**Module 1 – Overview of IT Industry**

**1. What is a Program?**

**THEORY EXERCISE:** A program is a sequence of instructions written in a programming language to perform a specific task. It instructs a computer to execute operations like calculations, data processing, or input/output handling. Programs function by converting human-readable code into machine code, allowing the hardware to carry out the desired tasks.

**LAB EXERCISE:** **Language 1: Python**

print("Hello, World!")

**Language 2: Java**

public class HelloWorld {

public static void main(String[] args) {

System.out.println("Hello, World!");

}

}

**Comparison:**

* Python is simpler and uses dynamic typing.
* Java requires class structure and static typing.

**2. What is Programming?**

**THEORY EXERCISE:** Programming is the process of designing and building an executable set of instructions to accomplish a specific task or solve a problem. It involves writing code, debugging, testing, and maintaining it.

**3. Key Steps in Programming Process**

**THEORY EXERCISE:**

1. Requirement Analysis
2. Designing Algorithm
3. Coding
4. Testing
5. Debugging
6. Deployment
7. Maintenance

**4. Types of Programming Languages**

**THEORY EXERCISE:**

* **High-Level Languages:** Easier to read and understand (e.g., Python, Java).
* **Low-Level Languages:** Closer to machine code, harder to understand (e.g., Assembly, Machine Code).

**5. World Wide Web & How Internet Works**

**THEORY EXERCISE:**

* **Client:** Sends requests to the server.
* **Server:** Processes requests and sends responses.

**LAB EXERCISE:** Diagram includes: Client -> Router -> ISP -> DNS -> Web Server -> Database Server -> Response

**6. Network Layers on Client and Server**

**THEORY EXERCISE:** TCP/IP Model Layers:

1. Application
2. Transport
3. Internet
4. Network Access

**LAB EXERCISE:** Simple Python HTTP Client-Server using sockets

**7. Client and Servers**

**THEORY EXERCISE:** Client-server communication involves clients sending service requests to a server, which processes and returns responses.

**8. Types of Internet Connections**

**THEORY EXERCISE:**

* **Broadband:** High-speed, widely available
* **Fiber:** Fastest, reliable, costly

**LAB EXERCISE:** Create a table listing broadband, fiber, satellite with pros and cons.

**9. Protocols**

**THEORY EXERCISE:**

* **HTTP:** Not encrypted
* **HTTPS:** Encrypted using SSL/TLS

**LAB EXERCISE:** Using curl:

curl http://example.com

curl -u username:password ftp://example.com

**10. Application Security**

**THEORY EXERCISE:** Encryption ensures data is securely transferred and accessed only by authorized users.

**LAB EXERCISE:** Common vulnerabilities:

* SQL Injection
* Cross-Site Scripting (XSS)
* Insecure Authentication Solutions: Input validation, HTTPS, multi-factor authentication

**11. Software Applications and Its Types**

**THEORY EXERCISE:**

* **System Software:** Runs hardware (e.g., OS)
* **Application Software:** User-specific tasks (e.g., Word Processor)

**LAB EXERCISE:** Classify: Chrome (Application), Windows (System), Antivirus (Utility), etc.

**12. Software Architecture**

**THEORY EXERCISE:** Modularity allows separation of concerns and easier maintenance.

**LAB EXERCISE:** Design a 3-tier: Presentation -> Logic -> Data Layer

**13. Layers in Software Architecture**

**THEORY EXERCISE:** Layers improve code reusability, scalability, and maintainability.

**LAB EXERCISE:** Case study with explanation of each layer’s role in an e-commerce app

**14. Software Environments**

**THEORY EXERCISE:** Development environments are isolated setups for writing and testing code before production.

**LAB EXERCISE:** Install VirtualBox and set up Ubuntu VM for development

**15. Source Code**

**THEORY EXERCISE:**

* **Source Code:** Human-readable
* **Machine Code:** Binary instructions

**LAB EXERCISE:** Write a C program, upload to GitHub

**16. Github and Introductions**

**THEORY EXERCISE:** Version control tracks changes, supports collaboration and backup.

**LAB EXERCISE:** Create repo, commit code:

git init

git add .

git commit -m "First commit"

git push origin main

**17. Student Account in Github**

**THEORY EXERCISE:** Benefits: Collaboration, portfolio building, learning real-world tools

**LAB EXERCISE:** Sign up, collaborate on a shared GitHub project

**18. Types of Software**

**THEORY EXERCISE:**

* **Open-source:** Free, modifiable (e.g., Linux)
* **Proprietary:** Licensed, not open (e.g., Windows)

**LAB EXERCISE:** Categorize: Ubuntu (System, Open), MS Word (Application, Proprietary)

**19. GIT and GITHUB Training**

**THEORY EXERCISE:** GIT supports collaboration through branching, merging, tracking code history.

**LAB EXERCISE:** Clone, branch, merge repo

**20. Application Software**

**THEORY EXERCISE:** Helps in productivity (e.g., Word processors, Spreadsheets)

**LAB EXERCISE:** Write report on MS Word, Google Docs, VLC, Excel, Notepad++

**21. Software Development Process**

**THEORY EXERCISE:** Stages: Planning → Analysis → Design → Development → Testing → Deployment → Maintenance

**LAB EXERCISE:** Flowchart of SDLC using draw.io

**22. Software Requirement**

**THEORY EXERCISE:** Critical because it defines scope, avoids rework, aligns user expectations

**LAB EXERCISE:** List requirements: Login, Issue Book, Return Book, Search

**23. Software Analysis**

**THEORY EXERCISE:** Defines functionalities, user roles, system behavior

**LAB EXERCISE:** Functional analysis of user roles, cart, payment system in shopping app

**24. System Design**

**THEORY EXERCISE:** Key elements: UI design, database schema, API interfaces

**LAB EXERCISE:** Architecture: User App → API → Delivery App → Admin Panel → DB

**25. Software Testing**

**THEORY EXERCISE:** Ensures functionality, reduces bugs, improves reliability

**LAB EXERCISE:** Test cases for calculator: add(2,3)=5, div(10,2)=5, etc.

**26. Maintenance**

**THEORY EXERCISE:** Types: Corrective, Adaptive, Perfective, Preventive

**LAB EXERCISE:** Case: Security patch for online banking software

**27. Development: Web vs Desktop**

**THEORY EXERCISE:**

* **Web:** Accessible, platform-independent
* **Desktop:** Performance, offline access

**28. Web Application**

**THEORY EXERCISE:** Advantages: Cross-platform, no installation, easy updates

**29. Designing**

**THEORY EXERCISE:** UI/UX ensures user-friendliness, aesthetic appeal, and usability

**30. Mobile Application**

**THEORY EXERCISE:**

* **Native:** Platform-specific (Swift for iOS, Kotlin for Android)
* **Hybrid:** Cross-platform (Flutter, React Native)

**31. DFD (Data Flow Diagram)**

**THEORY EXERCISE:** Shows data movement within the system, improves understanding and communication

**LAB EXERCISE:** DFD for hospital: Patient -> Reception -> Doctor -> Pharmacy -> Billing

**32. Flow Chart**

**THEORY EXERCISE:** Helps visualize logic, plan programs, debug easier

**LAB EXERCISE:** Flowchart: User Registers → Email Verified → Profile Created → Login