

## Lab Exercise

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

**Ans-**

1. Python Program:

```
# Hello World in Python  
Print ("Hello, World!")
```

2. C Program:

```
// Hello World in C  
#include <stdio.h>  
  
int main () {  
    printf("Hello, World!");  
    return 0;  
}
```

Feature	Python	C
Syntax Simplicity structured	Very Simple	More
Output Statement printf("Hello, World!");	print ("Hello, World!")	
Main Function Required	No	Yes
Semicolon Needed	No	Yes

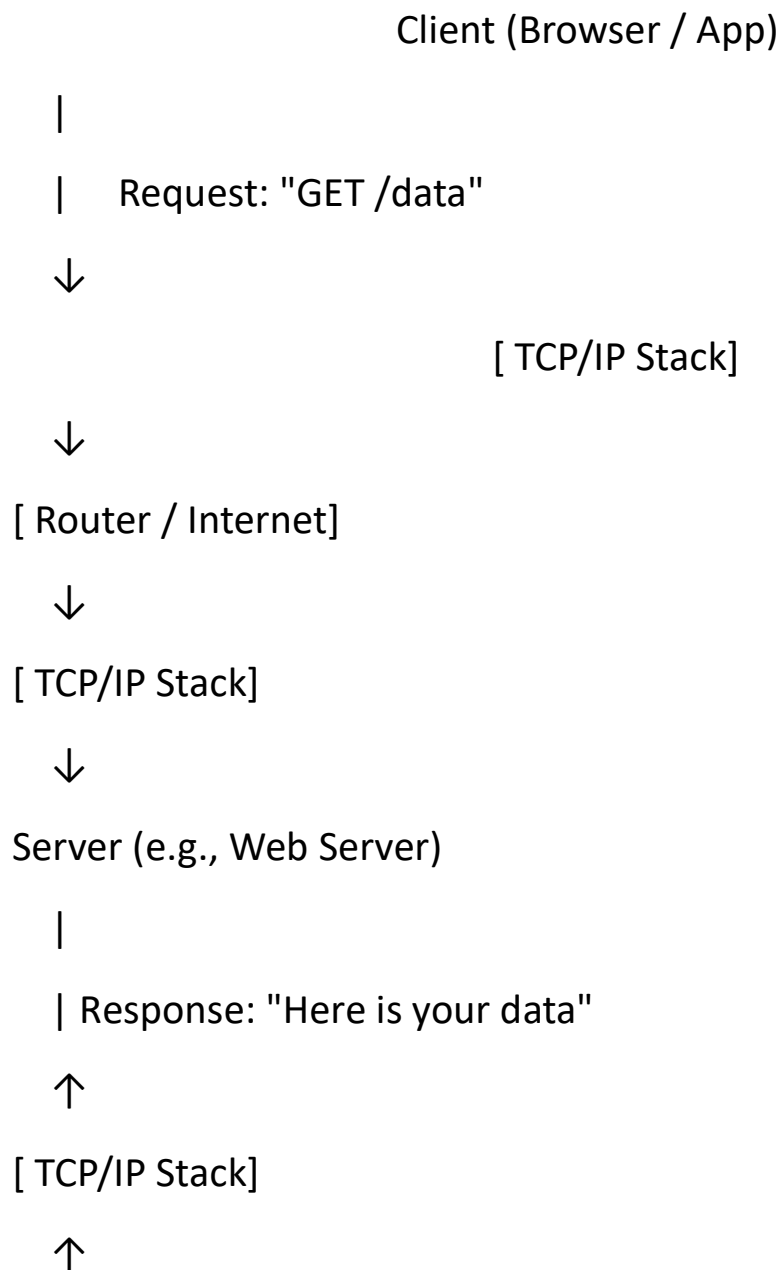
Header Files  
(#include <stdio.h>)

Not required

Required

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

**Ans-**



[ Router / Internet]



[ TCP/IP Stack]



Client receives data

### **3) Design a simple HTTP client-server communication in any language.**

**Ans-** Here's a simple HTTP client-server example using Python with its built-in http. Server (for the server) and requests library (for the client).

Server (Python)

```
from http. Server import BaseHTTPRequestHandler, HTTPServer
```

```
class SimpleHandler (BaseHTTPRequestHandler):
```

```
def do_GET(self):
```

```
self.send_response(200)
```

```
self.send_header("Content-type", "text/plain")
```

```
self.end_headers ()
```

```
self.wfile. write ("Hello from the server!")
```

```
# Start the server
```

```
if __name__ == "__main__":
```

```
server_address = ("localhost", 8000)
```

```
httpd = HTTPServer(server_address, SimpleHandler)
```

```
print ("Server running at http://localhost:8000")
```

```
httpd. serve_forever ()
```

Client (Python)

```
import requests
```

```
response = requests. Get("http://localhost:8000")
```

```
print ("Server says:", response. Text)
```

How it Works

1. Start the server script first.
2. The client sends a GET request to http://localhost:8000.
3. The server responds with "Hello from the server!".

**4) Research different types of internet connections (e.g., broadband, fibre, satellite) and list their pros and cons.**

**Ans-**

#### **1. Digital Subscriber Line (DSL)**

**Pros:**

- Widely available
- Allows internet and phone use at the same time

- Affordable for basic users

**Cons:**

- Speed depends on distance from service provider
  - Slower compared to modern options like fiber
- 

## **2. Cable Internet**

**Pros:**

- Faster than DSL
- Suitable for streaming and gaming
- Uses existing TV cable lines

**Cons:**

- Shared bandwidth can cause speed drops during peak hours
  - Limited availability in rural areas
- 

## **3. Fiber Optic**

**Pros:**

- Very high speed (up to 1 Gbps or more)
- Low latency and highly reliable
- Great for heavy users (streaming, gaming, work-from-home)

**Cons:**

- Limited availability in some regions

- Installation may be expensive
- 

#### **4. Satellite Internet**

**Pros:**

- Available in remote and rural areas
- Doesn't require cable or phone lines

**Cons:**

- High latency (delay), not good for gaming or video calls
  - Weather can affect signal quality
  - Data caps and slower speeds
- 

#### **5. Wireless Internet (Mobile Data / Wi-Fi)**

**Pros:**

- Convenient and portable
- Easy to set up
- Useful for smartphones and hotspots

**Cons:**

- Speed and reliability depend on signal strength
  - May have data limits or be costly
- 

#### **6. Broadband over Power Lines (BPL)**

**Pros:**

- Uses existing electrical infrastructure
- Easy access where other services are unavailable

**Cons:**

- Not widely available
- Interference issues can occur

**5) Simulate HTTP and FTP requests using command line tools (e.g., curl).****Ans-****1. Simulating an HTTP Request Using curl**

Command:

```
curl http://example.com
```

Explanation:

- This command sends an HTTP GET request to the server at example.com.
- The server responds with the HTML content of the page.
- Useful for testing websites or APIs.

**2. Simulating an FTP Request Using curl**

Command (to download a file):

```
curl ftp://ftp.example.com/file.txt --user username:
password
```

Explanation:

- Connects to an FTP server.
- Logs in with provided username and password.
- Downloads the file file.txt from the FTP server.

**6) Identify and explain three common application security vulnerabilities.**

**Suggest possible solutions.**

**Ans-**

### **1. SQL Injection**

- Problem: Hacker tricks the app to get into the database.
- Fix: Check and clean user input.

### **2. XSS (Cross-Site Scripting)**

- Problem: Hacker puts bad code in a website that runs on other people's screens.
- Fix: Don't show user input directly. Clean it first.

### **3. Weak Login System**

- Problem: Easy passwords or no security checks.
- Fix: Use strong passwords and add OTP or 2-step login.



**7) Identify and classify 5 applications you use daily as either system software**

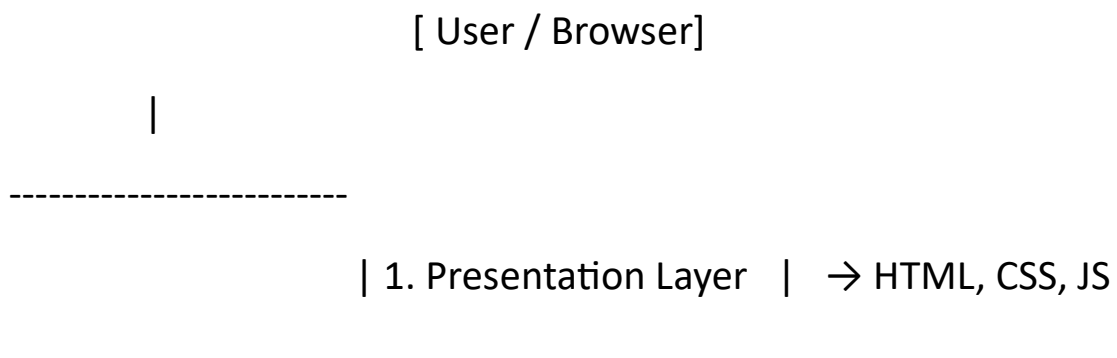
**Or application software.**

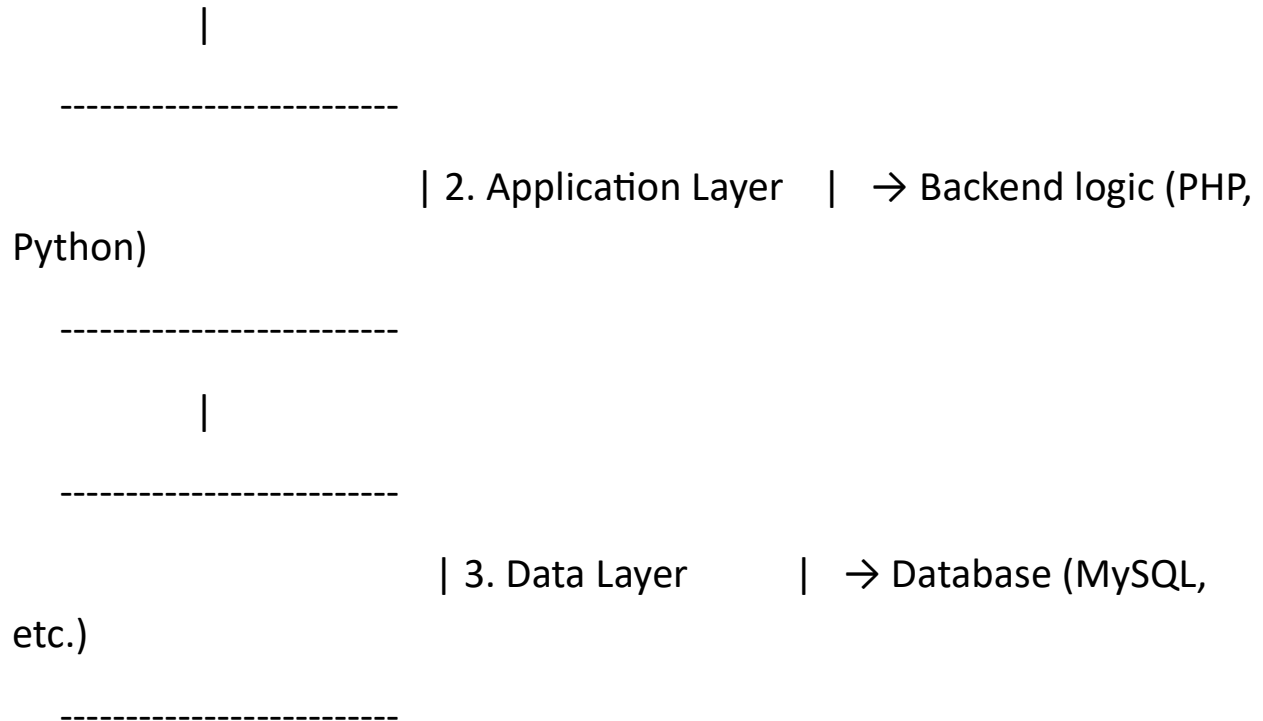
**Ans-**

- Google Chrome – Application Software
- Microsoft Word – Application Software
- Windows 10 – System Software
- VLC Media Player – Application Software
- Antivirus (like Quick Heal) – System Software

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

**Ans-**





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

**Ans-**

### **1. Presentation Layer (Frontend / UI)**

Role: This is what the user interacts with.

- User browses restaurants and food items
- Adds food to cart
- Enters delivery details
- Makes payment

Technologies Used:

HTML, CSS, JavaScript, React, Flutter (for mobile)

## **2. Business Logic Layer (Application Layer)**

Role: This handles all decision-making and rules.

- Processes order and verify payment
- Applies discounts and taxes
- Matches user with nearby delivery agents
- Calculates estimated delivery time

Technologies Used:

Node.js, Java, PHP, Python

## **3. Data Access Layer (Database Layer)**

Role: Deals with storing and retrieving data.

- Saves user profiles, orders, and payment info
- Fetches list of restaurants and menus
- Tracks real-time delivery status
- Stores feedback and reviews

Technologies Used:

MySQL, MongoDB, PostgreSQL

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

**Ans-**

Types of Software Environments:

**1. Development Environment**

- o Used by developers to write and build code
- o Contains IDEs, compilers, and debugging tools
- o Example: VS Code, Python, XAMPP

**2. Testing Environment**

- o Used by QA (testers) to test features
- o Isolated from development and production
- o Includes tools for automated/manual testing
- o Example: Selenium, Postman, JUnit

**3. Production Environment**

- o The live environment where real users access the application
- o Must be stable, secure, and monitored
- o Example: Hosted web server (Apache, Nginx), Cloud (AWS, Azure)

---

**Basic Virtual Machine Setup (Example using VirtualBox):**

1. Install VirtualBox or VMware
2. Create a new virtual machine
  - o Choose OS (e.g., Ubuntu or Windows)
  - o Allocate RAM and disk space

3. Install a development stack
  - o Example for web development:
    - ? Install Apache, MySQL, PHP (or use XAMPP)
    - ? Install code editor (e.g., VS Code)
4. Test a basic web page or script
  - o Create a hello.php file
  - o Run it in the browser from localhost

## **11) Write and upload your first source code file to Github..**

**Ans-**

### **1. Write a Simple Code File**

Create a simple file named hello.py:

```
# hello.py  
print("Hello, GitHub!")
```

### **2. Create a Repository on GitHub**

- Go to <https://github.com>
- Click New Repository
- Name it (e.g., first-code)

- Add a description (optional)
- Choose Public
- Click Create repository

### **3. Upload the Code Using Git (Command Line)**

Open terminal or Git Bash:

```
git init
```

```
git add hello.py
```

```
git commit -m "Add hello.py"
```

```
git branch -M main
```

```
git remote add origin https://github.com/your-username/first-code.git
```

```
git push -u origin main
```

**12) Create a Github repository and document how to commit and push code changes.**

**Ans-**

#### **Step 1: Create a GitHub Repository**

1. Go to <https://github.com>
2. Click on “New” to create a new repository
3. Enter a repository name (e.g., my-first-repo)
4. (Optional) Add a description

5. Choose Public or Private
6. Click Create repository

---

## Step 2: Prepare Your Project Locally

Create a folder and add a file (e.g., main.py):

python

Copy code

```
# main.py
```

```
print("This is my first commit!")
```

---

## Step 3: Use Git to Commit and Push Code

Open Git Bash or Terminal, then run:

bash

Copy code

```
git init          # Initialize Git in the folder
```

```
git add .         # Stage all files
```

```
git commit -m "Initial commit" # Commit changes with a message
```

```
git branch -M main      # Rename default branch to main
```

```
git remote add origin https://github.com/your-username/my-first-repo.git
```

```
git push -u origin main  # Push changes to GitHub
```

Replace your-username with your actual GitHub username.

**13) Create a student account on GitHub and collaborate on a small project with a classmate**

**Ans=**

**Objective: -**

To understand version control using GitHub and practice real-time collaboration on a basic project.

---

**Tasks to Perform: -**

1. Create a GitHub account by visiting <https://github.com>.
  2. Set up your profile with your real name and profile photo.
  3. Create a new repository named collab-project.
  4. Add a README.md file describing the project.
  5. Invite your classmate as a collaborator via repository settings.
  6. Both team members should commit at least one file each.
  7. Explore features like:
    - o Issues
    - o Pull requests
    - o Commit history
-



**Tools Required: -**

- GitHub account
- Web browser
- Basic internet connection

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

**Ans—**

Title: Create a list of software you use regularly and classify them into the following categories: System, Application, and Utility Software

---

**Objective: -**

To recognize and categorize commonly used software based on their function and purpose within a computing environment.

---

**Tasks to Perform**

1. Identify 9–12 software applications you use regularly on your computer or smartphone.
2. Organize the software into one of the three types:
  - o System Software
  - o Application Software
  - o Utility Software

3. Present the information in a table format with proper headings.

---

#### Table Format

Software Name	Category	Description / Purpose
---------------	----------	-----------------------

Windows 11	System Software	Manages hardware and provides system interface
------------	-----------------	--

Android OS	System Software	Mobile operating system
------------	-----------------	-------------------------

MS Word	Application Software	Word processing and document creation
---------	----------------------	---------------------------------------

Google Chrome	Application Software	Internet browsing
---------------	----------------------	-------------------

VLC Media Player	Application Software	Playing audio and video files
------------------	----------------------	-------------------------------

Tally ERP	Application Software	Accounting and financial management
-----------	----------------------	-------------------------------------

WinRAR	Utility Software	Compressing and extracting files
--------	------------------	----------------------------------

CCleaner	Utility Software	Cleaning junk files and optimizing performance
----------	------------------	--

Antivirus (e.g., Avast)	Utility Software	Protecting against malware and viruses
-------------------------	------------------	--

---

#### Tools Required: -

- Access to your device's installed software list
- Pen-paper or text editor for writing

- Internet (optional, for researching unfamiliar software)

**15) Follow a GIT tutorial to practice cloning, branching, and merging repositories.**

**Ans—**

**Objective: -**

To understand and apply the basic operations of Git for version control, including cloning a repository, creating branches, and merging code.

---

**Tasks to Perform: -**

**1. Cloning a Repository**

- o Use git clone to download a remote repository to your local machine.

**o Example:**

bash

code

```
git clone https://github.com/username/repository-name.git
```

**2. Creating a Branch**

- o Create a new branch to add features without affecting the main code.

**o Example:**

bash

code

git checkout -b feature-branch

### **3. Making Changes**

- o Edit files, commit the changes using git commit, and push to the new branch.

### **4. Merging Branches**

- o Switch to the main branch and merge the feature branch into it.
- o Example:

bash

code--

git checkout main

git merge feature-branch

### **5. Resolve Merge Conflicts (if any)**

- o Practice conflict resolution when Git highlights file conflicts.

---

#### **Tools Required: -**

- Git installed on your computer
- GitHub account with a repository
- Command-line interface or Git GUI (like Git Bash, GitHub Desktop)

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

**Ans—**

Title: Write a report on the various types of application software and how they improve productivity

---

**Objective: -**

To explore different types of application software and understand how they assist users in completing tasks more efficiently and effectively.

---

**Tasks to Perform: -**

1. Research and list different categories of application software.
  2. Provide examples of each category.
  3. Write a brief report explaining how each type improves user or organizational productivity.
- 

**Suggested Report Structure: -**

**1. Word Processing Software**

- Example: Microsoft Word, Google Docs
- Productivity Impact: Helps create, edit, format, and print text documents quickly and professionally.

**2. Spreadsheet Software**

- Example: Microsoft Excel, Google Sheets
- Productivity Impact: Allows data analysis, calculations, chart generation, and financial modeling.

### **3. Presentation Software**

- Example: PowerPoint, Canva
- Productivity Impact: Enables professionals to communicate ideas effectively with visual support.

### **4. Database Management Software (DBMS)**

- Example: Microsoft Access, MySQL
- Productivity Impact: Organizes and retrieves structured data efficiently, saving time and effort.

### **5. Multimedia Software**

- Example: Adobe Photoshop, VLC Media Player
- Productivity Impact: Facilitates content creation, editing, and consumption (videos, graphics, audio).

### **6. Web Browsers**

- Example: Google Chrome, Mozilla Firefox
- Productivity Impact: Provides access to information, tools, and web applications instantly.

### **7. Communication Software**

- Example: Zoom, Microsoft Teams, Slack
- Productivity Impact: Enables instant messaging, video conferencing, and team collaboration.

---

**Tools Required: -**

- Word processor (MS Word, Google Docs)
- Internet access for research

### **17) Create a flowchart representing the Software Development Life Cycle (SDLC).**

**Ans—**

#### **Objective: -**

To visually represent the phases of the Software Development Life Cycle (SDLC) using a standard flowchart and understand the role of each phase in structured software development.

---

#### **Tasks to Perform: -**

1. Study the six basic phases of SDLC.
  2. Use a diagram tool (e.g., Draw.io, Lucidchart, MS Visio, or even pen-paper) to design a flowchart.
  3. Ensure each SDLC phase is shown with correct flow and relationships.
- 

#### **Phases of SDLC to Include: -**

- 1. Requirement Analysis
- 2. System Design
- 3. Implementation (Coding)
- 4. Testing

- 5. Deployment
- 6. Maintenance

---

### **Sample Flowchart Structure: -**

sharp

Copy code

[Start]



[Requirement Analysis]



[System Design]



[Implementation]



[Testing]



[Deployment]



[Maintenance]



[End]



You can also include decision points (e.g., after testing: "Is software bug-free?" → Yes → Deploy / No → Return to Coding)

---

**Tools Required: -**

- Draw.io / Lucidchart / Pen-paper
- Internet (for SDLC references)

**18) Write a requirement specification for a simple library management system**

**Ans—**

Title: Write a requirement specification for a simple Library Management System

---

**Objective: -**

To prepare a clear and concise Software Requirement Specification (SRS) document that outlines the functional and non-functional requirements of a Library Management System.

---

**Tasks to Perform: -**

1. Define the purpose and scope of the system.
2. Identify the functional requirements (what the system should do).
3. Identify the non-functional requirements (system qualities like performance, security).
4. Present the specification in standard SRS format.

---

## **Sample Requirement Specification Document: -**

### **1. Introduction: -**

- **Purpose:**

To manage books, members, and borrowing activities in a digital format.

- **Scope:**

The system will allow librarians to add/remove books, register members, issue/return books, and generate reports.

### **2. Functional Requirements: -**

- The system shall allow the librarian to:
  - o Add, delete, and update book records.
  - o Register and manage members.
  - o Issue books to members.
  - o Return books from members.
  - o Generate overdue fine reports.
- The system shall display:
  - o Available and borrowed books.
  - o Member transaction history.
  - o Due date alerts.

### **3. Non-Functional Requirements: -**

- **Usability:** User-friendly UI for easy navigation.

- **Reliability:** System should handle simultaneous users and maintain data consistency.
- **Security:** Login credentials required for librarian and staff access.
- **Performance:** The system should perform all operations within 2 seconds.
- **Portability:** Should work on web browsers and desktop platforms.

#### **4. Assumptions: -**

- Users have basic computer literacy.
  - Database is regularly backed up.
- 

#### **Tools Required: -**

- Word processor (e.g., MS Word, Google Docs)
  - Internet (optional, for reference templates)
- 

#### **Learning Outcome**

After completing this lab, students will:

- Understand how to define functional and non-functional requirements
- Gain practice in writing technical documents
- Learn how proper specification prevents software development errors

**19) Perform a functional analysis for an online shopping system.**

**Ans—**

**Objective: -**

To identify and analyze the core functional components of an online shopping system and understand how each contributes to the overall system behavior.

---

**Tasks to Perform: -**

1. Identify key user roles and system actors (e.g., Customer, Admin).
  2. List core functional requirements and explain their purpose.
  3. Draw a functional block diagram (optional) for better understanding.
- 

**Functional Requirements of Online Shopping System**

**1. User Registration & Login: -**

- Users must be able to register and securely log in.
- Forgot password and user authentication features included.

**2. Product Browsing and Search: -**

- Users can browse by category, search for products using keywords, and filter results.

**3. Shopping Cart: -**

- Users can add/remove products, view totals, and update quantities.

#### **4. Checkout and Payment: -**

- System calculates total price with taxes and shipping.
- Supports payment gateways like UPI, Credit/Debit Cards, Net Banking.

#### **5. Order Management: -**

- Users can view order history, current status (shipped, delivered), and cancel orders.

#### **6. Admin Functionalities: -**

- Add/update/delete product listings
- Manage inventory, users, and process orders

#### **7. Feedback and Reviews: -**

- Customers can leave product ratings and reviews.

---

#### **Optional Functional Block Diagram: -**

A diagram showing the flow between:

User → Product Search → Cart → Checkout → Payment → Order Confirmation

---

#### **Tools Required: -**

- Word processor
  - Diagramming tool (for functional block diagram, optional)
-

**Learning Outcome: -**

**After completing this lab, students will be able to: -**

- Identify key functionalities of real-world systems
- Perform structured analysis of a complex application
- Understand how to break down large systems into manageable features

## **20) Design a basic system architecture for a food delivery app**

**Ans—**

**Objective: -**

To design a simple yet complete system architecture for a food delivery application, showing how different components interact in real-time to fulfill user requests.

---

**Tasks to Perform: -**

1. Identify the main system components and user roles.
2. Design a basic architecture diagram.
3. Describe the role of each component and how data flows through the system.

---

**Architecture Components: -**

**1. Frontend (User Interface): -**

- Customer App: Browse restaurants, place orders, track delivery.

- Restaurant Panel: Accept/prepare orders, update status.
- Delivery App: Accept delivery tasks, update real-time location.

## **2. Backend (Application Server): -**

- **Handles:**
  - o Order placement logic
  - o Authentication and user data
  - o Payment integration
  - o Notification system (push/SMS/email)
  - o Order status updates

## **3. Database Layer: -**

### **Stores:**

- User data (login, address, orders)
- Restaurant menus and availability
- Payment history and reviews
- Delivery logs

## **4. Payment Gateway API: -**

- Securely processes transactions via UPI, cards, wallets, etc.

## **5. Real-Time Tracking System: -**

- Uses GPS and mapping APIs (e.g., Google Maps)
- Tracks delivery location
- Shows ETA to customers

## 6. Notification System: -

- Sends order confirmations, delivery status, offers, etc.

---

### Sample Architecture Diagram (Text Representation)

Copy code

[Customer App] —> [Backend Server] <— [Restaurant Panel]



[Payment Gateway] [Database Layer]



[Real-time GPS] <— [Delivery App]

---

### Tools Required: -

- Drawing tool (Draw.io / Lucid chart / MS PowerPoint)
- Word processor for documentation

---

### Learning Outcome: -

After completing this lab, students will:

- Understand the structure of multi-user, real-time systems
- Be able to create and explain a basic system architecture



- Recognize the importance of APIs, data storage, and user interfaces in modern apps

## **21) Develop test cases for a simple calculator program**

**Ans—**

### **Objective: -**

To create structured test cases for a simple calculator program that performs basic arithmetic operations: addition, subtraction, multiplication, and division.

---

### **Tasks to Perform: -**

1. Identify the calculator functions to be tested.
  2. Define input values, expected output, and conditions.
  3. Organize test cases into a test case table.
- 

### **Calculator Functionalities to Test: -**

- Addition (+)
- Subtraction (-)
- Multiplication (\*)
- Division (/)
- Handling of invalid inputs
- Division by zero

---

**Tools Required: -**

- Calculator Program (Python/C/Java/Any Language)
  - Word processor or spreadsheet software to document test cases
- 

**Learning Outcome: -**

After completing this lab, students will:

- Understand the importance of test cases in software quality assurance
- Be able to write effective test cases for simple programs
- Learn how to validate correct and incorrect input handling

**22) Document a real-world case where a software application required critical maintenance**

**Ans—**

**Objective: -**

To understand the significance of software maintenance by analyzing a real-world case in which a software application required urgent or critical fixes due to bugs, performance issues, or changing requirements.

---

**Tasks to Perform: -**

1. Research a known software maintenance case.
  2. Describe the problem, its cause, and the maintenance performed.
  3. Summarize the outcome and lessons learned.
- 

### **1. Background: -**

WhatsApp, the popular messaging application owned by Meta, faced a global outage on 25th October 2022. Users were unable to send or receive messages for over two hours.

### **2. Problem Description: -**

- Messages were stuck on the “clock” icon.
- Groups and private chats were unresponsive.
- Web version also failed to connect.
- The issue impacted millions of users worldwide.

### **3. Cause: -**

- Internal server configuration changes triggered a major communication breakdown between WhatsApp servers.
- Load balancing failed due to improper update deployment.

### **4. Maintenance Actions Taken: -**

- The engineering team rolled back the latest deployment.
- Reconfigured server communication modules.
- Conducted an emergency round of system health checks and network traffic balancing.

## **5. Outcome: -**

- Services were gradually restored within 2.5 hours.
  - Meta issued a public apology and promised enhanced monitoring.
  - Internal deployment processes were revised to include stricter testing phases.
- 

## **Tools Required: -**

- Internet connection for research
  - Word processor for report writing
- 

## **Learning Outcome: -**

After completing this lab, students will:

- Gain awareness of real-world maintenance challenges
- Understand how maintenance impacts users and business reputation
- Learn best practices in error recovery and rollback strategy

## **23) Create a DFD for a hospital management system**

**Ans—**

## **Objective: -**

To understand and visualize the flow of data within a Hospital Management System (HMS) by creating a Level 0 and Level 1 DFD that includes key entities, processes, and data stores.

---

**Tasks to Perform: -**

1. Identify key processes and external entities in the hospital system.
  2. Create a Level 0 DFD (Context Diagram).
  3. Expand into a Level 1 DFD showing detailed interactions.
- 

**Level 0 DFD (Context Diagram): -**

External Entities:

- Patient
- Doctor
- Receptionist
- Admin

**Processes: -**

- Hospital Management System

**Data Flows: -**

- Patient provides registration details
- Doctor provides diagnosis
- Receptionist schedules appointments
- Admin manages records

Code---

[Patient] → (HMS) ← [Doctor]

[Receptionist] → (Hospital Management System) ← [Admin]

---

### **Level 1 DFD (Detailed Process Breakdown): -**

Processes:

1. Patient Registration
2. Appointment Scheduling
3. Medical Diagnosis
4. Billing and Discharge
5. Report Generation

### **Data Stores: -**

- Patient Records
- Appointment Database
- Billing Info
- Medical History

### **Example Flow: -**

code--

[Patient] → (1. Patient Registration) → [Patient Records]

[Receptionist] → (2. Appointment Scheduling) → [Appointment DB]

[Doctor] → (3. Medical Diagnosis) ↔ [Medical History]

(HMS) → (4. Billing & Discharge) → [Billing Info]

---

**Tools Required: -**

- Diagram tool (Draw.io / Lucid chart / Paper sketch)
  - Word processor for documentation
- 

**Learning Outcome: -**

After completing this lab, students will:

- Understand the structure of DFDs and how to read/create them
- Learn to break down a real-world system into logical data processes
- Be able to model data flow for complex systems like healthcare software

**24) Build a simple desktop calculator application using a GUI library**

**Ans—**

**Objective: -**

To design and develop a desktop calculator with basic arithmetic functionality (Addition, Subtraction, Multiplication, Division) using a Graphical User Interface (GUI) library such as Tkinter (Python), JavaFX (Java), or WinForms (C#).

---

**Tasks to Perform: -**

1. Design a calculator GUI with buttons for digits (0-9), operations (+, −, ×, ÷), clear, and equals.
  2. Implement logic to handle button clicks and perform operations.
  3. Display results and handle invalid inputs (e.g., division by zero).
- 

#### **Suggested Tech Stack: -**

- Language: Python (Recommended)
  - GUI Library: Tkinter
- 

#### **Design Notes: -**

- Use frames to organize buttons into rows
  - Validate inputs and handle edge cases
  - UI should be responsive and user-friendly
- 

#### **Tools Required: -**

- Python 3.x
  - Tkinter (comes built-in with Python)
  - Code editor (VS Code / PyCharm / IDLE)
- 

#### **Learning Outcome: -**

After completing this lab, students will:



- Understand GUI event handling and layout design
- Be able to create interactive desktop apps
- Learn how to integrate logic with GUI controls

**25) Draw a flowchart representing the logic of a basic online registration system.**

**Ans—**

**Objective: -**

To understand the logical flow of user interactions in an online registration system and visualize the process using a flowchart diagram.

---

**Tasks to Perform: -**

1. Identify the sequence of steps a user follows in an online registration form.
2. Define decision points such as validation and duplication check.
3. Draw a flowchart using standard flowchart symbols.

---

**Flowchart Logic Description: -**

1. Start
2. Display Registration Form
3. User Inputs Details
4. Validate Required Fields

- o If Invalid → Show Error → Go to Step 3
  - o If Valid → Proceed
  - 5. Check If User Already Exists
    - o If Yes → Show "User Exists" Message → End
    - o If No → Proceed
  - 6. Store User Data in Database
  - 7. Show Registration Success Message
  - 8. End
- 

### **Flowchart (Text Representation): -**

code

[Start]



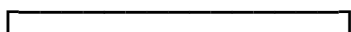
[Display Registration Form]



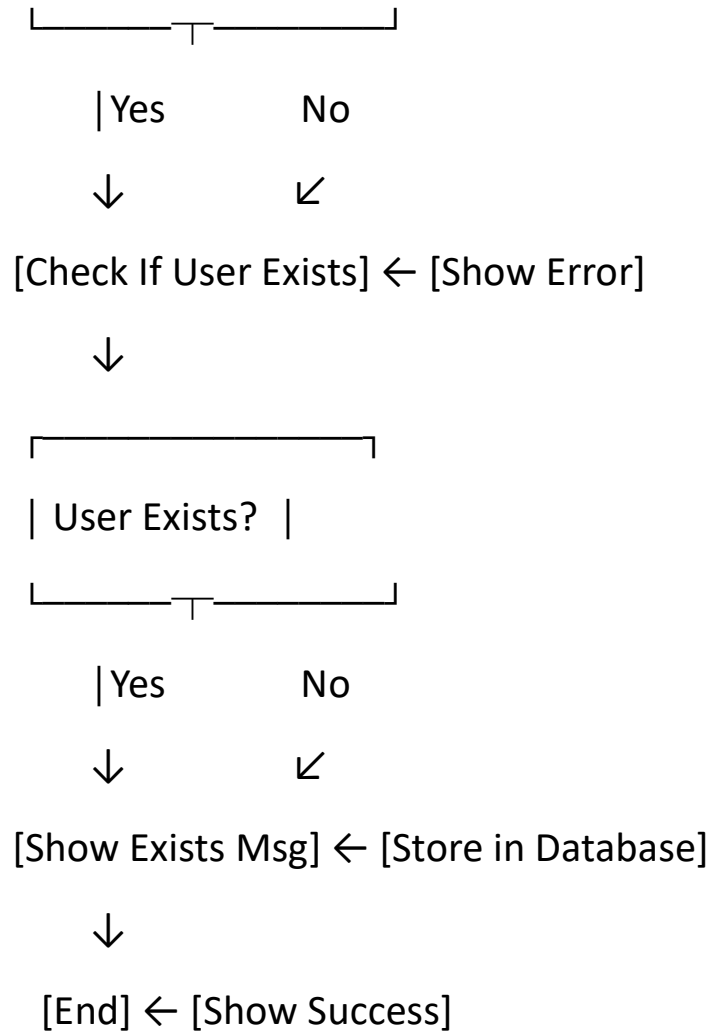
[User Enters Details]



[Validate Inputs]



| Inputs Valid? |




---

### Tools Required: -

- Paper & Pen (for manual diagram)
  - OR
  - Diagram Tools (Draw.io, Lucid chart, Creately, etc.)
- 

### Learning Outcome: -

After completing this lab, students will:

- Understand how to visualize decision-making in a system
- Learn flowchart components like decision, process, and input/output
- Gain experience mapping real-world processes into diagrams