Module 1 – Overview of IT Industry

1. *What is a Program?*

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

→ In Python Programming language we can write “Hello World” as:

SYNTAX-

print (“Hello World”)

In C Programming language we can write “Hello World” as:

SYNTAX-

#include <sdtio.h>

Int main () {

printf (“Hello World”);

return 0;

}

***THEORY EXERCISE**: Explain in your own words what a program is and how it functions.*

→ Program is refer to set of instructions or statements that tells us how to complete any task in step-by-step order. The function of program is to give instructions to perform a specific task.

2. *What is Programming?*

***THEORY EXERCISE**: What are the key steps involved in the programming process?*

→ The key steps involved in the programming process are:

1. Understanding the problem
2. Planning solution
3. Writing the program
4. Testing and Debugging
5. Execution of program
6. Maintenance of program

3. Types of Programming Languages

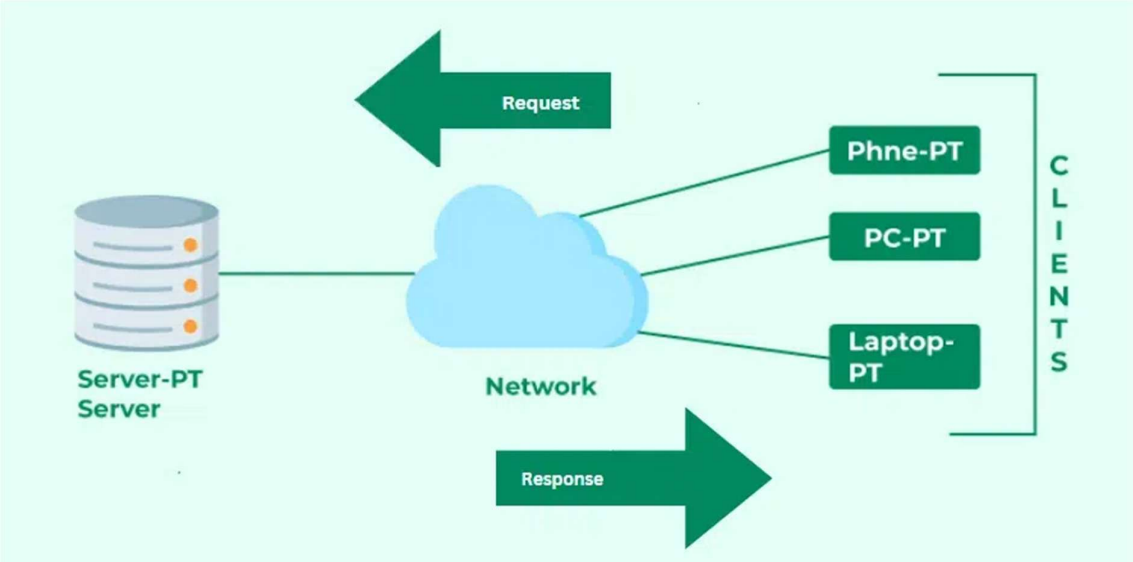
***THEORY EXERCISE**: What are the main differences between high-level and low-level programming languages?*

→ The main key difference between high-level and low-level programming language is that high-level languages are understandable by humans and is very convenient, while low-level languages are known as machine language so they are not friendly with humans so they are hard to understand.

4. World Wide Web & How Internet Works

***LAB EXERCISE**: Research and create a diagram of how data is transmitted from a client to a server over the internet.*

→ This is the diagram for Client-Server data transmission which shows the process of data transfer when requested by the client and response of that data from server.



***THEORY EXERCISE**: Describe the roles of the client and server in web communication.*

→ Roles of Client and Server in web communication are:

Client- The client is web browser or application that sends a request to the server asking for any data or services (e.g. Webpage).

Server- The server is a system that stores the data, processes them according to client requests and delivers the requested data or services to the client. It responses by sending back the data or services to the requests made by the client (e.g. HTML file, image, etc.).

5. Network Layers on Client and Server

***LAB EXERCISE**: Design a simple HTTP client-server communication in any language.*

→ Same as "*of how data is transmitted from a client to a server over the internet*".

THEORY EXERCISE*: Explain the function of the TCP/IP model and its layers.*

→ The TCP/IP model (Transmission Control Protocol/Internet Protocol) is used for network communication. It defines how data is sent, received, and routed over the Internet.

TCP/IP model has 4 layers and their function are:

1. Application Layer: Let apps use the network.
2. Transport Layer: Make sure data is sent correctly.
3. Internet Layer: Decide the route for the data.
4. Network Access Layer: Send data over actual cables or Wi-Fi.

6. Client and Servers

THEORY EXERCISE*: Explain Client Server Communication*

→ Client-Server communication is the process of data exchange between a client and a server over some network, mainly over the internet. The client is the device that requests a service or data, and the server is the system that stores the data provides that request from the client and send back a response.

7. Types of Internet Connections

LAB EXERCISE*: Research different types of internet connections (e.g., broadband, fiber, satellite) and list their pros and cons.*

→ The different types of internet connections are DSL, cable, fiber-optic, wireless, broadband and satellite.

Here are the pros and cons of each internet connections:

1. DSL:-
   Pros- Uses existing phone infrastructure so can be widely available; relatively cheap; decent for basic tasks (browsing, email).
   Cons- Speed drops if you're far from the phone exchange; upload speeds often much slower; not good for very high-bandwidth needs

2. Cable:-
   Pros- Much faster than DSL generally; widely available in urban/suburban areas; good for streaming, gaming
   Cons- Shared bandwidth means slowdowns during peak usage; latency isn't as low as fiber; performance can vary.

3. Fiber-optic:-
   Pros- Very high speeds, often symmetrical upload & download; low latency; reliable; good for future growth and heavy data use.

Cons- Expensive to deploy especially in remote/rural areas; not everywhere is connected yet; installation cost can be higher.

4. Wireless:-
   Pros- Faster installation (no need to dig long cables); can serve places where wired infrastructure is poor; good speeds in ideal conditions.
   Cons- Requires line-of-sight to the wireless tower; weather or obstacles can affect signal; speed can fluctuate.

5. Broadband:-
   Pros- Very flexible, you can use it on-the-go; newer tech (5G) offers high speeds, good for many modern uses.
   Cons- Data caps/plans may be restrictive; network congestion (many users sharing towers) can slow you; coverage isn't uniform (especially 5G) so performance varies.

6. Satellite:-
   Pros- Can reach very remote or rural areas; doesn't need many ground cables; good fall back or only option in some places.
   Cons- High latency (delay) because signals travel long distances; more expensive; weather can interfere; often data caps; slower upload speeds.

THEORY EXERCISE*: How does broadband differ from fiber-optic internet?*

→ Broadband differs from Fiber-optic as each carries different transmission medium, signal type, speed, reliability, cost and availability.

In Broadband we have:

1. Copper wires / Coaxial cables for transmission.
2. Signal type is electrical.
3. Speed is moderate.
4. Reliability affected by distance.
5. Cost is low.
6. Availability is wide.

In Fiber-optic we have:

1. Glass or plastic fiber cables for transmission.
2. Signal type is Light.
3. Speed is very high.
4. Reliability is very stable & consistent.
5. Cost is high.
6. Availability is limited.

8. Protocols

***LAB EXERCISE**: Simulate HTTP and FTP requests using command line tools (e.g., curl).*

→

***THEORY EXERCISE**: What are the differences between HTTP and HTTPS protocols?*

→ The difference between HTTP and HTTPS protocols are:

1. The HTTPS protocol is more secure then HTTP as it provides a layer of authentication and it is then only be accessed by the verified user.
2. The performance is better in HTTP is higher as it does have layer like the HTTPS protocol.

9. Application Security

***LAB EXERCISE**: Identify and explain three common application security vulnerabilities. Suggest possible solutions.*

→

***THEORY EXERCISE**: What is the role of encryption in securing applications?*

→ The role of encryption in securing applications is by protecting sensitive information from unauthorized access that only authorized user can only read it.

10. Software Applications and Its Types

***LAB EXERCISE**: Identify and classify 5 applications you use daily as either system software or application software.*

→ Google Chrome, YouTube, Spotify, Microsoft Word, Windows Defender.

***THEORY EXERCISE**: What is the difference between system software and application software?*

→ The difference between system software and application software is that System software runs the computer and manages hardware, while application software helps users perform specific tasks.

11. Software Architecture

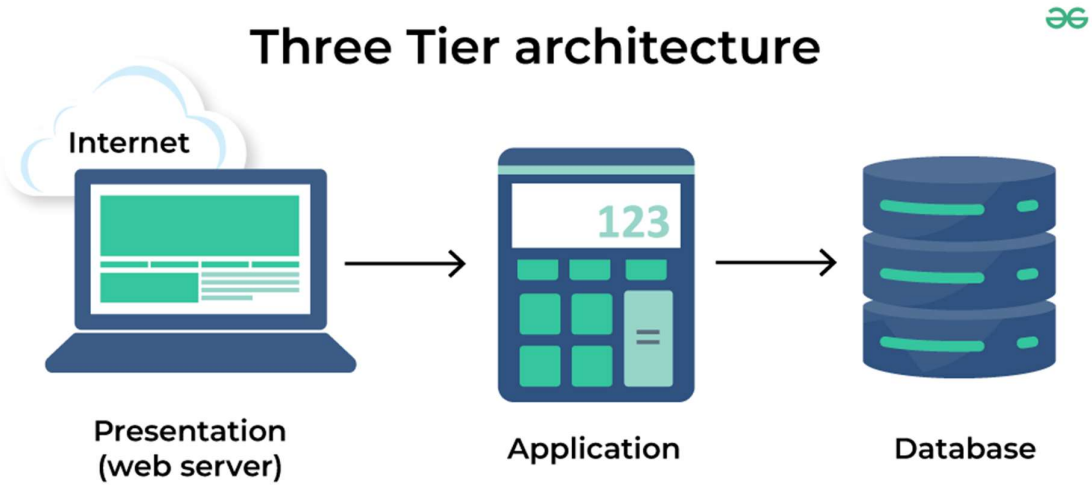
***LAB EXERCISE**: Design a basic three-tier software architecture diagram for a web application.*

→ The three-tier software architecture for a web application involves:

1. Presentation Tier
2. Application Tier

3. Database Tier

Below is the diagram for the three-tier software architecture.



***THEORY EXERCISE**: What is the significance of modularity in software architecture?*

→ The significance of modularity is that it makes software easier to understand, develop, test, and maintain. Since each module works independently, developers can make changes or fix issues in one part without affecting the whole system. Modularity helps build cleaner, more organized, and flexible software.

12. Layers in Software Architecture

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

→ Case Study: Fitness Tracker Application

1. Presentation Layer-

This layer is responsible for the user experience and interaction. It is what the user sees and interacts with on their screen.

- Displays dashboards, charts, and fitness progress.
- Allows users to log workouts, set goals, or track steps.
- Shows feedback messages like “Workout Added” or “Goal Achieved.”
- Sends requests to the client layer.

2. Application Layer-

The client layer acts as a bridge between the user interface and the business logic.

- Manages user requests and sends them to the business logic layer.
- Handles responses from the backend and delivers results back to the UI.
- Maintains session information such as login tokens and temporary data.

3. Business Layer-

This layer contains the core logic and processing rules of the app.

- Validates user data and ensures proper input.
- Calculates workout statistics like calories burned or steps achieved.
- Applies fitness rules (e.g., “Drink 2 litres of water daily”).
- Interacts with the data access layer for storing or retrieving data.

4. Persistence Layer-

The data access layer acts as a middleman between the business logic and the database. It manages all operations related to data storage and retrieval.

- Executes queries such as insert, update, delete, and fetch.
- Ensures data consistency and error handling.
- Converts raw database data into usable formats for the business layer.

5. Database Layer-

This is the lowest layer that stores all application data permanently and securely.

- Stores user profiles, workout logs, step counts, and nutrition data.
- Supports backup and recovery for data safety.
- Handles indexing and query optimization for faster data access.

THEORY EXERCISE*: Why are layers important in software architecture?*

→ Layers are important in software architecture because they organize the system into separate parts, each with a specific role. This makes the software easier to develop, understand, and maintain.

13. Software Environments

LAB EXERCISE*: Explore different types of software environments (development, testing and production). Set up a basic environment in a virtual machine.*

→

THEORY EXERCISE*: Explain the importance of a development environment in software production.*

→ Development environment in software production is important as it provides a controlled, efficient, and safe space where code is written, tested, and prepared before being moved to testing and production stages.

14. Source Code

LAB EXERCISE*: Write and upload your first source code file to GitHub.*

→

***THEORY EXERCISE**: What is the difference between source code and machine code?*

→ The difference between source code and machine code is that source code is human-readable and is written by the developers but machine code is computer-readable and is executed by the CPU.

15. GitHub and Introductions

***LAB EXERCISE**: Create a GitHub repository and document how to commit and push code changes.*

→ Steps to create GitHub repository:

- Go to GitHub and log in.
- Click + → New repository.
- Enter a Repository Name and optional description.
- Choose Public or Private.
- Click Create repository.

For commit and push code changes:

- Make changes to files.
- Check status: git status
- Stage changes: git add <file> or git add .
- Commit changes: git commit -m "Commit message"
- Push to GitHub: git push origin main

***THEORY EXERCISE**: Why is version control important in software development?*

→ Version control is important in software development because it tracks code changes, enables multiple developers to collaborate safely, allows recovery of previous versions, supports testing new features via branching, and ensures organized, reliable, and accountable project management.

16. Student Account in GitHub

***LAB EXERCISE**: Create a student account on GitHub and collaborate on a small project with a classmate.*

→

***THEORY EXERCISE**: What are the benefits of using GitHub for students?*

→ Benefits of using GitHub for students are:

- Portfolio Building
- Collaboration
- Version Control Practice
- Backup and Safety
- Learning Opportunities

- Project Management
- Resume Boost

17. Types of Software

***LAB EXERCISE**: Create a list of software you use regularly and classify them into the following categories: system, application, and utility software.*

→ List of software we use regularly and classify into system, application and utility software are:

SYSTEM SOFTWARE	APPLICATION SOFTWARE	UTILITY SOFTWARE
Windows	Microsoft Word	McAfee
iOS	Google Sheets	WinRAR
Printer Drivers	Chrome	Task Manager

***THEORY EXERCISE**: What are the differences between open-source and proprietary software?*

→

18. GIT and GITHUB Training

***LAB EXERCISE**: Follow a GIT tutorial to practice cloning, branching, and merging repositories.*

→

***THEORY EXERCISE**: How does GIT improve collaboration in a software development team?*

→

19. Application Software

***LAB EXERCISE**: Write a report on the various types of application software and how they improve productivity.*

→

***THEORY EXERCISE**: What is the role of application software in businesses?*

→ Application software plays a key role in business by helping companies work faster, stay organized, and make better decisions. It automates tasks, manages data, improves communication, supports customer service, and increases overall productivity.

20. Software Development Process

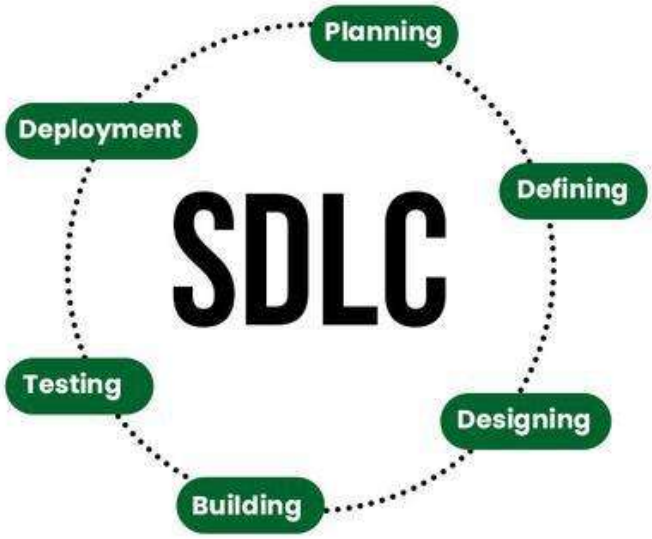
***LAB EXERCISE**: Create a flowchart representing the Software Development Life Cycle (SDLC).*

→ This flowchart represent the software development lifecycle (SDLC):

The process of SDLC has 6 phases of development-

1. Requirement gathering
2. Planning and design
3. Development

4. Testing
5. Deployment
6. Maintenance



THEORY EXERCISE*: What are the main stages of the software development process?*
→ The software development is a step-by-step process includes:

7. Requirement gathering
8. Planning and design
9. Development
10. Testing
11. Deployment
12. Maintenance

21. Software Requirement

LAB EXERCISE*: Write a requirement specification for a simple library management system.*
→

THEORY EXERCISE*: Why is the requirement analysis phase critical in software development?*
→ The requirement analysis phase is critical because it identifies the goals and features, avoid mistakes during development, saves time and cost, also plans correct and satisfies the user's need.

22. Software Analysis

LAB EXERCISE*: Perform a functional analysis for an online shopping system.*
→

***THEORY EXERCISE**: What is the role of software analysis in the development process?*

→ Software analysis plays a key role by acting like the planning phase before building the actual software. It helps the developer to understand the user's need, requirements of project, balance risk.

23. System Design

***LAB EXERCISE**: Design a basic system architecture for a food delivery app.*

→

***THEORY EXERCISE**: What are the key elements of system design?*

→

24. Software Testing

***LAB EXERCISE**: Develop test cases for a simple calculator program.*

→

***THEORY EXERCISE**: Why is software testing important?*

→ Software testing is important because it ensures that the software works correctly, reliable, and meets user expectations.

25. Maintenance

***LAB EXERCISE**: Document a real-world case where a software application required critical maintenance.*

→

***THEORY EXERCISE**: What types of software maintenance are there?*

→

26. Development

***THEORY EXERCISE**: What are the key differences between web and desktop applications?*

→ The difference between web and desktop applications are:

Web Applications – run in a browser, need internet, slow performance

Desktop Applications – installed on device, can work offline, fast performance

27. Web Application

***THEORY EXERCISE**: What are the advantages of using web applications over desktop applications?*

→ The advantages of using web applications over desktop applications are:

- No installation required
- No specific device is required

- Low maintenance cost

28. Designing

***THEORY EXERCISE**: What role does UI/UX design play in application development?*

→ UI/UX design plays a crucial role in application development by shaping how users interact with the app and how they feel while using it.

29. Mobile Application

***THEORY EXERCISE**: What are the differences between native and hybrid mobile apps?*

→ Native Apps:

Developed separately for each platform (Android or iOS). They offer better performance and full access to device features.

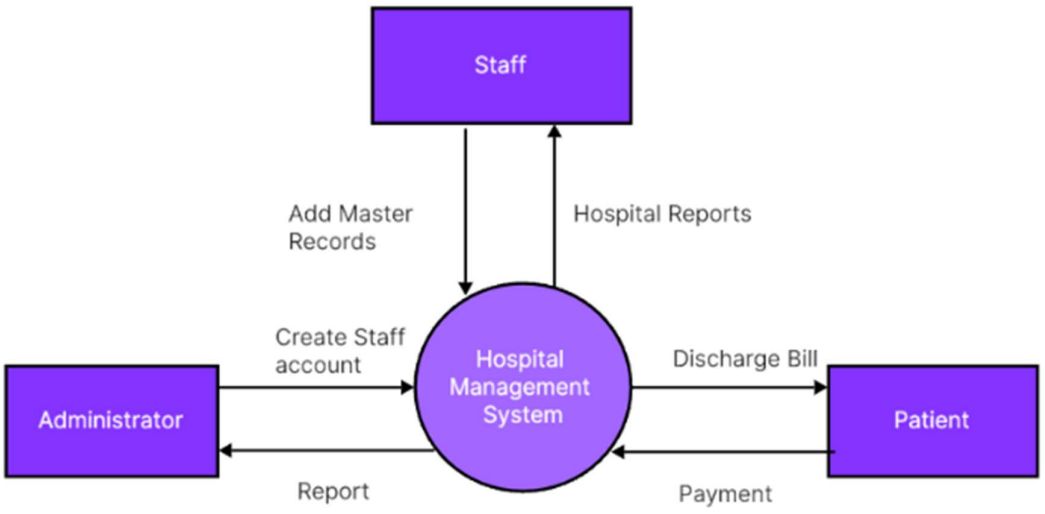
Hybrid Apps:

Developed using a single codebase for multiple platforms. They are faster and cheaper to build but may have slightly lower performance.

30. DFD (Data Flow Diagram)

***LAB EXERCISE**: Create a DFD for a hospital management system.*

→



***THEORY EXERCISE**: What is the significance of DFDs in system analysis?*

→ DFDs are important because they show the movement of data clearly, improve communication, and help build better and more efficient systems.

31. Desktop Application

***LAB EXERCISE**: Build a simple desktop calculator application using a GUI library.*

→

***THEORY EXERCISE**: What are the pros and cons of desktop applications compared to web applications?*

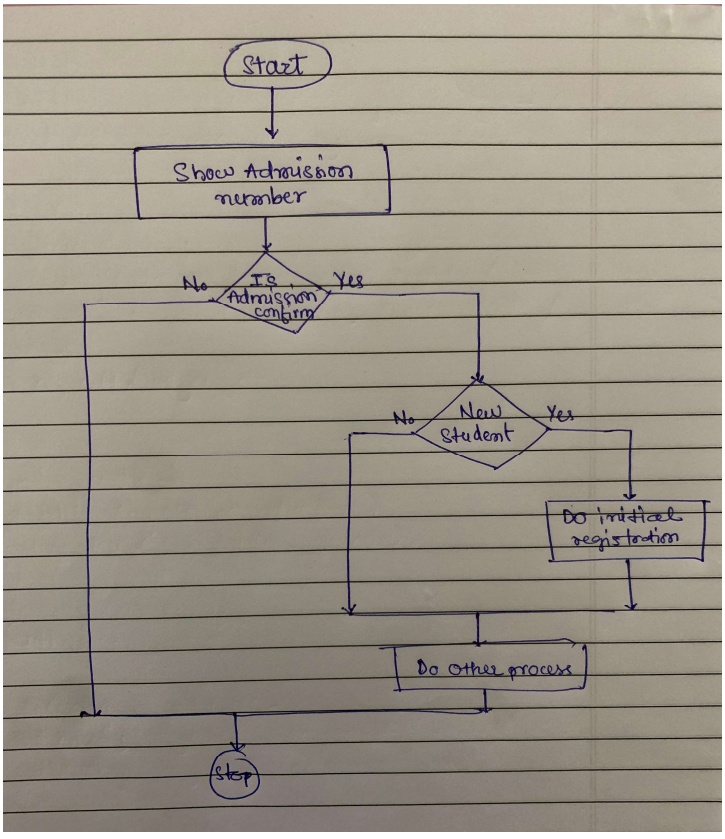
→ Compared to web applications the pros and cons of desktop applications are:

Web Applications	Desktop Applications
Faster performance	Slower performance
Works offline	Needs Internet
Installation required	No installation required

32. Flow Chart

***LAB EXERCISE**: Draw a flowchart representing the logic of a basic online registration system.*

→ This flowchart shows the student admission process:



***THEORY EXERCISE**: How do flowcharts help in programming and system design?*

→ Flowchart helps in programming and system design by giving a visual representation of how a program or system work. They use symbols and arrows to show the step-by-step flow of logic, making it easier to understand what happens first, next, and last.