

Module 1 – Overview of IT Industry

What is a Program?

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

- A program is basically a set of instructions that a computer follows to do a specific task.

Here's how it functions:

1. Written by a programmer – The instructions are written in a programming language (like Python, C++, or Java) that humans can understand.
2. Translated for the computer – Since computers only understand binary (0s and 1s), the program gets converted into machine code using a compiler or interpreter.
3. Execution – The computer's processor reads the machine code and carries out the instructions, one by one, in the correct order.
4. Output – The result of following those instructions is shown to the user, stored in memory, or sent to another device.

What is Programming?

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

| Feature | High-Level Language (HLL) | Low-Level Language (LLL) |
|-------------------|---|---|
| Definition | A language that is closer to human language and easier to understand. | A language that is closer to machine code (binary) and harder for humans to understand. |
| Examples | Python, Java, C++, JavaScript | Assembly language, Machine code |

| | | |
|----------------------------|---|--|
| Readability | Easy to read and write; uses words and symbols like print, if, while. | Hard to read; uses instructions like MOV A, 5 or binary numbers. |
| Portability | Can run on different types of computers with little or no change in code. | Usually tied to a specific type of processor or hardware. |
| Execution Speed | Slower than low-level languages (needs to be translated first). | Faster execution because it is closer to machine instructions. |
| Ease of Development | Faster to develop, requires less code. | Slower to develop, requires detailed hardware knowledge. |

World Wide Web & How Internet Works

1. Describe the roles of the client and server in web communication. Network Layers on Client and Server

Roles of Client and Server in Web Communication

- Client
 - The device or software (like a browser) that requests a service or resource from the server. ◦ Example: When you type a website address in Chrome, your browser (client) asks the server for the web page. ◦ Main role: Send requests and display the server's response to the user.

- Server
 - A computer or software that stores resources (webpages, files, databases) and responds to client requests. ◦ Example: The website's hosting machine that sends HTML, images, and data back to your browser. ◦ Main role: Receive requests, process them, and send the appropriate response.

Flow:

Client → Sends HTTP request → Server

Server → Processes request → Sends HTTP response → Client

Network Layers

Both client and server have the same layers, but data flows in opposite directions:

1. **Application** – What you see (websites, apps).
2. **Transport** – Breaks data into small parts (packets).
3. **Network** – Finds where to send (IP address).
4. **Data Link** – Sends over local network (MAC address).
5. **Physical** – Turns data into signals (wires, Wi-Fi).

Think of it like **ordering pizza**:

- **Client** → Orders pizza.
- **Server** → Makes pizza and delivers it.
- **Layers** → The steps to take the order, make it, pack it, send it, and deliver it.

Network Layers on Client and Server

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

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

- The TCP/IP model is a set of rules that explains how data travels from one computer to another over a network (like the internet).

Layers of TCP/IP Model (4 layers)

Layers of TCP/IP Model (4 layers)

1. **Application Layer** ○ What the user interacts with.
 - Examples: HTTP (websites), FTP (file transfer), SMTP (email).
 - **Function:** Provide services for applications to send/receive data.
2. **Transport Layer** ○ Controls the flow of data between devices.
 - Examples: TCP (reliable, ordered delivery), UDP (faster, no ordering).
 - **Function:** Splits data into segments and makes sure it arrives correctly.
3. **Internet Layer** ○ Decides the best path for data to travel.
 - Example: IP (Internet Protocol).
 - **Function:** Adds IP addresses so packets know where to go.
4. **Network Access Layer** (also called Link Layer) ○ Sends data physically through cables or Wi-Fi.
 - Examples: Ethernet, Wi-Fi.
 - **Function:** Moves data as electrical signals, light, or radio waves.

Client and Servers

1.Explain Client Server Communication

What it is

Client–server communication is how two devices talk over a network:

- **Client** → Asks for something.
- **Server** → Provides the requested thing.

Example:

You open Google in your browser → Browser (client) sends a request → Google's servers send back the webpage.

How it works (Steps)

1. **Client sends a request** – using a protocol like HTTP.
2. **Server receives the request** – checks what is needed.
3. **Server processes the request** – finds the data or runs code.
4. **Server sends a response** – sends the result back to the client.
5. **Client displays the result** – shows webpage, file, or data to the user.

Types of Internet Connections

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

| Feature | Broadband | Fiber-Optic |
|-------------|---|--|
| Meaning | General term for high-speed internet, usually through copper cables (DSL, cable) or wireless. | Type of broadband that uses thin glass fibers to send data as light. |
| Speed | Slower compared to fiber; often up to a few hundred Mbps. | Much faster; can reach 1 Gbps or more. |
| Reliability | Can slow down over long distances and in bad weather. | Very reliable; less affected by distance or weather. |
| Latency | Slightly higher delay. | Very low delay. |

| | | |
|---------|--|--|
| Cost | Usually cheaper and more widely available. | Can be more expensive and less available in rural areas. |
| Example | DSL, cable internet. | Airtel Xstream Fiber, JioFiber. |

Protocols

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

| Aspect | HTTP | HTTPS |
|-------------------|---|--|
| Full Form | HyperText Transfer Protocol | HyperText Transfer Protocol Secure |
| Port Number | Uses port 80 by default | Uses port 443 by default |
| Security | No encryption; data is sent in plain text | Encrypted using SSL/TLS, making data secure |
| Data Protection | Vulnerable to hacking, eavesdropping, and man-in-the-middle attacks | Protects against hacking and data theft |
| URL Prefix | http:// | https:// |
| Browser Indicator | No padlock icon in the address bar | Shows a padlock icon in the address bar |
| Use Cases | Suitable for non-sensitive data like blogs or general info pages | Suitable for sensitive data like banking, e-commerce, and logins |

| | | |
|-------|---|---|
| Speed | Slightly faster (no encryption process) | Slightly slower (due to encryption), but difference is minimal with modern tech |
|-------|---|---|

Application Security

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

Role of Encryption in Securing Applications

Encryption is a process of transforming data into a coded format using mathematical algorithms, so that it becomes unreadable to unauthorized individuals. Only those who possess the correct decryption key can convert the data back to its original form. It is one of the most fundamental and effective methods for securing modern applications against cyber threats.

1. Protection of Data in Transit

When data moves between a client and a server (e.g., in web applications), it travels through various network nodes. Without encryption, attackers can intercept and read this data (a process called **packet sniffing**).

- Example: Using **HTTPS** (SSL/TLS encryption) ensures that login credentials, financial transactions, and personal information cannot be read by a third party.

2. Protection of Data at Rest

Data stored in databases, file systems, or cloud storage can be at risk if the storage media is stolen or compromised. Encryption ensures that even if unauthorized individuals gain access to storage, the data remains unreadable without the decryption key.

- Example: **AES-256 encryption** is widely used to secure stored data in applications.

3. Confidentiality

Encryption ensures that sensitive information, such as passwords, personal details, and business secrets, is accessible only to authorized parties. This helps maintain privacy for users and organizations.

4. Integrity Verification

Some encryption algorithms work with hashing techniques to detect any alteration in data. This ensures that the data received is exactly what was sent, without unauthorized modifications.

- Example: Digital signatures combine encryption and hashing to provide integrity and authenticity.

5. Authentication and Access Control

Encryption plays a key role in verifying user identities through secure password storage, encrypted authentication tokens, and certificates. This ensures that only legitimate users can access certain parts of an application.

6. Regulatory Compliance

Many industries have strict data protection laws, such as:

- **GDPR** (General Data Protection Regulation)
 - **HIPAA** (Health Insurance Portability and Accountability Act)
 - **PCI-DSS** (Payment Card Industry Data Security Standard)
- Encryption is often a mandatory requirement under these regulations to ensure secure handling of personal and financial data.

7. Prevention of Data Breaches

In the event of a data breach, encrypted data is often useless to attackers without the decryption keys. This significantly reduces the impact of security incidents.

Software Applications and Its Types

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

| Aspect | System Software | Application Software |
|--------|-----------------|----------------------|
|--------|-----------------|----------------------|

| | | |
|---------------------|---|--|
| Definition | Software designed to manage and control computer hardware, and provide a platform for running application software. | Software designed to help users perform specific tasks or solve particular problems. |
| Purpose | Acts as a bridge between hardware and user applications; manages the overall functioning of the computer. | Performs specific tasks based on user requirements, such as writing documents or editing images. |
| Examples | Operating systems (Windows, Linux, macOS), device drivers, utility programs. | MS Word, Google Chrome, Photoshop, VLC Media Player. |
| Dependency | Runs in the background and is essential for the functioning of application software. | Runs on top of system software; cannot work without it. |
| User Interaction | Less direct user interaction; works automatically in the background. | Directly interacts with the user; requires user input to function. |
| Functionality Scope | General-purpose functions like managing files, memory, hardware devices, and system resources. | Specific-purpose functions like creating documents, browsing the internet, or playing music. |
| Installation | Usually comes pre-installed with the computer or device. | Installed by the user as per their needs. |
| Execution Time | Starts running when the system is turned on and stops when it is turned off. | Runs only when the user opens and uses the application. |

Software Architecture

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

Significance of Modularity in Software Architecture:

Modularity means dividing software into small, separate parts called modules.

Importance:

1. Easy to Maintain – Changes in one module don't affect others.
2. Reusability – A module can be used in other projects.
3. Easy Testing – Each module can be tested separately.
4. Faster Development – Different teams can work on different modules at the same time.
5. Scalable – New modules can be added without disturbing the old ones.

Example:

In a school management system, “Library” and “Examination” are separate modules. Changes in the library module don't affect the examination module.

Layers in Software Architecture

1. Why are layers important in software architecture?

Why Layers Are Important in Software Architecture

Layers in software architecture divide a system into different levels, each with a specific role (like presentation, business logic, and data).

Importance:

1. Organization – Keeps code neat and structured.
 2. Separation of Concerns – Each layer focuses on one task only.
 3. Easy Maintenance – Changes in one layer don't affect others.
 4. Reusability – Layers can be reused in other projects.
 5. Scalability – New features can be added without disturbing all layers.
-

Example:

In a banking app, the UI layer shows the balance, the logic layer calculates transactions, and the data layer stores records in the database.

Software Environments

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

A **development environment** is the setup—both in terms of software tools and configurations—that developers use to create, test, and debug applications.

Its importance in software production can be explained as follows:

1. Provides Necessary Tools for Development

- Includes code editors, compilers, debuggers, and libraries.
- Enables developers to write, run, and test code efficiently without constantly switching tools.

2. Ensures Consistency Across the Team

- A standard environment prevents the “it works on my computer” problem.
- By using the same programming language versions, dependencies, and configurations, all developers get consistent results.

3. Facilitates Faster Development and Debugging

- Integrated Development Environments (IDEs) offer features like code completion, syntax highlighting, and error detection.
- Debugging tools help quickly locate and fix errors.

4. Supports Testing Before Deployment

- Local environments allow developers to test new features or fixes without affecting the live application.
- This reduces the risk of bugs reaching production.

5. Encourages Safe Experimentation

- Developers can try new ideas or changes in a controlled space without damaging real data or systems.
- Version control integration in development environments helps track and revert changes if needed.

6. Improves Collaboration

- Shared environments and containerized setups (like Docker) make it easier for teams to work on the same codebase without compatibility issues.

Source Code

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

1. Source Code

- Definition: The set of instructions written by a programmer in a human-readable programming language like Python, C, or Java.
 - Form: Text files with keywords, symbols, and logic that humans can read and understand.
 - Purpose: To express the program's logic in a way that is easy to write, read, and modify.
 - Execution: Needs to be translated (compiled or interpreted) into machine code before the computer can run it.
 - Example (in C):
-

```
printf("Hello, World!");
```

2. Machine Code

- Definition: The low-level binary code that a computer's processor understands directly.
- Form: A sequence of 0s and 1s (binary numbers).
- Purpose: To instruct the CPU exactly what operations to perform.
- Execution: Can be executed directly by the computer without further translation.
- Example:

```
10111000 00000001
```

Github and Introductions

1. Why is version control important in software development?

Version control is important in software development because it helps teams and individuals manage changes to code over time in a safe, organized, and collaborative way.

1. Tracks Changes Over Time

- Records every modification made to the code.
- Allows developers to see *who* changed *what* and *when*.

2. Enables Collaboration

- Multiple developers can work on the same project without overwriting each other's work.
- Tools like Git merge contributions from different team members efficiently.

3. Protects Against Mistakes

- If a bug is introduced, you can revert to a previous working version of the code.

- Acts like a “time machine” for your project.

4. Supports Experimentation

- Developers can create *branches* to try new features or ideas without affecting the main codebase.
- If the experiment fails, it can be discarded safely.

5. Improves Project Management

- Makes it easier to review changes, approve updates, and ensure code quality.
- Integrates with issue trackers for better planning and tracking of development tasks.

Student Account in Github

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

1. Learn Real-World Development Practices

- GitHub uses **Git**, the most popular version control system in software development.
- Students get hands-on experience with tools and workflows used in professional projects.

2. Collaboration Skills

- GitHub allows multiple people to work on the same project at the same time.
- Students learn how to **collaborate, review code, and resolve conflicts**, important teamwork skills.

3. Showcase Your Work

- Students can create a public portfolio by hosting their projects on GitHub.
 - This helps when applying for internships, jobs, or college programs to show practical skills.
-

4. Access to Free Resources

- GitHub offers **free student developer packs** including free private repositories, cloud services, and tools like code editors and databases.
- This saves money and provides powerful tools for learning and building projects.

5. Project Management Tools

- GitHub includes issue tracking, project boards, and wikis, helping students organize their work and learn project management basics.

Types of Software

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

Here's a clear comparison between open-source software and proprietary software:

Open-Source Software (OSS)

- **Definition:** Software whose source code is freely available for anyone to view, modify, and distribute.
- **Access:** Users have full access to the source code.
- **Cost:** Usually free to use, but not always (some OSS have paid support or services).
- **Customization:** Highly customizable since users can change the code to fit their needs.
- **Community:** Developed and maintained by a community of developers around the world.
- **Examples:** Linux, Firefox, LibreOffice, Android (mostly).
- **Licensing:** Uses licenses like GPL, MIT, Apache that allow copying and modification under certain terms.

Proprietary Software

- **Definition:** Software owned by a company or individual, with source code kept secret.
- **Access:** Users do not have access to the source code.
- **Cost:** Usually paid software, requiring a license purchase or subscription.
- **Customization:** Limited or no customization allowed by users.
- **Development:** Developed and maintained by a single company or organization.
- **Examples:** Microsoft Windows, Adobe Photoshop, Microsoft Office.
- **Licensing:** Users get a license to use the software but cannot modify or share it.

GIT and GITHUB

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

Here's how **Git improves collaboration** in a software development team:

1. Tracks Changes Made by Everyone

- Git records who changed what and when in the codebase.
- This helps team members understand the history and purpose of changes.

2. Supports Multiple Contributors Simultaneously

- Multiple developers can work on different parts of the project at the same time without interfering with each other.

3. Uses Branching and Merging

- Developers create **branches** to work on new features or fixes independently.
 - Once ready, these branches are merged back into the main codebase, keeping work organized and isolated.
-

4. Reduces Conflicts with Version Control

- Git helps identify conflicting changes when two people edit the same file.
- It provides tools to resolve conflicts efficiently, preventing loss of work.

5. Enables Code Reviews and Quality Checks

- Teams can review changes (via pull requests) before merging them into the main project.
- This improves code quality and knowledge sharing.

6. Facilitates Rollbacks

- If a bug is introduced, Git makes it easy to revert to a previous stable version, minimizing disruption.

7. Integrates with Collaboration Platforms

- Git works seamlessly with platforms like GitHub, GitLab, and Bitbucket, which provide extra tools for issue tracking, documentation, and team communication.

Application Software

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

Role of Application Software in Businesses:

- **Automating Tasks:**
It automates routine and repetitive tasks like billing, payroll, and inventory management, reducing manual effort and errors.
- **Improving Productivity:**
Tools like word processors, spreadsheets, and project management apps help employees complete work faster and more accurately.

- **Facilitating Communication:**
Email clients, messaging apps, and video conferencing software enable smooth communication within teams and with clients.
- **Supporting Decision Making:**
Business intelligence and analytics software help analyze data and generate reports, aiding managers in making informed decisions.
- **Managing Customers:**
Customer Relationship Management (CRM) software helps track customer interactions, improve sales, and enhance customer service.
- **Handling Finances:**
Accounting software manages budgets, expenses, invoicing, and compliance with financial regulations.

Software Development Process

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

The **main stages of the software development process** typically include the following steps:

1. Requirement Analysis

- Understand and document what the users or clients need from the software.
- Define functional and non-functional requirements clearly.

2. Design

- Plan the software architecture and components.
- Create diagrams like flowcharts, data models, and user interfaces to guide development.

3. Implementation (Coding)

- Write the actual source code according to the design.
 - Developers build the software using programming languages and tools.
-

4. Testing

- Check the software for bugs, errors, and whether it meets the requirements.
- Types of testing include unit testing, integration testing, system testing, and user acceptance testing.

5. Deployment

- Release the software to the users or production environment.
- Make it available for use, often involving installation or distribution.

6. Maintenance

- Fix bugs and issues that appear after deployment.
- Update the software to improve features, security, or performance over time.

Software Requirement

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

Why is Requirement Analysis Important?

- It tells us what the software should do.
 - Helps everyone understand and agree on the software's goals.
 - Prevents mistakes by catching problems early.
 - Guides developers on what to build
 - Helps plan how much time and money is needed.
-

- Makes sure the final software meets the user's needs.

Software Analysis

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

Role of Software Analysis

- **Understand the Problem:** It helps figure out what the users really need and what the software should do.
- **Gather Requirements:** Collect all important details and features that the software must have.
- **Identify Constraints:** Find out any limits or rules that affect how the software will be built.
- **Create a Clear Plan:** Organize the information so developers know exactly what to design and build.
- **Prevent Mistakes:** Catch misunderstandings or missing parts early to avoid problems later.

System Design

1.What are the key elements of system design?

Here are the key elements of system design explained :

1. Architecture Design

- Defines the overall structure of the system — how different parts will interact and work together.

2. Data Design

- Organizes how data will be stored, managed, and accessed, usually through databases or data structures.

3. Interface Design

- Plans how users and other systems will interact with the software, including user interfaces and APIs.

4. Component Design

- Breaks down the system into smaller modules or components, each with specific responsibilities.

5. Security Design

- Defines how the system will protect data and operations from unauthorized access or attacks.

6. Performance Design

- Ensures the system meets speed, responsiveness, and resource use requirements.

Software Testing

1. Why is software testing important?

Why is Software Testing Important?

- **Finds Bugs Early:** Testing helps catch errors and problems before the software is used by real users.
-

- **Ensures Quality:** It makes sure the software works correctly and meets the requirements.
- **Improves User Experience:** Testing checks that the software is easy to use and doesn't crash or freeze.
- **Saves Time and Money:** Fixing bugs during testing is cheaper and faster than fixing them after release.
- **Increases Reliability:** Tested software is more dependable and less likely to fail in real situations.
- **Protects Security:** Testing helps find security weaknesses to keep data safe.

Maintenance

1.What types of software maintenance are there?

Here are the main types of software maintenance explained simply:

1. Corrective Maintenance

- Fixing bugs and errors found after the software is released.

2. Adaptive Maintenance

- Updating the software to work with new hardware, operating systems, or environments.
-

3. Perfective Maintenance

- Improving performance, adding new features, or making the software easier to use.

4. Preventive Maintenance

- Making changes to prevent future problems or improve software reliability.

Development

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

Here are the advantages of web applications over desktop applications in simple terms:

1. Access Anywhere

- Web apps can be used from any device with an internet connection and browser—no installation needed.

2. Easy Updates

- Updates happen on the server, so users always get the latest version without needing to download or install anything.

3. Cross-Platform Compatibility

- Web apps work on different operating systems (Windows, Mac, Linux) and devices (PC, tablet, phone) without extra work.

4. Lower Maintenance

- Since the app runs on a central server, it's easier to maintain and fix issues without needing to touch every user's device.
-

5. Cost-Effective

- No need to develop and distribute separate versions for different platforms.

6. Better Collaboration

- Web apps often allow multiple users to work together in real time, like Google Docs.

Web Application

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

Here are the advantages of web applications over desktop applications in a simple way:

1. Accessible Anywhere

You can use web apps from any device with an internet connection and a browser — no need to install anything.

2. Automatic Updates

Web apps update on the server side, so you always use the latest version without manual downloads.

3. Works on Multiple Devices

They work on different operating systems (Windows, Mac, Linux) and devices (PCs, tablets, phones) without extra effort.

4. Lower Maintenance

Since everything runs on the server, developers fix bugs and add features without bothering users.

5. Cost-Effective

No need to build different versions for different platforms.

6. Easy Collaboration

Web apps often let multiple users work together in real-time, like Google Docs.

Designing

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

What is UI/UX Design?

- **UI (User Interface):** How the app looks — buttons, colors, layout, fonts, and visual elements.
- **UX (User Experience):** How the app feels — how easy and enjoyable it is to use.

Role of UI/UX Design in Application Development

- 1. Makes the App Easy to Use**

- Good design helps users understand and navigate the app smoothly without confusion.

- 2. Improves User Satisfaction**

- A pleasant and intuitive interface keeps users happy and encourages them to keep using the app.

- 3. Increases Efficiency**

- Well-designed apps let users complete tasks quickly and with fewer errors.

- 4. Builds Brand Identity**

- Consistent UI/UX design creates a professional look and strengthens the brand's image.

- 5. Reduces Development Costs**

- Designing with users in mind early on prevents costly changes and fixes later.
-

6. Enhances Accessibility

- Good UX considers diverse users, making the app usable for people with different abilities.

Mobile Application

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

Native Mobile Apps

- **Built for:** A specific platform (like iOS or Android) using platform-specific languages (Swift/Objective-C for iOS, Java/Kotlin for Android).
- **Performance:** Fast and smooth because they are optimized for the platform.
- **Access to Features:** Full access to device features like camera, GPS, and sensors.
- **User Experience:** Better UI/UX because they follow platform design guidelines closely.
- **Development Cost:** Usually higher because you need to build separate apps for each platform.

Hybrid Mobile Apps

- **Built with:** Web technologies like HTML, CSS, and JavaScript, then wrapped in a native container.
 - **Performance:** Slightly slower compared to native apps because they run inside a web view.
 - **Access to Features:** Access to device features via plugins, but sometimes limited compared to native.
 - **User Experience:** Can be less smooth and may not fully match the platform's look and feel.
-

- **Development Cost:** Lower because you write one codebase that works on multiple platforms.

DFD (Data Flow Diagram)

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

What is a DFD?

A Data Flow Diagram (DFD) visually shows how data moves through a system — where it comes from, how it's processed, and where it goes.

Significance of DFDs in System Analysis

1. Simplifies Complex Systems

- Breaks down complicated processes into simple, understandable visuals.

2. Helps Understand Data Movement

- Shows how data flows between different parts of the system clearly.

3. Identifies System Boundaries

- Defines what is inside the system and what interacts from outside.

4. Facilitates Communication

- Provides a common visual language for analysts, developers, and users to discuss the system.

5. Detects Missing or Redundant Processes

- Helps spot errors or gaps in how data is handled.

6. Guides System Design and Development

- Acts as a blueprint to build or improve the system.
-

Desktop

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

Here's a simple comparison of desktop applications vs. web applications with their pros and cons:

Desktop Applications Pros:

- **Better Performance:** Runs directly on the computer, so often faster and more powerful.
- **Works Offline:** Doesn't need internet to function.
- **Access to Hardware:** Can use device features like printers, scanners, or USB devices easily.
- **More Control:** Can offer richer interfaces and features tailored to the OS.

Cons:

- **Installation Required:** Users must download and install the software.
- **Platform Dependent:** Separate versions needed for Windows, Mac, Linux, etc.
- **Updates are Manual:** Users need to download updates themselves unless auto-update is built-in.
- **Less Accessible:** Can only be used on the device where installed.

Web Applications Pros:

- **Accessible Anywhere:** Can be used on any device with a browser and internet.
- **No Installation:** No need to install or update software manually.
- **Cross-Platform:** Works across different operating systems and devices.
- **Easier Maintenance:** Updates happen on the server side, so users always have the latest version.

Cons:

- **Needs Internet:** Usually requires a stable internet connection.
- **Performance Limitations:** May be slower than desktop apps, especially for complex tasks.
- **Limited Hardware Access:** Less direct access to device features.
- **Security Risks:** Data travels over the internet, which can be vulnerable if not properly secured.

Flow Chart

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

How Flowcharts Help 1. Visualize Logic

- Flowcharts show the step-by-step flow of a program or system using symbols and arrows.
- This makes it easier to understand complex processes.

2. Plan Before Coding

- They help programmers plan the logic before writing code, reducing mistakes.

3. Simplify Communication

- Flowcharts provide a clear visual way to explain ideas to team members, clients, or stakeholders.

4. Identify Errors Early

- By visualizing the flow, it's easier to spot logical errors or missing steps before coding begins.

5. Documentation

- Flowcharts serve as useful documentation for future reference and maintenance.
-