NAME – Rahul koli

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

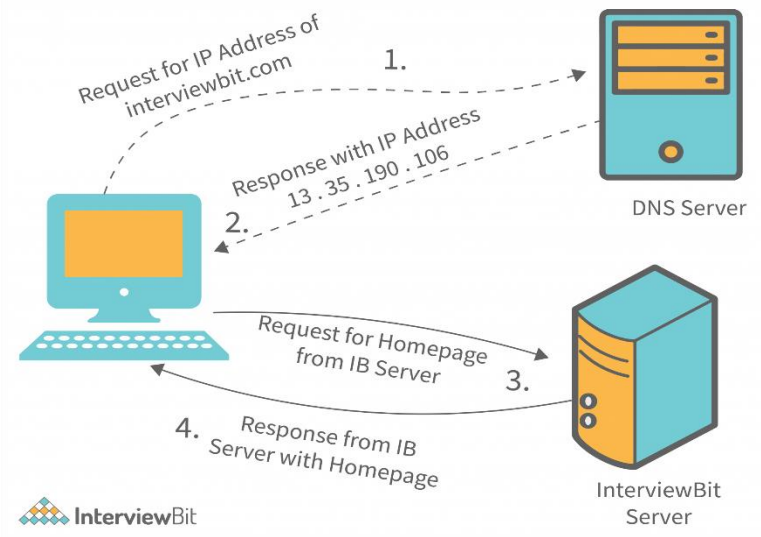
→ World wide web is a system of interlinked to web pages and web resources that can be accessed through the internet using a web browser. It allow users to view and share information like text, image, videos, websites, and other multimedia.

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

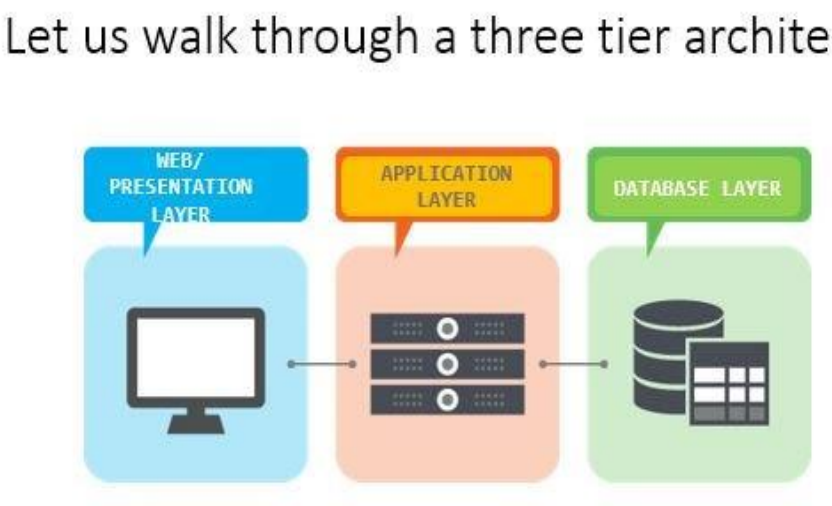
- Application software is designed to help users perform specific tasks.
- It runs on top of system software and cannot work without it.
- Used for activities like writing documents, browsing, messaging, or gaming.
- Examples: MS Word, Chrome, WhatsApp, Photoshop, VLC Player.
-

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.



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:

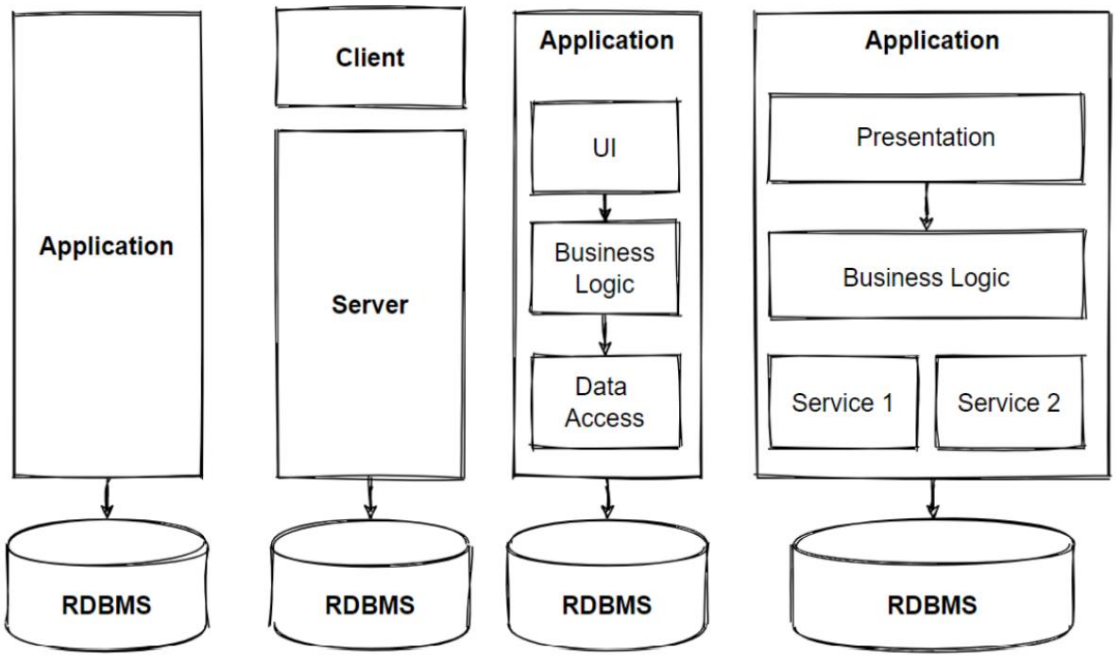
Layers in software architecture refer to organizing a system into hierarchical levels, where each layer has a specific responsibility and interacts only with adjacent layers.

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. What is GitHub and github training ?

GitHub is a cloud-based platform that hosts GIT repositories and helps developers collaborate online.

- Uses of GitHub
- Store and share code
- Collaborate with teammate
- Review code using pull requests
- Manage projects and issues

LAB EXERCISE:

- Follow a GIT tutorial to practice cloning, branching, and merging repositories.

To understand and practice basic GIT operations: cloning a repository, creating branches, and merging changes.

Requirements

- GIT installed on your system
- GitHub account
- Internet connection

Procedure

Clone a Repository

git clone https://github.com/username/sample-repo.git

cd sample-repo

Creates a local copy of the remote repository.

Check Repository Status

git status -v

THEORY EXERCISE:

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

  GIT improves collaboration in a software development team by providing an efficient and reliable way to manage source code and coordinate work among multiple developers. The key ways are explained below:

Version Control - GIT keeps a complete history of all changes made to the code. Every modification is saved with details such as the author, time, and message. This allows team members to track progress and revert to earlier versions if errors occur.

Parallel Development Using Branches - GIT allows developers to create separate branches to work on new features, bug fixes, or experiments independently. Multiple team members can work simultaneously without affecting the main codebase.

Safe Code Integration - Changes from different developers can be merged into the main branch. GIT automatically detects conflicts and helps resolve them, ensuring that code integration is controlled and safe.

Improved Team Coordination - shared repositories (e.g., on GitHub), developers can pull the latest updates, push their changes, and stay synchronized with the team's work.

Accountability and Transparency - Each commit records who made the change and why. This improves responsibility, simplifies code reviews, and makes debugging easier.

19. Application Software

Application software refers to programs designed to help users perform specific tasks or activities. These tasks can be related to work, education, entertainment, communication, or daily personal use.

Unlike system software (which runs and controls the computer), application software is user-oriented and runs on top of the operating system.

LAB EXERCISE:

- Write a report on the various types of application software and how they improve productivity.

Application software consists of programs designed to help users perform specific tasks. These tasks may include document creation, data analysis, communication, designing, or managing organizational activities. Application software plays a vital role in increasing efficiency, accuracy, and speed of work, thereby improving overall productivity.

Types of Application Software

- General-Purpose Application Software - These are commonly used by individuals and organizations for everyday tasks.

Examples:

- Word Processors (MS Word, Google Docs)
- Spreadsheets (MS Excel)

- Special-Purpose Application Software - Designed to perform specific tasks for particular professions or industries.

Examples:

- Accounting Software (Tally)
- Graphic Design Software (Photoshop)

- Customized (Tailor-Made) Application Software - Developed to meet the specific requirements of an organization or user.

Examples:

- School Management System
- Hospital Management Software
- Banking Systems

- Communication and Collaboration Software - Used for communication, meetings, and teamwork.

Examples:

- Email (Gmail)
- Video Conferencing (Zoom, Google Meet)

- Educational Application Software - Used for learning, teaching, and skill development.

Examples:

- Learning Management Systems (Google Classroom)
- Online Learning Platforms (Coursera)
- Educational Apps

- How They Improve Productivity:

- Easy access to learning resources
- Supports self-paced learning
- Enhances teaching efficiency
- Reduces dependency on physical materials

THEORY EXERCISE:

- What is the role of application software in businesses?

Application software plays a crucial role in modern businesses by helping organizations perform daily operations efficiently, accurately, and cost-effectively. It supports decision-making, communication, and overall business growth.

- Automation of Business Processes
- Improved Productivity and Efficiency
- Data Management and Analysis
- Better Communication and Collaboration
- Enhanced Customer Relationship Management
- Support for Marketing and Sales
- Cost Reduction
- Scalability and Business Growth

20. Software development process?

    The Software Development Process (also called Software Development Life Cycle – SDLC) is a structured approach used to design, develop, test, and maintain software applications. It ensures that software is developed systematically, on time, within budget, and with high quality.

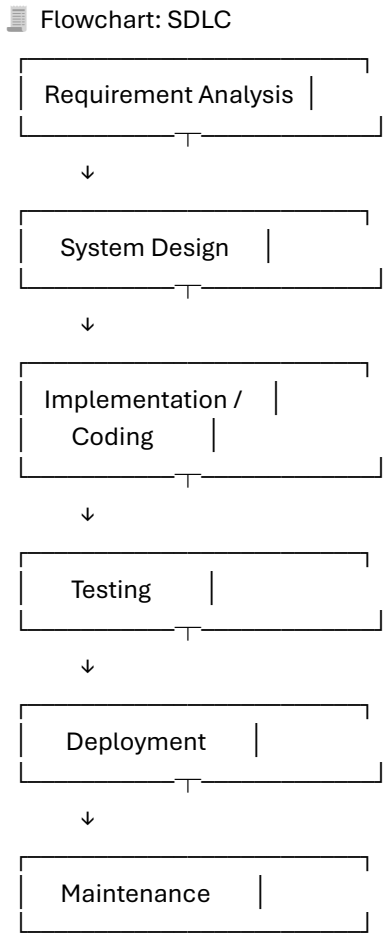
    It includes the following main stages:

- Requirement Analysis – understanding user needs
- Design – planning the system structure
- Implementation – coding the software
- Testing – finding and fixing errors
- Deployment & Maintenance – releasing and updating the software

LAB EXERCISE:

- Create a flowchart representing the Software Development Life Cycle (SDLC).

    To create a flowchart that represents the different stages of the Software Development Life Cycle (SDLC).



Requirement Analysis – Collects and insure user requirements
System Design – Prepares software architecture and design
Implementation – Converts design into code
Testing – Ensures software is error-free
Deployment – Releases software for use
Maintenance – Updates and fixes issues after release

THEORY EXERCISE:

- What are the main stages of the software development process? The Software Development Process, also known as the Software Development Life Cycle (SDLC), consists of the following main stages:

Requirement Analysis - In this stage, user needs are collected and analyzed. The requirements are clearly documented to understand what the software should do.

System Design - Based on the requirements, the system architecture, data structures, and interfaces are designed. This stage acts as a blueprint for development.

Implementation (Coding) - The actual software is developed by writing source code using suitable programming languages and tools according to the design.

Testing - The software is tested to identify and fix errors. It ensures the software meets user requirements and works correctly.

Deployment - After successful testing, the software is released and installed for users to use

Maintenance - This final stage involves fixing bugs, updating features, and improving performance after the software is in use.

21. Software Requirement

A software requirement is a clear statement that describes what a software system should do and the conditions under which it must operate. It defines the needs and expectations of users and stakeholders from the software.
Software requirements are the foundation of the software development process, because all design, coding, and testing are based on them.

LAB EXERCISE:

- Write a requirement specification for a simple library management system.

  The purpose of this document is to describe the requirements of a Simple Library Management System (LMS). The system helps manage library resources such as books, members, and book transactions efficiently.

Requirements

1. Functional Requirements

- The system shall allow the librarian to add, update, and delete book records.
- The system shall allow the librarian to register and manage members.
- The system shall allow users to search books by title or author.
- The system shall issue books to members and record issue/return dates.
- The system shall calculate fines for late returns.

2. Non-Functional Requirements

- The system shall be easy to use.
- The system shall provide fast response time.
- The system shall ensure data security and authorized access.
- The system shall be reliable and accurate.

THEORY EXERCISE:

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

  The requirement analysis phase is critical in software development because it clearly defines what the software should do and what the user expects before development begins.

Reasons Why It Is Critical

- Clear Understanding of User Needs

It helps developers understand the exact needs and expectations of users, reducing confusion.

- Prevents Errors and Rework

Well-defined requirements reduce mistakes in design and coding, saving time and cost.

- Acts as a Foundation for All Phases

Design, coding, testing, and maintenance depend on accurate requirements.

- Improves Project Planning

Helps in estimating cost, time, and resources correctly.

- Ensures Customer Satisfaction

Delivering software based on user requirements increases acceptance and satisfaction.

22. Software Analysis

Software analysis is the process of studying, understanding, and defining the requirements of a software system. It focuses on identifying what the software should

do, the problems it must solve, and the constraints under which it must operate.
Software analysis is usually performed at the early stage of software development and plays a key role in building the right system.

LAB EXERCISE:

- Perform a functional analysis for an online shopping system.

Aim

To perform a functional analysis for an online shopping system by identifying its key functions, inputs, outputs, and user interactions.

Tools Required

- Computer
- Text editor (Notepad / MS Word)

Problem Statement

Analyze the functions of an online shopping system to understand how users interact with it and what operations the system must perform.

THEORY EXERCISE:

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

Software analysis plays a crucial role in ensuring that the final software meets user needs and is developed efficiently. It is usually performed in the early stages of the software development process.

Key Roles of Software Analysis

- Understanding User Requirements
  - Collects and studies requirements from users and stakeholders
  - Identifies what the system must do and the constraints
- Defining System Functions
  - Determines the functional and non-functional requirements
  - Forms the basis for design and implementation
- Feasibility Assessment
  - Evaluates technical, economic, and operational feasibility
  - Ensures the project is practical before development begins
- Documentation
  - Prepares the Software Requirement Specification (SRS)
  - Serves as a reference for developers, testers, and stakeholders
- Reducing Development Risks
  - Identifies potential problems early
  - Minimizes errors, rework, and delays
- Improving Communication
  - Provides a clear understanding between users, analysts, and developers
  - Ensures everyone agrees on the system requirements

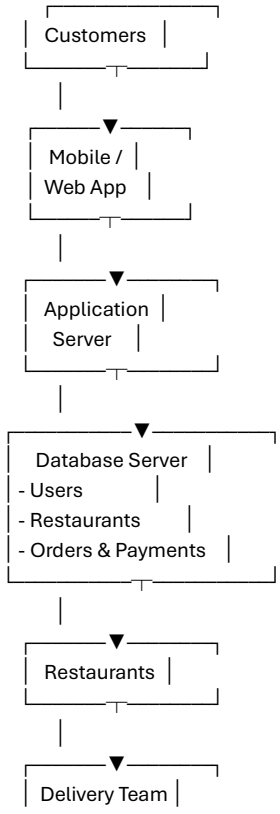
23. System Design?

System design is the phase in the software development process where the requirements gathered during analysis are translated into a blueprint for building the software. It focuses on how the system will be implemented rather than what it will do.

System design bridges the gap between requirement analysis and implementation.

LAB EXERCISE:

- Design a basic system architecture for a food delivery app?



The basic system architecture outlines how customers, restaurants, and delivery personnel interact through a centralized application and database, ensuring smooth operation of a food delivery app.

THEORY EXERCISE:

- What are the key elements of system design?

System design involves creating a blueprint for a software system based on the requirements gathered during analysis. The key elements ensure that the system is efficient, maintainable, and meets user needs.

The key elements of system design are:

1. Architecture Design – Overall structure and module interactions.
2. Data Design – How data is stored, accessed, and managed.
3. Interface Design – User interfaces and APIs for interaction.
4. Module / Component Design – Division of system into functional units.
5. Control / Process Design – Flow of logic and processes.
6. Security & Performance – Ensuring safety, reliability, and efficiency.

24. Software Testing

Software testing is the process of evaluating a software application to determine whether it meets the specified requirements and detects defects or bugs. Testing ensures the software is reliable, functional, and of high quality before release.

Software testing is a vital part of the software development process that ensures the software works correctly, is bug-free, and meets user expectations.

LAB EXERCISE:

- Develop test cases for a simple calculator program

Test Case ID	Function	Input	Expected Output	Result (Pass/Fail)
TC1	Addition	5 + 3	8	
TC2	Addition	-2 + 7	5	
TC3	Subtraction	10 - 4	6	
TC4	Subtraction	0 - 5	-5	
TC5	Multiplication	6 × 3	18	
TC6	Multiplication	-2 × 4	-8	
TC7	Division	8 ÷ 2	4	
TC8	Division	5 ÷ 0	Error / Cannot divide by zero	
TC9	Decimal Addition	2.5 + 3.2	5.7	
TC10	Large Number Multiplication	1000 × 2000	2,000,000	

Explanation

- Each test case checks a specific function of the calculator.
- Inputs include positive numbers, negative numbers, decimals, and edge cases (like division by zero).
- Expected output defines what the correct result should be.
- Result column is filled after executing the test.

THEORY EXERCISE:

Why is software testing important?

Software testing is a critical phase in the software development process because it ensures that the software works correctly, is reliable, and meets user expectations.

Software testing is important because it ensures that the software:

1. Detects and fixes errors before deployment.
2. Ensures quality and reliability, meeting user requirements.
3. Reduces maintenance costs and prevents failures.

25. Maintenance

Software maintenance is the process of modifying and updating software after it has been delivered to correct faults, improve performance, or adapt it to new requirements. It ensures the software remains useful, reliable, and efficient throughout its lifecycle.

Importance of Software Maintenance

- Keeps software reliable and error-free.
- Ensures compatibility with changing environments.
- Enhances user satisfaction by adding features and improvements.
- Reduces long-term costs of software operation.

LAB EXERCISE:

- Document a real-world case where a software application required critical maintenance.

Critical Maintenance Performed

Type of Maintenance	Actions Taken
Corrective Maintenance	Fixed bugs in billing calculations and appointment scheduling modules.
Adaptive Maintenance	Updated the software to work with the latest OS and hardware upgrades.
Perfective Maintenance	Optimized database queries to improve response time and reduced system crashes.
Preventive Maintenance	Implemented monitoring tools to predict future system failures and performed code refactoring for maintainability.

THEORY EXERCISE:

What types of software maintenance are there?

Software maintenance refers to modifying and updating software after its delivery to correct faults, improve performance, or adapt it to new requirements. There are four main types:

➢ Corrective Maintenance

- Purpose: Fixes bugs or defects found after software deployment.

- Example: Correcting errors in a billing module of an accounting software.

➢ Adaptive Maintenance

- Purpose: Modifies software to work in a changed environment, such as new hardware, OS, or regulations.
- Example: Updating a web application to support a new browser version.

➢ Perfective Maintenance

- Purpose: Improves software performance, usability, or adds new features based on user feedback.
- Example: Optimizing database queries to speed up a reporting module.

➢ Preventive Maintenance

- Purpose: Prevents potential future problems or failures by refactoring code or updating documentation.
- Example: Refactoring legacy code to make it more maintainable.

26. Development

Software development is the process of designing, coding, testing, and deploying a software application to meet specific user needs or solve a particular problem. It involves creating programs that are functional, reliable, and efficient.

➢ Objectives of Software Development

1. Solve problems – Provide software solutions for business, education, or personal use.

2. Meet user requirements – Develop software according to specifications.
3. Ensure quality and reliability – Build software that is error-free and dependable.
4. Maintainability and scalability – Software should be easy to update and extend in the future.

THEORY EXERCISE:

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

Key Differences Between Web and Desktop Applications

Feature	Web Application	Desktop Application
Definition	Runs on a web browser and accessed over the internet	Installed and runs directly on a computer
Installation	No installation required; accessed via URL	Requires installation on each device
Accessibility	Accessible from anywhere with an internet connection	Accessible only on the installed device
Updates	Updated on the server; users	Users must manually install updates

Feature	Web Application	Desktop Application
	automatically get the latest version	
Platform Dependency	Usually platform-independent (works on Windows, Mac, Linux)	Platform-dependent; must be compatible with OS
Performance	Dependent on internet speed and server response	Typically faster as it runs locally
Examples	Gmail, Google Docs, Amazon, Facebook	MS Word, VLC Media Player, Photoshop

27. Web Application

A web application is a software program that runs on a web browser and is accessed over the internet or an intranet. Unlike desktop applications, web applications do not need to be installed on a user's device; they are hosted on a web server.

- Key Features of Web Applications

1. Accessible Anywhere: Can be used on any device with an internet connection and a browser.
2. No Installation Needed: Runs directly from a URL.

3. Platform Independent: Works on different operating systems like Windows, macOS, Linux, or mobile platforms.
4. Centralized Updates: Updates are applied on the server; users automatically get the latest version.
5. Interactive: Supports dynamic user interfaces and real-time updates.

THEORY EXERCISE:

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

Advantages of Using Web Applications Over Desktop Applications

➢ Accessibility from Anywhere

- Web applications can be accessed from any device with an internet connection and browser.
- Users are not tied to a single device.

➢ No Installation Required

- Runs directly from a URL; users do not need to install or configure software.

➢ Platform Independence

- Works on multiple operating systems such as Windows, macOS, Linux, or mobile platforms without modification.

➢ Centralized Updates and Maintenance

- Updates are applied on the server; all users automatically get the latest version.
- Reduces the need for manual updates on individual machines.

➢ Collaboration and Real-Time Access

- Supports multiple users working together simultaneously.
- Enables real-time data sharing and collaboration.

➢ Reduced Cost and IT Management

- No need to maintain software on individual devices.
- Easier to manage and deploy for organizations.

28. Designing

Designing is the phase in software development where the requirements gathered during analysis are transformed into a blueprint for building the software. It defines how the software will work and how different components will interact.

Designing acts as a bridge between requirement analysis and implementation (coding).

Objectives of Designing

1. Plan the System Architecture – Decide the overall structure and organization of modules.
2. Define Data and Database Structures – Specify how data will be stored, accessed, and managed.
3. Create User Interfaces – Design user-friendly screens and interactions.
4. Specify Modules and Components – Define functionality, inputs, outputs, and interactions.
5. Ensure Quality Attributes – Consider performance, security, maintainability, and reliability.

➢ Types of Design

- High-Level Design (HLD)

- Provides an overview of the system architecture.
- Defines modules, their functions, and interactions.
- Example: Block diagrams, architecture diagrams.

- Low-Level Design (LLD)

- Details the design of each module.
- Specifies algorithms, data structures, and interfaces.
- Example: Flowcharts, pseudo-code, class diagrams.

THEORY EXERCISE:

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

  UI (User Interface) and UX (User Experience) design are critical in application development as they focus on how users interact with and perceive the software.

Key Roles of UI/UX Design

- Enhances Usability
- Ensures the application is easy to navigate and intuitive.
- Reduces learning curve for users.
- Improves User Satisfaction
- A well-designed interface makes the application pleasant and engaging to use.
- Encourages continued usage and loyalty.
- Increases Efficiency
- Streamlines workflows and reduces the number of steps needed to perform tasks.
- Helps users complete tasks faster and with fewer errors.
- Supports Accessibility
- Designs interfaces that are usable by people with varying abilities.
- Ensures inclusivity and wider reach.
- Boosts Application Success
- Attractive and functional UI/UX can differentiate an application in a competitive market.
- Leads to higher adoption and positive reviews.

- Guides Development Decisions

- Helps developers understand user needs and priorities.
- Influences functionality, layout, and navigation choices.

29. Mobile Application?

A mobile application (mobile app) is a software program designed to run on mobile devices such as smartphones and tablets. Mobile apps allow users to perform specific tasks, access information, or interact with services conveniently on-the-go.

A mobile application (mobile app) is a software program developed specifically for mobile devices such as smartphones, tablets, or wearable devices. Mobile apps allow users to perform tasks, access information, and interact with digital services conveniently anywhere and anytime.

THEORY EXERCISE:

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

Feature	Native Mobile Apps	Hybrid Mobile Apps
Definition	Developed for a specific platform (iOS, Android) using platform-specific languages (Swift, Kotlin, Java)	Developed using web technologies (HTML, CSS, JavaScript) and wrapped in a native container to run on multiple platforms

Feature	Native Mobile Apps	Hybrid Mobile Apps
Performance	High performance, faster, and smoother	Slightly slower due to web view dependency
Platform Dependency	Platform-specific; separate code for each platform	Cross-platform; single codebase for multiple platforms
Access to Device Features	Full access to device features (camera, GPS, sensors)	Limited or requires plugins to access device features
Development Cost	Higher; separate development for each platform	Lower; single codebase for multiple platforms
Updates and Maintenance	Updates need to be done separately for each platform	Updates can be done once and applied across platforms
Examples	WhatsApp (iOS & Android versions separately)	Instagram (partly hybrid), Gmail (web-based components)

30. DFD (Data Flow Diagram)

A Data Flow Diagram (DFD) is a graphical representation of the flow of data through a system. It shows how input data is transformed into output through processes, data stores, and interactions with external entities.

DFDs are widely used in system analysis and design to understand how data moves and is processed within a system.

Key Components of a DFD

➢ External Entities (Sources/Sinks)

- Represent people, organizations, or systems that interact with the system.
- Shown as rectangles.
- Example: Customer, Supplier, Bank.

➢ Processes

- Represent actions or operations that transform input data into output.
- Shown as circles or rounded rectangles.
- Example: Calculate Salary, Issue Book, Process Order.

➢ Data Stores

- Represent places where data is stored within the system.
- Shown as open-ended rectangles or parallel lines.
- Example: Employee Database, Book Inventory, Customer Records.

➢ Data Flows

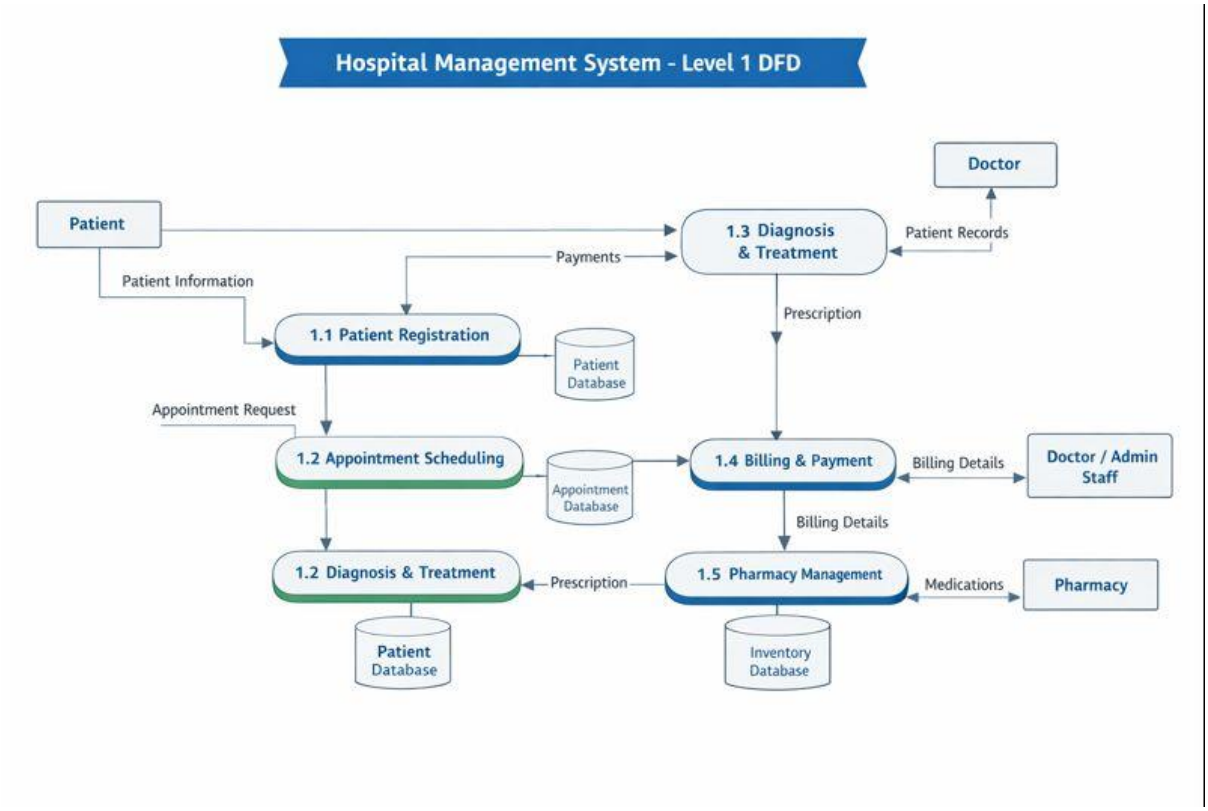
- Represent movement of data between entities, processes, and data stores.
- Shown as arrows.
- Example: Order Details, Payment Information, Customer Request.

LAB EXERCISE:

- Create a DFD for a hospital management system.

➢ Levels of DFD

- Level 0 (Context Diagram): Shows the whole system as a single process interacting with external entities.
- Level 1: Breaks the system into main sub-processes, showing major data stores.
- Level 2: Detailed view of sub-processes with finer data flows.



THEORY EXERCISE:

- What is the significance of DFDs in system analysis?

Significance of Data Flow Diagrams (DFDs) in System Analysis Data Flow Diagrams (DFDs) are essential tools in system analysis because they visually represent how data moves through a system, helping analysts, developers, and stakeholders understand and improve the system.

Key Significance

➢ Visualization of Data Flow

- DFDs show how data flows between processes, data stores, and external entities, making complex systems easier to understand.

➢ Clarifies System Requirements

- Helps identify what the system should do and how data should be processed, ensuring accurate requirement gathering.

➢ Identifies Redundancies and Bottlenecks

- Highlights unnecessary data movements, repeated processes, or inefficient workflows.

➢ Improves Communication

- Provides a common visual language for developers, analysts, and stakeholders to discuss the system clearly.

➢ Supports System Design

- Serves as a blueprint for designing databases, processes, and interfaces.
- Helps in creating efficient and well-structured systems.

➢ Facilitates Documentation

- Provides clear documentation of system processes and data flow for maintenance and future enhancements.

31. Desktop Application

A desktop application is a software program that is installed and runs locally on a personal computer or laptop, rather than being accessed through a web browser or over a network. Desktop applications are designed to take advantage of the computing power and resources of the local machine. Here's a detailed breakdown:

Key Features of Desktop Applications

1. Local Installation: Installed directly on the computer's hard drive.
2. Offline Access: Can often work without an internet connection.
3. High Performance: Can utilize system resources efficiently (CPU, GPU, RAM).
4. Rich User Interface: Can offer complex and responsive graphical interfaces.

5. Security: Data can be stored locally, giving more control over privacy.

LAB EXERCISE:

- Build a simple desktop calculator application using a GUI library?

Simple Desktop Calculator

1. Objective:
   Create a GUI-based desktop calculator that performs addition, subtraction, multiplication, and division.
2. Tools Used:
   - Python – programming language
   - Py library – GUI library (comes pre-installed with Python)
3. Steps to Build:
   - Step 1: Import Py library
   - Step 2: Create Main Window
   - Step 3: Add Entry Field
   - Step 4: Define Functions
   - Step 5: Add Buttons
   - Step 6: Run Application

32. What is flow chart?

A flowchart is a diagrammatic representation of a process or algorithm. It shows the step-by-step flow of a task using standard symbols connected by arrows. Flowcharts help us understand, plan, and analyze how a process works.

Purpose of a Flowchart

- To visualize a process clearly
- To plan programs before coding

- To identify errors or missing steps
- To explain logic easily to others

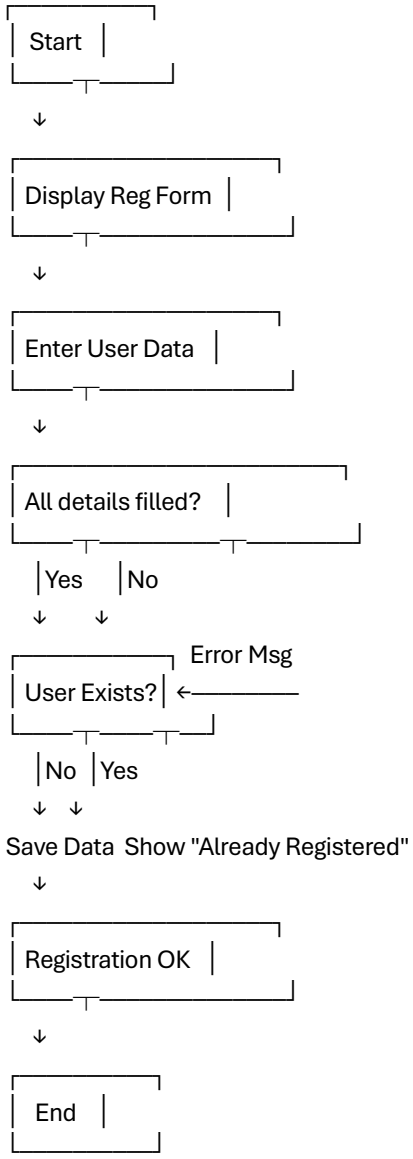
LAB EXERCISE:

- Draw a flowchart representing the logic of a basic online registration system.

  To draw a flowchart representing the logic of a basic online registration system.

Flowchart Logic (Step-by-Step)

1. Start
2. Display Registration Form
3. User Enters Details
   (Name, Email, Username, Password)
4. Decision: Are all details filled?
   - No → Show error message → Go back to form
   - Yes → Continue
5. Decision: Is username/email already registered?
   - Yes → Show "User already exists" → Go back to form
   - No → Continue
6. Save User Details in database
7. Display Registration Successful Message
8. End



THEORY EXERCISE:

- How do flowcharts help in programming and system design?

Flowcharts play an important role in programming and system design by providing a clear visual representation of the steps involved in a process or algorithm.

Firstly, flowcharts help programmers understand the logic of a problem before writing code. By breaking the problem into step-by-step actions, they make complex logic easier to analyze.

Secondly, in system design, flowcharts help in planning the structure of a system. Designers can see how different modules interact, where decisions are made, and how data flows through the system.
Thirdly, flowcharts make error detection easier. Logical mistakes, missing steps, or unnecessary processes can be identified early, reducing debugging time during implementation.
Additionally, flowcharts improve communication among team members. Since they use standard symbols, even non-technical stakeholders can understand the system workflow.
Finally, flowcharts serve as documentation. They act as a reference for future maintenance, updates, and training.

In summary, flowcharts simplify problem-solving, improve system planning, reduce errors, and enhance communication, making them a valuable tool in programming and system design.