**Module 01: Overview of IT Industry**

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

A program is a set of instructions that a computer follows to perform a specific task. It's essentially a set of commands written in a programming language.

Program takes input and process data and then produce output that’s how program functions.

2.what are the key steps involved in programing process?

1.Requirement : Define what to build

2.Planning : Determine how to build and manage

3.Design : Architect how it function and look

4.Devlopement : Write, integrate and document the code

5.Testing : Ensure quality and correctness

6.Deployment : Release to users

7.Maintanance : Keep software secure ,functional and evolving

3.what is the main differences between high-level and low-level programing languages?

**1. Abstraction Level**

High-level languages provide strong abstraction, letting you write code using human-friendly syntax and logic without worrying about how the hardware works.

Low-level languages are close to the hardware, offering minimal abstraction—you're dealing directly with memory, CPU registers, and machine instructions.

**2. Readability & Ease of Use**

**High-level**: Designed to be easy to learn and use, with clean syntax, automatic memory management, and robust debugging tools.

**Low-level**: Harder to read and write, requiring intimate knowledge of computer architecture and manual control of resources.

**3. Performance, Efficiency & Control**

**Low-level**: Offers maximumperformance and the tightest control over resources—code runs faster and uses less memory.

**High-level**: Adds overhead due to abstraction layers and interpreters/compilers, which can reduce performance.

**4. Portability**

**High-level**: Code is generally portable across multiple platforms with little or no changes.

**Low-level**: Tied closely to specific hardware or processor architectures—**not** portable without rewriting.

**5. Use Cases**

**High-level**: Ideal for **web** applications**,** mobileapps**,** scripting**,** dataprocessing—where productivity and readability matter.

**Low-level**: Used for system programming, embedded systems, device drivers, and performance-critical tasks.

**6. Translation Process**

**High-level**: Requires a compiler, interpreter, or byte-code engine to translate human-readable code into executable machine code.

**Low-level**:

Assembly → via assembler → converted to machine code

Machine code → directly executed by the CPU

Programmers often use them together—writing performance-sensitive parts in low-level code while building the rest with high-level languages for ease and maintainability.

4.describe the roles of client and server in web communication.

In web communication, the client and server play distinct, yet interdependent, roles. The client, typically a web browser, initiates requests for resources (like webpages, images, or data) from the server. The server, on the other hand, receives these requests, processes them, and then sends back the requested information or performs the requested action. This request-response cycle is the foundation of web interaction.

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

The TCP/IP model is a conceptual framework that structures how data is transmitted over networks. It divides the communication process into four layers: Application, Transport, Internet, and Network Access. Each layer handles specific tasks, ensuring reliable and efficient data exchange between devices.

**1. Application Layer:**

This layer is closest to the user and provides network services to applications like email (SMTP), web browsing (HTTP), and file transfer (FTP). It handles tasks like user interface, data formatting, and encoding.

**2. Transport Layer:**

This layer ensures reliable and ordered delivery of data between applications on different devices. It uses protocols like TCP (Transmission Control Protocol) for connection-oriented, reliable communication, and UDP (User Datagram Protocol) for connectionless, faster but less reliable communication.

**3. Internet Layer:**

This layer is responsible for addressing and routing packets of data across networks. The Internet Protocol (IP) is the core protocol here, assigning logical addresses (IP addresses) to devices and determining the best path for data to travel.

**4. Network Access Layer:**

This layer manages the physical transmission of data over the network medium (e.g., Ethernet, Wi-Fi). It handles tasks like framing data, addressing at the link level (e.g., MAC addresses), and controlling access to the physical medium.

**6.Explain Client Server Communication.**

Client-server communication is a fundamental model where clients request services or resources from servers, and servers provide those services or resources back to the clients. This interaction forms the backbone of how much of the internet and many software applications work today.

Here's a quick breakdown:

* Client: A device or application (like your web browser or phone app) that initiates a request for something.
* Server: A computer or system that listens for client requests, processes them, and then sends a response.
* Network: The connection that allows the client and server to exchange requests and responses.
* Protocols: Standardized rules (like HTTP for web browsing) that ensure both sides understand how to communicate.

For example, when you type a website address in your browser (the client), it sends a request to the web server that hosts the site. The server then sends back the webpage files (like HTML and images), which your browser displays.

This model allows for centralized data management, scalability, and enhanced security, but it can also face challenges like server overload or vulnerability to attacks if not properly managed.

**7.How does broadband differ from Fiber-optic internet?**

Broadband is a type of internet service that provides its customers with high-speed internet access. Broadband is capable of sending multiple signals over a wide bandwidth. Broadband uses optical fibre, coaxial cable, twisted pair, and radio as its medium. It can be used by organisations as well as by individuals. The fibre cables used in fibre optic are used to transmit data through them. Superfast broadband is quicker than standard broadband, and the connection is also more reliable.

**8. What are the differences between HTTP and HTTPS protocols?**

Data in HTTP transmitted without encryption, meaning anyone intercepting the traffic can easily read the information. While HTTPS Uses TLS/SSL (TLS=Transport Layer Security, SSL=Secure sockets Layer) to encrypt data, ensuring confidentiality and integrity.

HTTP is Susceptible to eavesdropping, man-in-the-middle attacks, and data breaches. While HTTPS Protects sensitive information like passwords, credit card details, and personal data

HTTP Typically uses port 80 as default. While HTTPS uses port 443 as it’s default.

HTTP Operates at the application layer of the network model. While HTTPS Operates at the transport layer, providing a secure channel for data exchange.

HTTPS Requires an SSL/TLS certificate to establish trust and enable encryption and also Search engines like Google Favors HTTPS sites, often ranking them higher.

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

1. Protecting sensitive data
2. Securing data in different states
3. Ensuring data integrity and authenticity
4. Compliance with data protection regulations
5. Mitigating the impact of data breaches
6. Strengthening authentication and access control

**10.What is the difference between system software and application software?**

System software and application software are two primary categories of computer programs, both essential but with distinct purposes and functions.

System Software: This acts as the foundation of a computer system, managing and controlling the hardware and providing a platform for application software to run. It's crucial for the basic operation of the computer and runs primarily in the background.

Application Software: These programs are designed for end-users to perform specific tasks, ranging from creating documents to playing games. They depend on system software to function and interact directly with users.

**11. What is the significance of modularity in software architecture?**

Modularity is a core concept in software architecture that involves breaking down a large and complex software system into smaller, independent, and interchangeable parts called modules. Each module is designed to handle a specific function or feature and interacts with other modules through well-defined interfaces.

This approach offers numerous benefits that are crucial for building robust, scalable, and maintainable software systems:

**12. Why are layers important in software architecture?**

Layered architecture is a powerful and widely adopted approach to structuring software systems. It involves organizing the application into distinct, horizontal layers, each with specific responsibilities and functionalities.

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

A well-configured development environment is paramount in software production, serving as the essential workspace where developers design, build, test, and maintain applications and websites. It provides a safe, enclosed area for developers to experiment with new code, features, and settings without impacting the live, working version, also known as the production environment.

**14.What is the difference between source code and machine code?**

Source code is the human-readable instructions written by programmers in a specific programming language (like Python, Java, C++, etc.). It's essentially the blueprint of a program, outlining how it should function. Source code is easy to read, understand, and modify for humans.

Machine code, on the other hand, is the raw, binary instructions (sequences of 0s and 1s) that a computer's Central Processing Unit (CPU) can directly understand and execute. It's the lowest level of programming language and is hardware-specific, meaning it differs depending on the processor type (e.g., Intel, AMD, ARM). Machine code is not human-readable or easily understandable.

**15. Why is version control important in software development?**

 version control is not just a tool; it's a fundamental part of a modern software development workflow. According to GeeksforGeeks it enables effective teamwork, protects against data loss, promotes code quality, and helps ensure the timely delivery of high-quality software.

**16. What are the benefits of using GitHub for students?**

1.Learn industry-standard skills

2.Collaborate and learn

3.Build a portfolio

4.Free tools and resources

5.Prepare for careers

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

Open-Source Software (OSS):

Source code is public: Anyone can view, modify, and distribute it.

Focus on freedom: Users have more freedom to use, customize, and share the software.

Community-driven: Often developed and supported by a global community.

Often free (but not always): The software itself might be free, but support or customization may cost money.

Proprietary Software (PS):

Source code is private: Owned by a company or individual, kept secret.

Focus on control: Use is restricted by licenses, often requiring payment.

Vendor-driven: Developed and supported by the owning company.

Usually paid: Requires buying licenses or subscriptions to use.

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

Git is a distributed version control system that significantly enhances collaboration within software development teams through its core features. These capabilities streamline teamwork, improve code quality, and boost overall productivity.

Key ways Git fosters collaboration include:

* 1. Parallel Development and Isolation: Git allows developers to create independent "branches" to work on features or bug fixes in isolation from the main codebase. This eliminates interference between team members, enabling parallel development efforts and reducing conflicts.
* 2. Seamless Merging and Conflict Resolution: Once work on a branch is complete, Git's merging features integrate the changes back into the main codebase. Git highlights areas where conflicts arise (when the same part of a file is modified differently in separate branches), providing tools and clear markers to help developers manually resolve these conflicts efficiently.
* 3. Pull Requests and Code Review: Platforms integrating Git, like GitHub and GitLab, utilize pull requests (or merge requests). This mechanism allows developers to propose changes, enabling other team members to review, discuss, and suggest improvements before the code is merged. This peer review process significantly enhances code quality and fosters knowledge sharing.
* 4. Distributed Nature and Offline Work: Git's distributed model gives every developer a full, local copy of the repository's history. This allows team members to work offline and commit changes independently, synchronizing them with the central repository when connected. This flexibility is particularly beneficial for remote or distributed teams.
* 5. Version History and Accountability: Git meticulously tracks every change, including who made it, when, and the commit message explaining the change. This detailed history provides traceability, aids debugging, and promotes accountability within the team.

**19.What is the role of application software in businesses?**

Application software is essential to modern businesses, supporting various operations, enhancing efficiency, and fostering growth. These specialized programs are designed to meet specific organizational needs, unlike system software, which manages computer hardware.

1.Automation

2.Data Management and Insights

3.Improved Communication and Collaboration

4.Enhanced Productivity

5.Customer Relationship Management

6.Financial Management

7.Resource Management

8.Marketing and Sales

9.Customization and Flexibility

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

1.Planning and Requirements Analysis

2.Design

3.Development/Coding

4.Testing

5.Deployment

6.Maintenance

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

The requirement analysis phase is arguably the most crucial stage in the Software Development Life Cycle (SDLC) because it lays the foundation for the entire project. This phase involves understanding, documenting, and validating the needs and expectations of all stakeholders (clients, users, and the development team) before any coding or design takes place.

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

software analysis (often synonymous with "requirements analysis" or the analysis phase of the Software Development Life Cycle (SDLC)) is a crucial stage that bridges the gap between initial ideas and a working software product. It's about thoroughly understanding what needs to be built before building it.

**23.What are the key elements of system design?**

1. Architecture: The overall structure and organization, like choosing between a single "monolith" or separate "microservices".
2. Components: Breaking the system into smaller, reusable parts (modules or services) and defining how they interact (interfaces).
3. Data Design: Deciding how data is stored, organized, and accessed (e.g., choosing database types, schema design).
4. Scalability: Ensuring the system can handle growing user numbers and data volumes without sacrificing performance, often using techniques like horizontal scaling (adding more servers) and load balancing.
5. Performance: Making the system fast and efficient, which involves optimizing code, utilizing caching, and efficient resource management.
6. Reliability: Designing the system to work consistently and recover gracefully from failures, using strategies like redundancy, replication, and failover.
7. Security: Protecting the system and data from unauthorized access, including authentication, authorization, and encryption.
8. Maintainability and Extensibility: Designing the system to be easy to update, fix bugs, and add new features in the future.

**24. Why is software testing important?**

Software testing is crucial because it ensures the delivery of high-quality, reliable, and secure software that meets user expectations and business requirements. By systematically identifying and fixing bugs, defects, and vulnerabilities early in the development lifecycle, testing prevents costly rework and delays, enhances user satisfaction, protects brand reputation, and ensures compliance with industry standards and regulations. This proactive approach mitigates risks, improves software performance, and ultimately contributes to the long-term success of any software project.

**25. What types of software maintenance are there?**

There are 4 types of software maintenance:

1.corrective maintenance

2.adaptive maintenance

3.perfective maintenance

4.preventive maintenance

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

The fundamental differences between web and desktop applications stem from their architectural design and operational models. Web applications run within a web browser and typically require an active internet connection to function, as their code and data reside on remote servers. This architecture makes them platform-independent, accessible from any device with a compatible browser, and allows for centralized updates managed by the service provider, ensuring all users are always on the latest version. In contrast, desktop applications are installed directly onto a specific operating system, often enabling them to function offline and leverage the local device's full processing power, leading to potentially higher performance for demanding tasks. However, this also makes them platform-dependent, often requiring separate versions for different operating systems, and necessitates users to manually download and install updates. The choice between the two often depends on factors like required performance, offline access needs, desired platform compatibility, and update management preferences.

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

Web applications offer significant advantages over desktop applications, primarily due to their accessibility and ease of management. They can be accessed from any device with an internet connection and a web browser, making them highly flexible for users working from various locations or devices. This also translates to simpler maintenance and updates, as changes are deployed centrally on the server, ensuring all users automatically benefit from the latest features and security patches without manual intervention. Furthermore, web applications excel in collaboration, allowing multiple users to work on projects simultaneously and share data seamlessly through cloud storage. Their cross-platform compatibility eliminates the need for developing separate versions for different operating systems, potentially leading to faster development times and reduced costs.

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

In the competitive landscape of application development, User Interface (UI) and User Experience (UX) design play a pivotal role, determining not just an app's aesthetics but its overall success and longevity. They are two distinct yet intertwined disciplines:

* UI (User Interface) Design: Focuses on the visual and interactive elements of an application. This includes everything the user sees and interacts with, such as buttons, icons, layout, typography, colour schemes, and visual hierarchy. UI design aims to create a visually appealing, intuitive, and consistent interface that makes navigation and interaction effortless.
* UX (User Experience) Design: Encompasses the entire experience a user has while interacting with the application. It's about understanding user behaviour, needs, and motivations to create a seamless, satisfying, and frustration-free experience. UX design considers the overall user journey, from initial interaction to task completion, ensuring the application is functional, usable, and enjoyable.

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

Native apps are built specifically for one platform (like iOS or Android) using platform-specific languages (e.g., Swift for iOS, Kotlin for Android). They offer the best performance, user experience, and full access to device features but are more costly and time-consuming to develop since separate codebases are needed for each platform.

Hybrid apps are built using web technologies (like HTML, CSS, JavaScript) and wrapped in a native container, allowing them to run on multiple platforms from a single codebase. They are generally faster to develop and more cost-effective but may have slightly lower performance, less access to specific device features, and potentially a less refined user experience compared to native apps.

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