+

# **🡪What is a Program?**

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

* A program is a set of instructions written in a programming language that a computer can understand and execute to perform a specific task or solve a problem.
* **Program** = the result (the code or app that runs)

-->How a Program Functions:

1)**Writing the Code** – A programmer writes instructions (code) in a programming language like Python, Java, or C++

2)**Compiling** - A compiler (for languages like C++) translates the entire program into machine code (binary) before execution.

3)**Execution** – The CPU (Central Processing Unit) reads the machine code and performs the operations:

4)**Output** – The program produces results, such as displaying graphics, saving files, or controlling hardware.

# **🡪What is Programming**?

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

* **Simple Steps in the Programming Process**

1. **Understand the Problem**  
    What do you want the program to do?
2. **Plan the Solution**  
    Think of the steps needed to solve the problem.
3. **Write the Code**  
    Use a programming language to write the steps.
4. **Test the Code**  
   Run the program to see if it works. Fix any mistakes.
5. **Finish and Save** Once it works, save your program and it's ready to use!

# **🡪*Types of Programming Languages***

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

* **High-level languages** **Low-level languages**

1. **High-level languages** are easy to read and write. 1. **Low-level languages** are hard to understand.
2. **High-level** is close to human language. 2.**Low-level** is close to computer language.
3. **High-level** is good for beginners. 3.**Low-level** is used by experts.
4. **High-level examples:** Python, Java. 4.**Low-level examples:** Assembly, Machine code.

# 🡪**World Wide Web & How Internet Works**

* The **World Wide Web (WWW)** is a system of web pages and websites connected through the **Internet**. It allows users to access information using a **web browser** like Chrome or Firefox.
* **How the Internet Works (Simple)**

1. **You Type a Web Address (URL)**  
   e.g., [www.google.com](http://www.google.com)
2. **Browser Sends a Request**  
   Your browser asks the server (another computer) for the webpage.
3. **Server Sends Back Data**   
   The server sends back files that make up the webpage.
4. **Browser Displays the Page**  
   Your browser shows the webpage on your screen.

4) Describe the roles of the client and server in web communication.

* **Client**: The client is usually a web browser. It sends a request to the server asking for a web page or data.

**• Responsibilities:**

o Sends HTTP requests (e.g., GET, POST) to a server.

o Requests web pages, files, data, etc.

o Renders and displays content (like HTML, CSS, JavaScript) received from the server.

o Interacts with users and may send further requests based on user actions (e.g., clicking a button).

* **Server**: The server is a computer that receives the request, finds the correct information, and sends it back to the client.

• **Responsibilities:**

o Listens for and accepts incoming requests from clients.

o Processes the request (e.g., queries a database, runs scripts).

o Sends back appropriate HTTP responses (e.g., web pages, JSON data).

o Hosts resources like websites, APIs, and files.

**🡪Network Layers on Client and Server**

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

* **Function of TCP/IP Model**
* Provides a framework for data transmission.
* Ensures accurate delivery of data between devices.
* Allows different systems to communicate using a standard set of protocols.
* **Layers of the TCP/IP Model:**

**1.**Application Layer

**2.** Transport Layer

**3.** Internet Layer

**4.** Network Access Layer

# 🡪**client and Servers.**

## 6) Explain Client Server Communication

1. **Client sends request**  
   Example: A web browser (client) asks a website server for a page.
2. **Server receives request**  
   The server gets the request and checks what the client needs.
3. **Server processes request**  
   It finds the right data or performs the required task.
4. **Server sends response**  
   The server sends back the data (e.g., a webpage, a file, or confirmation).
5. **Client receives and displays**  
   The client gets the response and shows it to the user or uses it.

# 🡪**Types of Internet Connections**

* **Dial-Up**
* **DSL (Digital Subscriber Line)**
* **Cable**
* **Fiber Optic**
* **Satellite**
* **Wi-Fi**
* **Mobile (4G/5G)**

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

| **Feature** | **Broadband** | **Fiber-Optic** |
| --- | --- | --- |
| Type | General term | Specific type |
| Speed | Medium to Fast | Very Fast |
| Wires Used | Phone or TV cables | Glass/light cables |
| Availability | Almost everywhere | Limited areas |
| Best For | Everyday use | Fast, reliable internet |

# **🡪Protocols**

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

* **1. Security:**
* **HTTP**: **Not secure**. Data is sent as plain text and can be intercepted.
* **HTTPS**: **Secure**. Data is encrypted using SSL/TLS, making it harder for hackers to intercept or tamper with the information.
* **2.Encryption:**
* **HTTP**: No encryption. Data travels openly.
* **HTTPS**: Encrypts the data between the client and the server, ensuring confidentiality.
* **3. Port Number:**
* **HTTP**: Uses port **80** by default.
* **HTTPS**: Uses port **443** by default.
* **4.URL Prefix:**
* **HTTP**: The URL starts with http://.
* **HTTPS**: The URL starts with https:// (the "s" stands for **secure**).
* **5.SSL/TLS Certificates:**
* **HTTP**: Does **not require** an SSL/TLS certificate.
* **HTTPS**: Requires an **SSL/TLS certificate** to establish a secure connection.
* **6.Performance:**
* **HTTP**: Faster because there’s no encryption overhead.
* **HTTPS**: Slightly slower due to the encryption/decryption process, but modern hardware handles this very well.

# **🡪Application Security**

## 9) What is the role of encryption in securing applications?.

* Encryption helps keep information safe in applications. It changes readable data into a secret code so that only people with the right key can read it.
* **Key Roles of Encryption:**

1. **Data Confidentiality**
   * Prevents unauthorized access to sensitive data (e.g., passwords, financial info, personal details).
   * Ensures that even if data is intercepted, it cannot be understood.
2. **Data Integrity**
   * Helps detect tampering by ensuring that the data has not been altered during storage or transmission.
3. **Secure Communication**
   * Encrypts data in transit (e.g., HTTPS using SSL/TLS), protecting information exchanged between users and servers.
4. **Regulatory Compliance**
   * Helps meet legal and industry standards (e.g., GDPR, HIPAA, PCI-DSS) that require data protection.
5. **User Trust**
   * Builds trust by safeguarding user information, particularly in web and mobile applications.

# 🡪Software Applications and Its Types

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

| **Feature** | **System Software** | **Application Software** |
| --- | --- | --- |
| Purpose | Runs and manages the computer system | Helps users perform specific tasks |
| Runs on | Directly on hardware | On top of system software |
| Examples | Windows, Linux, macOS, device drivers | MS Word, Chrome, WhatsApp, Photoshop |
| User Interaction | Works mostly in the background | Used directly by the user |

# ***🡪Software Architecture***

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

* Modularity in software architecture means breaking down a system into smaller, independent parts called **modules**. Each module performs a specific function and can be developed, tested, and maintained separately.
* **Significance of Modularity:**

**1.Improves Maintainability**

* + Easier to update or fix one part without affecting the entire system.

**2.Enhances Reusability**

* + Modules can be reused in other projects or applications.

**3.Supports Parallel Development**

* + Different teams can work on different modules at the same time.

**4.Simplifies Testing and Debugging**

* + Easier to test smaller, isolated components.

**5.Increases Scalability**

* + New features or modules can be added without major changes to the whole system.

# ***🡪Layers in Software Architecture***

## 12)Why are layers important in software architecture?

* **1. Separation of Work**

Each layer has a clear role (e.g., one shows the screen, one handles rules, one talks to the database).

* **2. Easier to Fix or Change**

You can change one part (like the design) without breaking the rest.

* **3.** **Better Teamwork**

Teams can work on different layers at the same time.

* **4. Reusability**

The same logic can be used with different interfaces (like web and mobile).

* **5.** **Testing is Easier**

You can test one layer without needing the whole system.

# ***🡪Software Environments***

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

* A **development environment** is the setup where programmers write and test their code. It includes tools like a code editor, compiler, debugger, and sometimes a virtual server.

**1.Helps Everyone Work the Same Way**  
 It makes sure all developers use the same tools and settings, so the code works the same for everyone.

**2.Saves Time**  
 With the right tools ready to use, developers can write and test code faster.

**3.Finds and Fixes Problems Early**  
 Built-in tools help catch mistakes in the code before the software goes live.

**4.Safe to Try New Ideas**  
 Developers can test new features without breaking the real software.

**5.Works with Other Tools**  
 It connects with systems that help automatically build and deliver the software.

# **🡪*Source Code***

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

| **Aspect** | **Source Code** | **Machine Code** |
| --- | --- | --- |
| **Definition** | Human-readable instructions written by a programmer in a programming language (e.g., Python, Java, C) | Binary instructions (0s and 1s) that a computer's processor can understand and execute directly |
| **Format** | Text (e.g., print("Hello")) | Binary (e.g., 10110000 01100001) |
| **Readable By** | Humans (programmers) | Computers (CPU) |
| **Examples** | Python, Java, C, JavaScript code | Executable files like .exe or .bin |
| **Needs To Be** | Compiled or interpreted to run | Already ready to run by the machine |
| **Editable** | Yes — easy to modify and understand | No — very hard to read or change manually |

# ***🡪Git hub and Introductions***

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

* **1. Keep Track of Changes**

It saves every version of the code, so you can go back to any previous version if needed.

* **2. Work as a Team**

Many developers can work on the same project without messing up each other’s code.

* **3. Fix Mistakes Easily**

If something breaks, you can quickly undo the changes and go back to a working version.

* **Try New Ideas Safely**

You can test new features in a separate branch without affecting the main project.

* **Know Who Did What**

It shows who made each change and why, which is helpful during teamwork or code reviews.

# **🡪Student Account in Git hub**

## 

## 16)What are the benefits of using Git hub for students?

* **🔹 1. Collaboration and Teamwork**

Students can work together on projects in real-time, even remotely.

GitHub tracks changes made by each contributor, making team collaboration organized and transparent.

* **🔹 2. Version Control**

GitHub uses Git to keep a history of all changes, allowing students to revert to previous versions of their code easily.

This reduces the risk of losing work or overwriting a teammate’s changes.

* **🔹 3. Portfolio Building**

Students can showcase their projects publicly, which helps when applying for internships or jobs.

Recruiters and employers often check GitHub to evaluate a candidate's coding skills and activity.

* **🔹 4. Learning Industry Standards**

GitHub is widely used in the software industry, so students gain hands-on experience with tools used by professional developers.

Skills like pull requests, branching, and issue tracking are valuable in real-world software development.

* **🔹 5. Access to Free Developer Tools**

With the **GitHub Student Developer Pack**, students get free access to premium tools and services like:

GitHub Copilot (AI code assistant)

Free domain names (via Namecheap)

Cloud platforms (like Heroku, DigitalOcean)

Design and project management tools (like Canva, Trello)

* **🔹 6. Practice Open Source Contribution**

Students can contribute to real-world open-source projects, gain experience, and connect with the global developer community.

* **🔹 7. Documentation and Project Management**

GitHub provides built-in features for writing documentation (README files, wikis).

Issues and project boards help manage tasks and bugs efficiently.

# ***🡪Types of Software***

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

| **Feature** | **Open-Source Software** | **Proprietary Software** |
| --- | --- | --- |
| **Source Code Access** | Freely available to the public | Restricted; only available to the developer/company |
| **Cost** | Usually free | Usually paid or licensed |
| **Modification** | Users can modify and customize the code | Users cannot legally modify the code |
| **Distribution** | Can be freely shared and redistributed | Sharing is restricted by license |
| **Support** | Community-driven (forums, volunteers) | Official support from the company |
| **Examples** | Linux, Firefox, LibreOffice, GIMP | Windows, Microsoft Office, Adobe Photoshop |
| **Security** | Code is open to inspection, bugs can be fixed quickly | Security relies on the company’s response |
| **Licensing** | Uses open licenses (e.g., GPL, MIT) | Uses proprietary licenses (e.g., EULA) |

# ***🡪GIT and GITHUB Training***

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

* **1.Version Control**
* Git tracks all changes made to the codebase, so developers can see who made what changes and when.
* It allows reverting to previous versions if something breaks, minimizing risk.
* **2. Branching and Merging**
* Developers can work on isolated branches for new features, bug fixes, or experiments without affecting the main codebase.
* Once work is complete and tested, branches can be merged—enabling parallel development.
* **3. Collaboration Without Conflict**
* Multiple developers can work on different parts of the project at the same time without overwriting each other's work.
* Git helps identify and resolve conflicts if two people modify the same part of a file.
* **Code Review and Quality Control**
* Tools like GitHub, GitLab, and Bitbucket allow **pull requests** or **merge requests**, which let teammates review and discuss code before it's merged.
* This improves code quality and helps spread knowledge across the team.
* **Accountability and Transparency**
* Every commit is attributed to a specific user, creating a clear audit trail.
* You can see who introduced a bug and who fixed it, making team contributions more visible.
* **Remote Collaboration**
* Git makes it easy for teams distributed across locations to collaborate on the same codebase in real-time.
* Combined with platforms like GitHub, teams can manage code, issues, and documentation in one place.

## 🡪Application Software

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

**1. Automating Business Processes**

* Software like accounting systems, payroll tools, and inventory management automates routine tasks.
* Reduces manual effort and human error in operations.

**2. Enhancing Communication**

* Email clients, video conferencing tools, and messaging apps (e.g., Outlook, Zoom, Slack) improve internal and external communication.
* Facilitates real-time collaboration, especially for remote teams.

**3. Supporting Decision-Making**

* Business Intelligence (BI) tools and spreadsheet applications help analyze data and generate reports.
* Enables managers to make informed, data-driven decisions.

**4. Managing Customer Relationships**

* CRM software (e.g., Salesforce, HubSpot) helps track customer interactions, sales pipelines, and support requests.
* Improves customer service and boosts client satisfaction.

**5. Boosting Productivity**

* Project management tools (e.g., Trello, Asana) help teams plan, track, and complete work efficiently.
* Encourages accountability and time management.

**6. Ensuring Data Management and Security**

* Database management systems (e.g., MySQL, Oracle) organize and secure business data.
* Helps maintain data accuracy, access control, and compliance.

**7. Facilitating Marketing and Sales**

* Marketing automation software and e-commerce platforms (e.g., Shopify, Mailchimp) help reach and engage customers effectively.
* Increases reach and sales opportunities.

## 🡪Software Development Process

## **20)What are the main stages of the software development process?**

**1. Requirement Gathering and Analysis**

* Understand the needs of users and stakeholders.
* Define what the software should do.
* Output: Requirements Specification Document.

**2. System Design**

* Plan the system architecture, components, interfaces, and data flow.
* Design both high-level (system) and low-level (detailed) aspects.
* Output: Design Documents, UI/UX mockups, database schemas.

**3. Implementation (Coding)**

* Developers write code according to design documents.
* Follows coding standards and best practices.
* Output: Source Code.

**4. Testing**

* Test the software for bugs, security issues, and performance.
* Types: Unit Testing, Integration Testing, System Testing, User Acceptance Testing (UAT).
* Output: Test Reports and Bug Logs.

**5. Deployment**

* Software is released to the production environment.
* May be done in phases (e.g., beta testing, full release).
* Output: Live application or system.

**6. Maintenance**

* Post-deployment support: bug fixes, updates, feature enhancements.
* Ensures the software remains useful and secure over time.

# ***🡪Software Requirement***

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

* **✅ 1. Defines Clear Goals and Expectations**

It helps identify what the client or users truly need.

Prevents misunderstandings by documenting precise, agreed-upon requirements.

* **✅ 2. Reduces Risk of Rework and Cost Overruns**

Catching issues or gaps early is much cheaper than fixing them later during development or after release.

Well-defined requirements prevent costly rework.

* **✅ 3. Guides System Design and Development**

Requirements serve as a blueprint for system design, architecture, and coding.

Ensures the final product aligns with user expectations and business goals.

* **✅ 4. Improves Communication**

Provides a common reference for developers, testers, stakeholders, and project managers.

Enhances collaboration and transparency across the team.

* **✅ 5. Enables Better Testing and Validation**

Test cases can be designed directly from the requirements.

Helps ensure the system is built correctly and meets user needs.

* **✅ 6. Supports Project Planning**

Accurate requirements help estimate time, cost, resources, and scope.

Enables better project scheduling and management.

# ***🡪Software Analysis***

## **:22)What is the role of software analysis in the development process?**

**1. Understanding User Requirements**

* Gathers detailed functional and non-functional needs from stakeholders.
* Ensures that the system solves the right problem and meets business goals.

**2. Defining System Scope**

* Clearly outlines what the software will and won’t do.
* Prevents scope creep by setting boundaries early in the process.

**3. Providing a Blueprint for Design**

* Translates user needs into structured specifications.
* Guides developers in system architecture and feature development.

**4. Identifying Constraints and Risks**

* Uncovers technical, legal, or operational limitations.
* Helps plan for potential roadblocks early on.

**5. Improving Communication**

* Serves as a reference document for developers, testers, designers, and stakeholders.
* Minimizes misunderstandings and rework.

**6. Supporting Testing and Validation**

* Provides the basis for test case development.
* Helps verify that the software behaves as expected.

**7. Enabling Better Project Planning**

* Helps in estimating time, cost, and resources.
* Supports more accurate scheduling and resource allocation.

# ***🡪System Design***

## **23)What are the key elements of system design?**

**1. Architecture Design**

* High-level structure of the system (e.g., client-server, microservices, layered architecture).
* Determines how components interact and are deployed.

**2. Data Design**

* Defines how data is structured, stored, and accessed.
* Includes database schema, data models, relationships, indexing, and data flow.

**3. Component Design**

* Breakdown of the system into smaller, manageable modules or services.
* Each component has a defined responsibility and interface.

**4. Interface Design**

* Defines how users interact with the system (UI/UX design).
* Also includes APIs between system components or with third-party services.

**5. Security Design**

* Involves authentication, authorization, encryption, data privacy, and secure communications.
* Protects the system from unauthorized access and data breaches.

**6. Scalability and Performance**

* Ensures the system can handle growth in users or data without performance loss.
* Includes caching, load balancing, and database optimization.

**7. Reliability and Fault Tolerance**

* Designs for consistent uptime and graceful handling of errors or failures.
* Uses redundancies, backups, and failover mechanisms.

**8. Maintainability and Modularity**

* Encourages clean code, documentation, and modular design for easier updates and debugging.
* Supports long-term evolution of the system.

**9. Integration Design**

* Planning how the system connects with external systems, APIs, or services.
* Ensures smooth data exchange and system interoperability.

**10. Deployment and Infrastructure**

* Defines hosting environment, CI/CD pipelines, cloud services, and containerization (e.g., Docker, Kubernetes).

# ***🡪Software Testing***

## **24)Why is software testing important?**

* **Finds Bugs**: Helps detect errors early, making it easier and cheaper to fix.
* **Ensures Quality**: Verifies that the software works as expected and meets requirements.
* **Improves Security**: Identifies vulnerabilities to protect data and privacy.
* **Boosts Customer Satisfaction**: Ensures the product is reliable, functional, and user-friendly.
* **Saves Time and Money**: Prevents issues in production, reducing long-term maintenance costs.

# ***🡪Maintenance***

## **25)What types of software maintenance are there?**

**1. Corrective Maintenance**

* **Definition**: Fixes defects or bugs in the software that were not discovered during the initial development phase.
* **Example**: Addressing an issue where a program crashes due to an unhandled error or fixing incorrect calculations in a feature.

**2. Adaptive Maintenance**

* **Definition**: Involves making changes to the software to ensure it remains compatible with new environments, operating systems, hardware, or software.
* **Example**: Updating a software application to be compatible with a new version of the operating system or integrating it with a new database management system.

**3. Perfective Maintenance**

* **Definition**: Improves the software's performance, adds new features, or enhances existing features based on user feedback or evolving requirements.
* **Example**: Adding new functionality to an app (like introducing new payment methods), or improving the software's response time to handle more users.

**4. Preventive Maintenance**

* **Definition**: Involves making changes to the software to prevent future problems or to improve maintainability. This type of maintenance ensures the software remains efficient, stable, and easier to manage in the long run.
* **Example**: Refactoring the code to make it cleaner, removing deprecated functions, or updating libraries to improve software security and future adaptability.

# ***🡪Development***

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

| **Feature** | **Web Application** | **Desktop Application** |
| --- | --- | --- |
| **Access** | Through a browser | Installed on a computer |
| **Internet Needed** | Yes (usually) | No (usually) |
| **Installation** | Not required | Required |
| **Platform** | Works on any device with a browser | Works only on the installed system |
| **Speed** | Slower (depends on internet) | Faster (uses local resources) |
| **Updates** | Automatic (on server) | Manual or auto-updates |
| **Offline Use** | Limited | Full access |
| **Security** | Exposed to online threats | Exposed to local threats |

# ***🡪Web Application***

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

* **No Installation Needed**  
  – Users can access them directly through a browser without downloading anything.
* **Cross-Platform Compatibility**  
  – They work on any device (Windows, macOS, Linux, mobile) as long as there's a browser.
* **Easy to Update**  
  – Developers can update the app on the server, and users always see the latest version without doing anything.
* **Accessible Anywhere**  
  – Can be used from any device with internet access, making them ideal for remote work or travel.
* **Centralized Data Storage**  
  – User data is stored on the server/cloud, which reduces the risk of data loss due to local hardware failure.
* **Lower System Requirements**  
  – They don't need high-performance hardware since most processing happens on the server.
* **Easier Maintenance**  
  – Developers maintain one version on the server instead of multiple installations across devices.

# ***🡪Designing***

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

* **1. User Interface (UI) Design**

**Focus**: Looks and layout.

**Role**: Ensures the app is visually appealing, intuitive, and consistent. This includes buttons, menus, icons, colors, fonts, and layout.

**Goal**: Make interactions clear and easy to understand.

* **2. User Experience (UX) Design**

**Focus**: Usability and experience.

**Role**: Ensures the app is easy to use, efficient, and satisfying. UX design considers user journeys, pain points, and overall flow.

**Goal**: Make the app functional and enjoyable, reducing frustration.

* **Key Benefits of Good UI/UX Design:**

**Improves user satisfaction and engagement**

**Reduces learning curve and support needs**

**Increases retention and conversion rates**

**Boosts brand perception and trust**

**Minimizes development rework by addressing usability early**

# ***🡪Mobile Application***

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

| **Feature** | **Native Apps** | **Hybrid Apps** |
| --- | --- | --- |
| **Technology Used** | Platform-specific (e.g., Swift for iOS, Kotlin for Android) | Web technologies (HTML, CSS, JavaScript) inside a native wrapper |
| **Performance** | Fast and responsive | Slower than native (especially for complex apps) |
| **Access to Device Features** | Full access to hardware (camera, GPS, etc.) | Limited access (depends on plugins/frameworks) |
| **Platform Support** | Built separately for each platform | One codebase runs on multiple platforms |
| **Development Time** | Longer (one app per platform) | Faster (one app for all platforms) |
| **Maintenance** | More effort (multiple codebases) | Easier (single codebase) |
| **User Experience** | Best UX (platform-optimized) | Slightly less smooth UX |

# 🡪***DFD (Data Flow Diagram)***

## ***30)What is the significance of DFDs in system analysis?***

1. **Visual Representation**  
   – DFDs provide a clear, graphical view of the system, showing processes, data stores, data flows, and external entities.
2. **Better Understanding**  
   – Helps developers, analysts, and clients understand how the system works — especially how data is input, processed, stored, and output.
3. **Improved Communication**  
   – Acts as a communication bridge between technical and non-technical stakeholders.
4. **Identifies Redundancies and Gaps**  
   – Reveals unnecessary processes or missing data flows, helping optimize the system.
5. **Simplifies Complexity**  
   – Breaks down a complex system into smaller, manageable parts (especially through levels 0, 1, 2...).
6. **Foundation for System Design**  
   – Forms the basis for designing databases, user interfaces, and software architecture.
7. **Documentation and Maintenance**  
   – Serves as documentation that helps in future upgrades, debugging, and training.

# ***🡪Desktop Application***

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

**Desktop Applications**

* **Pros:**

1. **Better Performance**  
   – Runs directly on the hardware; great for heavy tasks (e.g., video editing, 3D design).
2. **Offline Access**  
   – Works without an internet connection.
3. **Full System Integration**  
   – Can use system resources (e.g., file system, peripherals) more effectively.
4. **Customizable UI/UX**  
   – Better control over the user interface and experience.

* **Cons:**

1. **Platform Dependent**  
   – Separate versions often needed for Windows, macOS, Linux, etc.
2. **Installation Required**  
   – Users must download and install the software.
3. **Harder to Update**  
   – Updates require user action or an update system.
4. **Limited Accessibility**  
   – Tied to the installed device.

**Web Applications**

* **Pros:**

1. **Accessible Anywhere**  
   – Just need a browser and internet connection.
2. **Platform Independent**  
   – Runs on any OS (Windows, macOS, Linux, mobile).
3. **No Installation Needed**  
   – Accessed via URL; no need to install.
4. **Easy to Update**  
   – Updates happen on the server and reflect instantly for all users.

* **Cons:**

1. **Internet Dependent**  
   – Most features require an active connection.
2. **Lower Performance**  
   – Limited access to system resources; slower for heavy tasks.
3. **Security Risks**  
   – More exposed to online threats like XSS, CSRF.
4. **Browser Compatibility Issues**  
   – Might behave differently across browsers.

# ***🡪Flow Chart***

## ***32)How do flowcharts help in programming and system design?***

**1. Clarifies Process Flow**

* Flowcharts break down complex processes into smaller, manageable steps.
* They show how the program or system moves from one step to the next, helping developers understand the logic.

**2. Simplifies Debugging and Troubleshooting**

* When problems occur in a program or system, a flowchart helps pinpoint where the process breaks down.
* It allows developers to trace and visualize the sequence of steps leading to an issue.

**3. Improves Communication**

* They offer a common language for technical and non-technical stakeholders, making it easier to explain and share ideas.
* Useful for team collaboration in understanding and designing systems.

**4. Aids in Code Structure**

* Flowcharts act as a blueprint for the program, giving developers a clear roadmap to follow when writing code.
* Helps ensure that no important steps are missed, ensuring a more structured approach.

**5. Design and Planning**

* During system design, flowcharts show how data flows through the system and how different components interact.
* They help identify potential inefficiencies and optimize system design before development starts.

**6. Documentation**

* Flowcharts can serve as documentation for future reference. They can guide developers working on a system in the future and provide insights into the original design.

**7. Facilitates Testing and Validation**

* When testing a system, flowcharts can help identify test cases by mapping out the various decision points and possible paths in the system.

## 