTTPS.
- **Receives and Processes Responses:** Displays content to users or processes received data.

Server

- **Waits for Requests:** Listens for incoming connections from clients.
- **Processes Requests:** Retrieves data, performs tasks, or runs applications as needed.
- **Sends Responses:** Delivers requested information or confirmation back to the client.

14. Client and Servers.

-> **Client and Server** are the two main components in a **networked system** that communicate to provide and use services.

Client:

- A client is a device or program that **requests services or resources** from a server.
- Examples: Web browsers, email apps, or mobile apps.
- It **initiates communication** and depends on the server to respond.

Server:

- A server is a device or program that **provides services or resources** to clients.
- Examples: Web servers, database servers, file servers.
- It **waits for client requests** and responds with the requested data or service.

15. Explain Client Server Communication.

-> Client-server communication is a network model in which a client, such as a computer or browser, requests services or data from a server, which is a system that provides resources or information. The client initiates the communication by sending a request over the network. The server receives the request, processes it, and sends back an appropriate response. This interaction allows the client to access resources, such as web pages, files, or database information, while the server handles multiple client requests simultaneously. Communication between the client and server typically follows standard protocols like TCP/IP to ensure data is transmitted accurately and reliably.

16.Types of Internet Connections.

->

1. Dial-Up
2. DSL (Digital Subscriber Line)
3. Cable Internet
4. Fiber-Optic Internet
5. Satellite Internet
6. Mobile/Cellular Internet
7. Wireless Broadband (Wi-Fi)
8. Leased Line / Dedicated Line

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

->**1. Dial-Up Internet**

- **Pros:** Cheap, widely available in remote areas.
- **Cons:** Very slow, occupies the phone line.

2. DSL (Digital Subscriber Line)

- **Pros:** Faster than dial-up, always connected, uses existing phone lines.
- **Cons:** Speed decreases with distance from provider, moderate speed.

3. Cable Internet

- **Pros:** High-speed broadband, supports multiple devices.
- **Cons:** Shared bandwidth can reduce speed, limited availability in some areas.

4. Fiber-Optic Internet

- **Pros:** Extremely fast, reliable, low latency.
- **Cons:** Expensive, not widely available in rural areas.

5. Satellite Internet

- **Pros:** Accessible in remote locations, no physical cables needed.
- **Cons:** High latency, expensive, affected by weather conditions.

6. Mobile/Cellular Internet (3G/4G/5G)

- **Pros:** Portable, wireless, good coverage in urban areas.
- **Cons:** Data limits, speed varies with network congestion.

7. Wireless Broadband (Wi-Fi)

- **Pros:** Convenient, can connect multiple devices, mobile within range.
- **Cons:** Limited range, dependent on underlying broadband connection.

18. How does broadband differ from fiber-optic internet?

-> **Key Differences:**

Feature	Broadband (DSL/Cable)	Fiber-Optic Internet
Medium	Copper cables, coaxial cables	Fiber-optic cables (light signals)
Speed	Moderate to high (up to ~1 Gbps)	Very high (up to several Gbps)
Reliability	Can be affected by distance and interference	Highly reliable, low interference
Latency	Moderate	Very low
Availability	Widely available	Limited in rural areas

19. Protocols.

-> Protocols are sets of rules and standards that define how data is transmitted and received over a network. They ensure that devices and applications can communicate reliably and efficiently.

Key Points:

- Protocols determine the format, timing, sequencing, and error-checking of data exchange.
- They are essential for interoperability between different devices and software.

Common Types of Protocols:

- HTTP/HTTPS: Used for transferring web pages over the internet.
- FTP (File Transfer Protocol): Used for transferring files between computers.
- TCP/IP (Transmission Control Protocol/Internet Protocol): Fundamental protocol for internet communication.
- SMTP/POP3/IMAP: Used for sending and receiving emails.

20. Simulate HTTP and FTP requests using command line tools (e.g., curl).

- -> **GET request:**

curl http://example.com

- **POST request:**

curl -X POST -d "username=user&password=pass" http://example.com/login

- **Save response to file:**

curl -o response.html http://example.com

Simulating FTP Requests with curl

- **Download a file:**

curl -u username:password ftp://ftp.example.com/sample.txt -O

- **Upload a file:**

curl -u username:password -T localfile.txt ftp://ftp.example.com/

curl -u username:password ftp://ftp.example.com/

21. What are the differences between HTTP and HTTPS protocols?

->

Feature	**HTTP (Hypertext Transfer Protocol)**	**HTTPS (Hypertext Transfer Protocol Secure)**
Security	Not secure; data is sent in plain text	Secure; data is encrypted using SSL/TLS
Port Number	80	443
Encryption	No encryption	Encrypts data to prevent interception

Feature	HTTP (Hypertext Transfer Protocol)	HTTPS (Hypertext Transfer Protocol Secure)
Authentication	No authentication	Server authentication via SSL/TLS certificates
Data Integrity	Data can be modified during transmission	Ensures data is not altered in transit
Usage	Basic websites or non-sensitive data	Online banking, e-commerce, login pages

22. Application Security.

->Application Security refers to the process of protecting software applications from threats, vulnerabilities, and attacks throughout their lifecycle. Its goal is to ensure that applications are safe, reliable, and trustworthy for users.

Key Points:

- Focuses on preventing unauthorized access, data breaches, and misuse of applications.
- Involves techniques such as encryption, authentication, authorization, input validation, and secure coding practices.
- Applies to web, mobile, desktop, and cloud applications.
- Regular security testing and updates are crucial to protect against evolving threats.

23. Identify and explain three common application security vulnerabilities. Suggest possible solutions.

-> 1. SQL Injection (SQLi)

Explanation:

- Occurs when an attacker inserts malicious SQL code into input fields to manipulate a database.
- Can lead to unauthorized access, data theft, or data deletion.

Solution:

- Use **parameterized queries or prepared statements**.
- Validate and sanitize user input.
- Limit database permissions to only what is necessary.

2. Cross-Site Scripting (XSS)

Explanation:

- Happens when an attacker injects malicious scripts into a website, which then run in other users' browsers.
- Can steal cookies, session tokens, or deface websites.

Solution:

- **Sanitize and encode user input/output**.
- Use Content Security Policy (CSP) headers.
- Avoid directly inserting user data into HTML or JavaScript.

3. Insecure Authentication

Explanation:

- Weak or improper authentication allows attackers to guess passwords, hijack sessions, or bypass login.

Solution:

- Enforce **strong password policies** and multi-factor authentication (MFA).
- Use secure password storage (hashed and salted).
- Implement account lockout policies after repeated failed login attempts.

24. What is the role of encryption in securing applications?

-> **Key roles include:**

- **Data Confidentiality:** Ensures that sensitive information cannot be read by unauthorized users.
- **Data Integrity:** Helps detect if data has been tampered with during transmission or storage.
- **Authentication:** Verifies the identity of users or systems by using encryption-based certificates or keys.
- **Secure Communication:** Protects data exchanged between clients and servers (e.g., HTTPS uses TLS/SSL encryption).

25. Software Applications and Its Types .

-> **Software Applications**

A **software application** is a program or set of programs designed to perform specific tasks for users. Applications help users accomplish tasks such as creating documents, browsing the web, managing data, or playing games.

Types of Software Applications :

1. System Software
2. Application Software
3. Programming Software
4. Utility Software

26.Identify and classify 5 applications you use daily as either system software or application software.

-> Below are 5 applications I use daily:

1. **Windows 11 – System Software**
   It’s the operating system on my laptop; without it, nothing runs.

2. **Google Chrome – Application Software**
   I open it every day to browse the internet, watch YouTube, and complete assignments.
3. **Microsoft Edge / Windows Defender – System Software**
   Windows Defender runs in the background all the time to protect my PC.
4. **WhatsApp Desktop – Application Software**
   I use it daily to chat with friends and group members for projects.
5. **VLC Media Player – Application Software**
   I use it to watch downloaded lectures and movies every day.

27. What is the difference between system software and application software?

->

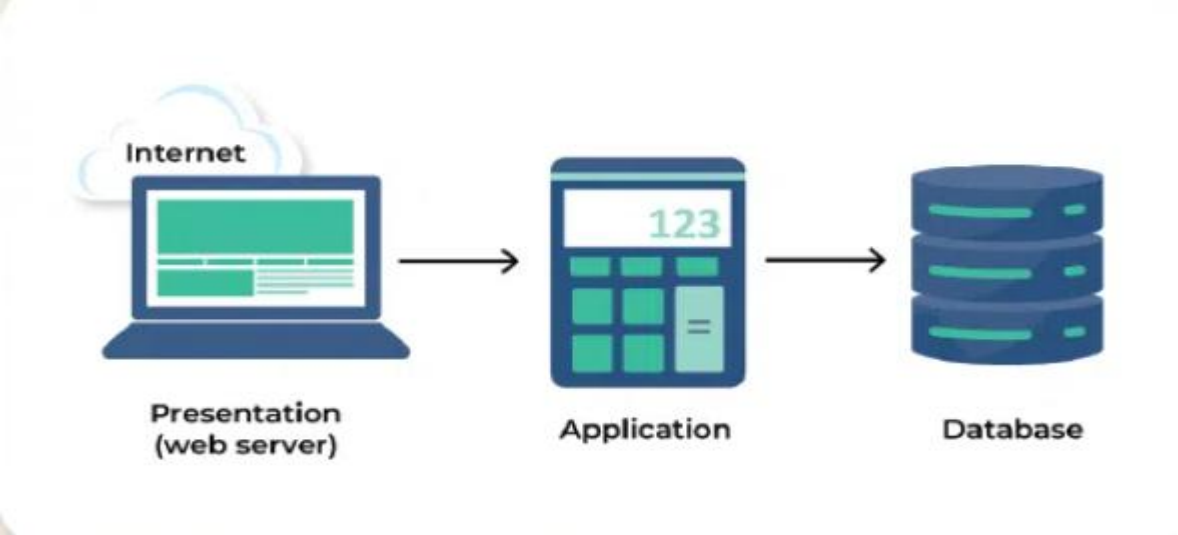
Feature	System Software	Application Software
Purpose	Manages and controls computer hardware and provides a platform for applications	Performs specific tasks for the user, like creating documents or browsing the web
Examples	Operating systems (Windows, macOS, Linux), device drivers, utilities	MS Word, Chrome, VLC Media Player, WhatsApp
Interaction	Runs in the background and interacts directly with hardware	Runs on top of system software and interacts with the user
Necessity	Essential for the computer to function	Not essential; installed as per user needs
User Control	Minimal user control; works automatically	Fully controlled and used by the user

28. Software architecture.

-> Software architecture refers to the overall structure and organization of a software system. It describes how the system is divided into major components and how these components interact with each other to perform the required functions. Architecture acts like a blueprint that guides developers, helping them understand the system's design, technologies used, and the way different parts communicate. A good software architecture ensures that the system is easy to maintain, secure, scalable, and capable of handling changes in the - future.

29. Design a basic three-tier software architecture diagram for a web application.

->



30. What is the significance of modularity in software architecture?

-> Modularity in software architecture is significant because it breaks a large system into smaller, independent, and manageable parts called modules. Each module focuses on a specific function, making the overall system easier to understand, develop, and maintain. When the system is modular, changes in one part do not heavily affect other parts, which improves flexibility and reduces the chance of errors. It also allows developers to work on different modules simultaneously, speeding up development and improving productivity. Modularity enhances reusability as well, since well-designed modules can be reused in other projects or systems. Overall, modularity leads to cleaner architecture, better scalability, and simpler debugging.

31. Layers in Software architecture.

-> A layered architecture is a design pattern where the software system is divided into hierarchical layers, each interacting only with the layer directly below it. Each layer has a specific role and is responsible for a certain aspect of the system.

Types of Layers:

- **Presentation Layer:** Handles user interface and interactions.
- **Business Logic Layer:** Implements core functionality and rules.
- **Data Access Layer:** Manages communication with databases.
- **Database Layer:** Stores and retrieves persistent data.

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

-> **Case Study: Online Bookstore System**

1. Presentation Layer:
The user interacts with the system through a web interface or mobile app. Customers can browse books, search by title or author, and add items to their cart. The presentation layer collects user input and displays search results, book details, and the shopping cart.

2. Business Logic Layer:
This layer processes user requests and applies the application rules. For example, when a user adds a book to the cart, the system checks stock availability, calculates the total price including discounts or taxes, and updates the order summary. It ensures all operations follow the bookstore's business rules.

3. Data Access Layer:
The data access layer communicates with the database to retrieve and store information. It fetches book details, user account info, and order history from the database, and updates inventory when an order is placed. It abstracts the database operations so the business layer doesn't deal with raw queries.

33. Why are layers important in software architecture?

-> Layers are important in software architecture because they organize a system into separate, manageable sections, each with a specific responsibility. This separation of concerns improves modularity, maintainability, and scalability, allowing developers to update or replace one layer without affecting others. Layers also make the system easier to understand, test, and debug, and they help reuse components across different parts of the application.

34. Software Environments.

-> Software Environments refer to the platforms and settings in which software is developed, tested, and executed. They provide the necessary tools, libraries, and configurations for software to run correctly.

Types of Software Environments:

- **Development Environment:** Where developers write and test code, often includes IDEs, compilers, and debugging tools.
- **Testing/QA Environment:** Used to test software for bugs and performance before release; simulates real-world usage.
- **Staging/Pre-Production Environment:** Mirrors the production environment closely to validate final release readiness.
- **Production Environment:** The live environment where end-users interact with the fully deployed software.

35. Explore different types of software environments (development, testing, production).Set up a basic environment in a virtual machine.

-> **Types of Software Environments**

1. **Development Environment:**
    - Used by developers to write, compile, and debug code.
    - Tools include IDEs (like Visual Studio, Eclipse), compilers, version control (Git), and local databases.

2. **Testing/QA Environment:**
    - Used to test software for bugs, performance, and security.
    - Mimics real-world usage but is separate from live data.
3. **Production Environment:**
    - The live environment where end-users interact with the software.
    - Must be stable, secure, and fully monitored.

Setting up a Basic Environment in a Virtual Machine (VM)

Step 1: Install a VM software

- Use **VirtualBox** or **VMware**.

Step 2: Create a new virtual machine

- Choose OS (e.g., Ubuntu, Windows).
- Allocate CPU, RAM, and disk space.

Step 3: Install the OS

- Boot the VM with the OS ISO file and follow installation instructions.

Step 4: Set up the environment

- **Development:** Install IDE (VS Code, Eclipse), Git, Python/Java runtime, and database (MySQL/PostgreSQL).
- **Testing:** Install testing tools (Selenium, JUnit, Postman).
- **Production:** Configure web server (Apache/Nginx), database, and deploy application.

Step 5: Test connectivity and functionality

- Ensure code runs, database connections work, and web services are accessible.

36. Explain the importance of a development environment in software production.

-> A development environment is crucial in software production because it provides a controlled space where developers can write, test, and debug code without affecting the live system. It ensures that the software is built correctly, efficiently, and safely, allowing developers to experiment, detect errors early, and integrate new features. By simulating real-world conditions, it helps maintain code quality, consistency, and collaboration among team members, ultimately reducing bugs and improving the reliability of the final product.

37. What is the difference between source code and machine code?

-> **Source Code:**

- Written by humans in high-level programming languages (e.g., Python, Java, C++).
- Readable and understandable by programmers.
- Needs to be compiled or interpreted to run on a computer.

Machine Code:

- Consists of binary instructions (0s and 1s) that the computer's CPU can execute directly.
- Not human-readable.
- Generated from source code through a compiler or assembler.

38. Github and Introductions.

-> GitHub is a web-based platform used for version control and collaborative software development. It allows developers to store, manage, and track changes in their code using Git, a distributed version control system.

Introduction to GitHub:

- Developers can create repositories to store projects.

- Supports branching and merging, enabling multiple people to work on the same project without conflicts.
- Provides collaboration features like pull requests, issues, and code reviews.
- Hosts both public (open-source) and private projects.

39. Why is version control important in software development?

-> **Version control** is important in software development because it **tracks and manages changes** to the code over time. It allows developers to **collaborate safely**, revert to previous versions if errors occur, and maintain a history of modifications. Version control also **prevents code conflicts**, supports branching for experimenting with new features, and ensures the software is **reliable, maintainable, and organized** throughout its development lifecycle.

40. What are the benefits of using Github for students?

-> **Benefits of using GitHub for students:**

1. **Version Control Practice:** Students learn to track changes, revert code, and manage multiple versions of projects.
2. **Collaboration Skills:** Enables teamwork through branching, pull requests, and code reviews.
3. **Portfolio Building:** Students can showcase their projects publicly to potential employers.
4. **Access to Open Source:** Explore and contribute to real-world projects for learning experience.
5. **Integration with Tools:** Works with IDEs, CI/CD tools, and project management platforms.
6. **Backup and Accessibility:** Stores code in the cloud, accessible from anywhere.

41. Types of Software.

-> **System Software:**

- Manages and controls computer hardware.
- Examples: Operating Systems (Windows, Linux), device drivers, utility programs.

Application Software:

- Performs specific tasks for users.
- Examples: Word processors, web browsers, media players.

Programming Software:

- Provides tools to write, test, and maintain other software.
- Examples: Compilers, interpreters, IDEs (Visual Studio, Eclipse).

Embedded Software:

- Runs on specialized hardware to control devices.
- Examples: Software in microwaves, cars, medical devices.

42. Create a list of software you use regularly and classify them into the following categories: system, application, and utility software.

-> **1. System Software:**

- Windows 10/11 (Operating System)
- Linux (for development)

2. Application Software:

- Google Chrome (Web browser)
- Microsoft Word (Word processor)
- VLC Media Player (Media player)

3. Utility Software:

- WinRAR (File compression)
- CCleaner (System cleanup)

43. What are the differences between open-source and proprietary software?

-> Open-source software is software whose source code is freely available, and anyone can use, modify, or share it. Examples include Linux and Mozilla Firefox.

Proprietary software is owned by a company or individual, and its source code is not accessible. Users must buy a license to use it and cannot modify it. Examples include Windows and Adobe Photoshop.

44. How does GIT improve collaboration in a software development team?

-> **How Git improves collaboration in a software development team:**

- **Version Tracking:** Keeps a history of all code changes for easy review and rollback.
- **Branching and Merging:** Allows developers to work on features independently and merge them safely.
- **Conflict Resolution:** Helps manage and resolve code conflicts when multiple people edit the same file.
- **Collaboration Across Locations:** Enables team members to contribute from anywhere in the world.

45. Application Software.

-> **Application Software** is a type of software designed to **help users perform specific tasks** or solve particular problems, unlike system software which manages hardware.

Key Points:

- Serves end-user needs such as **productivity, communication, or entertainment**.
- Can be **desktop, web, or mobile-based**.

- Examples include **Microsoft Word, Excel, Google Chrome, WhatsApp, and Adobe Photoshop**.
- Enhances **efficiency and productivity** by automating tasks and simplifying complex processes.

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

-> **Introduction:**
Application software consists of programs designed to help users perform specific tasks efficiently. They improve productivity by automating processes, simplifying tasks, and enhancing communication.

Types of Application Software:

1. **Word Processing Software:**
    - Examples: Microsoft Word, Google Docs
    - **Productivity Benefit:** Allows fast creation, editing, and formatting of documents, reducing manual effort.
2. **Spreadsheet Software:**
    - Examples: Microsoft Excel, Google Sheets
    - **Productivity Benefit:** Enables data organization, calculations, and analysis using formulas and charts, saving time on manual computations.
3. **Database Software:**
    - Examples: MySQL, Microsoft Access
    - **Productivity Benefit:** Helps store, retrieve, and manage large amounts of data efficiently, supporting decision-making.
4. **Presentation Software:**
    - Examples: Microsoft PowerPoint, Prezi

- **Productivity Benefit:** Assists in creating engaging presentations quickly, enhancing communication and clarity.

5. **Web Browsers:**
    - Examples: Google Chrome, Mozilla Firefox
    - **Productivity Benefit:** Provides fast access to online resources and cloud-based tools for research and collaboration.

47. What is the role of application software in businesses?

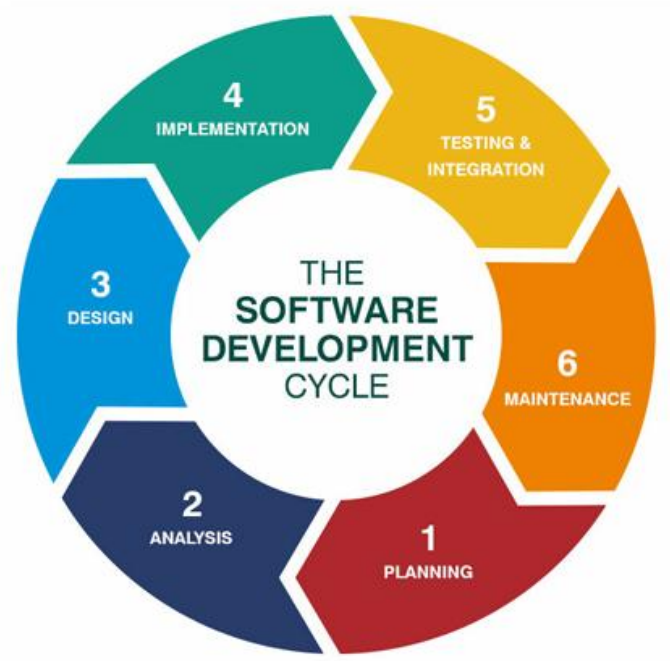
-> The role of application software in businesses is to automate tasks, improve efficiency, and support decision-making. It helps in managing data, creating documents, analyzing information, and communicating effectively within and outside the organization. By streamlining operations like accounting, inventory management, customer relationship management, and reporting, application software reduces manual effort, minimizes errors, and increases overall productivity, enabling businesses to operate smoothly and make informed decisions.

48. Software Development Process.

-> The **Software Development Process** is a structured set of activities followed to **plan, create, test, and maintain software**. It ensures that software is delivered efficiently, meets user requirements, and is of high quality.

49. Create a flowchart representing the Software Development Life Cycle (SDLC).

->



50. What are the main stages of the software development process?

-> The main stages of the **software development process** are:

1. **Requirement Analysis:** Understanding and documenting what the users need from the software.
2. **System Design:** Planning the software architecture, modules, and data flow.
3. **Implementation / Coding:** Writing the actual program code based on the design.
4. **Testing:** Checking the software for errors, bugs, and verifying it meets requirements.
5. **Deployment:** Installing and making the software available for users.
6. **Maintenance:** Updating, fixing issues, and improving the software after release.

51. Software requirement.

-> Software Requirement refers to the specifications of what a software system should do and how it should perform. It defines the features, functions, and constraints that the software must satisfy to meet user needs.

Types of Software Requirements:

1. Functional Requirements: Describe what the software should do, e.g., “The system must allow users to log in.”
2. Non-Functional Requirements: Define how the software performs tasks, e.g., performance, security, usability, and reliability.

52. Write a requirement specification for a simple library management system.

-> **Purpose:**
To manage books, members, and book issues/returns in a library efficiently.

Functional Requirements:

- Add, update, or remove library members.
- Add, update, or remove books.
- Issue and return books.
- Search books by title, author, or category.
- Track overdue books and fines.

Non-Functional Requirements:

- Easy-to-use interface.
- Secure access for librarians.
- Data must be saved correctly and recoverable.

Constraints:

- Works on Windows or Linux.
- Uses a database (e.g., MySQL) to store records.

53. Why is the requirement analysis phase critical in software development?

-> The requirement analysis phase is critical in software development because it defines what the software must do before design and coding begin. Clear and accurate requirements ensure that developers build the right system, meeting user needs and expectations. Mistakes or gaps in this phase can lead to costly errors, delays, and rework later. It also helps in planning, estimating resources, and reducing misunderstandings between developers and stakeholders.

54. Software analysis.

-> Software Analysis is the process of studying and understanding a software system's requirements, functions, and objectives before design and development begin. It involves examining user needs, system constraints, and business processes to ensure the software will meet its intended purpose.

Key Points:

- Identifies functional and non-functional requirements.
- Helps detect potential problems early.
- Serves as a foundation for system design and development.

55. Perform a functional analysis for an online shopping system.

-> **Functional Analysis of Online Shopping System**

1. User Registration and Login:

- Users can create accounts and log in securely.
- Supports password recovery and profile management.

2. Product Catalog:

- Displays available products with details like name, price, description, and images.
- Allows users to search and filter products by category, price, or brand.

3. Shopping Cart Management:

- Users can add, update, or remove products from the cart.
- Calculates total price, taxes, and shipping costs.

4. Order Placement:

- Users can place orders for items in the cart.
- Supports multiple payment options (credit card, debit card, digital wallets).

5. Order Tracking:

- Users can view order status (processing, shipped, delivered).
- Sends notifications for updates.

6. Reviews and Ratings:

- Users can rate products and write reviews.
- Helps other users make informed purchase decisions.

7. Admin Functions:

- Add, update, or remove products.
- Manage orders, track inventory, and generate sales reports.

56. What is the role of software analysis in the development process?

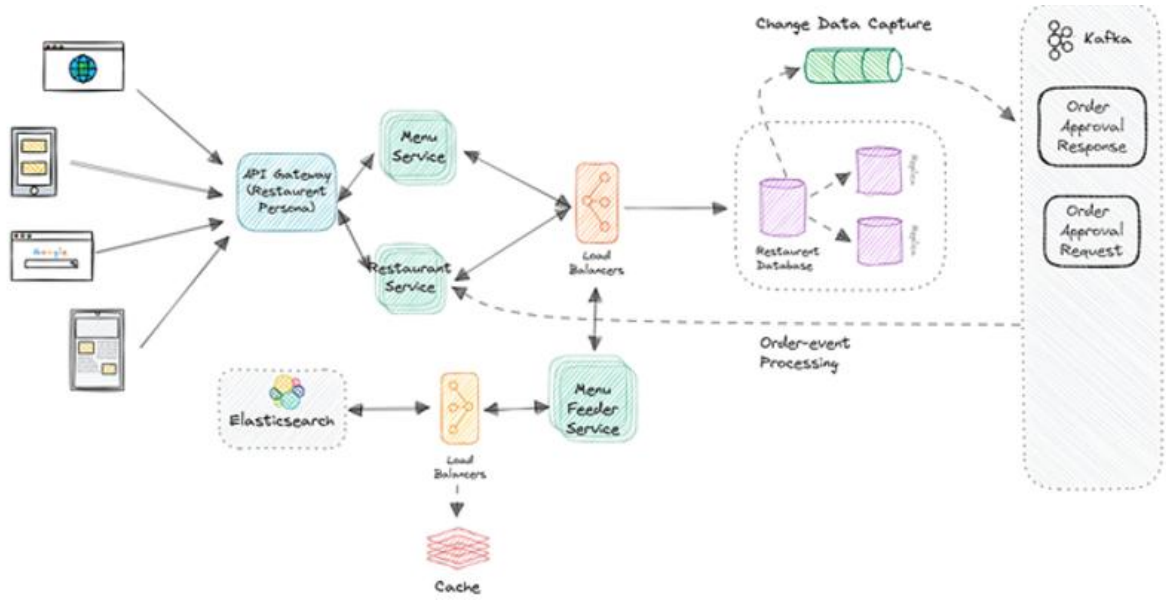
-> The role of software analysis in the development process is to understand and define what the software must achieve before design and coding begin. It involves gathering user requirements, identifying functional and non-functional needs, and examining system constraints. This ensures that the software is aligned with user expectations, feasible to build, and less prone to errors, providing a clear foundation for the design, implementation, and testing phases.

57. System design.

-> System Design is the phase in software development where the requirements gathered during analysis are translated into a detailed blueprint for building the system. It defines the architecture, components, modules, interfaces, and data flow needed to implement the software.

58. Design a basic system architecture for food delivery system.

->



59. What are the key elements of system design?

-> The key elements of **system design** include:

1. **Architecture Design:** Defines the overall structure of the system, including components and their interactions.
2. **Interface Design:** Specifies how different modules and users will interact with the system.
3. **Database Design:** Organizes how data will be stored, retrieved, and managed efficiently.

4. **Module/Component Design:** Breaks the system into smaller, manageable units with specific functions.
5. **Data Flow and Control Design:** Shows how data moves through the system and how processes are controlled.
6. **Security Design:** Ensures the system is protected against unauthorized access and vulnerabilities.
7. **Performance and Scalability Considerations:** Ensures the system meets speed, efficiency, and growth requirements.

60. Software testing.

-> Software Testing is the process of evaluating a software application to identify defects, ensure quality, and verify that it meets the specified requirements. It helps detect bugs, errors, or gaps before the software is released to users.

61. Why is software testing important?

-> Software testing is important because it ensures that the software is reliable, functional, and meets user requirements. It helps identify and fix bugs before release, preventing costly errors and system failures. Testing also improves software quality, performance, security, and user satisfaction, and ensures the product is maintainable and safe for real-world use.

62. Maintenance.

-> Maintenance in software development is the process of updating, fixing, and improving software after it has been deployed. Its purpose is to ensure the software continues to function correctly, meets changing user needs, and adapts to new environments or technologies.

63. Document a real world case where a software application required critical maintenance.

-> **Case: Microsoft Windows 10 – WannaCry Ransomware Response (2017)**

Background:
In May 2017, the **WannaCry ransomware attack** affected hundreds of thousands of computers worldwide, exploiting a vulnerability in Microsoft Windows operating systems. The ransomware encrypted users' files and demanded payment to unlock them.

Critical Maintenance Required:

- Microsoft released an **emergency security patch (update)** for supported and even some unsupported versions of Windows.
- The maintenance was **corrective and preventive**:
    - **Corrective:** Fixed the exploited vulnerability to stop further attacks.
    - **Preventive:** Ensured that similar ransomware attacks could not exploit the same flaw.
- IT teams globally had to **deploy the update immediately** to prevent system failures and data loss.

Impact:

- The patch prevented further spread of WannaCry.
- Highlighted the importance of **regular software maintenance and updates** for security-critical applications.

Conclusion:
This case shows that **critical software maintenance** can be essential to protect systems from **security threats, data loss, and operational disruption**, emphasizing that maintenance is not just about adding features but also about **safeguarding users and systems**.

64. What types of software maintenance are there?

-> There are **four main types of software maintenance**:

1. **Corrective Maintenance:** Fixes bugs and errors found after the software is released.
2. **Adaptive Maintenance:** Updates the software to work with new hardware, operating systems, or environments.
3. **Perfective Maintenance:** Adds new features or improves existing functionality and performance.
4. **Preventive Maintenance:** Modifies the software to prevent potential future problems and improve maintainability.

65. Developement.

-> Development in software engineering refers to the phase where actual coding and creation of the software take place based on the design specifications. It transforms requirements and designs into a working software system.

66. What are the key differences between web and desktop applications?

Feature	Web Application	Desktop Application
Access	Runs in a web browser via the internet.	Installed and runs directly on a computer.
Platform Dependency	Platform-independent (works on any device with a browser).	Platform-dependent (may require Windows, macOS, or Linux).
Installation	No installation needed; accessed via URL.	Requires installation on each device.

Feature	Web Application	Desktop Application
Updates	Updated centrally on the server; users always access the latest version.	Updates must be installed on each device individually.
Connectivity	Usually requires internet connection.	Can work offline (most cases).
Examples	Gmail, Google Docs, Facebook	Microsoft Word, Adobe Photoshop, VLC Media Player

67. Web application.

-> A Web Application is a software program that runs on a web server and is accessed through a web browser over the internet or an intranet. Unlike desktop applications, users do not need to install it on their device.

Key Points:

- Platform-independent: works on any device with a browser.
- Can be updated centrally on the server.
- Often requires an internet connection to function.
- Examples: Gmail, Google Docs, Facebook, Amazon.

68. What are the advantages of using web applications over desktop applications?

-> The advantages of web applications over desktop applications are:

1. **Platform Independence:** Works on any device with a web browser, regardless of operating system.
2. **No Installation Required:** Users can access it directly via a URL.
3. **Automatic Updates:** Updates are applied on the server, so all users get the latest version instantly.

4. **Accessibility:** Can be accessed from anywhere with an internet connection.
5. **Reduced Storage Needs:** Data and processing are handled on the server, saving local storage.
6. **Easy Maintenance:** Centralized management simplifies troubleshooting and feature upgrades.

69. Designing.

-> Designing in software development is the process of planning how a software system will work and how its components will interact before actual coding begins. It transforms requirements into a blueprint for implementation.

Key Points:

- Defines system architecture, modules, interfaces, and data flow.
- Includes high-level design (overall system structure) and low-level design (detailed module and database design).
- Ensures the software is efficient, maintainable, and scalable.

70. What role does UI/UX design play in application development?

-> UI/UX design plays a crucial role in application development by ensuring the software is user-friendly, intuitive, and engaging.

- UI (User Interface) Design: Focuses on the visual elements—layout, colors, buttons, and overall look—so users can interact with the application easily.
- UX (User Experience) Design: Focuses on how users feel and navigate the application, ensuring smooth workflows, efficiency, and satisfaction.

Good UI/UX design improves usability, increases user engagement, reduces errors, and enhances overall product success.

71. Mobile application.

-> A Mobile Application is a software program designed to run on mobile devices such as smartphones and tablets. It provides specific functionality and services to users directly on their devices.

Key Points:

- Can be native (built for a specific platform like Android or iOS) or cross-platform (runs on multiple platforms).
- Installed via app stores like Google Play or Apple App Store.
- Can work offline or online, depending on functionality.
- Examples: WhatsApp, Instagram, Google Maps, Spotify.

72. What are the differences between native and hybrid mobile apps?

->

Feature	Native App	Hybrid App
Platform	Built for a specific platform (Android or iOS).	Works on multiple platforms using a single codebase.
Performance	High performance; faster and smoother.	Slightly slower due to additional abstraction layers.
Development	Requires separate code for each platform.	Single codebase for all platforms; faster development.
Access to Device Features	Full access to device hardware (camera, GPS, sensors).	Limited access; may require plugins.
User Experience	Offers the best UI/UX, optimized for platform.	UI/UX may feel less native and less smooth.
Examples	WhatsApp, Instagram	Twitter Lite, Uber (some parts use hybrid technology)

73. DFD (Data Flow Diagram).

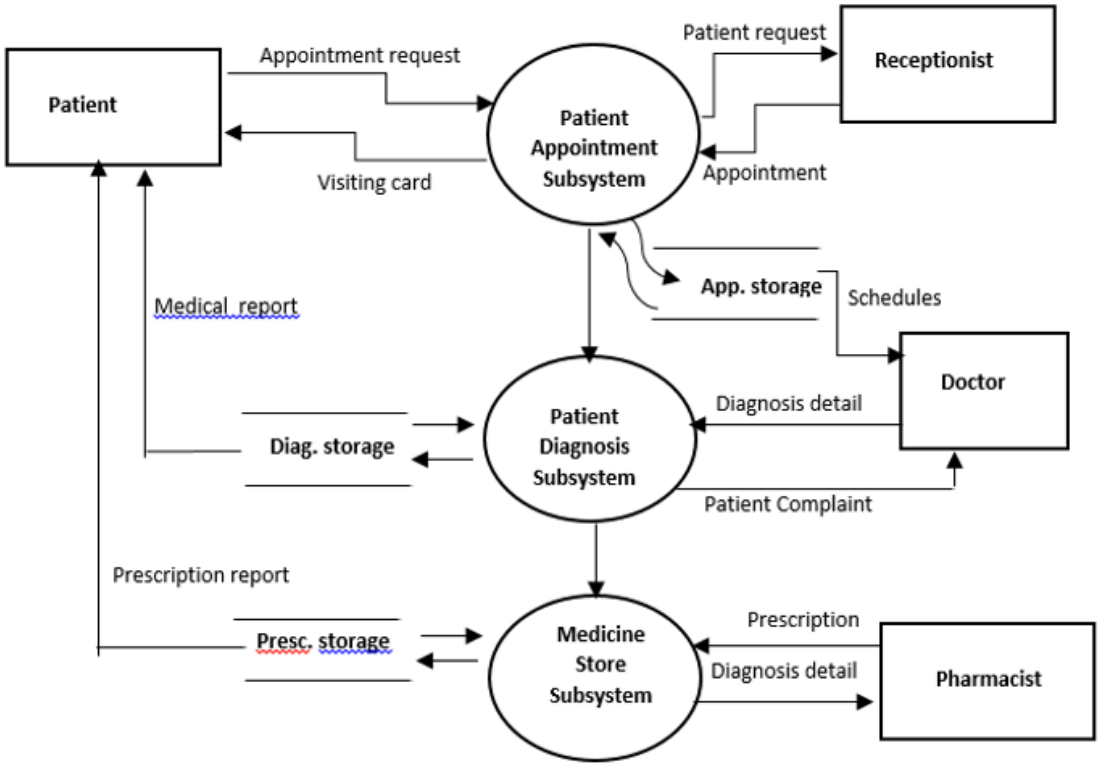
-> A Data Flow Diagram (DFD) is a graphical representation of how data moves through a system. It shows the flow of information between processes, data stores, and external entities without detailing program logic.

Key Points:

- Illustrates inputs, processes, outputs, and storage in a system.
- Helps in understanding system functionality and data movement.
- Uses standard symbols:
    - Rectangle: External entity (source or destination of data)
    - Circle/Oval: Process (transforms data)
    - Arrow: Data flow (movement of data)
    - Open-ended Rectangle: Data store (where data is stored)

74. Create a DFD for Hospital Management System.

->



75. What is the significance of DFDs in system analysis?

-> The significance of Data Flow Diagrams (DFDs) in system analysis is that they visually represent how data moves within a system, making it easier to understand, analyze, and communicate system functionality.

Key Points:

- Helps identify processes, data sources, and data destinations.
- Reveals redundancies, inefficiencies, or missing components in the system.
- Provides a clear, simple view for both technical and non-technical stakeholders.
- Serves as a blueprint for system design and development, ensuring requirements are accurately implemented.

76. Desktop application.

-> A Desktop Application is a software program that is installed and runs directly on a personal computer or laptop. It performs specific tasks and does not require a web browser to function.

Key Points:

- Platform-dependent: Usually designed for a specific operating system (Windows, macOS, Linux).
- Works offline: Can run without an internet connection.
- Direct access to hardware: Can use local resources like files, printers, and devices efficiently.
- Examples: Microsoft Word, Adobe Photoshop, VLC Media Player, Excel.

77. What are the pros and cons of desktop applications compared to web applications?

->

Pros of Desktop Applications:

- Can work **offline** without an internet connection.
- **Faster performance** and better access to hardware resources.
- Often **richer features** and more control over the system.

Cons of Desktop Applications:

- **Platform-dependent**; may need different versions for Windows, macOS, etc.
- **Manual updates** required on each device.
- Limited **accessibility**; cannot be used remotely easily.

Pros of Web Applications:

- **Platform-independent**; accessible from any device with a browser.
- **Automatic updates** on the server; users always access the latest version.
- Can be accessed **anywhere** with an internet connection.

Cons of Web Applications:

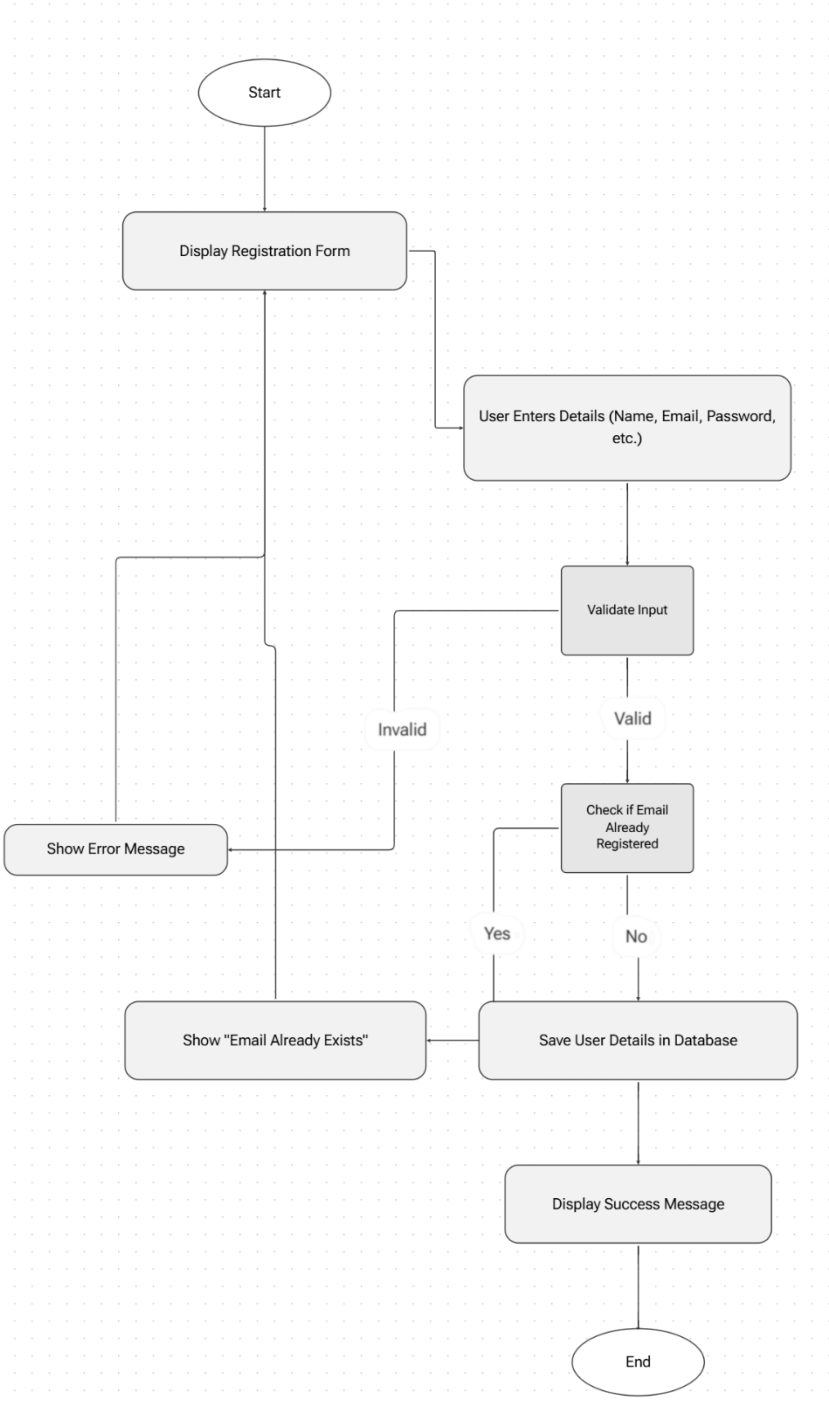
- Usually **requires internet** to function.
- Slightly **slower performance** compared to desktop apps.
- Limited access to some device hardware features.

78. Flowchart.

-> A flowchart is a visual diagram that shows the step-by-step flow of a process, using symbols like boxes, arrows, diamonds, and ovals. It helps you understand how a system, program, or workflow operates.

79. Draw a flowchart representing the logic of a basic online registration system.

->



80. How do flowcharts help in programming and system design?

->

Flowcharts are an essential tool in programming and system design because they provide a graphical representation of the sequence of steps, decisions, and processes involved in a system or algorithm. By translating complex logic into visual diagrams, flowcharts make it easier to understand how a system works and how different components interact.

In programming, flowcharts help in several ways:

1. **Clarifying Logic:** Before writing code, developers can visualize the flow of operations, making it easier to plan algorithms and understand the sequence of actions.
2. **Error Identification:** By mapping out the process, potential logic errors, redundancies, or inefficiencies can be spotted early, reducing costly mistakes during coding.
3. **Guiding Development:** Flowcharts act as a roadmap, helping programmers follow a structured approach to coding. This ensures that each step is implemented correctly and in the proper order.
4. **Improving Collaboration:** Flowcharts provide a common visual language that developers, designers, and stakeholders can understand, facilitating better communication and teamwork.
5. **Documentation and Maintenance:** They serve as a reference for future updates or troubleshooting, making it easier for new team members to understand the system.

In system design, flowcharts are equally important:

- They help designers plan the structure and workflow of the system.
- Assist in breaking down complex systems into manageable components.
- Support decision-making by showing possible paths and outcomes in processes.