Assignment – 1

Module 1 – Overview of IT Industry

## What is Program?

* A program is a set of instructions written in a programming language that a computer can understand and execute to perform a specific task.

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

1. C Example: -

* #include <stdio.h> // Library for input/output

int main () {

printf ("Hello, World!"); // Output text

}

1. Python Example: -

* print ("Hello, World!")

|  |  |  |
| --- | --- | --- |
| Comparison of Structure and Syntax | | |
| Aspect | **Python** | **C** |
| Length | 1 line | Multiple lines |
| Entry Point | Direct Execution | Required main () |
| Ease of Writing | Very beginner-friendly | Require Syntax |
| Semicolons | Not required | Required after each statement |

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

* A **program** is a set of instructions written by a person that tells a computer what to do and how to do it.

When you run a program, here’s what happens:

1. **You write the code** in a programming language (like Python, Java, or C++).
2. **The computer translates it** into a form it understands — machine code made of 1s and 0s.
3. **The CPU executes each instruction** in order, step-by-step.
4. **The program interacts** with data, the user, or other systems to produce the desired result — such as showing text, playing music, or calculating numbers.

## What is Programming?

* **Programming** is the process of creating a set of instructions (a program) that a computer can follow to perform a specific task.

It involves:

1. **Thinking of a solution** to a problem.
2. **Writing the solution** in a programming language (like Python, Java, C++).
3. **Testing and fixing** the code so it works correctly.

### What are the key steps involved in the programming process?

1. **Problem Definition**

* Understand what you want the program to do.
* Identify the inputs, the expected outputs, and the purpose.

1. **Planning the Solution**

* Break the problem into smaller, logical steps.
* Use **flowcharts** or **pseudocode** to outline the process.

1. **Writing the Code**

* Translate the planned steps into a programming language.
* Follow syntax rules of the chosen language.

1. **Testing the Program**

* Run the program to check if it works correctly.
* Use sample data to verify outputs.

1. **Debugging**

* Find and fix errors (syntax errors, logical errors, runtime errors).

1. **Documentation**

* Write explanations for how the code works.
* Add comments inside the code for future reference.

**7. Maintenance & Updates**

* Improve the program over time.
* Fix bugs found after release or add new features.

## Types of Programming Languages

1. Low-Level Languages

* These are closer to the computer’s hardware and are harder for humans to read.
* *Example:* Assembly, x86 Assembly

1. High-Level Languages

* Closer to human language, easier to write and understand, and independent of hardware.
* *Example:* Python, Java, C++, JavaScript, C#

1. Object-Oriented Languages (OOP)

* Organize code into **objects** containing data (attributes) and behaviour (methods)
* *Example:* Java, Python, C++, Ruby

1. Scripting Languages

* Often used for automation, quick tasks, and web development.
* *Example:* JavaScript, Python, PHP, Perl

1. Markup & Query Languages

* Markup**:** Defines data structure and presentation (HTML, XML).
* Query**:** Retrieves or manipulates data in databases (SQL).

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

|  |  |  |
| --- | --- | --- |
| Feature | High-level language | Low-level language |
| Definition | Languages close to human language, easy to read and write. | Languages close to machine language, harder for humans to understand. |
| Examples | Python, Java, C++, JavaScript | Assembly language, Machine code |
| Abstraction | |  | | --- | |  |   High — hides hardware details. | Low — directly interacts with hardware. |
| Ease of Use | Easy to learn and use. | Difficult to learn, requires understanding of computer architecture. |
| Execution Speed | Slower | Faster |
| Portability | Portable across different systems. | Hardware-specific, not easily portable |
| Translation | Needs compiler or interpreter. | Assembly needs an assembler; machine code is executed directly. |
| Development Time | Shorter — simpler syntax and tools. | Longer — more complex and detailed instructions. |

## World Wide Web & How Internet Works

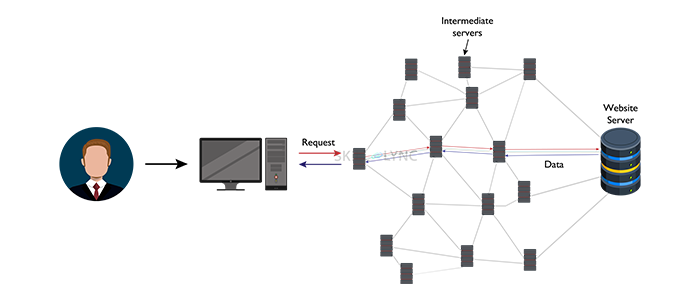
1. **World Wide Web (WWW)**

* The World Wide Web is a collection of web pages and other resources stored on servers and accessed via the internet using a web browser.

1. **How the Internet Works**

* You type a web address (URL) in your browser.
* DNS Lookup → Your browser contacts a Domain Name System (DNS) server to find the IP address of that domain (like turning a name into a phone number).
* Request Sent → Your computer sends an HTTP/HTTPS request to the web server using the IP address.
* Server Response → The web server processes the request and sends back the requested data (HTML, CSS, JS, images, etc.).
* Rendering → Your browser interprets the code and displays the website on your screen.

### Research and create a diagram of how data is transmitted from a client to a server over the internet



**How Data Moves: Simple Flow**

1. **You make a request**  
   You enter a website address (URL) in your browser. The browser sends a request to the server hosting that site.
2. **DNS translates the address**  
   The browser asks a DNS server to turn the human-readable domain (like example.com) into an IP address, which is the server’s numeric location.
3. **Data travels through the Internet**  
   The request is broken into **packets**, sent via protocols like **TCP/IP**, and routed through various routers until it reaches the server
4. **Server processes and responds**  
   The server receives your request, processes it, and sends back the requested data—like a webpage or file.
5. **Packets return to you**  
   The response travels back across the Internet and arrives at your browser, which reconstructs the packets and displays the content

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

* **Client**
* **Who it is:** The device or software (e.g., browser, app) making a request.
* **Role:**
  + Initiates communication.
  + Sends requests for resources (HTML pages, images, data).
  + Displays or processes the server’s response for the user.
* **Examples:** Chrome browser, mobile banking app, email client.
* **Server**
* **Who it is:** A computer system hosting websites, apps, or data.
* **Role:**
  + Waits for incoming requests from clients.
  + Processes the request (may involve database queries, computations, etc.).
  + Sends the requested content or an error message back to the client.
* **Examples:** Apache web server, Nginx, cloud hosting servers.

## Network Layers on Client and Server

* **Network Layers on Client and Server** refer to the step-by-step process by which data is prepared, sent, received, and understood when two computers communicate over a network — usually explained using models like **TCP/IP** or **OSI**.

1. **Client Side (Sending a Request)**
2. **Application Layer** – The app (browser, email, etc.) creates the request.
3. **Transport Layer** – Breaks the request into smaller chunks (segments) and adds port numbers.
4. **Internet Layer** – Adds IP addresses so the data knows where to go.
5. **Network Access Layer** – Turns the data into signals to send through cables, Wi-Fi, etc.
6. **Server Side (Receiving the Request)**
7. **Network Access Layer** – Receives the signals and turns them back into data packets.
8. **Internet Layer** – Checks the IP address to ensure the packet is for this server.
9. **Transport Layer** – Reassembles the data and sends it to the right application.
10. **Application Layer** – The server software (like a web server) reads the request and responds.

### Explain the function of the TCP/IP model and its layers.

* The **TCP/IP model** is a framework that explains how devices communicate over a network like the Internet. It breaks the process into **layers**, where each layer has a specific role, and together they make sure data gets from one device to another reliably.
* **Function of the TCP/IP Model**
* **Defines rules** for how data should be packaged, addressed, transmitted, routed, and received.
* **Ensures compatibility** between different types of devices and networks.
* **Separates responsibilities** so each layer handles a specific part of the communication process.
* **Supports reliability** (making sure data arrives correctly) and flexibility (working with many types of networks).
* **Layers of the TCP/IP Model**

**1. Application Layer**

* **Purpose**: Provides services and interfaces for applications to communicate over the network.
* **Examples**: Web browsing (HTTP/HTTPS), email (SMTP, IMAP), file transfer (FTP).
* **Real-world analogy**: Like the language you use to talk to someone — both people must understand it.

**2. Transport Layer**

* **Purpose**: Ensures data is delivered error-free, in sequence, and without losses or duplication.
* **Key Protocols**:
  + **TCP (Transmission Control Protocol)** → Reliable, connection-oriented (e.g., web pages, emails).
  + **UDP (User Datagram Protocol)** → Faster, connectionless (e.g., video streaming, gaming).
* **Real-world analogy**: Like a courier who makes sure your package arrives in order and without damage.

**3. Internet Layer**

* **Purpose**: Handles addressing, routing, and delivering data packets across networks.
* **Key Protocol**: **IP (Internet Protocol)** → assigns unique addresses to devices (IP addresses).
* **Real-world analogy**: Like writing an address on a letter so the postal service knows where to send it.

**4. Network Access Layer (also called Link Layer)**

* **Purpose**: Deals with physical transmission of data over the network (hardware, cables, Wi-Fi).
* **Examples**: Ethernet, Wi-Fi, ARP (Address Resolution Protocol).
* **Real-world analogy**: Like the road or delivery path your package travels on.

## Client and Servers

* A **Client** and a **Server** are two main roles in a network that work together to exchange data and services.

**1. Client**

* **Definition**: A device or program that requests services or resources from a server.
* **Role**: Sends requests and waits for a response.
* **Examples**:
  + Your web browser (Chrome, Firefox) asking for a webpage.
  + A mobile app retrieving data from a cloud service.
* **Real-world analogy**: Like a customer at a restaurant who places an order.

**2. Server**

* **Definition**: A device or program that provides services or resources to clients.
* **Role**: Listens for client requests, processes them, and sends back responses.
* **Examples**:
  + Web server hosting a website (Apache, Nginx).
  + Email server sending and receiving messages.
* **Real-world analogy**: Like the chef in a restaurant who prepares and serves food.

### Explain Client Server Communication

1. **Client sends a request**

* The client (e.g., a browser or app) starts the communication by sending a request over the internet or a local network.
* Example: You type www.google.com and press enter — your browser sends a request for Google’s homepage.

1. **Server processes the request**

* The server (a computer running special software like Apache, Nginx, or Node.js) receives the request, understands it, and figures out the right response.

1. **Server sends a response**

* The server sends data back to the client (like HTML for a webpage, a file download, or an API response).

1. **Client displays or uses the data**

* The client shows the user the requested information or uses it internally (e.g., an app updates its data).

## Types of Internet Connections

* + 1. **Dial-up** – Very slow, old, uses telephone lines.
    2. **DSL** – Faster than dial-up, still uses telephone lines.
    3. **Cable** – Uses TV cables, good speed.
    4. **Fiber Optic** – Very fast, modern (up to 1 Gbps+).
    5. **Wireless (Wi-Fi)** – Internet without cables, short range.
    6. **Mobile Data (3G/4G/5G)** – Internet via cellular networks.
    7. **Satellite** – Internet via satellites, good for remote areas but higher delay.

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

1. **Broadband (DSL / Cable)**

**Pros**

* Always on (no need to dial in).
* Faster than dial-up.
* Widely available in cities and towns.
* Affordable compared to newer tech

**Cons**

* Speed depends on distance from provider’s exchange.
* Slower than fiber optic.
* Can get congested when many users are online.

1. **Fiber Optic Internet**

**Pros**

* Very high speed (up to 1 Gbps or more).
* Reliable, with low lag (latency).
* Great for streaming, gaming, video calls, large downloads.

**Cons**

* Expensive installation.
* Not available everywhere (mainly in urban areas).
* Requires special cables and infrastructure.

1. **Satellite Internet**

**Pros**

* Works in remote/rural areas where cables aren’t available.
* Can cover large geographic regions.
* Newer providers (e.g., Starlink) offer much better speeds than older ones.

**Cons**

* Higher latency (delay), not ideal for gaming or real-time video calls.
* More expensive than other options.
* Weather (rain, storms) can affect performance.

1. **Mobile Data (3G, 4G, 5G)**

**Pros**

* Portable (works anywhere with cell coverage).
* No cables needed.
* 5G offers very high speeds and low latency.

**Cons**

* Coverage can be poor in rural areas.
* Data limits and expensive plans.
* Speed varies depending on signal strength.

1. **Wi-Fi (Wireless Broadband inside Homes/Offices)**

**Pros**

* Convenient — no wires for devices.
* Multiple devices can connect at once.
* Widely used for home/office networks.

**Cons**

* Limited range (only works near the router).
* Speed depends on broadband connection quality.
* Can be affected by interference (walls, other signals).

### How does broadband differ from fiber-optic internet?

* **Broadband (DSL/Cable):**
  + Uses copper/TV cables
  + speed 10–500 Mbps
  + slows with distance & heavy usage
  + high distance effect
  + widely available
  + cheaper.
* **Fiber-Optic:**
  + Uses glass fiber cable
  + very fast speed 100 Mbps–1 Gbps+
  + stable
  + little distance effect
  + less available
  + costlier.

## Protocols

* **Protocols** are like **rules or languages** that computers follow to communicate with each other over a network. They make sure data is sent, received, and understood correctly.
* **Main Types:**
  + **HTTP/HTTPS** – Web browsing
  + **FTP** – File transfer
  + **SMTP/IMAP/POP3** – Email
  + **DNS** – Website name → IP address
  + **TCP** – Reliable data delivery
  + **UDP** – Fast, less reliable
  + **IP** – Addressing & routing
  + **Ethernet/Wi-Fi** – Physical connection

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

* **HTTPS Request (Secure Website)**

curl https://www.google.com

This fetches Google’s homepage over HTTPS.

* **FTP Request (File Transfer)**

If you have an FTP server (username: user, password: pass):

**Download a file**

curl -u user: pass ftp://ftp.example.com/file.txt -o file.txt

### What are the differences between HTTP and HTTPS protocols?

|  |  |  |
| --- | --- | --- |
| Features | HTTP | HTTPS |
| Full Form | Hypertext Transfer Protocol | Hypertext Transfer Protocol Secure |
| Security | Not secure (plain text) | Secure (encrypted with SSL/TLS) |
| Port | Uses **port 80**. | Uses **port 443**. |
| Data Safety | Vulnerable to attacks | Protects from eavesdropping & MITM |
| Usage | Old/non-sensitive sites | Almost all modern websites |

## Application Security

* Application Security means **protecting software applications** (like websites, mobile apps, or desktop apps) from threats, attacks, and data breaches.
* It includes **practices, tools, and measures** that make sure apps are **safe, reliable, and trustworthy**.

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

**1. SQL Injection (SQLi)**

* **Problem:** Hackers type special commands into forms (like login boxes) to trick the database.
* **Impact:** They can see, change, or delete your data.
* **Fix:** Always check and clean user input, and use **safe database queries** (prepared statements).

**2. Cross-Site Scripting (XSS)**

* **Problem:** Hackers put bad code (JavaScript) into a website, which then runs in other people’s browsers.
* **Impact:** Can steal cookies, show fake pop-ups, or redirect users to bad sites.
* **Fix:** Clean and escape all user input, and block untrusted scripts with security rules (CSP).

**3. Cross-Site Request Forgery (CSRF)**

* **Problem:** Hackers trick a logged-in user into doing something without knowing (like sending money or changing password).
* **Impact:** Unauthorized actions on your account.
* **Fix:** Use **CSRF tokens** in forms, secure cookies, and ask users to confirm sensitive actions.

### What is the role of encryption in securing applications?

* **Encryption** means changing readable data (**plain text**) into unreadable data (**cipher text**) using a secret key. Only someone with the correct **key** can unlock (decrypt) it.

## Software Applications and Its Types

* Software applications (or just “apps”) are **programs designed for users** to perform specific tasks on a computer or mobile device.

**Types of Software Applications: -**

Communication Applications, Media Applications, Business Applications, Gaming Applications

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

MS Word, Gmail, WhatsApp, Instagram, Zoom

### What is the difference between system software and application software?

|  |  |  |
| --- | --- | --- |
| Feature | System Software | application software |
| Purpose | Runs/manages computer hardware | Helps users do tasks |
| Runs on | Background (always active) | Front-end (when user opens) |
| User interaction | Less (works behind the scenes) | Direct (user uses it) |
| Examples | Windows, Linux, Drivers | MS Word, Chrome, WhatsApp |

## Software Architecture

* the plan of how software is built and how parts talk to each other, just like a blueprint for a house.

**Types of Software Architecture (in simple words)**

1. **Monolithic** → All in one piece. Easy to start, hard to change later.
2. **Layered (N-Tier)** → Divided into layers (UI, Logic, Database). Very common.
3. **Client-Server** → One computer (client) asks, another (server) answers. Used in websites.
4. **Microservices** → Split into small services. Flexible, used in big apps like Netflix.
5. **Event-Driven** → Works by reacting to events (like clicks, messages, notifications).

### What is the significance of modularity in software architecture?

**Modularity** means breaking a big software system into **smaller, separate parts (modules)**. Each module does **one job** (like login, payment, search, or reporting).

**Significance**

1. **Easy to Understand** – Small pieces are simpler to learn and manage.
2. **Easier to Fix** – If one module has a bug, you don’t need to touch the whole system.
3. **Reusability** – A module can be reused in other apps (e.g., payment system reused in shopping apps).
4. **Teamwork** – Different teams can work on different modules at the same time.
5. **Scalability** – New features can be added by just adding new modules.
6. **Flexibility** – You can replace or upgrade one module without changing the whole software.

## Layers in Software Architecture

* In software architecture, **layers** are like **levels in a building**. Each layer has a specific role, and together they make the whole system work smoothly.

### Why are layers important in software architecture?

* Layers are important because they make software **organized, easier to fix, reusable, and scalable** — like building a house with separate floors (kitchen, bedroom, office) instead of mixing everything in one room.

## Software Environments

* + - A **software environment** is the setup or space where software runs.

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

1. **Safe Place to Build**
   * Developers can try ideas and write code without affecting the real system (production).
2. **Debugging & Testing**
   * Bugs can be found and fixed early in a controlled space.
3. **Consistency**
   * All developers use the same tools, libraries, and settings → fewer errors when moving code to production.
4. **Faster Development**
   * Provides tools like compilers, debuggers, and editors that make coding easier.
5. **Experimentation**
   * Developers can test new features or technologies without risking real users’ data.
6. **Team Collaboration**
   * Shared environments (like Git + local dev setups) help teams work smoothly together.

## Source Code

* + - **Source Code** is the **human-readable instructions** written by a programmer in a programming language (like C, Java, Python).
    - It tells the computer **what to do** step by step.

## What is the difference between source code and machine code?

|  |  |
| --- | --- |
| Source Code | Machine code |
| Human-readable (Python, Java) | Only 1s and 0s (binary) |
| Written by Programmers | Generated by compiler/interpreter |
| Humans understand it | Computer (CPU) understand it |
| Example: print(“Hello”) | Example: 01001000 01101001 |

## GitHub and Introductions

* + - **GitHub** is a **platform** where programmers can **store, share, and work together on code**.
    - It uses **Git** (a version control system) to keep track of changes in code.

### Why is version control important in software development?

* Version control is important because it makes software development **organized, safe, and collaborative**, especially when many people are working on the same project.

## Student Account in GitHub

### What are the benefits of using GitHub for students?

* **GitHub helps students** store and manage code, track changes, and work together on projects. It also lets them build a portfolio to show skills to recruiters, access free tools through the Student Developer Pack, and learn from real-world open-source projects. It improves coding, teamwork, and career opportunities.

## Types of Software

There are three types of software:

* **System Software** → Runs the computer.
* **Application Software** → Helps users do tasks.
* **Programming Software** → Helps developers build programs.

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

|  |  |  |
| --- | --- | --- |
| Feature | Open-Source software | Proprietary software |
| Source Code | Open to everyone | Closed, company-owned |
| Cost | Mostly free | Usually paid (license fee) |
| Customization | Users can modify | Users cannot modify |
| Support | Community-driven | Company-provided |
| Examples | Linux, Firefox | Windows, MS Office |

## GIT and GITHUB Training

### How does GIT improve collaboration in a software development team?

* Git improves collaboration by letting developers **work together safely, track changes, avoid conflicts, and merge work smoothly** — like a team writing a shared Google Doc but with much more control.

## Application Software

* Application software is a type of software designed to help users perform **specific tasks** or solve problems. Unlike system software (which runs the computer), application software focuses on the **end user’s needs**.

## What is the role of application software in businesses?

* Application software in businesses is used to **automate work, improve communication, manage data, serve customers, and boost productivity** — making the business run more efficiently and competitively.

## Software Development Process

### What are the main stages of the software development process?

**Main Stages of the Software Development Process (SDLC)**

1. **Planning**
   * Understand the problem and decide what the software should do.
   * Example: A company plans to build a shopping app.
2. **Requirement Analysis**
   * Collect and document what the users and business need.
   * Example: App must have login, cart, and payment features.
3. **Design**
   * Create the blueprint of the software (UI design, architecture, database design).
4. **Implementation (Coding)**
   * Developers write the actual source code using programming languages.
5. **Testing**
   * Check for bugs and errors, make sure the software works as expected.
6. **Deployment**
   * Release the software for real users (production environment).
7. **Maintenance**
   * Fix issues, update features, and improve performance after release.

## Software Requirement

### Why is the requirement analysis phase critical in software development?

* Requirement analysis is critical because it lays the **foundation** of the whole project. If this step is wrong, the entire software can fail to meet user needs.

## Software Analysis

### What is the role of software analysis in the development process?

* The role of software analysis is to **bridge the gap between user needs and technical design**, ensuring that the final software actually solves the right problem.

## System Design

### What are the key elements of system design?

* The key elements of system design are **architecture, data, interface, components, security, performance, and reliability** — all planned to make the system effective and user-friendly.

## Software Testing

### Why is software testing important?

* Software testing is important because it ensures the software is **working correctly, safely, and reliably** before it is given to users.
* Software testing is important because it **checks quality, finds bugs, improves security, saves cost, and ensures users are happy** with the final product.

## Maintenance

## What types of software maintenance are there?

* There are 4 types of software maintenance:

1. Corrective maintenance → Fix bugs.
2. Adaptive maintenance → Update for new environments.
3. Perfective maintenance → Add/improve features.
4. Preventive maintenance → Avoid future issues.

## Development

### What are the key differences between web and desktop applications?

|  |  |  |
| --- | --- | --- |
| Feature | Web Applications | Desktop Applications |
| Installation | Run directly in a web browser | Need to be installed on the computer. |
| Accessibility | Can be accessed from anywhere with internet. | Only works on the computer where installed. |
| Updates | Updated automatically on the server side. | Must be updated manually on each device. |
| Performance | Depends on internet speed and browser. | Usually faster and can use full system resources. |
| Platform Dependency | Works on any device with a browser (cross-platform). | Often OS-specific (Windows, Mac, Linux). |
| Storage | Data usually stored on remote servers (cloud). | Data stored locally on the computer. |
| Examples | Gmail, Google Docs, Facebook | MS Word, Photoshop, VLC Media Player |

## Web Application

### What are the advantages of using web applications over desktop applications?

* Web applications are **easier to access, update, and share**, making them more flexible and convenient than desktop applications.

## Designing

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

* UI/UX design plays a key role in making applications **easy, enjoyable, and efficient** to use, which increases user satisfaction and success of the app.

## Mobile Application

### What are the differences between native and hybrid mobile apps?

|  |  |  |
| --- | --- | --- |
| Features | Native Apps | Hybrid Apps |
| Definition | Built specifically for one platform (Android or iOS). | Built using web technologies (HTML, CSS, JS) and wrapped to run on multiple platforms. |
| Performance | Very fast and smooth | Slightly slower |
| User Experience | Best UI/UX, feels like part of the device. | UI/UX may not be as smooth as native. |
| Development Cost & Time | More expensive & time-consuming | Cheaper & faster |
| Examples | WhatsApp, Instagram | Uber, Gmail, Twitter |

## DFD (Data Flow Diagram)

### What is the significance of DFDs in system analysis?

* A **Data Flow Diagram (DFD)** shows how data moves through a system — where it comes from, how it is processed, and where it goes.
* DFDs are important in system analysis because they **visually explain how data flows in a system**, making it easier to understand, analyse, and design efficient solutions.

## Desktop Application

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

|  |  |  |
| --- | --- | --- |
| Aspect | Desktop Application | Web Application |
| Pros | -Usually faster and more powerful  - Works offline (no internet needed)  - Can fully use device resources (CPU, GPU, storage)  - Often more secure (data stored locally) | - Accessible from anywhere with internet  - No installation required, runs in browser  - Easy to update (done on server side)  - Cross-platform (works on Windows, Mac, Mobile) |
| Cons | - Must be installed separately on each device  - Updates need to be downloaded manually  - Usually tied to one OS (Windows, Mac, Linux)  - Can be expensive to distribute | - Needs internet to work properly - Slower performance compared to desktop apps  - Limited access to hardware features - Security risks (data on cloud/server) |

## Flow Chart

### How do flowcharts help in programming and system design?

* A **flowchart** is a diagram that shows the **steps of a process** using symbols and arrows.
* Flowcharts help by **visually representing the logic and flow of a system**, making programming and system design easier to plan, understand, and debug.