

Assignment :- theory

Module 1 – Overview of IT Industry :-

1. What is a Program and How Does it Function?

➡ A program is a set of instructions written in a programming language to perform a specific task. The program runs when it's either compiled or interpreted into machine code that the computer understands.

2. What is Programming?

➡ **Programming** is the process of writing, testing, and maintaining code that tells a computer what to do. It involves solving problems using logic, algorithms, and programming languages.

3. Key Steps in the Programming Process :

- ➡
1. Problem Analysis
 2. Designing Algorithms (flowcharts/pseudocode)
 3. Writing Code (implementation)
 4. Testing and Debugging
 5. Documentation
 6. Maintenance and Updates

4. Types of Programming Languages:

- ➡
- **High-Level:** Python, Java (human-readable)
 - **Low-Level:** Assembly, Machine code (hardware-oriented)
 - **Procedural:** C, Pascal
 - **Object-Oriented:** Java, C++
 - **Functional:** Haskell, Lisp

→	Feature	High-Level	Low-Level
	Readability	Easy for humans	Hard to read
	Portability	High	Low
	Speed	Slower	Faster
	Hardware Access	Abstracted	Direct access

5. High-Level vs Low-Level Languages:

6. Roles of Client and Server in Web Communication:

→ **Client:** Requests data (e.g., browser)

Server: Responds with requested resources (e.g., website, file)

7. TCP/IP Model and Its Layers:

- 1. **Application Layer** – HTTP, FTP
 - 2. **Transport Layer** – TCP/UDP
 - 3. **Internet Layer** – IP
 - 4. **Network Access Layer** – Ethernet/Wi-Fi
- Each layer plays a role in packaging and transmitting data reliably.

8. Client-Server Communication:

→ The **client** initiates a request; the **server** processes it and sends back a response. Common in web apps, databases, and email services.

9. Types of Internet Connections:

→ **Broadband:** Uses telephone or cable lines; moderate speed.

Fiber-Optic: Uses light signals; extremely fast and stable.

10. HTTP vs HTTPS:

➡ **HTTP:** Unsecured data transfer

HTTPS: Uses SSL/TLS for encryption, ensuring data privacy and integrity.

11. Role of Encryption in Application Security:

➡ Encryption turns readable data into ciphertext, protecting it from unauthorized access and ensuring secure communication (e.g., banking apps).

12. System vs Application Software:

➡ **System Software:** Manages hardware (e.g., OS, drivers)

Application Software: Performs tasks for users (e.g., Word, browsers)

13. Modularity in Software Architecture:

➡ Breaking a system into smaller, independent **modules** improves maintainability, testing, and scalability.

14. Importance of Layers in Software Architecture:

➡ **Presentation Layer** – UI/UX

Business Logic Layer – Application rules

Data Access Layer – Interacts with database

This separation enhances clarity and maintainability.

15. Importance of a Development Environment:

➔ A good development environment provides tools (e.g., editors, debuggers, compilers) that make coding, testing, and deployment efficient.

16. Source Code vs Machine Code:

➔ **Source Code:** Human-readable instructions

Machine Code: Binary instructions executed by the CPU

17. Why Version Control is Important:

➔ Tracks code changes, supports teamwork, and allows rollback in case of mistakes. Git is a widely used system.

18. Open-Source vs Proprietary Software:

➔ **Open-Source:** Free to view, use, and modify (e.g., Linux)

Proprietary: Owned and licensed (e.g., Windows)

19. How Git Improves Collaboration:

➔ Allows multiple developers to contribute to the same codebase, manage branches, and merge changes without losing work.

20. Role of Application Software in Businesses:

➔ Helps manage business operations like accounting, CRM, HR, etc., increasing efficiency and decision-making.

21. Main Stages of Software Development:

- ➔ 1. Requirement Gathering
- 2. System Design
- 3. Implementation (Coding)
- 4. Testing
- 5. Deployment
- 6. Maintenance

22. Importance of Requirement Analysis:

- ➔ Defines what the software should do. It helps avoid costly changes later and sets clear goals for development.

23. Role of Software Analysis:

- ➔ Involves examining requirements and determining the best approach to meet them through design and implementation.

24. Key Elements of System Design:

- ➔
 - Software Architecture
 - Database Design
 - Data Flow Diagrams
 - User Interface Layout

25. Importance of Software Testing:

- ➔ Ensures the software works correctly, is secure, and performs well before being released to users.

26. Types of Software Maintenance:

- ➔
- **Corrective:** Fix bugs
 - **Adaptive:** Adjust to changes (e.g., OS update)
 - **Perfective:** Improve performance or features
 - **Preventive:** Avoid future problems

27. Web vs Desktop Applications:



Feature	Web App	Desktop App
Access	Browser	Installed locally
Updates	Server-side	Manual
Performance	Depends on internet	Generally faster

28. Advantages of Web Applications:

- ➔
- No installation needed
 - Accessible anywhere
 - Easy to update and maintain

29. Role of UI/UX Design:

➔ Good design makes applications easy to use and visually appealing, improving user satisfaction and engagement.

30. Native vs Hybrid Mobile Apps:

➔ **Native:** Platform-specific, better performance

Hybrid: Single codebase for multiple platforms, cost-effective

31. Significance of DFDs:

➔ **Data Flow Diagrams** show how data moves through a system. Useful in system analysis and design.

32. How Flowcharts Help:

➔ Flowcharts visually represent algorithms and logic flow, making it easier to plan, debug, and explain code.

Let me know if you'd like:

- Diagrams added
- Translations
- A summarized version for revision

