Python – Full Stack Assignment

Module 1 – Overview of IT Industry

<u>What is a Program?</u>

<u>LAB EXERCISE: Write a simple "Hello World" program in two different programming languages of</u>

Program 1: Python

```python
# Hello World in Python

print("Hello World")
```

Program 2: HTML

```html
<!-- Hello World in HTML -->
<!DOCTYPE html>
<html>
<head>
   <title>Hello Page</title>
</head>
<body>
   <h1>Hello World</h1>
</body>
</html>
```

<u>THEORY EXERCISE: Explain in your own words what a program is and how it functions.</u>

<u>What is Programming?</u>

What is Programming?
A program is a set of instructions that tells a computer what to do.

<u>How does it function?</u>
The computer reads the instructions one by one and performs the task accordingly.

<u>What is Programming?</u>
Programming is the process of writing those instructions to solve problems using a computer.

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

Types of Programming Languages

1. Key Steps in the Programming Process

1. Problem Definition – Understand what needs to be solved.
2. Planning (Algorithm/Flowchart) – Design the logic.
3. Coding – Write the program using a programming language.
4. Testing – Check the program for errors (bugs).
5. Debugging – Fix the errors.
6. Execution – Run the program and check the output.
7. Documentation – Write details about how the program works.
8. Maintenance – Update and improve the program over time.

2. Types of Programming Languages

1. Low-Level Languages
    a. *Machine Language*
    b. *Assembly Language*
2. High-Level Languages
    a. Easy to read and write
    b. Examples: Python, Java, C, C++
3. Scripting Languages
    a. Used for automation or web tasks
    b. Examples: JavaScript, PHP, Python
4. Object-Orien**ted Languages**
    a. Focus on objects and classes
    b. Examples: Java, C++, Python
5. Functional Languages
    a. Based on mathematical functions
    b. Examples: Haskell, Lisp

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

World Wide Web & How Internet Works

1. Differences Between High-Level and Low-Level Programming Languages

Feature	High-Level Language	Low-Level Language

Ease of Use	Easy to read and write	Hard to understand
Syntax	Similar to human language	Closer to machine code
Examples	Python, Java, C++	Assembly, Machine Language
Portability	Works on different computers	Specific to hardware
Speed	Slower than low-level	Very fast and efficient
Abstraction	High abstraction	Low abstraction

2. World Wide Web (WWW)

The World Wide Web is a system of web pages connected through the internet. It uses web browsers (like Chrome) and protocols like HTTP to access websites.

- It was invented by Tim Berners-Lee in 1989.
- It includes websites, links, images, videos, and more.
- Every website has a unique URL (Uniform Resource Locator).

3. How the Internet Works

1. User Request – You type a URL in a browser.
2. DNS Lookup – Domain Name System converts URL to IP address.
3. Server Contacted – Your request goes to the website’s server.
4. Data Transfer – Server sends data (website files) back to you.
5. Browser Display – Your browser shows the website content.

LAB EXERCISE: Research and create a diagram of how data is transmitted from a client to a server

over the internet.

1. Client (like your browser) sends a request (e.g., open google.com).
2. DNS Server translates domain name to IP address.
3. Internet Routers forward the request to the correct server.
4. Web Server receives the request, processes it, and sends a response.

5. Client receives the response (like HTML page) and displays it.

Diagram:

plaintext

```
[Client Device]
    |
    v
[DNS Server]
    |
    v
[Internet Routers]
    |
    v
[Web Server]
    |
    v
[Response Back to Client]
```

You can draw this in your notebook or on a computer with arrows like this:

User Request ---> DNS ---> Internet ---> Web Server ---> Response to Client

Labels to include in your diagram:

- Client (browser or app)
- DNS Server
- Internet (routers/switches)
- Web Server
- Data Request & Response Flow

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

Network Layers on Client and Server

1. Roles of Client and Server in Web Communication

- Client:
  A client is a device or software (like a web browser) that requests data from a server.
  Example: Your browser requesting to open google.com.
- Server:
  A server is a computer that receives the request from the client and sends back the requested data.
  Example: Google's server sending back the webpage.

2. Network Layers on Client and Server (OSI Model)

Layer No.	Layer Name	Role (Simple)
7	Application Layer	User interactions (e.g., browser, HTTP)
6	Presentation Layer	Data formatting, encryption
5	Session Layer	Opens and closes connections
4	Transport Layer	Breaks data into packets (TCP/UDP)
3	Network Layer	Adds IP address for routing
2	Data Link Layer	Adds MAC address for local delivery
1	Physical Layer	Sends actual bits through wires or wireless

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.

Client and Servers

1. Function of the TCP/IP Model and Its Layers

TCP/IP (Transmission Control Protocol/Internet Protocol) is the basic communication model used on the internet. It defines how data is sent and received between devices.

Layers of TCP/IP Model:

Layer No.	Layer Name	Function
4	Application Layer	Provides services like HTTP, FTP, Email
3	Transport Layer	Ensures reliable data transfer (e.g., TCP, UDP)

2	Internet Layer	Handles addressing and routing (IP addresses)
1	Network Access Layer	Deals with hardware, cables, and physical transfer

Summary:
TCP/IP model helps break data into packets, send them over the internet, and reassemble them correctly at the destination.

2. Client and Servers

- **Client:**
  A client is a device or program that **requests data or services** from a server.
  Example: A web browser is a client.
- **Server:**
  A server is a system that **receives requests** and **responds with data or services**.
  Example: A website's server sends pages to your browser.

THEORY EXERCISE: Explain Client Server Communication

Types of Internet Connections

1. Client-Server Communication

Client-server communication is a model where:

- Client: Sends a request (like opening a website).
- Server: Receives the request, processes it, and sends a response back.

Example:
When you open google.com, your browser (client) sends a request to Google's server. The server replies with the webpage content.

2. Types of Internet Connections

Type	Description
Dial-Up	Uses telephone lines; very slow speed
DSL (Broadband)	Faster than dial-up; uses telephone lines
Cable	Uses TV cable; fast and reliable
Fiber Optic	Very high-speed using light signals

Wireless (Wi-Fi)	Wireless access via router; limited by range
Mobile (3G/4G/5G)	Uses cellular network; works on smartphones
Satellite	Used in remote areas; slower and costly

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

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

Protocols

1. How does Broadband differ from Fiber-Optic Internet?

Feature	Broadband	Fiber-Optic Internet
Speed	Normal speed	Very fast speed
Cables Used	Copper (metal) wires	Glass wires (fiber cables)
Quality	May get slow sometimes	Very stable and reliable
Cost	Cheaper	More costly
Use	Common in many places	Best for fast internet needs

2. Protocols

Protocols are rules that computers follow to share data over the internet.

Some Common Protocols:

- **HTTP** – Used to open websites.
- **HTTPS** – Like HTTP, but more secure.
- **FTP** – Used to send or receive files.
- **TCP/IP** – Helps send and receive data correctly.
- **SMTP** – Used to send emails.

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?

Application Security

1. Differences between HTTP and HTTPS Protocols

Feature	HTTP	HTTPS
Full Form	Hypertext Transfer Protocol	Hypertext Transfer Protocol Secure
Security	Not secure (data can be hacked)	Secure (data is encrypted)
Port	Uses port 80	Uses port 443
Speed	Slightly faster	Slightly slower (due to encryption)
Use	For normal websites	For secure websites (like banking)

In short: HTTPS is just like HTTP but with extra security.

2. Application Security

- **Meaning:**
  Application security is protecting apps (like websites, mobile apps) from hackers or data leaks.
- **How is it done?**
  - Using **passwords** and **authentication**
  - Using **HTTPS** for secure data transfer
  - Keeping software updated
  - Using **firewalls** and **antivirus**

LAB EXERCISE: Identify and explain three common application security vulnerabilities.

Suggest possible solutions.

THEORY EXERCISE: What is the role of encryption in securing applications?Software Applications and Its Types

1. Role of Encryption in Securing Applications

- Encryption means changing data into a secret code so that no one can read it without a key.
- It keeps sensitive information (like passwords, banking details) safe from hackers.

- Used in apps, websites (HTTPS), and messaging apps (like WhatsApp).
- Even if data is stolen, encryption makes it useless without the correct key.

In short: Encryption protects data and ensures privacy.

2. Software Applications and Its Types

Software Applications are programs made to perform specific tasks for users.
Example: MS Word, WhatsApp, Google Chrome.

Types of Software Applications:

1. Web Applications – Run on browsers. (Example: Gmail)
2. Mobile Applications – For smartphones. (Example: WhatsApp)
3. Desktop Applications – Installed on computers. (Example: MS Office)
4. Enterprise Applications – For business use. (Example: ERP, SAP)
5. Games and Multimedia Apps – For entertainment. (Example: PUBG, VLC)

LAB EXERCISE: Identify and classify 5 applications you use daily as either system software

or application software.

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

Software Architecture

1. Difference between System Software and Application Software

Feature	System Software	Application Software
Purpose	Runs and controls computer hardware	Helps the user do specific tasks
Examples	Windows, Linux, Mac OS	MS Word, WhatsApp, Chrome
Installation	Comes pre-installed with computer	Installed as per user needs
Interaction	Works in the background	Works directly for user tasks

In short:
System software runs the computer, while application software is used by users for tasks like writing, browsing, etc.

2. Software Architecture

- **Definition:**
  Software architecture is the **design and structure** of a software system.
  It defines how different **components of the software** work together.
- **Key Points:**
  - It shows the **overall framework** of the software.
  - Helps developers build **organized and scalable** programs.
  - Example: In a web app, the **frontend**, **backend**, and **database** are parts of the architecture.

<u>LAB EXERCISE: Design a basic three-tier software architecture diagram for a web application.</u>

<u>THEORY EXERCISE: What is the significance of modularity in software architecture?</u>

<u>Layers in Software Architecture</u>

1. Significance of Modularity in Software Architecture

- Modularity means dividing software into small, independent parts called modules.
- Each module performs a specific function, making the software easy to understand, test, and update.
- If one module has an error, other modules are not affected.
- It improves teamwork because different developers can work on different modules.

2. Layers in Software Architecture

Software architecture is usually divided into layers, where each layer has a specific role.

Common Layers:

1. Presentation Layer – The user interface (UI).
2. Application Layer – Handles logic and operations.
3. Business Layer – Contains rules and workflows.
4. Data Layer – Manages databases and storage.

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?

Software Environments

1. Why are Layers Important in Software Architecture?

- Layers separate the software into different levels (UI, logic, data).
- Each layer focuses on a specific job, making the software organized.
- It is easier to debug, test, and update each layer independently.
- Improves security because data and logic are managed separately.
- Makes the system scalable and easier to maintain.

2. Software Environments

- A **software environment** is the setup where software runs, including **hardware, operating system, and tools**.
- It helps developers test and run applications properly.

Types of Software Environments:

1. **Development Environment** – For writing and testing code (e.g., VS Code).
2. **Testing Environment** – For testing apps before release.
3. **Staging Environment** – Pre-release environment similar to production.
4. **Production Environment** – The final environment where users use the software.

LAB EXERCISE: Explore different types of software environments (development, testing,

production). Set up a basic environment in a virtual machine.

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

Source Code

Importance of a Development Environment in Software Production

- A **development environment** provides all tools (compiler, editor, debugger) needed to create software.
- It helps developers **write, test, and debug** code easily.
- Ensures that the software works correctly before moving to testing or production.
- Saves **time and effort** by providing an organized workspace.
- Examples: **Visual Studio, Eclipse, PyCharm**.

Source code : print("Hello World")

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

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

Github and Introductions

1. Difference between Source Code and Machine Code

Feature	Source Code	Machine Code
Definition	Human-readable code written by programmers	Code understood by computers (binary)
Language	Written in languages like C, Java, Python	Written in 0s and 1s (binary)

Editability	Easy to read and modify	Hard for humans to understand
Execution	Needs to be compiled or interpreted	Directly executed by the computer
Example	print ("Hello World")	01101000 01100101 01101100

2. GitHub and Introductions

- **GitHub:**
  GitHub is an online platform for storing, sharing, and managing **source code** using version control (Git).
  - Helps developers **collaborate** on projects.
  - Used to track changes in code and host open-source projects.
- **Introduction to GitHub:**
  - GitHub works with **repositories** (places where code is stored).
  - You can upload, edit, and share code with teams.
  - Example: Developers worldwide share projects like Linux on GitHub.

<u>LAB EXERCISE: Create a Github repository and document how to commit and push code changes.</u>

<u>THEORY EXERCISE: Why is version control important in software development?</u>

<u>Student Account in Github</u>

1. Why is Version Control Important in Software Development?

- Version control keeps track of changes made to code over time.
- It allows developers to undo mistakes by going back to an older version.
- Helps teamwork, as many developers can work on the same project without conflicts.
- Tools like Git and GitHub make version control easy and organized.
- Ensures that the project is safe and backed up.

2. Student Account in GitHub

- GitHub offers **free student accounts** with extra benefits.
- Students can access **private repositories** and free tools for coding projects.
- To create one:
    - Sign up on **github.com** with your student email.
    - Apply for **GitHub Student Developer Pack** by uploading your student ID.
    - Get access to free coding resources and offers.

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?

Types of Software

1. Benefits of Using GitHub for Students

1. Free Student Developer Pack with premium tools.
2. Create private repositories to store projects safely.
3. Learn version control (Git) and teamwork skills.
4. Explore open-source projects and contribute.
5. Build a portfolio to show coding skills to employers.

2. Types of Software

Software is divided into two main types:

1. **System Software** – Runs and controls computer hardware.
   Examples: Windows, Linux, macOS.
2. **Application Software** – Helps users perform tasks.
   Examples: MS Word, WhatsApp, Chrome.
3. **Utility Software** – Maintains and protects the system.
   Examples: Antivirus, Disk Cleanup.

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

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

GIT and GITHUB Training

1. Differences between Open-Source and Proprietary Software

- Open-Source Software:
  - Source code is free to view and modify.
  - Mostly free of cost.
  - Developed by a community of developers.
  - Examples: Linux, VLC Media Player.
- Proprietary Software:
  - Source code is hidden (closed).
  - Usually paid and requires a license.
  - Developed and controlled by companies.
  - Examples: Windows, MS Office.

2. GIT and GITHUB Training

- **GIT:**
  Git is a **version control system** used to track changes in code. It allows multiple developers to work together on the same project.
- **GITHUB:**
  GitHub is an **online platform** where you can store code, share projects, and collaborate using Git.

Training Importance:

- Helps students learn **version control**.
- Useful for **team projects** and open-source contributions.
- Builds a **portfolio** for jobs and internships.

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? Application Software

1. How does GIT improve collaboration in a software development team?

- Git allows multiple developers to work on the same project at the same time.
- It keeps track of all changes made to the code (version control).
- Developers can create branches to work on features without disturbing the main code.
- If mistakes happen, Git allows teams to revert to an older version.
- It helps in merging code from different team members smoothly.

In short: Git makes teamwork easy, organized, and error-free in coding projects.

2. Application Software

- **Definition:**
  Application software is a type of software designed to help users perform **specific tasks**.
- **Examples:** MS Word (document creation), WhatsApp (messaging), VLC (media player).
- **Types of Application Software:**
  - **Word Processing Software** – MS Word, Google Docs.
  - **Web Browsers** – Chrome, Firefox.
  - **Media Players** – VLC, Windows Media Player.
  - **Communication Apps** – Zoom, WhatsApp.
  - **Games** – PUBG, Chess Apps.

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?

Software Development Process

1. Role of Application Software in Businesses

- Helps in managing daily tasks like billing, accounting, and communication.
- Increases productivity by automating work (e.g., Excel for data).
- Improves communication with tools like Zoom or email apps.
- Stores and manages business data safely.
- Supports decision-making with reporting and analysis tools.

2. Software Development Process

The **software development process** is a step-by-step method to build software.

Main Steps:

1. **Requirement Analysis** – Understand what the user needs.
2. **Design** – Plan the software structure (flowcharts, diagrams).
3. **Coding** – Write the program using a programming language.
4. **Testing** – Find and fix errors (bugs).
5. **Deployment** – Release the software for users.
6. **Maintenance** – Update and improve the software after release.

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

THEORY EXERCISE: What are the main stages of the software development process?

Software Requirement

1. Main Stages of the Software Development Process

1. Requirement Analysis – Understand what the user wants.
2. Design – Plan the structure and look of the software.
3. Coding (Implementation) – Write the program using a programming language.
4. Testing – Check for errors (bugs) and fix them.
5. Deployment – Deliver the software to the user.
6. Maintenance – Update and improve the software over time.

2. Software Requirement

- **Definition:**
  Software requirement is a detailed description of **what a software should do**.
  It includes all the **features, functions, and goals** of the software.

- **Types of Software Requirements:**
    - **Functional Requirements** – What the software should do (e.g., login feature).
    - **Non-Functional Requirements** – Quality aspects like speed, security, or reliability.

<u>LAB EXERCISE: Write a requirement specification for a simple library management system.</u>

<u>THEORY EXERCISE: Why is the requirement analysis phase critical in software development? Software Analysis</u>

1. Why is the Requirement Analysis Phase Critical in Software Development?

- It clearly defines what the user needs from the software.
- Prevents errors, confusion, or rework during development.
- Saves time and cost by creating a proper plan.
- Ensures the final product matches user expectations.
- Acts as the base for design, coding, and testing.

2. Software Analysis

- **Definition:** Software analysis is the process of **studying and understanding requirements** to design the correct software solution.
- It identifies **problems, goals, and features** needed in the software.
- **Example:** Analysing what features a shopping app should have (login, cart, payment, etc.).

<u>LAB EXERCISE: Perform a functional analysis for an online shopping system.</u>

<u>THEORY EXERCISE: What is the role of software analysis in the development process?</u>

<u>System Design</u>

1. Role of Software Analysis in the Development Process

- Software analysis defines what the software should do.

- It helps in understanding user needs and system requirements.
- Acts as a blueprint for design and coding.
- Detects possible problems or missing features early.
- Ensures that the final software is useful and correct.

2. System Design

- **Definition:** System design is the process of **planning the structure and components** of a software or system.
- It decides **how the system will work**, including data flow, user interface, and modules.
- **Types of System Design:**
    - **High-Level Design (HLD):** Overview of the system (modules, architecture).
    - **Low-Level Design (LLD):** Detailed design of each part (algorithms, database).

<u>LAB EXERCISE: Design a basic system architecture for a food delivery app.</u>

<u>THEORY EXERCISE: What are the key elements of system design?</u>

<u>Software Testing</u>

1. Key Elements of System Design

1. Architecture Design – Overall structure of the system.
2. Data Design – How data will be stored and managed (databases).
3. Interface Design – User interface (UI) and interactions.
4. Module Design – Dividing the system into small, manageable parts.
5. Security Design – Protecting data and ensuring secure operations.

2. Software Testing

- **Definition:** Software testing is the process of **checking and finding errors (bugs)** in the software to ensure it works correctly.
- **Purpose:** To verify that the software meets requirements and is **reliable and error-free**.
- **Types of Testing:**
    - **Manual Testing** – Done by humans.
    - **Automation Testing** – Done using testing tools.

    - **Unit Testing** – Tests small parts (modules).
    - **System Testing –** Tests the whole system.

<u>LAB EXERCISE: Develop test cases for a simple calculator program.</u>

<u>THEORY EXERCISE: Why is software testing important?</u>

<u>Maintenance</u>

1. Why is Software Testing Important?

- Ensures the software is free from errors (bugs).
- Verifies that the software works as expected.
- Improves quality and reliability of the product.
- Saves time and cost by finding issues early.
- Increases user satisfaction by delivering a smooth product.

2. Maintenance

- **Definition:** Maintenance is the process of **updating and improving software** after it is delivered to the user.
- **Types of Maintenance:**
    - **Corrective:** Fixing bugs or errors.
    - **Adaptive:** Updating software for new environments (e.g., OS updates).
    - **Perfective:** Adding new features or improvements.
    - **Preventive:** Making changes to avoid future problems.

<u>LAB EXERCISE: Document a real-world case where a software application required critical maintenance.</u>

<u>THEORY EXERCISE: What types of software maintenance are there?</u>

<u>Development</u>

1. Types of Software Maintenance

1. Corrective Maintenance – Fixing bugs and errors found after release.

2. Adaptive Maintenance – Updating software to work with new hardware or operating systems.
3. Perfective Maintenance – Adding new features or improving performance.
4. Preventive Maintenance – Making changes to avoid future problems or failures.

2. Development

- **Definition:** Development is the phase where programmers **write the actual code** for the software.
- It follows the **design plan** and uses programming languages like C, Java, or Python.
- After development, the code is tested to ensure it works correctly.

<u>THEORY EXERCISE: What are the key differences between web and desktop applications?</u>

Feature	Web Applications	Desktop Applications
Access	Runs on web browsers (Chrome, Edge).	Installed and runs on a computer.
Internet	Needs internet connection.	Can work offline (no internet).
Updates	Updates happen automatically.	Must update manually.
Platform	Works on any device with a browser.	Works only on the installed system.
Examples	Gmail, Google Docs.	MS Word, VLC Player.

<u>27. Web Application</u>

<u>THEORY EXERCISE: What are the advantages of using web applications over desktop applications?</u>

1. **No Installation Needed** – Web apps run directly in a browser.
2. **Accessible Anywhere** – Can be used from any device with internet.
3. **Automatic Updates** – Always up to date, no manual update needed.
4. **Cross-Platform** – Works on Windows, Mac, mobile, etc.
5. **Cost-Effective** – No need for heavy hardware or software installation.
6. **Easy Collaboration** – Multiple users can work together online (e.g., Google Docs).

28. Designing

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

- UI (User Interface) focuses on how the application looks (colors, buttons, layout).
- UX (User Experience) focuses on how the application feels (easy to use, smooth navigation).
- Good UI/UX design makes the app simple, attractive, and user-friendly.
- It improves user satisfaction and encourages people to use the app.
- A well-designed UI/UX reduces errors and makes tasks faster.

29. Mobile Application

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

Feature	Native Apps	Hybrid Apps
Platform	Built for a specific OS (Android or iOS).	Works on multiple platforms.
Performance	Faster and smoother.	Slightly slower than native.
Development	Uses platform-specific languages (Java, Swift).	Uses web technologies (HTML, CSS, JS).
Cost	More expensive to develop.	Cheaper (single code for all).
Examples	WhatsApp, Instagram.	Gmail, Uber.

30. DFD (Data Flow Diagram)

- Definition:
  A DFD is a visual diagram that shows how data flows through a system.
- It describes inputs, processes, and outputs in a simple way.
- Components of DFD:
  - Processes – Show how data is processed.
  - Data Stores – Where data is stored.
  - Data Flows – Show movement of data.
  - External Entities – Sources or destinations of data.

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

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

- DFD (Data Flow Diagram) helps to visualize how data moves within a system.
- It makes complex systems easy to understand using simple diagrams.
- Shows the flow of information between processes, databases, and users.
- Helps in finding problems or missing steps in a system.
- Useful for planning, designing, and documenting software 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?

Pros of Desktop Applications:

1. Work Offline – No internet required.
2. Better Performance – Usually faster and smoother.
3. More Features – Can use full hardware power.
4. Security – Data is stored locally, less exposed to online threats.

Cons of Desktop Applications:

1. **Installation Needed** – Must be installed on each device.
2. **Updates Are Manual** – User has to download updates.
3. **Limited Access** – Can only be used on the installed device.
4. **Platform Dependent** – Separate versions for Windows, Mac, etc.

32. Flow Chart

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

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

- Flowcharts show the step-by-step flow of a program or process using symbols.
- They make complex logic easy to understand.
- Help programmers in planning and designing the structure before coding.
- Useful for debugging and finding errors in logic.
- Provide clear documentation for future reference.