Module 1 : SE Overview of IT industry

1. **What is a Program?**

→ Program is a set of Instructions.

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

→ A program is instructions for a computer to execute specific tasks.
→ It's functions :-

- Written in programming language(C, Dart, Python, etc.)
- Compilation/Interpretation
- Execution
- Input–Processing–Output

3. **What is Programming?**

→ Programming is the process of writing instructions that a computer can understand and execute to perform specific tasks.

4. **What are the key steps involved in the programming process?**

→ **Problem Definition :-** Clearly identify the problem you want to solve with the program.
→ **Design :-** Plan the solution, including algorithms, data structures, and the overall program structure.
→ **Coding :-** Write the actual program code using a chosen programming language, following its syntax and conventions.
→ **Testing :-** Thoroughly test the program for errors and bugs, ensuring it functions as expected.
→ **Documentation :-** Create documentation for the program, including user guides, code comments, and other relevant information.
→ **Maintenance :-** Maintain the program over time, including bug fixes, updates, and improvements.

5. **Types of Programming Languages**

→ Procedural Languages
- C, Pascal, etc.

→ Object-Oriented Languages
- C++, Java, etc.

→ Logical Languages
- Prolog, etc.

→ Functional Languages
- Python, etc.

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

→ **High-Level Language**

- Easy to read and write.
- Looks like English.
- Used for building apps, websites, games, etc.
- Example: Python, Java, C++

→ **Low-Level Language**

- Hard to read, looks like numbers or symbols.
- Directly talks to the computer's hardware.
- Used for making operating systems, device drivers.
- Example: Machine Language, Assembly

7. World Wide Web & How Internet Works

→ The World Wide Web is a collection of web pages and websites that you can access using the Internet through a web browser like Chrome or Firefox.
→ Uses HTTP/HTTPS to transfer data.
→ Pages are written in HTML**.**
→ You access it by typing a URL (like www.google.com).

→ **How the Internet Works (Simple Steps)**

1. You enter a website URL

- Like www.youtube.com in a browser.

2. DNS (Domain Name System)

- It converts the name www.youtube.com into an IP address (like 142.250.190.206), which computers understand.

3. Your device contacts the server

- Using the IP address, your device sends a request to the web server that hosts the website.

4. Server sends data

- The web server sends back HTML, CSS, images, etc. for the website.

5. Browser displays the page

- Your browser puts it all together and shows you the webpage.

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

→ **Client Side**

- Sends a request to a server for a web page or file.
- Displays the information (like a website) received from the server.
- Example : When you type www.google.com and press Enter, your browser (client) sends a request.

→ **Server Side**

- Receives requests from clients.
- Processes the request (e.g., find the web page or run a program).
- Sends back a response (like HTML, images, or data).
- Example : The Google server receives your request and sends back the Google homepage.

9. Network Layers on Client and Server

Layer Number	Layer Name	Function
7	Application	User interaction (e.g., browser, app)
6	Presentation	Data format, encryption, compression
5	Session	Start/end communication (session)
4	Transport	Breaks data into packets (e.g., TCP)
3	Network	Chooses the route for data (IP address)
2	Data Link	Moves data between devices on the same network
1	Physical	Actual hardware (cables, Wi-Fi, etc.)

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

→ The TCP/IP model is a **set of rules** that allow computers to communicate over the Internet.

→ It helps in sending, receiving, and routing data from one device to another.

→ It's the foundation of the Internet.

TCP/IP Layer	Function	Example Protocols
Application	User services (web, email)	HTTP, FTP, SMTP, DNS
Transport	Reliable data transfer	TCP, UDP
Internet	Routing and addressing	IP, ICMP
Network Access	Physical transmission (hardware)	Ethernet, Wi-Fi

11. Client and Servers

→ **Client :** A client is a device or software (like a web browser) that requests services or data from a server.

→ **Server :** A server is a powerful computer or software that provides services or data to clients.

Client	**Server**
Sends requests	Responds to requests
Uses services	Provides services
Example: Browser	Example: Web server

12. Explain Client Server Communication

1. Client Sends a Request

→ A user performs an action (e.g., types a URL, clicks a button).

→ The client (browser/app) sends a request to the server.

→ The request includes:

- Type of request (GET, POST, etc.)
- Data (like login info or search term)
- Destination (URL or IP address)

2. Server Receives the Request

→ The server checks what the client is asking for.

→ It may:

- Retrieve data from a database
- Run a program or process logic
- Return a web page**,** file, or message

3. Server Sends a Response

→ The server prepares the result (HTML, JSON, image, etc.).

→ Sends it back to the client over the network.

4. Client Displays the Response

→ The client receives the data.

→ Browser or app displays it to the user (like showing a web page or a success message).

13. Types of Internet Connections

Type	Speed	Used In
Dial-up	Very Slow	Old homes (rare)
DSL	Moderate	Homes/offices
Cable	Fast	Homes (shared areas)
Fiber Optic	Very Fast	Cities, tech hubs
Satellite	Moderate	Remote villages, rural areas
Wi-Fi	Varies	Homes, cafes, offices
Mobile Data	Fast (4G/5G)	Phones, on-the-go use

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

→ **Broaband :**

- A general term for high-speed internet.
- Uses existing infrastructure (like phone or cable lines).
- Speed depends on the type (DSL is slower, cable is faster).
- More **common** in homes.

→ **Fiber :**

- A type of broadband that uses thin glass fibers to transmit data as light signals**.**
- Much **f**aster and more reliable than other broadband types.
- Speed can reach 1 Gbps or more.
- Best for gaming**,** streaming, video calls, etc.

15. Protocols

→ A protocol is a set of rules that computers follow to communicate with each other over a network like the Internet.

→ The protocol can be described as an approach to rules that enable a couple of entities of a communication program to transfer information through any type of variety of a physical medium.

→ **Types of Protocols**

1. HTTP or HTTPS
2. FTP(File Transfer protocols)
3. Email Protocols(POP3,SMTP)
4. TCP(Transmission control protocol) and UDP(User Datagram Protocol)

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

Feature	HTTP	HTTPS
Full Form	HyperText Transfer Protocol	HyperText Transfer Protocol **Secure**
Security	Not secure (data is sent as plain text)	Secure (data is encrypted)
Encryption	No encryption	Uses SSL/TLS for encryption
Risk	Can be hacked or intercepted	Safer from hackers and attacks
URL Format	http://www.example.com	https://www.example.com
Use Case	Used for non-sensitive websites	Used for login, payments, personal info
Indicator	No lock icon	Lock icon in browser address bar

17. Application Security

→ Application Security means protecting software applications from threats and attacks.

→ It ensures that your apps are safe to use**,** and that data is not stolen or damaged.

→ its ultimate purpose is to improve security practices and, as a result, detect, repair, and, ideally, avoid security flaws in applications.

→ It covers the entire application life cycle, including requirements analysis, design, implementation, testing, and maintenance.

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

→ Encryption is the process of converting readable data into unreadable code so that only authorized users can understand it.

→ **Roles :**

- **Protects sensitive data :-** Usernames, passwords, bank details
- **Secures communication :-** Chat apps, payment apps use encryption (like WhatsApp or PayTM)
- **Data storage safety :-** Encrypts files and databases

19. Software Applications and Its Types

→ It is a type of software application that helps in the automation of the task based on the Users Input.

→ It can perform single or multiple tasks at the same period of time.

→ There are the different application which helps us in our daily life to process our instructions based on certain rules and regulations.

→ Application Software helps in providing a graphical user interface to the user to operate the computer for different functionality**.**

→ **Types of Application Software**

- Application software
- System software
- Driver software
- Middleware
- Programming software

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

Feature	System Software	Application Software
Purpose	Runs and manages the hardware and system	Helps users perform specific tasks
User Interaction	Works in the background, not directly used	Directly used by the user
Dependency	Needed to run the computer	Depends on system software to work
Examples	Windows, Linux, Mac OS, Device Drivers	MS Word, Google Chrome, VLC, WhatsApp
Installation Time	Installed when OS is installed	Installed as needed by the user
Included Tools	OS tools, file manager, utilities	Games, browsers, office tools

21. Software Architecture

→ Software Architecture is the structure or blueprint of a software system.

→ It shows how different parts of the software work together**.**

→ This helps the software development team to clearly communicate how the software is going to be built as per the requirements of customers.

→ **Layer in Software Architecture**

1. Presentation layer
2. Application layer
3. Business layer
4. Persistence layer
5. Database layer

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

→ Modularity means breaking down a software system into separate, independent components or modules, each responsible for a specific piece of functionality. Here's why it is significant in software architecture:

✓ **Improved Maintainability**

- Each module handles one functionality.
- Easier to update, debug, or replace without affecting the entire system.

✓ **Code Reusability**

- Modules can be reused in different parts of the same project or in different projects.

✓ **Parallel Development**

- Different teams can work on different modules simultaneously.

✓ **Better Scalability**

- New features can be added by integrating new modules without rewriting the whole system.

✓ **Easy Testing**

- Modules can be tested independently (unit testing).
- Helps find bugs faster and improves software quality.

✓ **Enhances Readability and Organization**

- Code is more organized and easier to understand.
- Useful for on boarding new developers.

✓ **Supports Separation of Concerns**
  - Each module has a clear responsibility.
  - Encourages cleaner design and better abstraction.

✓ **Facilitates Integration and Deployment**
  - Modules can be deployed or updated independently in micro services or plugin-based architectures.

23. Layers in Software Architecture

Layer	Role	Responsibility	Examples / Technologies
1. Presentation Layer	User Interface	Displays data to the user and captures user input	HTML, CSS, JavaScript, React, Angular, WinForms
2. Business Logic Layer (BLL)	Processes application logic	Applies business rules, calculations, validations, and workflows	C#, Java, Python logic, .NET Services
3. Data Access Layer (DAL)	Communicates with the database	Performs CRUD operations and handles connections to the database	ADO.NET, Entity Framework, JDBC, Hibernate
4. Database Layer	Stores data persistently	Manages structured data storage, indexing, and querying	SQL Server, MySQL, Oracle, MongoDB
5. Service Layer / API (Optional)	Connects client and server	Provides API endpoints for communication between frontend and backend	REST API, GraphQL, gRPC, Web API (.NET)

24. Why are layers important in software architecture?

→ **Separation of Concerns**
- Each layer has a specific role (UI, logic, data access).
- Makes code easier to understand, develop, and manage

→ **Improved Maintainability**
- Changes in one layer (e.g., database) don’t affect others (e.g., UI).
- Easier to fix bugs, update features, or improve performance.

→ **Code Reusability**

- Logic and functions in the business or data layer can be reused across different parts of the application or in other projects.

→ **Scalability**

- Layers allow applications to grow smoothly in size and complexity.
- You can scale specific layers (e.g., database or API) independently.

→ **Testability**

- Each layer can be tested in isolation (unit testing).
- Makes automated testing easier and more effective.

→ **Team Collaboration**

- Developers can work on different layers at the same time.
    - UI developers work on the Presentation Layer.
    - Backend developers handle Business Logic and Data Layers.

→ **Better Security**

- Sensitive data access is restricted to specific layers (e.g., only DAL interacts with the database).
- Reduces risk of exposing business logic or data handling details.

→ **Flexibility and Adaptability**

- You can replace or upgrade a layer (like switching from SQL Server to MongoDB) without affecting other layers.

25. Software Environments

→ A software environment refers to the combination of tools, platforms, libraries, frameworks, and configurations used to develop, test, deploy, and run software applications. These environments ensure that software operates correctly and efficiently across different stages of the software development lifecycle.

→ **Type of environments**

- The analysis and design environment
- The development environment
- The common build environment
- The testing environment
- The production environment

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

Reason	Explanation
1. Code Writing and Editing	Provides tools like IDEs (e.g., Visual Studio, Eclipse, VS Code) for writing and managing code.
2. Testing and Debugging	Allows developers to test small sections of code (unit tests) and fix bugs before integration.
3. Safe Experimentation	Developers can try out new features or improvements without affecting the live product.
4. Tool Integration	Supports integration with version control (Git), linters, formatters, and build systems.
5. Consistency and Configuration	Ensures developers work in a consistent environment with the correct versions of tools and libraries.
6. Faster Development Workflow	Speeds up the development process by offering auto-complete, code suggestions, and live previews.
7. Reduces Production Errors	Catching and fixing bugs early in development avoids costly issues in later stages like production.
8. Supports Collaboration	Enables teams to share environments or use containerized setups (like Docker) for consistency.

27. Source Code

→ Source code is the set of human-readable instructions and statements written by a programmer using a programming language like C, Java, Python, or C#.

→ It is the original form of a computer program before it is compiled or interpreted into machine code.

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

Aspect	Source Code	Machine Code
Definition	Human-readable instructions written in a programming language	Binary code (0s and 1s) that a computer's CPU understands
Readability	Easy for humans to read and write	Only readable by machines (not humans)
Languages	High-level (C, C++, Java, Python, etc.)	Low-level binary or hexadecimal (e.g., 10100011)
Execution	Needs to be compiled or interpreted	Directly executed by the CPU
Modifiability	Can be easily modified and maintained	Very difficult to modify without original source code
Purpose	For software development and maintenance	For actual program execution by the hardware
Storage Format	Stored in text files with language-specific extensions (.c, .java, .py)	Stored in executable files (.exe, .bin)

29. Github and Introductions

→ GitHub is a web-based platform used for version control**,** collaboration, and hosting source code using Git**.**

→ It helps developers track changes, work together on projects, and manage software efficiently.

→ **Basic GitHub Workflow**

- Create a repository on GitHub.
- Clone it to your local machine using Git.
- Make changes in your code editor.
- Commit the changes.
- Push the changes back to GitHub.
- Create a pull request (if collaborating).
- Review and merge changes into the main branch.

30. Why is version control important in software development?

Reason	Explanation
1. Tracks Changes Over Time	Every modification is recorded, allowing you to view and revert changes.
2. Collaboration Support	Multiple developers can work on the same codebase without interfering with each other.
3. Rollback Capability	Easily revert to previous versions if bugs or errors are introduced.
4. Experimentation with Branches	Developers can create separate branches to test features without affecting the main code.
5. Prevents Code Loss	Code is safely stored and backed up, avoiding accidental overwrites or deletions.
6. Enhances Code Quality	Encourages peer reviews, testing, and bug tracking through pull requests and issues.
7. Maintains Project History	Provides a complete log of who made changes, when, and why.
8. Facilitates Continuous Integration	Works well with automated build and deployment tools (CI/CD).

31. Student Account in Github

→ A GitHub Student Account gives students free access to premium tools and resources for software development, including private repositories, cloud services, and professional-grade development tools — all for free under the GitHub Student Developer Pack.

→ **Eligibility Requirements**

- You must be currently enrolled in a degree- or diploma-granting course.
- You must have a school-issued email address or upload proof of enrollment (e.g., student ID or transcript).
- You must have a GitHub account.

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

Benefit	Explanation
Hands-on Experience	Helps students learn version control (Git), a key skill in the software industry.
Portfolio Building	Projects on GitHub act as a public portfolio to showcase your work to employers.
Collaboration Skills	Enables team projects, collaboration through branches, pull requests, and reviews.
Track Progress	Version history helps review your coding progress and revert if needed.
Student Developer Pack	Gives access to free premium tools like GitHub Copilot, JetBrains, Canva Pro, etc.
Free Private Repositories	You can host unlimited private projects at no cost.
Professional Readiness	Builds familiarity with industry-standard workflows used in real jobs.
Open-Source Contributions	Participate in real-world open-source projects and get recognized by the community.
Project Hosting	Use GitHub Pages or other integrations to deploy websites or apps for free.
Backup and Security	Your work is safely stored and versioned in the cloud.

33. Types of Software

Type	Purpose	Example
System Software	Manages hardware & system functions	Windows, BIOS
Application Software	End-user tasks	MS Word, Chrome
Programming Software	Code creation and execution	GCC, VS Code
Development Software	Complete development environment	IntelliJ, Android Studio
Embedded Software	Device-specific functions	Smart TV OS, Car system
Enterprise Software	Business and operations	SAP, Salesforce
Entertainment Software	Fun and creativity	Games, Audio/Video Editors

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

Aspect	Open-Source Software	Proprietary Software
Source Code Access	Freely available to anyone	Not available to users; only the developer/vendor has access
Customization	Can be modified and redistributed	Cannot be legally modified or redistributed
Cost	Usually free (though support may cost money)	Often requires purchasing a license or subscription
Community Involvement	Developed collaboratively by a community	Developed and maintained by a specific company
License Type	GNU GPL, MIT, Apache, etc.	EULA (End User License Agreement)
Updates	Frequent and community-driven	Released by the vendor, may be limited to certain versions/users
Support	Community forums, some offer paid support	Official support from the vendor
Security	Transparent; anyone can inspect for vulnerabilities	Hidden; trust depends on the vendor
Examples	Linux, Firefox, LibreOffice, VLC	Windows, Microsoft Office, Adobe Photoshop, macOS

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

→ **Distributed Version Control**	Every team member has the full project history locally, enabling work offline and easy syncing.
→ **Branching & Merging**	Developers can create separate branches to work on features/bugs independently without affecting the main codebase.
→ **Concurrent Workflows**	Multiple people can work on different parts of the code simultaneously, reducing bottlenecks.
→ **Change Tracking**	Git records who changed what and when, helping with accountability and understanding the evolution of code.
→ **Conflict Detection & Resolution**	Git identifies merge conflicts early, making it easier to resolve overlapping changes collaboratively.
→ **Pull Requests (via GitHub)**	Enables code review and discussion before integrating changes, improving code quality and knowledge sharing.
→ **Revert and History**	If issues arise, teams can easily roll back to previous stable versions without losing work.
→ **Blame Feature**	Shows who last modified each line, useful for debugging and understanding code decisions.
→ **Integration with CI/CD**	Automated testing and deployment workflows trigger on code changes, ensuring smooth collaboration.

36. Application Software

→ Application Software is a type of computer program designed to help users perform specific tasks.

→ Unlike system software (which runs the computer itself), application software focuses on helping the user accomplish useful tasks such as writing documents, browsing the web, or managing data.

→ **Features of Application Software**

- User-oriented
- Easy-to-use interfaces
- Performs specific functions
- Requires a system platform (like an OS) to run

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

→ Application software plays a crucial role in helping businesses operate efficiently**,** improve productivity, and stay competitive.

→ It automates tasks, manages data, enhances communication, and supports decision-making processes.

→ **Data Management**	Store, retrieve, and analyze large volumes of data using tools like databases.
→ **Accounting & Finance**	Manage invoices, payroll, tax, and budgeting using software like Tally or QuickBooks.
→ **Productivity & Office Work**	Create documents, spreadsheets, and presentations with MS Office, Google Workspace.
→ **Inventory & Supply Chain**	Track stock, sales, and suppliers using ERP and inventory management tools.
→ **Customer Relationship Management (CRM)**	Manage customer data, sales, and support with tools like Salesforce or Zoho.
→ **Communication & Collaboration**	Use email, chat, and video conferencing software like Outlook, Teams, or Zoom.
→ **Security & Access Control**	Protect sensitive data using software for encryption, access control, etc.
→ **Project Management**	Plan, assign, and track tasks with tools like Trello, Asana, or Jira.
→ **E-Commerce & Marketing**	Run online stores, automate marketing campaigns using Shopify, Mailchimp, etc.

38. Software Development Process

→ The Software Development Process is a structured sequence of stages that guide the planning, creation, testing, and deployment of a software product.

→ It ensures that software meets user requirements, is reliable, and is delivered on time.

→ **Example**

- **Requirement Analysis**: Understand features like login, view attendance, mark attendance.
- **Design**: Create database structure and UI layout.
- **Coding**: Build the portal using HTML, CSS, PHP, and MySQL.
- **Testing**: Ensure login, data entry, and reports work properly.
- **Deployment**: Launch the system for teachers and students.
- **Maintenance**: Add features like report cards or SMS alerts later.

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

→ **Requirement Analysis**

- **Goal:** Understand what the software should do.
- **Activities:** Meetings with clients, gathering user needs, creating requirement documents.
- **Output:** Software Requirement Specification (SRS) document.

→ System Design

- **Goal:** Plan how the system will meet the requirements.
- **Activities:** Create system architecture, data models, flowcharts, UI designs.
- **Output:** Design Documents (high-level and low-level designs).

→ **Implementation / Coding**

- **Goal:** Convert the design into actual code.
- **Activities:** Developers write code using chosen programming languages and tools.
- **Output:** Working source code for each module/component.

→ **Testing**

- **Goal:** Verify the software is free of bugs and meets requirements.
- **Activities:** Perform unit testing, integration testing, system testing, acceptance testing.
- **Output:** Test reports, bug logs, and a verified system ready for deployment.

→ **Deployment**

- **Goal:** Release the software to users.
- **Activities:** Host the application on a server or deliver it to clients.
- **Output: Live** application or installed software on user systems.

→ **Maintenance**

- **Goal:** Fix issues and enhance the software after it's live.
- Activities**:** Apply updates, patches, and add new features based on feedback.
- **Output:** Improved and up-to-date software system.

40. Software Requirement

→ Software requirements are detailed descriptions of a system's functions, features, and constraints.

→ They define what the software should do and how it should perform, ensuring that the development team and stakeholders are aligned.

→ **Why Are Requirements Important**

- Provide a **clear understanding** of what to build.
- Help in **estimating cost and time**.
- Serve as the **basis for design, development, and testing**.
- Reduce the risk of **miscommunication or feature creep**.
- Are essential for **client satisfaction**.

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

→ The Requirement Analysis phase is one of the most important steps in the Software Development Life Cycle (SDLC) because it sets the foundation for the entire project. Any mistake or misunderstanding at this stage can lead to project delays, cost overruns, or even failure.

Reason	Explanation
→ **Clear Understanding of Needs**	Ensures that developers, clients, and users all have the same expectations.
→ **Foundation for Design & Development**	All future phases like design, coding, and testing depend on accurate requirements.
→ **Cost and Time Estimation**	Helps in planning resources, timelines, and budgeting more accurately.
→ **Avoids Rework**	Reduces chances of building the wrong features, which leads to costly rework.
→ **Improves Client Satisfaction**	Delivering what was expected ensures higher satisfaction and better relations.
→ **Supports Effective Testing**	Testing teams use requirements to verify whether the product meets expectations.

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

Role	Explanation
Understanding User Needs	Collects and analyzes what the users or clients expect from the software.
Defining Requirements	Clearly specifies functional and non-functional requirements.
Feasibility Study	Checks if the software is technically, financially, and legally viable.
Lays the Foundation for Design	Provides a base for creating system architecture and interface design.
Problem Identification	Detects inconsistencies, conflicts, or gaps in user needs early on.
Creates Documentation (SRS)	Results in a Software Requirements Specification (SRS) document used throughout the project.
Supports Testing & Validation	Requirements become a baseline for future testing and validation of the system.

43. What are the key elements of system design?

→ **Architecture Design**

- Defines the overall structure and interaction between components (e.g., client-server, MVC).

→ **Data Design**

- Specifies data structures, databases, data flow, and storage strategies.

→ User Interface Design

- Designs how users will interact with the system (UI layout, navigation, inputs, outputs).

→ **Module Design**

- Breaks the system into smaller modules or components, defining their responsibilities.

→ **Interface Design**

- Describes how different modules or systems will communicate (APIs, data formats, protocols)
.

→ **Security Design**

- Ensures user authentication, data encryption, access control, and secure communication.

→ **Hardware & Network Design**

- Identifies hardware requirements and networking needs (e.g., servers, bandwidth, devices)
.

→ **Performance & Scalability**

- Considers system speed, load handling, and future scalability options.

→ **Error Handling & Fault Tolerance**

- Plans how the system will handle errors and recover from failures.

44. Why is software testing important?

→ **Ensures Software Quality**

- Confirms the software meets specified requirements and works as intended.

→ **Identifies and Fixes Bugs Early**

- Detects errors before the software is released, saving time and cost.

→ **Improves User Experience**
- Helps deliver a smooth, bug-free, and user-friendly interface.

→ **Validates Functionality**
- Checks if all features perform correctly under different scenarios.

→ **Enhances Performance**
- Tests how the software behaves under load, stress, and various conditions.

→ **Ensures Security**
- Identifies vulnerabilities and prevents data breaches or hacking risks.

45. What types of software maintenance are there?

→ **Corrective Maintenance**
- **Purpose:** Fixes bugs, errors, or faults discovered after the software is in use.
- **Example:** Repairing a login failure or fixing a crash issue.

→ **Adaptive Maintenance**
- **Purpose:** Modifies the software to keep it compatible with a changing environment (e.g., new operating systems, hardware, or platforms).
- **Example:** Updating software to run on a new version of Windows or Android.

→ **Perfective Maintenance**
- **Purpose:** Enhances existing features or adds new ones based on user feedback or changing requirements.
- **Example:** Improving the UI layout or adding a search function.

→ **Preventive Maintenance**
- **Purpose:** Makes changes to prevent future problems or improve maintainability.
- **Example:** Refactoring code, updating documentation, or optimizing performance.

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

Feature	Web Applications	Desktop Applications
Accessibility	Browser-based, global	Local device only
Installation	Not required	Required
Internet Dependence	Usually required	Not always required
Platform Compatibility	Cross-platform	Platform-specific
Update Method	Server-side (automatic)	Manual or user-initiated

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

→ **Accessibility Anywhere**

- Can be accessed from any device with a browser and internet connection — no need to be on a specific computer.

→ **No Installation Required**

- Users don't need to install or update anything; the latest version is always available online.

→ **Cross-Platform Compatibility**

- Works on any operating system (Windows, macOS, Linux) as long as a browser is available.

→ **Easy Maintenance and Updates**

- Developers can update the application on the server, and all users get the update instantly.

→ **Cost-Efficient Deployment**

- No need to distribute installation files or handle OS-specific compatibility — saves time and money.

→ **Centralized Data Storage**

- Data is stored on the cloud or central servers, making it easier to back up, secure, and share.

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

→ **UI (User Interface) Design**

- **Role:** Focuses on the look and layout of the application — the visual elements like buttons, icons, colors, typography, and screen structure.
- **Purpose:** To create an interface that is aesthetically pleasing, consistent, and easy to navigate.
- **Impact:** Good UI increases usability and keeps users engaged by providing a smooth and intuitive experience.

→ **UX (User Experience) Design**

- **Role:** Focuses on the overall feel of the application and how efficiently users can complete their goals.
- **Purpose:** To ensure the app is useful, easy to use, and enjoyable.
- **Impact:** Strong UX design leads to higher user satisfaction, reduced frustration, and better retention.

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

Feature	Native Apps	Hybrid Apps
Platform Support	One platform per app (iOS/Android)	One app for multiple platforms
Development Time	Longer (separate codebases)	Faster (single codebase)
Performance	High	Moderate
Access to Device Features	Full	Limited/depends on framework
Cost	Higher	Lower
User Experience (UX)	Superior	Moderate to good

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

→ **Visual Clarity of System Processes :-** DFDs provide a clear, graphical view of the system's processes, data inputs, outputs, storage, and flows — making complex systems easier to understand.

→ **Improves Communication** **:-** Helps bridge the gap between technical and non-technical stakeholders by presenting information in a simple, easy-to-understand format.

→ **Identifies Data Sources and Destinations :-** Clearly shows where data comes from, where it goes, and how it is processed, ensuring nothing important is missed.

→ **Supports Requirement Gathering :-** DFDs help in eliciting and validating user requirements by showing what the system should do and how it should handle data.

→ **Aids in System Design :-** Acts as a blueprint for designing databases, modules, and interfaces by laying out data flow and system logic.

→ **Modular and Scalable Representation :-** DFDs can be decomposed into levels (Level 0, Level 1, etc.) to represent the system at different depths, allowing focus on both the big picture and the details.

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

→ **Desktop Applications**

- **Pros :**

✓ **Works Offline**
  - Does not require an internet connection to function.

✓ **High Performance**
  - Can access system resources more efficiently and run faster, especially for heavy processing tasks (e.g., video editing, 3D rendering).

✓ **Better Integration with Hardware**
  - Can directly interact with printers, scanners, and other local devices.

✓ **Rich User Interface**
  - Can offer complex UI/UX without browser limitations.

- **Cons:**

✓ **Platform-Dependent**
  - Must be developed separately for Windows, macOS, Linux, etc.

✓ **Requires Installation & Updates**
  - Needs to be installed and manually updated by users.

✓ **Limited Accessibility**
  - Only accessible on the device where it's installed.

→ **Web Applications**

- **Pros :**

✓ **Accessible Anywhere**
  - Can be used from any device with a browser and internet connection.

✓ **Cross-Platform Compatibility**
  - Works on any operating system without modification.

✓ **No Installation Needed**
  - Runs directly in the browser, saving storage space and time.

✓ **Easier to Update**
  - Updates are applied on the server side — no need for users to install them.

- **Cons:**

✓ **Requires Internet Connection**
  - Most web apps don’t work offline or have limited functionality offline.

✓ **Performance Limitations**
  - Slower for heavy computations or large data processing.

✓ **Security Risks**
  - More exposed to cyber threats if not properly secured.

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

→ **Simplify Complex Logic**
- Break down complex processes into easily understandable steps, using symbols to represent decisions, actions, and flows.

→ **Help in Planning Algorithms**
- Assist programmers in designing algorithms before writing code, reducing logical errors and improving efficiency.

→ **Document the System Clearly**
- Provide a clear visual documentation of a system’s workflow for current and future reference, aiding in maintenance.

→ **Improve Code Structure**
- Guide developers in writing well-structured, modular, and readable code by having a clear plan to follow.

→ **Useful for Teaching and Learning**
- Help beginners understand programming concepts and logic flow without diving directly into complex code syntax.