MODULE -1: OVERVIEW OF IT INDUSTRY

❖ WHAT IS PROGRAM?

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

Example: A web browser, calculator, or video game is a program.

❖ WHAT IS PROGRAMMING?

Ans.

→Programming, or coding, is the process of creating a set of instructions for a computer to execute, enabling it to perform specific tasks.

→Programming refers to a technological process for telling a computer which tasks to perform in order to solve problems.

→ The key steps in the programming process are:

i. problem definition
ii. planning the solution
iii. coding
iv. testing
v. documentation

→theory exercise:

❖ types of programming language:

1.low level language---0 and 1(binary/machine language)

2.intermidiate level language (assembly level language)
3.high level language

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

Low level language	High level language
Easy to understand for machine, but less human readable.	Human readable and easier to use and understand.
Not Portable	Portable
High execution speed	Comparatively lower execution speed
Uses binary code	Uses syntax
Examples:assembly language,machine language	Example:java,python, c++

Que-4Describe the roles of the client and server in web communication. Network Layers on Client and Server.

Roles of Client and Server in Web Communication:

- **Client**:
  The client is typically a **web browser** or app used by a user. It **sends a request** to the server for a web page, data, or service.
- **Server**:
  The server is a computer that **hosts websites or services**. It **receives the client request**, processes it, and **sends back a response**, such as a web page or data.

Network Layers on Client and Server (in Simple Words & Short)

Both the client and server use the **same 5 layers** in the network model. Here's a simple breakdown:

1. **Application Layer**
    - What we see (websites, apps).
    - Example: HTTP, HTTPS, DNS.
2. **Transport Layer**
    - Ensures data gets sent/received correctly.
    - Example: TCP, UDP.
3. **Network Layer**
    - Chooses the best path for data.
    - Example: IP (Internet Protocol).
4. **Data Link Layer**
    - Moves data between devices on the same network.
    - Example: Ethernet, Wi-Fi.
5. **Physical Layer**
    - Actual hardware (cables, signals, etc.).
    - Example: Fiber optics, copper cables.

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

Function of the TCP/IP Model (Simple & Short):

The **TCP/IP model** helps computers **communicate over the internet** by **breaking data into layers**, so it can be sent, received, and understood properly.

Layers of TCP/IP Model (Simple Explanation):

1. **Application Layer**
    - Interacts with the user.
    - Example: Browsing, email, file transfer.
    - Protocols: HTTP, FTP, SMTP.
2. **Transport Layer**
    - Breaks data into smaller pieces and ensures correct delivery.
    - Protocols: TCP (reliable), UDP (fast but less reliable).
3. **Internet Layer**
    - Finds the best path to send data across networks.
    - Protocol: IP (Internet Protocol).
4. **Network Access Layer**
    - Moves data over physical hardware (cables, Wi-Fi).
    - Includes both Data Link & Physical layer functions

Que-6Explain Client Server Communication Types of Internet Connections

Client-Server Communication (Simple & Short):

- The **client** (like a browser or app) sends a **request** for information.
- The **server** receives the request, **processes it**, and sends back a **response**.
- Example: You open a website → browser (client) asks for the page → server sends it back → you see it.

Types of Internet Connections (Simple & Short):

1. **Dial-Up**
    - Old, slow; uses phone line.
2. **DSL (Digital Subscriber Line)**
    - Faster than dial-up; also uses phone line but keeps it free for calls.
3. **Cable**
    - Uses TV cables; fast and common in homes.
4. **Fiber Optic**
    - Very fast; uses light through glass cables.
5. **Satellite**
    - Connects via satellite; good for remote areas, but can be slower.
6. **Mobile Data (3G, 4G, 5G)**
    - Internet from cell towers; used on phones.
7. **Wi-Fi**

- Wireless connection, usually from a router at home or in public places.

Que-7How does broadband differ from fiber-optic internet?

Difference Between Broadband and Fiber-Optic Internet

- **Broadband**
  - A general term for **high-speed internet**.
  - Includes types like **DSL, cable, fiber, satellite**, etc.
- **Fiber-Optic Internet**
  - A **type of broadband**.
  - Uses **light signals through glass cables**.
  - **Much faster and more reliable** than DSL or cable.

Que-8: What are the differences between HTTP and HTTPS protocols?

Difference Between HTTP and HTTPS :

- **HTTP (HyperText Transfer Protocol)**
  - Sends data **without encryption**.
  - **Not secure** – can be read by hackers.
  - URL starts with http://.
- **HTTPS (HTTP Secure)**
  - Sends data **with encryption** (using SSL/TLS).
  - **Secure** – protects passwords, personal info.
  - URL starts with https:// and shows a **lock icon** in the browser.

- **HTTP** = not secure.
- **HTTPS** = secure, encrypted, safer for websites

Que-9 What is the role of encryption in securing applications?

Role of Encryption in Securing Applications :

- **Encryption** changes data into a **secret code** so only the right person or system can read it.
- It protects **sensitive info** like passwords, messages, and credit card details.

Why It Matters:

- Stops **hackers** from reading stolen data.
- Keeps **data private** during transfer or storage.
- Builds **trust** in apps and websites.

- **Encryption = locks your data**.
- Only the right key can unlock and read it.

Que-10 What is the difference between system software and application software?

Difference Between System Software and Application Software

- **System Software**
    - Runs the computer and manages hardware.
    - Example: **Operating System** (Windows, macOS, Linux).

- **Application Software**
    - Helps you do specific tasks.
    - Example: **Word processors**, browsers, games.

- **System software** = runs the computer.
- **Application software** = lets you do work or have fun.

Que-11 : What is the significance of modularity in software architecture?

Significance of Modularity in Software Architecture :

- **Modularity** means breaking software into **smaller, separate parts** (modules).
- Each module does **one specific job** and can work **independently**.

Why It Matters:

- **Easier to understand** and manage.
- **Faster to develop and test** each part.
- **Simple to update or fix** without affecting the whole system.
- Encourages **code reuse**.

- **Modularity = divide and conquer** in software.
- Makes code **cleaner, flexible, and easier to maintain**.

Que-12 Why are layers important in software architecture?

Why Layers Are Important in Software Architecture :-

- **Layers** organize software into **levels**, each with a specific role (like UI, logic, data).
- They help **separate concerns**, so each layer focuses on **one thing**.

Why It Matters:

- **Easier to build, test, and update**.
- **Improves teamwork**—different people can work on different layers.
- **More secure and flexible**.
- **Reduces bugs**, since changes in one layer don't break others.

- **Layers = organized structure** in software.
- Makes development **clearer, safer, and easier to manage**.

Que-13 : Explain the importance of a development environment in software production.

Importance of a Development Environment in Software Production

- A **development environment** is the setup where programmers **write, test, and debug code**.

Why It Matters:

- Provides the **tools needed** to build software (editors, compilers, debuggers).
- Helps **find and fix errors early**.
- Allows **safe testing** without breaking the real system.

- Speeds up development with **automation and version control**.

- **Development environment = workspace for coders**.
- It makes building software **faster, safer, and more efficient**.

Que-14 What is the difference between source code and machine code?

Difference Between Source Code and Machine Code :-

- **Source Code**
  - Written by programmers in **human-readable languages** (like Python, Java).
  - Easy to understand and edit.
- **Machine Code**
  - The **binary code (0s and 1s)** that the computer's processor understands directly.
  - Not readable by humans.

- **Source code = human language for programming**.
- **Machine code = computer language to run programs**.

Que-15 Why is version control important in software development?

Why Version Control Is Important in Software Development :-

- **Version control** keeps track of all changes made to the code over time.

Why It Matters:

- Helps **recover previous versions** if something breaks.
- Allows **multiple developers to work together** without conflicts.
- Keeps a **history of who changed what and when**.
- Makes it easier to **test and release updates** safely.

- **Version control = safety net and teamwork tool**.
- It keeps code **organized, safe, and manageable**.

Que-16 : What are the benefits of using Github for students?

Benefits of Using GitHub for Students :-

- **Free access** to coding tools and projects.
- Helps **learn version control** and collaboration.
- Easy to **share and showcase projects** to others.
- Supports **teamwork** on coding assignments.
- Access to lots of **open-source code to learn from**.
- GitHub helps students **code, collaborate, and build their portfolios** easily.

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

Differences Between Open-Source and Proprietary Software

Open-Source Software

    - Code is **public and free to use, modify, and share**.
    - Community-driven development.
    - Example: Linux, Firefox.
- **Proprietary Software**
    - Code is **closed and owned by a company**.
    - You must **buy or get permission** to use it.
    - Example: Microsoft Windows, Adobe Photoshop.

- **Open-source = free and open to everyone**.
- **Proprietary = owned, restricted, usually paid**.

Que-18 : How does GIT improve collaboration in a software development team?

How GIT Improves Collaboration in Software Development

- Allows **multiple developers to work on the same project** at the same time.
- Tracks **who made what changes and when**.
- Helps **merge changes smoothly** without losing work.
- Enables **review and rollback** if mistakes happen.

- GIT keeps teamwork **organized, efficient, and safe** when coding together.

Que-19 :- What is the role of application software in businesses?

Role of Application Software in Businesses

- Helps businesses **perform specific tasks** like accounting, communication, and data management.
- Improves **productivity and efficiency**.
- Enables **better decision-making** with tools like spreadsheets and databases.
- Supports **customer service** through apps like email and CRM.

- Application software helps businesses **work smarter and faster**.

Que-20 : What are the main stages of the software development process?

Main Stages of the Software Development Process

1. **Planning**
    - Understand what the software should do.
2. **Design**
    - Plan how the software will work and look.
3. **Development**
    - Write the actual code.
4. **Testing**
    - Check for bugs and fix problems.
5. **Deployment**
    - Release the software for users.
6. **Maintenance**

- Update and improve the software over time.

- Software development = **Plan → Design → Build → Test → Release → Maintain**.

Que-21 Why is the requirement analysis phase critical in software development?

Why Requirement Analysis Is Critical in Software Development

- It **defines what the software must do** based on user needs.
- Helps avoid **misunderstandings and mistakes** later.
- Ensures the final product **meets expectations**.
- Saves time and money by **planning correctly upfront**.

- Requirement analysis = **foundation for successful software**.
- Get it right to build the right product

Que-22 What is the role of software analysis in the development process?

Role of Software Analysis in Development (Simple & Short):

- **Software analysis** studies what the software needs to do.
- It gathers and understands user requirements.
- Identifies problems and solutions before coding starts.
- Helps create a clear plan for design and development.

- Software analysis = **understanding and planning** before building software.
- It ensures the software solves the right problems.

Que-23 What are the key elements of system design?

Key Elements of System Design (Simple & Short):

1. **Architecture**
    - Overall structure and how parts connect.
2. **Components**
    - Individual modules or pieces of the system.
3. **Interfaces**
    - How components interact with each other.
4. **Data Flow**
    - How data moves through the system.
5. **Security**
    - Protecting the system and data.
6. **Performance**
    - Making sure the system works fast and efficiently.

- System design = **plan for building and connecting parts** to work well together.

Que-24 Why is software testing important?

Why Software Testing Is Important (Simple & Short):

- Finds **bugs and errors** before users see them.
- Ensures the software **works correctly**.
- Improves **quality and reliability**.
- Helps make software **safer and easier to use**.
- Saves **time and money** by catching problems early.

- Testing = **making sure software works well and is bug-free**.

Que-25 What types ofsoftware maintenance are there?

Types of Software Maintenance (Simple & Short):

1. **Corrective Maintenance**
    - Fixes bugs and errors.
2. **Adaptive Maintenance**
    - Updates software to work with new environments (like new OS).
3. **Perfective Maintenance**
    - Improves features and performance.
4. **Preventive Maintenance**
    - Prevents future problems by improving code.

- Maintenance = **fix, update, improve, and protect software** over time.

Que-26 : What are the key differences between web and desktop applications?

Key Differences Between Web and Desktop Applications (Simple & Short):

- **Web Applications**
  - Run in a **web browser** (like Chrome, Firefox).
  - Need an **internet connection**.
  - Can be used on any device with a browser.
  - Updated on the server—no user installation needed.
- **Desktop Applications**
  - Installed directly on a **specific computer**.
  - Can work **without internet**.
  - Usually designed for one operating system (Windows, macOS).
  - Users must update or install new versions manually.

- Web apps = **browser-based, online, easy access**.
- Desktop apps = **installed locally, offline, device-specific**.

Que-27 What are the advantages of using web applications over desktop applications?

Advantages of Web Applications Over Desktop Applications:

- **Accessible anywhere** with an internet connection.
- No need to **install or update** on each device.
- Works on **different devices and operating systems**.

- Easier to **collaborate and share** with others.
- Updates happen **automatically on the server**.

- Web apps = **flexible, easy to maintain, and accessible** compared to desktop apps.

Que-28 What role does UI/UX design play in application development?

Role of UI/UX Design in Application Development

- **UI (User Interface)** is how the app looks—buttons, colors, layout.
- **UX (User Experience)** is how easy and enjoyable the app is to use.

Why It Matters:

- Makes the app **attractive and user-friendly**.
- Helps users **find what they need quickly**.
- Increases **user satisfaction and engagement**.
- Reduces mistakes and frustration.

- UI/UX design = **makes apps look good and work well for users**.

Que-29 What are the differences between native and hybrid mobile apps?

Differences Between Native and Hybrid Mobile Apps

- **Native Apps**
    - Built for **one platform** (like Android or iOS).

    - Use platform-specific languages (like Swift or Kotlin).
    - **Faster performance** and full access to device features.
    - Must build **separate apps** for each platform.
- **Hybrid Apps**
    - Built using **web technologies** (like HTML, CSS, JavaScript).
    - Work on **multiple platforms** with one codebase.
    - **Slightly slower** than native apps.
    - Easier and cheaper to develop.

- **Native = faster, one platform**.
- **Hybrid = one app, multiple platforms**.

Que-30 : What is the significance of DFDs in system analysis?

Significance of DFDs in System Analysis :

- **DFD (Data Flow Diagram)** shows **how data moves** through a system.
- Helps **understand the system's processes** clearly.
- Shows where data **comes from, goes to, and how it's processed**.
- Useful for **planning, improving, and communicating** system design.

- DFD = **visual map of data movement** in a system.
- Makes systems easier to **analyze, design, and explain**

Que-31 What are the pros and cons of desktop applications compared to web applications?

Pros and Cons of Desktop Applications vs Web Applications

→ **Pros of Desktop Applications**

- Work **without internet**.
- Usually **faster** and more powerful.
- Can access **full system resources** (like files, devices).

→ **Cons of Desktop Applications**

- Must be **installed** on each device.
- **Updates** need to be done manually.
- Work only on **specific operating systems**.

→ **Pros of Web Applications**

- **Accessible anywhere** with internet.
- **No installation** needed.
- Work on **any device or OS** with a browser.
- **Auto updates** from the server.

→ **Cons of Web Applications**

- Need **internet to work** (in most cases).
- Can be **slower** than desktop apps.
- Limited access to **device features**.

- **Desktop apps = powerful, offline, but less flexible**.
- **Web apps = easy access, no install, but need internet**

Que-32 How do flowcharts help in programming and system design?

How Flowcharts Help in Programming and System Design

- **Flowcharts** are diagrams that show the **steps of a process**.
- Help **visualize the logic** before coding.
- Make it easier to **plan, understand, and debug** programs.
- Useful for **communicating ideas** with others clearly.

- Flowcharts = **visual guide to how a program or system works**.
- Help with **planning, problem-solving, and communication**.