Module 1 – SE -Overview of IT Industry

1.What is a Program?

Answer:

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

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

Answer:

A **program** is like a set of instructions that tells a computer what to do. Just like you might follow steps in a recipe to bake a cake, a computer follows the steps in a program to complete a task.

Humans write these instructions using special languages called **programming languages** (like Python, C, or Java), because computers don’t understand human language.

**How Does a Program Work?**

1. **Writing the Instructions**: A programmer writes a list of steps (called code) that tells the computer what to do.
2. **Running the Program**: The computer reads the code one line at a time and carries out each instruction.
3. **Processing Data**: The program can take input (like your name), do something with it (like count the letters), and then show the result (like “Your name has 6 letters”).
4. **Producing Output**: After following the instructions, the computer gives you the result—this could be text, images, sounds, or actions like saving a file.

3. What is Programming?

Answer:

**Programming** is the process of **writing instructions** (called code) that a computer can understand and follow to perform specific tasks.

It’s how humans **communicate with computers** to make them do things—like displaying a webpage, solving math problems, running games, or controlling machines.

4.Types of Programming Languages

Answer:

| **Language Type** | **Examples** | **Main Use** |
| --- | --- | --- |
| Low-Level | Assembly, Machine Code | Hardware control, OS |
| High-Level | Python, Java, C++ | General programming |
| Procedural | C, Pascal | Step-by-step programs |
| Object-Oriented | Java, C++, Python | Reusable, modular programs |
| Scripting | JavaScript, PHP, Python | Web, automation, quick tasks |
| Functional | Haskell, Lisp | Pure functions, data flow |
| Markup/Query (Special) | HTML, CSS, SQL | Web design, data handling |
|  |  |  |

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

Answer:

| **Feature** | **High-Level Language** | **Low-Level Language** |
| --- | --- | --- |
| 🗣️ **Ease of Use** | Easy to read, write, and understand (like English) | Hard to read; closer to binary/machine code |
| 💡 **Abstraction Level** | High (hides hardware details) | Low (gives direct access to hardware) |
| ⚙️ **Examples** | Python, Java, C++, JavaScript | Assembly, Machine Language |
| 🔄 **Portability** | Code can run on different systems with little change | Code is hardware-specific |
| 🧩 **Memory Management** | Handled automatically or simply | Done manually by the programmer |
| 🧪 **Execution Speed** | Slower (needs to be translated into machine code) | Faster (closer to machine language) |
| 🔧 **Control Over Hardware** | Less control | Full control |
| 🧑‍💻 **Best For** | App development, websites, data processing | Operating systems, device drivers, embedded systems |

6.What is World Wide Web & How Internet Work

Answer:

The **World Wide Web (WWW)** is a system of **interconnected web pages** and resources that you can access using a web browser like Chrome, Firefox, or Safari.

**Here's internet how it works:**

1. **You type a website address (URL)**
   * Example: www.amazon.com
2. **DNS Translates it**
   * DNS (Domain Name System) changes the web address into an **IP address** (like 192.168.1.1) so the computer knows where to find the website.
3. **Request is sent to a Web Server**
   * Your device sends a request over the Internet to the server where the website is stored.
4. **Server Sends Data Back**
   * The server sends back the website’s files (text, images, layout) in small pieces called **data packets**.
5. **Your Browser Displays the Page**
   * Your browser collects all the packets, reassembles them, and shows you the web page.

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

Answer:

**Client:**

* The **client** is usually **your device** (computer, phone, tablet) that requests data or services from a server.
* The client uses a **web browser** (like Chrome or Firefox) to send requests to servers using **HTTP or HTTPS**.
* Example: When you type www.youtube.com, your browser (client) asks the server to show the YouTube homepage.

**🖥️ Server:**

* The **server** is a **powerful computer** that stores websites, files, databases, or services.
* It **listens for requests** from clients, processes them, and sends back the correct response.
* Example: When your browser requests www.youtube.com, YouTube’s server sends back the video data, layout, and images.

**🧱 Network Layers on Client and Server**

Web communication follows the **OSI Model** (7 layers) or the simplified **TCP/IP Model** (4 layers). Let's focus on the **TCP/IP model**, which is most practical.

**✅ TCP/IP Layers (Both Client and Server):**

| **Layer** | **Role** |
| --- | --- |
| **Application Layer** | Client: Sends HTTP request via browser.  Server: Responds with HTML page. |
| **Transport Layer** | Manages **reliable delivery** using TCP (e.g., breaks data into packets). |
| **Internet Layer** | Adds IP address info; routes packets through the network. |
| **Network Access Layer** | Uses physical hardware (like cables, Wi-Fi, router) to send/receive data. |

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

Answer:

The **TCP/IP model** is a set of rules (protocols) that **allow computers to communicate over the Internet**. It breaks down communication into **layers**, with each layer having a specific job.

TCP/IP stands for:

* **TCP** – Transmission Control Protocol
* **IP** – Internet Protocol

Together, they handle how data is sent, routed, and received.

**🧱 Layers of the TCP/IP Model**

The TCP/IP model has **4 layers**, and data flows from top to bottom when sending, and bottom to top when receiving.

**1. 🔗 Application Layer**

* **Topmost layer** that interacts with the user or application.
* Deals with things like **HTTP**, **HTTPS**, **FTP**, **DNS**, and **Email (SMTP, POP3)**.
* Provides **user services** like web browsing, file transfer, email, etc.

✅ **Example:** When you open a website, your browser uses HTTP to request a page from a server.

**2. 📦 Transport Layer**

* Responsible for **breaking data into smaller parts (packets)** and making sure they get to the destination **safely and in order**.
* Uses protocols like:
  + **TCP (Transmission Control Protocol)** – Reliable, connection-based (used in web, email).
  + **UDP (User Datagram Protocol)** – Faster, but less reliable (used in video streaming, games).

✅ **Example:** TCP ensures that your entire web page is delivered correctly, without missing parts.

**3. 🌐 Internet Layer**

* Handles **routing and addressing** so data reaches the correct destination.
* Uses **IP (Internet Protocol)** to assign unique addresses to devices (IP addresses).
* Also manages data packet movement between networks.

✅ **Example:** Ensures your request to google.com goes to the correct Google server by using its IP address.

**4. 📡 Network Access Layer (Link Layer)**

* Deals with the **physical connection** between devices (like cables, Wi-Fi, network cards).
* Converts packets into signals (electrical, light, or radio) that can travel across hardwar

9. : Explain Client Server Communication

**1. Client Sends a Request**

* You (the client) open a browser and type www.example.com.
* The browser sends an **HTTP request** to the server asking for the website.

**2. Server Receives the Request**

* The server (like Google or Facebook’s computer) receives the request.
* It looks for the correct web page or data.

**3. Server Sends a Response**

* The server sends back a response (HTML, CSS, images, etc.).
* It reaches your device over the Internet.

**4. Client Displays the Result**

* Your browser reads the response and **shows the website**

**10.** **How does broadband differ from fiber-optic internet? Protocols**

**1. Broadband Internet**

* **Broadband** is a general term for high-speed internet that’s always on.
* It includes several technologies like:
  + DSL (Digital Subscriber Line)
  + Cable internet
  + Satellite internet
* Uses traditional copper wires or coaxial cables.
* Speed varies widely (usually from a few Mbps to hundreds of Mbps).
* More prone to signal loss and interference.

**2. Fiber-Optic Internet**

* Uses **thin strands of glass or plastic fibers** to transmit data as **light signals**.
* Provides **much faster speeds** (up to Gbps - gigabits per second).
* Has very low latency and is less affected by interference.
* Can handle large amounts of data over long distances without degradation.
* Considered the most advanced and reliable internet connection today.

11. What are the differences between HTTP and HTTPS protocols? Application Security

| **Feature** | **HTTP (HyperText Transfer Protocol)** | **HTTPS (HTTP Secure)** |
| --- | --- | --- |

|  |  |  |
| --- | --- | --- |
| **Full Form** | HyperText Transfer Protocol | HyperText Transfer Protocol Secure |

|  |  |  |
| --- | --- | --- |
| **Security** | **Not secure**; data is sent in plain text | **Secure**; data is encrypted using SSL/TLS |

|  |  |  |
| --- | --- | --- |
| **Port Used** | Port **80** | Port **443** |

|  |  |  |
| --- | --- | --- |
| **Encryption** | No encryption; vulnerable to interception | Encrypts data to prevent eavesdropping |

|  |  |  |
| --- | --- | --- |
| **Certificate** | No certificate required | Requires SSL/TLS certificate from a trusted authority |

|  |  |  |
| --- | --- | --- |
| **Data Integrity** | No guarantee data isn’t altered | Ensures data integrity and authenticity |

|  |  |  |
| --- | --- | --- |
| **Performance** | Slightly faster (less overhead) | Slightly slower due to encryption process |

|  |  |  |
| --- | --- | --- |
| **Use Case** | Suitable for public information | Necessary for sensitive transactions (login, payment) |

12. What is the role of encryption in securing applications Software Applications and Its Types

**What is Encryption?**

Encryption is the process of **converting readable data (plaintext) into an unreadable format (ciphertext)** using algorithms, so only authorized parties can decode and read it.

**Why Encryption is Important in Application Security:**

1. **Protects Sensitive Data:**  
   Encrypts passwords, personal information, financial data so attackers can’t read them even if they intercept the data.
2. **Secures Data in Transit:**  
   When data moves between clients and servers (like in HTTPS), encryption prevents eavesdropping or tampering.
3. **Secures Data at Rest:**  
   Data stored in databases or files can be encrypted to protect it in case of unauthorized access.
4. **Ensures Data Integrity and Authenticity:**  
   Encryption can help verify that data hasn’t been altered or forged during transmission.
5. **Builds User Trust:**  
   Users feel safe when their data is protected by encryption, essential for online banking, e-commerce, etc.

13. : What is the difference between system software and application software? Software Architecture

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

|  |  |  |
| --- | --- | --- |
| **Purpose** | Manages hardware and provides a platform for other software | Performs specific tasks to help users |

|  |  |  |
| --- | --- | --- |
| **Interaction** | Directly interacts with hardware | Runs on top of system software |

|  |  |  |
| --- | --- | --- |
| **Examples** | Operating Systems (Windows, Linux), Device Drivers, Utilities | Word processors, browsers, games, media players |

|  |  |  |
| --- | --- | --- |
| **Users** | Mostly used by the system and developers | Used directly by end-users |

|  |  |  |
| --- | --- | --- |
| **Dependency** | Essential for running the computer | Depends on system software to function |

14. What is the significance of modularity in software architecture? Layers in Software Architecture

**significance of Modularity in Software Architecture**

**Modularity** means designing a software system as a set of **separate, independent modules or components** that can be developed, tested, and maintained independently.

**Why is Modularity Important?**

1. **Easier Maintenance:**  
   Bugs or updates can be fixed or made in one module without affecting the entire system.
2. **Reusability:**  
   Modules can be reused in different parts of the application or even in other projects.
3. **Parallel Development:**  
   Different teams can work on different modules simultaneously, speeding up development.
4. **Improved Understandability:**  
   Smaller modules are easier to understand than a big monolithic codebase.
5. **Scalability:**  
   Modular systems can be expanded or enhanced by adding new modules without disturbing existing ones.
6. **Isolation:**  
   Faults in one module are less likely to affect others, improving system robustness.

**🏢 Layers in Software Architecture**

**Layered architecture** is a common style where the system is divided into distinct layers, each with specific responsibilities.

**Typical Layers:**

| **Layer Name** | **Description** |
| --- | --- |
| **Presentation Layer (UI Layer)** | User interface; handles user input and display |
| **Business Logic Layer (Application Layer)** | Processes data, enforces rules and logic |
| **Data Access Layer** | Manages data storage and retrieval (databases) |
| **Database Layer** |  |

15. Why are layers important in software architecture?

**Importance of Layers in Software Architecture**

1. **Separation of Concerns**  
   Layers divide the system into distinct parts, each handling a specific responsibility. This makes the system easier to understand and manage.
2. **Improved Maintainability**  
   Changes in one layer (e.g., the user interface) can be made without affecting other layers (e.g., data storage). This reduces the risk of bugs and makes updates simpler.
3. **Reusability**  
   Layers can be reused in different projects or parts of the same project. For example, the data access layer can be used by multiple applications.
4. **Scalability and Flexibility**  
   You can scale or replace individual layers independently. For example, upgrading the database layer without changing the business logic.
5. **Better Testing**  
   Layers allow testing parts of the system independently, making it easier to identify and fix issues.
6. **Encapsulation**  
   Each layer hides its internal workings from others, exposing only what is necessary through well-defined interfaces.
7. **Parallel Development**  
   Different teams can work on different layers simultaneously, speeding up development.

16. Explain the importance of a development environment in software production. Source Code

**Importance of a Development Environment in Software Production**

A **development environment** is the set of tools, software, and configurations that developers use to write, test, and debug their code before it’s released.

**Why is it important?**

1. **Safe Testing Space**  
   Developers can write and test code without affecting the live (production) system, preventing bugs from impacting users.
2. **Consistency**  
   Provides a consistent setup for all developers, so code behaves the same way on everyone’s machine.
3. **Efficiency and Productivity**  
   Includes helpful tools like code editors, debuggers, and version control, speeding up development.
4. **Error Detection and Debugging**  
   Allows developers to find and fix errors early through debugging and automated testing.
5. **Collaboration**  
   Supports version control systems (like Git), enabling multiple developers to work together smoothly.
6. **Environment Configuration**  
   Simulates real-world scenarios (databases, servers, APIs) so the application behaves as expected.

**💻 What is Source Code?**

* **Source code** is the human-readable set of instructions written by programmers using a programming language.
* It is the **blueprint** of the software.
* The source code is then **compiled or interpreted** into machine code that the computer can execute.
* Maintaining clean and well-documented source code is essential for software quality and future updates.

17. What is the difference between source code and machine code? Github and Introductions

| **Feature** | **Source Code** | **Machine Code** |
| --- | --- | --- |
| **Definition** | Human-readable instructions written by programmers in languages like Python, Java, C++ | Binary code (0s and 1s) that a computer’s CPU can directly execute |
| **Readability** | Readable and understandable by humans | Not readable by humans without special tools |
| **Language** | Written in high-level or low-level programming languages | Written in binary, specific to processor architecture |
| **Purpose** | To define the program’s logic and behavior | To be executed directly by the computer hardware |
| **Conversion** | Needs to be compiled or interpreted into machine code | Executed directly by the CPU |
| **Example** | print("Hello, World!") (Python) | 10101000 00001111 ... (binary instructions) |

**GitHub and Introduction**

**What is GitHub?**

* **GitHub** is a **web-based platform** for version control and collaboration.
* It uses **Git**, a distributed version control system, to track changes in source code during software development.
* Allows multiple developers to work on a project simultaneously, manage versions, and merge code changes.
* Provides features like issue tracking, project management, code review, and hosting repositories.

**Why Use GitHub?**

* **Collaboration:** Teams can work together from anywhere.
* **Version Control:** Keeps history of code changes, making it easy to revert or track progress.
* **Open Source Hosting:** Many open-source projects are hosted on GitHub.
* **Integration:** Supports many tools and continuous integration pipelines.

20. Why is version control important in software development?

1. **Tracks Changes Over Time**  
   Version control systems keep a detailed history of every change made to the codebase, including who made the change and why. This helps in understanding the evolution of the project.
2. **Enables Collaboration**  
   Multiple developers can work on the same project simultaneously without overwriting each other's work. Changes can be merged carefully.
3. **Facilitates Bug Tracking and Fixes**  
   If a bug is introduced, version control allows developers to revert to a previous stable version quickly.
4. **Supports Experimentation**  
   Developers can create branches to try out new features or fixes without affecting the main codebase, ensuring stability.
5. **Improves Code Quality**  
   Code reviews and pull requests are easier with version control, leading to better-reviewed and more reliable code.
6. **Provides Backup and Recovery**  
   The codebase is stored safely in repositories, protecting against accidental data loss.
7. **Simplifies Release Management**  
   Helps manage different versions of software for production, testing, or development environments.

21. : What are the benefits of using Github for students?

 **Learn Real-World Version Control**  
GitHub teaches students how to use **Git**, the industry-standard version control system, which is essential for collaborative software development.

 **Collaborate Easily**  
Students can work together on projects, practice teamwork, and manage code changes smoothly.

 **Build a Portfolio**  
Students can showcase their projects publicly, which can impress potential employers or for college applications.

 **Access to Open Source Projects**  
GitHub hosts millions of open-source projects, giving students opportunities to contribute, learn from others, and improve their skills.

 **Free Resources and Tools**  
GitHub offers free private repositories, GitHub Pages (to host websites), and learning labs tailored for students.

 **Track Progress and History**  
Students learn to document their coding journey, track changes, and revert mistakes easily.

 **Integrates with Many Tools**  
Supports integration with code editors, CI/CD pipelines, and project management tools.

 **Community and Networking**  
Connect with other developers, join organizations, and participate in coding events or hackathons.

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

| **Feature** | **Open-Source Software** | **Proprietary Software** |
| --- | --- | --- |
| **Source Code Access** | Source code is freely available to view, modify, and distribute | Source code is closed and not available to users |
| **Licensing** | Usually under licenses like GPL, MIT, Apache that allow free use and modification | Licensed with restrictions; users pay for usage and cannot modify the code |
| **Cost** | Mostly free to use | Usually requires purchasing a license or subscription |
| **Customization** | Highly customizable by users or organizations | Limited customization; only the vendor can make changes |
| **Support** | Community-driven support; some projects offer paid support | Vendor provides official support, updates, and maintenance |
| **Development Model** | Collaborative and transparent, with contributions from many developers | Developed and maintained by a single company or organization |
| **Examples** | Linux, Mozilla Firefox, LibreOffice | Microsoft Windows, Adobe Photoshop, Microsoft Office |
| **Security** | Transparent code allows many eyes to find/fix bugs, but risks depend on community involvement | Vendor is responsible for security patches; users depend on vendor’s responsiveness |

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

 **Distributed Version Control**  
Every team member has a complete copy of the project’s history, so they can work independently and offline, then sync changes later.

 **Branching and Merging**  
Developers create separate branches to work on features or bug fixes without affecting the main codebase. Once ready, branches can be merged smoothly.

 **Conflict Resolution**  
Git helps identify and manage conflicts when multiple people edit the same files, ensuring changes don’t accidentally overwrite each other.

 **Clear History and Accountability**  
Every change is tracked with author info and timestamps, making it easy to see who changed what and why.

 **Pull Requests & Code Reviews**  
Git platforms like GitHub support pull requests where team members review and discuss changes before merging, improving code quality.

 **Backup and Recovery**  
Since the full code history is stored locally and remotely, it reduces risks of data loss.

 **Facilitates Continuous Integration**  
Git works well with automated testing and deployment pipelines, ensuring code changes are integrated and tested frequently.

24. What is the role of application software in businesses? Software Development Process

 **Automates Tasks:** Helps automate routine business tasks such as accounting, inventory management, payroll, and customer relations, increasing efficiency.

 **Improves Productivity:** Enables employees to perform tasks faster and more accurately (e.g., word processing, spreadsheets, email).

 **Supports Decision Making:** Provides tools for data analysis and reporting, helping management make informed decisions.

 **Enhances Communication:** Facilitates communication through email clients, messaging apps, and collaboration platforms.

 **Manages Resources:** Helps track and optimize resources such as time, finances, and human resources.

 **Customer Engagement:** Supports marketing, sales, and customer support through CRM (Customer Relationship Management) software.

 **Competitive Advantage:** Custom or specialized application software can give businesses an edge by tailoring solutions to specific needs.

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

1. **Requirement Analysis**
   * Gathering and understanding what the users and stakeholders need from the software.
   * Documenting detailed functional and non-functional requirements.
2. **Design**
   * Planning the software’s architecture and how the system components will work together.
   * Creating design documents, UI/UX layouts, and database schemas.
3. **Implementation (Coding)**
   * Writing the actual source code based on the design.
   * Developers build the software modules and features.
4. **Testing**
   * Checking the software for bugs, errors, and verifying that it meets the requirements.
   * Includes unit testing, integration testing, system testing, and user acceptance testing.
5. **Deployment**
   * Releasing the software to the production environment where users can access it.
   * May include installation, configuration, and rollout planning.
6. **Maintenance**
   * Ongoing updates, bug fixes, and improvements after deployment.
   * Adapting the software to changing user needs or environments.

**26.** **E: Why is the requirement analysis phase critical in software development? Software Analysis**

 **Foundation of the Project**  
It defines **what** the software should do—clear, detailed requirements guide the entire development process.

 **Avoids Misunderstandings**  
Helps ensure that developers, clients, and stakeholders have a shared understanding of the project goals.

 **Reduces Risks and Costs**  
Identifying requirements early helps catch potential problems or missing features before coding begins, saving time and money.

 **Guides Design and Development**  
Good requirements enable precise system design and coding that meets user needs.

 **Facilitates Testing**  
Clear requirements provide the basis for creating test cases to verify if the software works correctly.

 **Improves Customer Satisfaction**  
When the software meets documented requirements, users are more likely to be satisfied with the final product.

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

 **Understanding the Problem**  
Software analysis helps gather and clarify the exact needs and problems that the software must address. It ensures that the development team fully understands what the users and stakeholders want.

 **Defining Requirements Clearly**  
It transforms vague ideas or business needs into clear, detailed, and actionable requirements that guide design and development.

 **Identifying Constraints and Risks**  
Analysis uncovers any limitations such as technical, legal, or business constraints and potential risks early in the project.

 **Improving Communication**  
Acts as a bridge between stakeholders (clients, users) and developers by translating business language into technical specifications.

 **Facilitating Better Design**  
By thoroughly analyzing requirements and processes, it supports designing an effective and efficient system architecture.

28. : What are the key elements of system design?

 **Architecture Design**

* Defines the overall structure of the system.
* Decides how different components (modules, services, databases) interact.
* Includes decisions on client-server setup, layers, and system patterns.

 **Data Design**

* Focuses on how data is stored, organized, and accessed.
* Includes database schemas, data models, and data flow.

 **Interface Design**

* Designs how users and other systems interact with the software.
* Includes user interfaces (UI), APIs, and communication protocols.

 **Component Design**

* Details the internal logic and behavior of individual modules or components.
* Specifies responsibilities, inputs, outputs, and internal workflows.

 **Security Design**

* Incorporates measures to protect data and operations from unauthorized access or attacks.
* Defines authentication, authorization, encryption, and auditing.

 **Performance Design**

* Ensures the system meets required performance criteria such as speed, scalability, and responsiveness.

 **Scalability and Maintainability**

* Plans for future growth and easy updates/maintenance.
* Designs modular, loosely coupled components.

29. Why is software testing important?

1. **Ensures Quality**  
   Testing verifies that the software works as intended and meets all specified requirements.
2. **Detects Bugs Early**  
   Identifies errors, defects, or issues before the software is released, saving time and cost on fixes later.
3. **Improves User Experience**  
   Helps ensure the software is reliable, stable, and user-friendly, which keeps users satisfied.
4. **Increases Security**  
   Finds vulnerabilities or weaknesses that could be exploited, helping protect the system and data.
5. **Validates Performance**  
   Confirms that the software performs well under expected loads and conditions.
6. **Facilitates Maintenance**  
   Testing during development helps maintain code quality, making future updates easier and safer.
7. **Supports Compliance**  
   Ensures the software meets industry standards, legal, or regulatory requirements.

30. What types of software maintenance are there?

1. **Corrective Maintenance**
   * Fixes bugs and errors discovered after the software is deployed.
   * Addresses issues that affect functionality or performance.
2. **Adaptive Maintenance**
   * Updates the software to work in a changed environment, such as new operating systems, hardware, or third-party software.
   * Keeps the software compatible with evolving platforms.
3. **Perfective Maintenance**
   * Enhances or improves existing features based on user feedback or new requirements.
   * Focuses on performance improvements, usability, or adding new functionalities.
4. **Preventive Maintenance**
   * Anticipates potential future problems and modifies the software to prevent them.
   * Involves code refactoring, updating documentation, and improving maintainability.

31. What are the key differences between web and desktop applications? • Web Application

| **Aspect** | **Web Application** | **Desktop Application** |
| --- | --- | --- |
| **Installation** | Runs in a web browser; no installation needed | Requires installation on individual computers |
| **Access** | Accessible from anywhere with internet and browser | Access limited to the installed machine |
| **Updates** | Updated centrally on the server; users always get latest version | Updates must be installed on each machine manually or via update software |
| **Platform Dependency** | Platform-independent; works on any OS with a browser | Platform-specific; needs versions for Windows, macOS, Linux, etc. |
| **Performance** | Depends on internet speed and browser capabilities | Generally faster and can use more system resources |
| **Storage** | Data stored on servers or cloud | Data stored locally on the computer |
| **Security** | Relies on web security measures like HTTPS, firewalls | Relies on local machine security and OS protection |

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

1. **Accessibility Anywhere**  
   Users can access web apps from any device with an internet connection and browser—no installation required.
2. **Cross-Platform Compatibility**  
   Web apps run on any operating system (Windows, macOS, Linux, mobile OS) without needing separate versions.
3. **Easy Updates and Maintenance**  
   Updates are made on the server side and instantly available to all users, no need for manual installations or downloads.
4. **Lower Cost of Deployment**  
   No need to distribute software physically or manage individual licenses per machine.
5. **Centralized Data Storage**  
   Data is stored on remote servers or cloud, enabling easier backups, data sharing, and synchronization across devices.
6. **Simplified Collaboration**  
   Many web apps support real-time collaboration and sharing among multiple users seamlessly.
7. **Reduced Hardware Requirements**  
   Since processing is done server-side, client devices require less computing power.
8. **Scalability**  
   Web applications can be scaled easily to support more users without needing to upgrade individual user devices.

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

**Enhances User Experience (UX)**

* Focuses on making the application **easy, intuitive, and enjoyable** to use.
* Considers user behavior, needs, and pain points to design smooth workflows.

**2. Creates an Attractive Interface (UI)**

* Designs the **look and feel** — colors, typography, buttons, icons, layouts.
* Makes the app visually appealing and consistent with brand identity.

**3. Improves Usability**

* Ensures users can accomplish tasks **efficiently and without confusion**.
* Reduces errors, learning curve, and frustration.

**4. Boosts User Engagement and Retention**

* Good UI/UX keeps users coming back and promotes positive feedback and word-of-mouth.

**5. Supports Accessibility**

* Designs for diverse users including those with disabilities, making the app inclusive.

**6. Reduces Development Costs**

* Early UI/UX research and prototyping help catch design flaws before coding, reducing costly rework.

**7. Aligns with Business Goals**

* A well-designed app can increase conversions, customer satisfaction, and overall business success

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

| **Aspect** | **Native Apps** | **Hybrid Apps** |
| --- | --- | --- |
| **Development** | Built specifically for one platform (iOS, Android) using platform-specific languages (Swift/Objective-C for iOS, Java/Kotlin for Android) | Built using web technologies (HTML, CSS, JavaScript) wrapped in a native container |
| **Performance** | High performance, fast, smooth user experience, direct access to device features | Slightly slower performance due to an extra abstraction layer, but improving with frameworks |
| **Access to Device Features** | Full access to device hardware and APIs (camera, GPS, notifications) | Access to many device features via plugins but sometimes limited or delayed support |
| **User Interface** | Uses platform-specific UI components, offering a look and feel consistent with the OS | UI is often uniform across platforms, may not always feel “native” |
| **Development Time & Cost** | Longer development time and higher cost as separate codebases are needed for each platform | Faster development and lower cost since one codebase works across platforms |
| **Maintenance** | Requires separate updates and maintenance per platform | Single codebase makes maintenance and updates easier and quicker |
| **Offline Capability** | Usually better offline support | Depends on how the app is built; offline functionality can be limited |

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

 **Visualizes System Processes Clearly**  
DFDs graphically represent how data moves through a system, showing inputs, processes, storage, and outputs in an easy-to-understand way.

 **Simplifies Complex Systems**  
Breaks down complex processes into smaller, manageable parts, helping analysts and stakeholders grasp the system structure.

 **Facilitates Communication**  
Provides a common language for developers, analysts, and clients to discuss system requirements and workflows without technical jargon.

 **Identifies System Boundaries**  
Helps define what is inside the system and what interacts with it externally (e.g., users, other systems).

 **Detects Redundancies and Inefficiencies**  
By mapping data flow, DFDs can uncover duplicated processes or unnecessary data movements to optimize system design.

 **Supports Requirement Validation**  
Stakeholders can review DFDs to verify that all required data flows and processes are captured correctly.

 **Guides System Design and Development**  
Acts as a blueprint for developers to understand data interactions and build the system accordingly.

36. What are the pros and cons of desktop applications compared to webapplications?

**Desktop Applications**

**Pros:**

* **Better Performance:** Run directly on the device, often faster and more responsive.
* **Offline Access:** Can work without an internet connection.
* **Full Access to System Resources:** Better integration with hardware like printers, scanners, or GPUs.
* **Enhanced Security:** Data can be stored locally, reducing exposure to some online threats.

**Cons:**

* **Platform Dependency:** Need separate versions for Windows, macOS, Linux, etc.
* **Installation Required:** Users must download and install the software.
* **Update Management:** Updates must be installed manually or via update tools.
* **Limited Accessibility:** Only accessible on the installed device unless remote access is set up.

**Web Applications**

**Pros:**

* **Cross-Platform:** Works on any device with a web browser and internet.
* **No Installation Needed:** Access instantly without setup.
* **Easier Updates:** Updated on the server; users always get the latest version.
* **Centralized Data Storage:** Easier backup and sharing.

**Cons:**

* **Dependent on Internet:** Requires a stable internet connection.
* **Performance Limitations:** Can be slower due to network latency and browser restrictions.
* **Limited Access to Hardware:** Restricted access to device resources compared to desktop apps.
* **Security Concerns:** Data is transmitted over the internet, potentially vulnerable if not properly secured

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

1. **Visual Representation of Logic**  
   Flowcharts use symbols and arrows to depict the step-by-step flow of a program or system process, making complex logic easier to understand.
2. **Simplify Problem-Solving**  
   Breaking down processes visually helps identify the sequence of operations, decision points, and possible paths, aiding clear problem analysis.
3. **Improve Communication**  
   Flowcharts provide a universal language that developers, analysts, and stakeholders can understand, facilitating better collaboration.
4. **Plan Before Coding**  
   They help programmers design the algorithm and system workflow before writing actual code, reducing errors and confusion.
5. **Debugging and Maintenance**  
   Visual flowcharts make it easier to trace the flow of a program, helping locate logic errors or inefficiencies during debugging or later updates.
6. **Documentation**  
   Serve as useful documentation for the system or program logic, helping new developers quickly understand the design.