Module 1: Overview of IT Industry

1. What is program?

- Program is a set of Instruction. Program in IT industry is a set of instruction written in a programming language for performing a task. It performs specific task like calculations, data processing, displaying information, running apps.

LAB EXERCISE:

- Write a simple "Hello World" program in two different programming languages of your choice. Compare the structure and syntax.

```
#include<stdio.h>
main(){
        printf( “hello world ”) ;
}
```

THEORY EXERCISE:

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

→ Program is set of instruction which works to run a task in computing language. This instruction are converted into machine language that CPU understand.

A program functions on these steps.

→ We write a program then the program is converted into machine language then the CPU executes the instruction after that Program will take input and give output to the User.

2. What is Programming?

→ Programming is process of giving instruction to a computer to make it perform a task. It is a process pf writing, testing and maintaining code using a computer language.

THEORY EXERCISE:

- What are the key steps involved in the programming process?

→ Problem definition, analysis, algorithm design, coding, testing, etc. are involved in programming process.

3. Types of Programming Languages?

→ There are 4 types of programming language

- Procedural programming language
- Object oriented programming language
- Logical programming language
- Functional programming language

THEORY EXERCISE:

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

→ High level language: This language is easy and closer to human level language. They allow programmers to write instructions in a simple and readable form.

For example: python, java , c++, c#, Javascript these are the high-level language.

→ Low level language: This language is closer to machine language and difficult for humans to read. They directly interact with hardware.

4. World Wide Web & How Internet Works

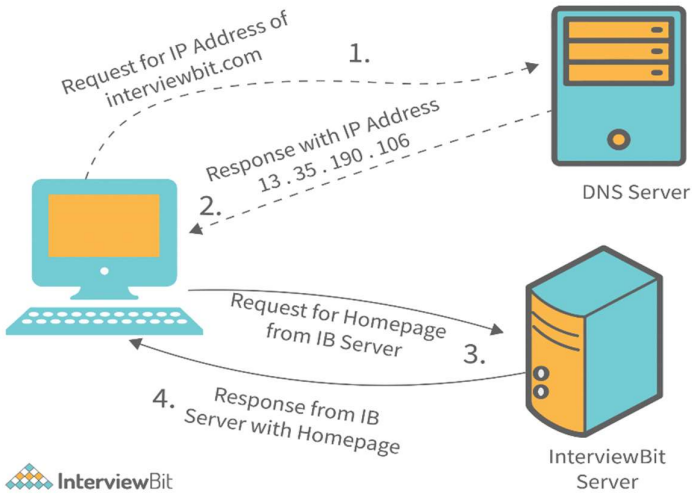
→ World wide web is a system of interlinked to web pages and web resources that can be accessed through the internet using a web browser. It allow users to view and share information like text, image, videos, websites, and other multimedia.

How internet works:

→ Firstly we can send a request then the request goes to your ISP (internet service provider) then the it convert websites into IP address then sever respond back and lastly the browser display the page.

LAB EXERCISE:

→ Research and create a diagram of how data is transmitted from a client to a server over the internet.



This diagram shows the data flow to client to a server over the internet.

THEORY EXERCISE:

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

→ Client are the request maker and the client is a device or software that request information from a web server.

Server are the Response provider and the server is a powerful computer that stores websites, data, and resources and send them to client on request.

5. Network Layers on Client and Server

→ In computer network both the client and server use the same set of network layers to communicate.

Network layers used by client and server:

Application layer, Presentation layer, Session layer, Transport layer, Network layer, Data Link layer, Physical layer.

LAB EXERCISE:

Design a simple HTTP client-server communication in any language.

THEORY EXERCISE:

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

→ The TCP/IP (transmission control protocol / Internet protocol) is a framework used for communication over the internet. It defines how the data is transmitted, from one device to another.

Types of layers:

Application layers, Transport layers, Internet layers, Network Access layer.

6. Client and Servers?

→ A client is a device or software that request a service or information from another computer on the network.

Server – server is a program that provides services, shares resources or send data to a client.

THEORY EXERCISE:

- Explain Client Server Communication

→ Client server communication is a network model where a client sends a request to a server and the server process the request and send back a response.

A. Client Sends a Request

The client could be:

- A web browser (Chrome, Firefox)
- A mobile app
- Any software needing data

It sends a request using a communication protocol (usually **HTTP/HTTPS**).

B. Server Receives and Processes

The server:

- Reads the request
- Fetches data from a database
- Performs logic (authentication, calculations, etc.)

C. Server Sends a Response

The response can contain:

- Data (JSON, XML, HTML)
- Images
- Error messages (404, 500)

D. Client Displays or Uses the Data

The client:

- Shows it on the screen
- Uses it in the app workflow
- Stores it temporarily

7. Types of Internet Connections

- Dial-up
- DSL
- Cable
- Fiber
- Satellite
- Mobile
- Wi-Fi
- Broadband

THEORY EXERCISE:

- How does broadband differ from fiber-optic internet?

  Broadband is a general term for all high-speed internet connections, including DSL, cable, satellite, and fiber. It can use copper wires or wireless signals to deliver data. Fiber-optic internet, on the other hand, is a specific type of broadband that uses **f**iber cables to transmit data as light. This makes fiber much faster, more reliable, and lower in latency compared to other broadband types that rely on electrical signals.

8. What Are Protocols?

Protocols are a set of rules and standards that define how data is transmitted and communicated between devices over a network. They ensure that computers, despite having different hardware or software, can understand each other and exchange information correctly. Protocols specify data format, timing, error handling, and communication methods. Examples include HTTP, HTTPS, FTP, TCP/IP, SMTP, and DNS. Without protocols, communication over the internet would be unorganized and unreliable.

THEORY EXERCISE:

- What are the differences between HTTP and HTTPS protocols?

  HTTP (Hypertext Transfer Protocol)

- Data is transferred in plain text
- Not secure — data can be intercepted by attackers
- Uses port 80
- No encryption

HTTPS (Hypertext Transfer Protocol Secure)

- Uses SSL/TLS encryption to secure communication
- Protects data from hacking, tampering, and eavesdropping
- Uses port 443
- Shows a padlock in the browser and a valid certificate

9. Application Security

Application Security refers to all the processes, tools, and practices used to protect software applications from threats, vulnerabilities, and attacks. It ensures that applications function safely, even when attackers try to exploit weaknesses.

THEORY EXERCISE:

- What is the role of encryption in securing applications?
  Encryption plays a crucial role in securing applications by converting sensitive data into an unreadable format so that unauthorized users cannot access it. It protects data in transit (while being sent over networks) and at rest (when stored in databases or files). Even if attackers intercept or steal the data, they cannot understand it without the correct decryption key. Encryption prevents data breaches, protects user privacy, secures login credentials, and ensures safe communication between clients and servers. Overall, encryption ensures confidentiality, integrity, and trust in applications.

10. Software Applications and its Types:

Software Applications (or Application Software) are programs designed to help users perform specific tasks. These tasks can include writing documents, browsing the

internet, playing games, editing photos, managing data, and more.
They are different from system software (like Windows or Linux), which only supports the computer's functioning.
Application software exists to help users do their work easily.

LAB EXERCISE:

- Identify and classify 5 applications you use daily as either system software or application software.

System Software

- Windows OS
- Android OS / iOS

Application Software

- Google Chrome
- WhatsApp
- MS Word

THEORY EXERCISE:

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

System Software

- System software controls and manages the hardware of a computer.
- It acts as a bridge between the user, applications, and the hardware.
- It runs the computer and provides a platform for application software.
- Examples: Windows, Linux, Android, macOS, Device Drivers.

Application Software

- Application software is designed to help users perform specific tasks.
- It runs on top of system software and cannot work without it.
- Used for activities like writing documents, browsing, messaging, or gaming.
- Examples: MS Word, Chrome, WhatsApp, Photoshop, VLC Player.
-

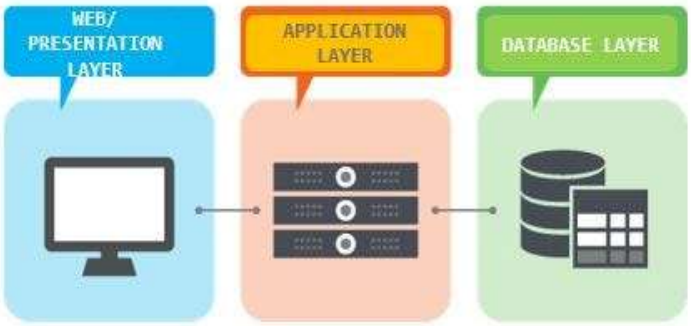
11. Software Architecture:

Software Architecture is the high-level structure of a software system—how its components are organized, how they interact, and the principles guiding its design and evolution. It's the blueprint that helps teams build systems that are scalable, maintainable, secure, and reliable.

LAB EXERCISE:

- Design a basic three-tier software architecture diagram for a web application.

Let us walk through a three tier architecture :



THEORY EXERCISE:

- What is the significance of modularity in software architecture?

→ Modularity is the practice of dividing a software system into smaller, independent, and well-defined components (modules), each responsible for a specific functionality.

The significance of modularity in software architecture lies in its ability to break a system into independent modules, which improves maintainability, scalability, reusability, testing, flexibility, and overall system reliability.

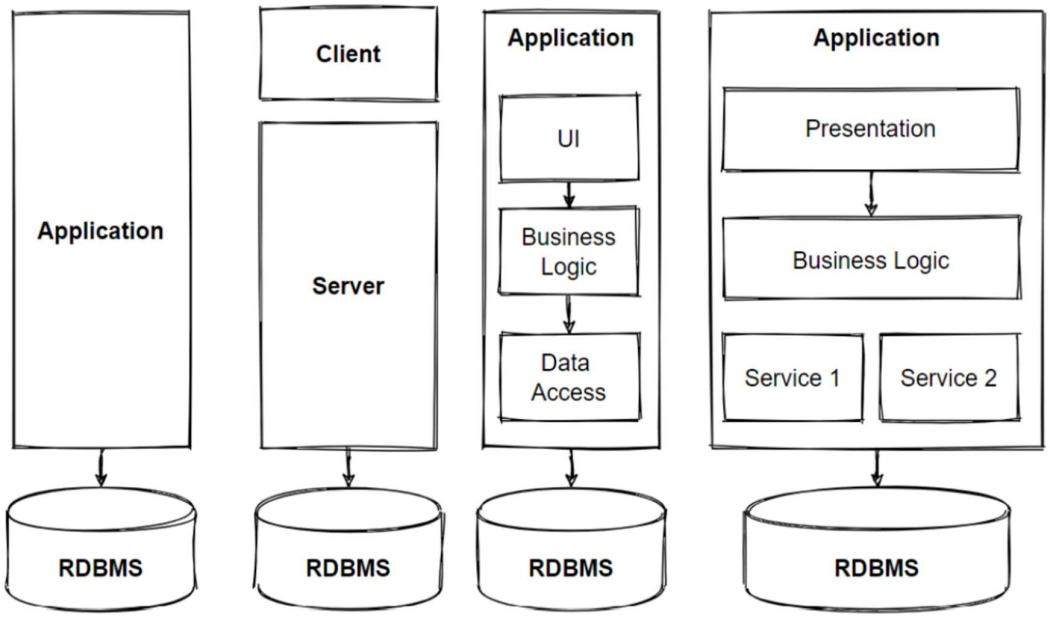
12. Layers in Software Architecture:

Layers in software architecture refer to organizing a system into hierarchical levels, where each layer has a specific responsibility and interacts only with adjacent layers.
Common Layers

1. Presentation Layer (UI Layer)
    - Handles user interaction and display
    - Examples: Web pages, mobile app screens
2. Application / Business Logic Layer
    - Processes data and applies business rules
    - Controls the flow between UI and data
3. Data Access Layer
    - Manages communication with databases
    - Performs CRUD (Create, Read, Update, Delete) operations
4. Database Layer
    - Stores and retrieves application data
    - Examples: MySQL, PostgreSQL, MongoDB

LAB EXERCISE:

Create a case study on the functionality of the presentation, business logic, and data access layers of a given software system.



THEORY EXERCISE:

- Why are layers important in software architecture?

  → Layers are important in software architecture because they separate concerns, allowing each part of the system to focus on a specific responsibility. This separation makes software easier to understand, develop, test, maintain, and scale. Layers also improve security, enable parallel development, and allow changes in one layer without affecting others, resulting in a more reliable and flexible system.

  It also improves security by restricting direct access to sensitive data and business logic. Additionally, layers enable parallel development, simplify testing and debugging, and enhance the overall reliability and flexibility of the software system.

13. Software Environments:

Software environments refer to the combination of hardware, operating system, software tools, libraries, configurations, and settings in which an application is developed, tested, deployed, and run. They play a crucial role in the software development life cycle (SDLC).

Environment in industry:

a. Analysis and Design Environment

b. Development Environment

c. Common build Environment

d. The testing Environment

e. Production Environment

LAB EXERCISE:

- Explore different types of software environments (development, testing, production). Set up a basic environment in a virtual machine.

Setting Up a Basic Environment in a Virtual Machine

Step 1: Install Virtualization Software

Step 2: Create a New Virtual Machine

Step 3: Install Operating System

Step 4: Set Up Development Tools

Step 5: Snapshot and Testing

THEORY EXERCISE:

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

  → A development environment is important in software production because it provides a controlled space where developers can write, test, and modify code safely without affecting live systems. It includes essential tools such as IDEs, compilers, debuggers, libraries, and version control systems, which improve productivity and code quality. The development environment allows early detection of errors, supports experimentation and innovation, and enables collaboration among developers. By isolating changes from testing and production environments, it reduces risks, ensures stability, and helps deliver reliable, secure, and high-quality software efficiently.

14. Source Code:

Source code is a set of instructions written by a programmer using a programming language such as Python, Java, C, or C++. It is human-readable and describes what the program is supposed to do. Source code is the starting point of any software application, which is later converted into machine code so that the computer can execute it.

- Written in high-level languages (easy for humans to read).
- Can be edited, debugged, and maintained.
- Needs a compiler or interpreter to convert into machine code.

LAB EXERCISE:

- Write and Upload Your First Source Code File to GitHub

1. Write a simple program (e.g., in Python):
2. print("Hello, World!")
3. Save the file as hello.py.
4. Create a GitHub repository:
    - Go to GitHub → Click New Repository → Enter name → Create.
5. Upload the file:
    - Open terminal or Git Bash:
    - git init
    - git add hello.py
    - git commit -m "First commit"
    - git branch -M main
    - git remote add origin <your-repo-URL>
    - git push -u origin main
6. Verify the file appears in your GitHub repository.

THEORY EXERCISE:

- Difference between Source Code and Machine Code

Aspect	Source Code	Machine Code
Readability	Human-readable	Binary (0s and 1s), not readable by humans
Purpose	Written by programmers to define program logic	Executed by the computer directly
Language	High-level languages like Python, Java, C++	Low-level binary instructions
Conversion	Needs compiler or interpreter to execute	Runs directly on CPU
Example	print("Hello, World!")	10110100 11001010 ...

Source code is the original program written by humans, while machine code is the computer-readable version of the same program.

15. GitHub and Introductions:

GitHub is a web-based platform that uses Git version control to store, manage, and track changes in source code. It allows developers to collaborate, share code, and maintain project history in a centralized repository.

Key Points:

- Supports collaboration among multiple developers.
- Tracks changes to code over time.
- Facilitates branching, merging, and issue tracking.
- Helps maintain project history and backup.

LAB EXERCISE:

- Create a GitHub Repository and Document How to Commit and Push Code Changes

1. Create a Repository on GitHub:
    - Log in to GitHub → Click New Repository → Enter a repository name → Choose Public or Private → Click Create repository.
2. Clone Repository to Local System (Optional):
3. git clone <repository-URL>
4. cd <repository-name>
5. Add Your Code Files:
6. git add <filename>
7. # Example:
8. git add hello.py
9. Commit Changes:

10. git commit -m "Initial commit with hello.py"
11. Push Changes to GitHub:
12. git push origin main
13. Verify:
    - Open your GitHub repository in the browser.
    - Check that the file is uploaded successfully.

THEORY EXERCISE:

- Why is Version Control Important in Software Development?

Version Control is crucial for software development because it:

1. Tracks Changes: Records every modification made to code files, allowing developers to see who changed what and when.
2. Supports Collaboration: Multiple developers can work simultaneously without overwriting each other's changes.
3. Enables Rollback: Allows reverting to previous versions if a bug or issue occurs.
4. Maintains History: Provides a history of project evolution for accountability and auditing.
5. Improves Productivity: Facilitates organized workflow with branching and merging.
6. Reduces Errors: Helps resolve conflicts and prevents loss of code during development.

16. Student Account in GitHub:

A GitHub student account is a free account for students that provides access to GitHub services and educational tools. Students can create repositories, collaborate on projects, and use features like GitHub Classroom, GitHub Pages, and private repositories.

- Free access to GitHub Pro features for learning.
- Can create and manage repositories.
- Supports collaboration and version control practice.
- Provides tools for learning and showcasing projects.

LAB EXERCISE:

- Create a Student Account and Collaborate on a Project

1. Create a GitHub Student Account:
    - Go to GitHub Education → Click Get benefits → Sign up with your student email → Verify student status.
2. Create a Repository:
    - Click New Repository → Give a name → Choose Public or Private → Click Create repository.
3. Collaborate with a Classmate:
    - Go to Settings → Manage Access → Invite Collaborator → Enter your classmate's GitHub username → Click Add.
4. Work Together:
    - Both students can clone the repository locally.
    - Make changes, commit them, and push to GitHub.

- Use pull requests to review and merge code.

5. Verify Changes:
    - Open the repository on GitHub → Check that both students' commits are visible.

THEORY EXERCISE:

- Benefits of Using GitHub for Students

1. Collaboration Skills: Students learn to work in teams on shared projects.
2. Version Control Experience: Understand how to manage code changes, branches, and merges.
3. Portfolio Building: Showcase projects and code to potential employers.
4. Learning Resources: Access tutorials, open-source projects, and GitHub Classroom exercises.
5. Professional Skills: Prepares students for real-world software development workflows.

17. Types of Software:

Software is a collection of programs and instructions that tell a computer how to perform tasks. It can be classified based on its functionality and usage.

Type of Software

1. System Software:
2. Application Software:

3. Utility Software:

LAB EXERCISE:

- Create a List of Software You Use and Classify Them

Software Name	Category	Purpose/Use
Windows 10	System Software	Operating system
Google Chrome	Application	Web browsing
MS Word	Application	Document creation
VLC Media Player	Application	Media playback
Avast Antivirus	Utility Software	Protects from malware
WinRAR	Utility Software	File compression/un compression

THEORY EXERCISE:

- Differences between Open-Source and Proprietary Software:

  Open-source software promotes collaboration, transparency, and free usage. Proprietary software is owned by a company, often paid, and limits user modifications.

18.