

## LAB EXERCISE:

### 1. Create Github account & make repository

**Ans**

**Step 1:** Create a GitHub Account

Go to: <https://github.com>

Click "Sign up" in the top right corner.

Enter:

- Your email address
- Create a strong password
- Choose a username

### 2.Explain phases of SDLC life cycle

**Ans.** Phases of the SDLC Life Cycle

---

#### 1. Requirement Gathering and Analysis

- Purpose: Understand what the client/user wants.
  - Activities:
    - Collecting requirements from stakeholders.
    - Analyzing the feasibility (technical, operational, economic).
  - Output: Software Requirement Specification (SRS) document.
-

## 2. System Design

- Purpose: Plan how the software will be built.
  - Activities:
    - Create architectural designs (high-level and detailed design).
    - Define system architecture, technology stack, data flow, UI mockups.
  - Output: Design documents, wireframes, database schema.
- 

## 3. Implementation / Coding

- Purpose: Actual development of the software.
  - Activities:
    - Developers write code based on design documents.
    - Use of programming languages, frameworks, and tools.
  - Output: Source code, build files.
- 

## 4. Testing

- Purpose: Ensure the software is bug-free and meets requirements.
- Activities:
  - Perform unit testing, integration testing, system testing, and user acceptance testing (UAT).

- Output: Test reports, bug lists.
- 

#### 5. Deployment

- Purpose: Release the software to users.
  - Activities:
    - Deploy the application to a live/production environment.
    - May include pilot runs or staged rollout.
  - Output: Live running software.
- 

#### 6. Maintenance

- Purpose: Keep the software updated and bug-free after release.
- Activities:
  - Fix bugs, update features, security patches.
  - Performance monitoring.
- Output: Updated versions, support documentation.

**3. Write a simple "Hello World" program in two different programming languages of your choice.**

**Compare the structure and syntax.**

**Ans**

## 1. Python Program

python

CopyEdit

```
# Hello World in Python
```

```
print("Hello, World!")
```

## 2. C Program

c

CopyEdit

```
// Hello World in C
```

```
#include <stdio.h>
```

```
int main() {  
    printf("Hello, World!\n");  
    return 0;  
}
```

**4. Research and create a diagram of how data is transmitted from a client to a server over the Internet.**

**Ans.** Steps: How Data Travels from Client to Server

## 1. Client Request

- A user opens a web browser and enters a URL (e.g., [www.google.com](http://www.google.com)).

## 2. DNS Resolution

- The browser contacts a DNS server to convert the domain name into an IP address.

## 3. TCP/IP Connection

- Using the IP address, the client establishes a connection with the server using the TCP/IP protocol.

## 4. Data Packet Transmission

- Data is broken into packets and sent over the internet.
- Packets pass through routers and switches.

## 5. Server Response

- The server processes the request and sends a response back to the client in the form of data packets.

## 6. Client Receives Data

- The browser reassembles the packets and displays the content (e.g., a webpage).

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

**Ans.**

Server Code (Python)

**# server.py**

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

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

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

```
        # Send response status code
```

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

```
        # Set headers
```

```
        self.send_header('Content-type', 'text/html')
```

```
        self.end_headers()
```

```
        # Write message
```

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

```
# Define server address and port
```

```
server_address = ('localhost', 8080)
```

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

```
print("Server running on http://localhost:8080")
```

```
httpd.serve_forever()
```

**6. .Research different types of internet connections (e.g., broadband, fiber, satellite)and list their**

**pros and cons.**

**Ans**

Different type

Connectio n Type	Speed	Cost	Availabi lity	Best Use Case
Broadband	Medium	Low-Med	Urban/Su burban	General home & office use
Fiber Optic	Very High	Medium- High	Limited areas	Heavy usage, gaming, 4K video

Satellite	Medium	High	Remote/Rural	Areas without wired internet
Mobile (4G/5G)	Medium-High	Medium	Wide coverage	On-the-go use
Dial-Up	Very Low	Very Low	Rare	Last resort

## 7. Simulate HTTP and FTP requests using command line tools (e.g., curl).

**Ans**

Task	Command Example
Open website (HTTP)	<code>curl http://example.com</code>
View HTTP headers	<code>curl -I http://example.com</code>



Download via FTP      `curl ftp://... --user user:pass`

Upload via FTP      `curl -T file.txt ftp://... --user user:pass`

**8. Identify and explain three common application security vulnerabilities. Suggest possible solutions.**

**Ans.**

Vulnerability	Description	Solution
SQL Injection (SQLi)	Injecting SQL code into inputs	Use prepared statements, validate input
Cross-Site Scripting	Injecting scripts into web pages	Escape/encode output, use CSP
Broken Authentication	Weak login/session management	Strong passwords, MFA, secure sessions

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

**Ans.**

Google Chrome	Application Software	Surfing the web
---------------	----------------------	-----------------

Application	Type	Function/Purpose
Microsoft Word	Application Software	Document creation/editing
Windows OS	System Software	Running and managing system resources
Windows Defender	System Software	Security and virus protection
WhatsApp	Application Software	Communication and chatting

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

**ans.Layer Descriptions**

♦ 1. Presentation Layer

- What: The front-end interface users interact with (e.g., browser, mobile app).
- Technologies: HTML, CSS, JavaScript, React, Angular.

♦ 2. Application Layer

- What: Business logic and server-side processing.
- Technologies: Node.js, Django, Flask, Spring Boot, ASP.NET.

♦ 3. Data Layer

- What: Stores and manages data.
- Technologies: MySQL, MongoDB, PostgreSQL, Oracle.

**11.Create a case study on the functionality of the presentation, business logic, and dataaccess layers**

**of a given software system.**

**Ans**

## 1. Presentation Layer (User Interface)

### What it does:

- Allows users to interact with the system.
- Displays food items, menus, cart, etc.
- Takes user input (e.g., login, food selection, address).

### Example:

- A customer opens a website or app and sees a list of food items.
  - They select "Pizza" and click **"Add to Cart"**.
- 

## 2. Business Logic Layer (Processing Rules)

### What it does:

- Handles **how** the system works.
- Applies rules like checking if the food is available, calculating total price, applying discounts, etc.
- Connects the user's request to the database.

### Example:

- After clicking "**Add to Cart**", the system:
    - Checks if pizza is in stock.
    - Calculates the price.
    - Adds it to the user's cart in memory or session.
- 

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

#### What it does:

- Communicates with the **database** to save or get data.
- Handles food list, user details, order history, etc.

#### Example:

- When the order is placed:
  - This layer **saves** the order to the database.
  - **Updates** stock (one less pizza).
  - **Retrieves** past orders if the user wants to see them.



## Simple Summary Table:

Layer	What It Does	Example Action
Presentation	Shows info to users, takes input	User sees menu and selects food
Business Logic	Processes rules, calculations	Adds food to cart, calculates total
Data Access	Connects to the database	Saves order, fetches food list

---



## Final Example Flow:

1. **User** selects food on website
  2. **System** checks availability, calculates price
  3. **Database** saves the order and updates stock.
- 

12. Explore different types of software environments (development, testing,

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

## ans. Development Environment

### Development Environment

- **Who uses it?** Developers (programmers)
  - **Purpose:** To write and test new code.
  - **Example Tools:** VS Code, Git, Python, Node.js
  - **Notes:** Not stable, changes happen often.
- 

### 2 Testing Environment

- **Who uses it?** Testers or QA team
  - **Purpose:** To test the application for bugs.
  - **Example Tools:** Selenium, Postman, JUnit
  - **Notes:** Similar to real (live) environment but used only for testing.
- 

### 3 Production Environment

- **Who uses it?** Real users or customers
- **Purpose:** The actual application people use (live website or app)
- **Example Tools:** Web server (Apache/Nginx), database

- **Notes:** Very stable, no direct changes made here.

## Set Up a Basic Environment in a Virtual Machine (Simple)

### What You Need:

- A computer with at least 4 GB RAM
  - A Virtual Machine software like **VirtualBox**
  - A Linux OS file (ISO), like **Ubuntu**
- 

### Steps to Setup:

#### Step 1: Download Tools

- Download **VirtualBox** from [virtualbox.org](https://www.virtualbox.org)
- Download **Ubuntu ISO** from [ubuntu.com](https://ubuntu.com)

#### Step 2: Create Virtual Machine

- Open VirtualBox → Click **New**
- Name it (e.g., "MyDevEnv")
- Choose **Linux** → **Ubuntu (64-bit)**



- Give it 2048 MB RAM and 20 GB hard disk

✅ Step 3: Install Ubuntu

- Start the VM and select the Ubuntu ISO file
- Follow on-screen steps to install Ubuntu

✅ Step 4: Install Basic Tools Inside VM

Open Terminal in Ubuntu and run:

```
bash
```

```
CopyEdit
```

```
sudo apt update
```

```
sudo apt install git curl build-essential
```

Optional: Install code editor

```
bash
```

```
CopyEdit
```

```
sudo snap install code --classic # Installs  
Visual Studio Code
```

**13. 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. Log in to your account.
  3. Click the + icon (top right) → Click "New repository"
  4. Fill in:
    - Repository name: (e.g., **my-project**)
    - Choose Public
    - Check "Add a README file"
  5. Click Create repository
- 

## ✅ Step 2: Set Up Git on Your Computer

👉 If Git is not installed, download it from <https://git-scm.com> and install it.

---

## ✅ Step 3: Clone the Repository

In GitHub, go to your repository page and copy the URL.

Then in your terminal:

```
bash
CopyEdit
git clone
https://github.com/your-username/my-project.git
cd my-project
```

---


### ✅ Step 4: Add a Code File

Example: Create a simple file

```
bash
CopyEdit
echo "print('Hello, GitHub!')" > hello.py
```

---

### ✅ Step 5: Commit and Push Code Changes

 Step-by-step:

```
bash
CopyEdit
git add hello.py                # Step 1: Stage the
file                             file
git commit -m "Add hello.py"    # Step 2: Commit
with a message                  with a message
git push origin main            # Step 3: Push to
GitHub                          GitHub
```

**15. Create a student account on Github and collaborate on a small project with aclassmate**  
**Ans.**

### **Step 1: Create a GitHub Student Account**

1. Go to: <https://education.github.com/students>
  2. Click "Get Student Benefits"
  3. Sign in with your GitHub account (or create one)
  4. Verify with your school email or student ID
  5. Wait for approval (usually within 1-2 days)
- 

### Step 2: Create a Project Repository

1. Log in to GitHub → Click "+" → "New repository"
  2. Name it (e.g., `school-project`)
  3. Choose Public or Private
  4. Click Create repository
- 

### Step 3: Add Your Classmate as Collaborator

1. Go to your repository → Click "Settings"
2. Select "Collaborators"

3. Add your classmate's GitHub username

4. Click Invite

---

✅ Step 4: Work Together

Now both of you can:

- Clone the repo
- Add files
- Commit changes
- Push to GitHub

**16. Create a list of software you use regularly and classify them into the following categories:**

**system, application, and utility software.**

**Ans System Software:**

Software Name	Use
Windows 10/11	Operating system
Android OS	Mobile operating system
Device Drivers	Allows hardware to work with system
BIOS/UEFI	Starts the computer

**Application Software:**

<b>Software Name</b>	<b>Use</b>
Google Chrome	Internet browsing
Microsoft Word	Typing documents
WhatsApp	Chat and messaging
VLC Media Player	Watching videos
Facebook/Instagram	Social networking

### **3. Utility Software**

<b>Software Name</b>	<b>Use</b>
Antivirus (e.g., Avast)	Protects from viruses
WinRAR / 7-Zip	Compress and extract files
Disk Cleanup	Frees up hard drive space
CCleaner	Cleans junk files
Task Manager	Monitors running programs

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

Ans

## Introduction

Application software is a type of computer program designed to help users perform specific tasks. These tasks can range from creating documents to managing data, communicating, or even editing photos and videos. Different types of application software are used in daily life to make work easier, faster, and more efficient.

---

## ✓ Types of Application Software

### 1. Word Processing Software

- **Example:** Microsoft Word, Google Docs
- **Use:** Writing and editing documents.
- **Productivity Benefit:** Helps create professional documents quickly with features like spell check, formatting tools, and templates.

### 2. Spreadsheet Software

- **Example:** Microsoft Excel, Google Sheets
- **Use:** Managing data, making calculations, creating charts.
- **Productivity Benefit:** Speeds up data analysis, financial planning, and tracking.

### 3. Presentation Software

- **Example:** Microsoft PowerPoint, Google Slides

- **Use:** Creating visual slideshows for meetings or teaching.
- **Productivity Benefit:** Communicates ideas clearly and quickly with visuals and animations.

#### 4. Database Management Software

- **Example:** Microsoft Access, MySQL
- **Use:** Storing, organizing, and managing large amounts of data.
- **Productivity Benefit:** Makes it easy to search, sort, and manage business or academic information.

#### 5. Communication Software

- **Example:** Zoom, Microsoft Teams, Gmail, WhatsApp
- **Use:** Video calls, emails, and messaging.
- **Productivity Benefit:** Enables fast communication and collaboration, especially for remote teams.

#### 6. Graphic Design Software

- **Example:** Adobe Photoshop, Canva
- **Use:** Creating and editing images, posters, social media content.
- **Productivity Benefit:** Helps produce creative content faster with built-in tools and templates.

#### 7. Web Browsers



- **Example:** Google Chrome, Mozilla Firefox
- **Use:** Browsing the internet.
- **Productivity Benefit:** Provides quick access to information, online tools, and research.

## 8. Project Management Software

- **Example:** Trello, Asana, Microsoft Project
- **Use:** Planning and managing tasks or team projects.
- **Productivity Benefit:** Helps track progress, meet deadlines, and stay organized.

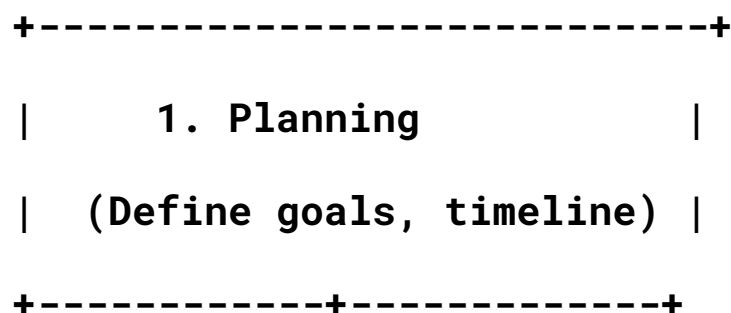
---

## ✅ Conclusion

Application software plays a vital role in improving productivity in both personal and professional life. It automates routine tasks, reduces errors, saves time, and enhances the quality of work. Choosing the right application software.

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

Ans.



|

v

+-----+

|        2. Requirements        |

| (Gather and analyze needs) |

+-----+

|

v

+-----+

|        3. Design        |

| (Create system architecture)|

+-----+

|

v

+-----+

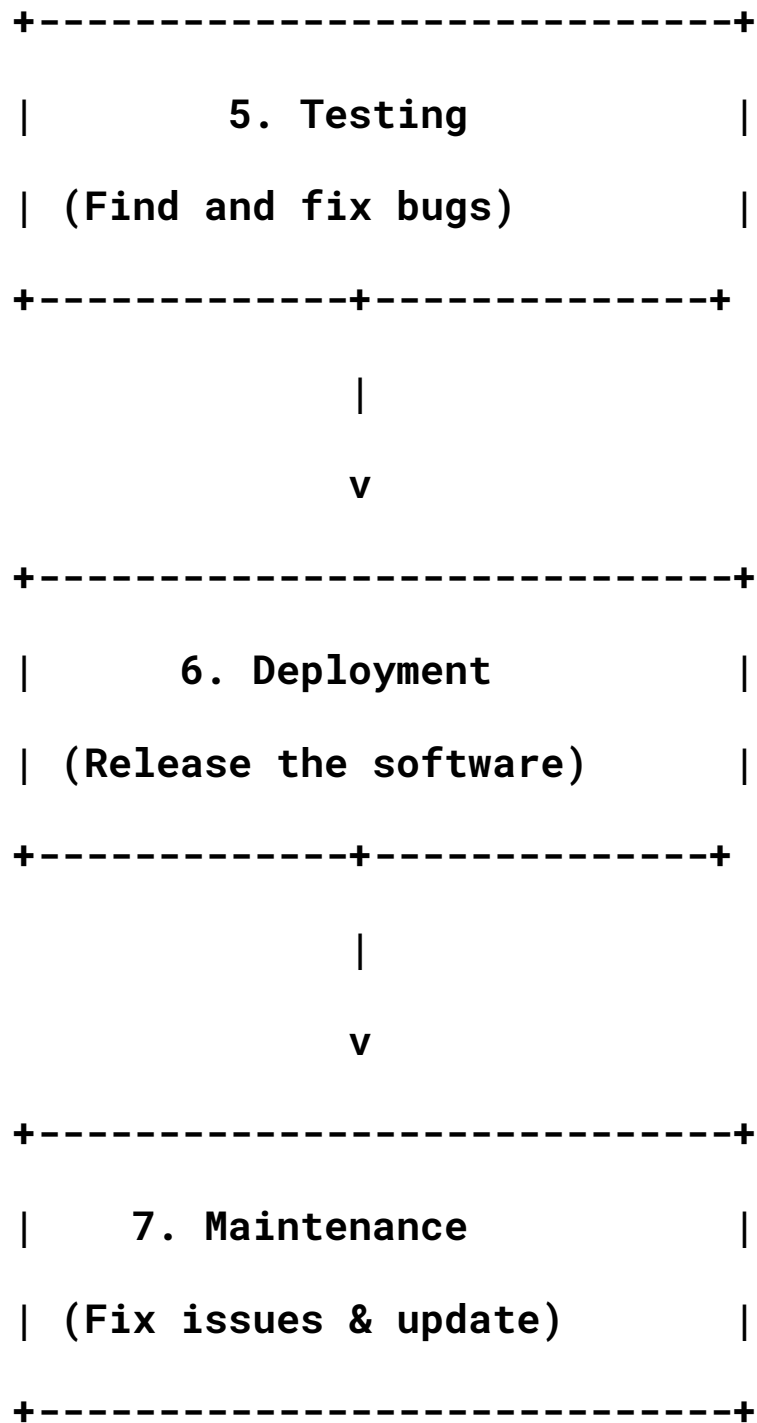
|        4. Development        |

| (Write the actual code)        |

+-----+

|

v



**20. Write a requirement specification for a simple library management system.**

Ans

# Software Requirement Specification (SRS)

## Project: Simple Library Management System

---

### 1. Introduction

#### 1.1 Purpose:

The purpose of this Library Management System is to manage book records, member information, issue and return of books, and generate reports efficiently in a library.

#### 1.2 Scope:

This system will allow:

- Admins to add/remove books and members.
  - Users to search and borrow books.
  - Maintain records of issued and returned books.
  - Generate overdue and availability reports.
- 

### 2. Functional Requirements

#### 2.1 Admin Features:

- Add/edit/delete book records.
- Add/edit/delete member records.
- Issue and return books.
- View all borrowed and returned books.

## 2.2 User Features:

- Search for books by title, author, or category.
- View book availability.
- Request to borrow a book (handled by admin).

## 2.3 Book Management:

- Store details like title, author, ISBN, category, quantity, status (available/issued).

## 2.4 Member Management:

- Store details like name, member ID, contact, and borrowing history.

## 2.5 Issue/Return Management:

- Record issue date, due date, return date.
- Show overdue status and fine if applicable.

---

## 3. Non-Functional Requirements

- **Usability:** Simple and user-friendly interface.
  - **Performance:** Fast response time for book searches.
  - **Security:** Admin login required for management features.
  - **Portability:** Should run on any standard PC or browser.
-

## ✓ 4. System Requirements

### 4.1 Hardware Requirements:

- 2 GHz processor, 4 GB RAM, 20 GB free disk space

### 4.2 Software Requirements:

- Operating System: Windows/Linux
  - Database: MySQL or SQLite
  - Backend: Python / PHP / Java
  - Frontend: HTML/CSS/JavaScript (for web) or simple GUI (for desktop)
- 

## ✓ 5. Assumptions & Constraints

- Each member can borrow a limited number of books (e.g., 3).
- Book return should happen within 14 days.
- Admin has higher access rights than normal users.

**21. Perform a functional analysis for an online shopping system.**

**Ans.**

### 1. User Functions

- **Register/Login:** Users create an account or log in.

- **Browse Products:** Users can search, filter, and view products.
  - **View Product Details:** See price, description, images, and reviews.
  - **Add to Cart:** Add selected items to shopping cart.
  - **Place Order:** Checkout, enter address, and choose payment.
  - **Track Orders:** View order status (Processing, Shipped, Delivered).
  - **Write Reviews:** Rate and review purchased products.
- 

## ✓ 2. Admin Functions

- **Manage Products:** Add, update, or delete products.
  - **Manage Orders:** View and update order status.
  - **Manage Users:** Handle user accounts and permissions.
  - **View Reports:** Sales reports, stock reports, etc.
  - **Moderate Reviews:** Approve or remove user reviews.
- 

## ✓ 3. System Inputs

- User login info (email, password)
- Search keywords

- Product selection
  - Shipping address
  - Payment details
- 

#### ✓ 4. System Outputs

- Product list
  - Cart summary
  - Order confirmation
  - Payment success/failure message
  - Order tracking updates
- 

#### ✓ 5. Main Actors

- **Customer:** Uses the site to shop.
  - **Admin:** Manages the site and content.
  - **Payment System:** Processes online payments.
  - **Delivery Staff:** Delivers the product.
- 

#### ✓ 6. Basic Process Flow

pgsql

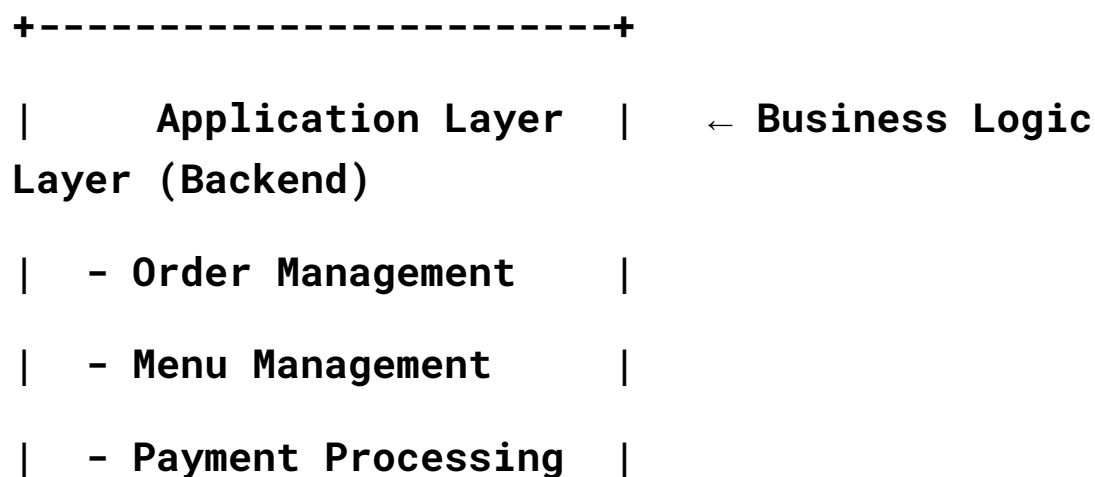
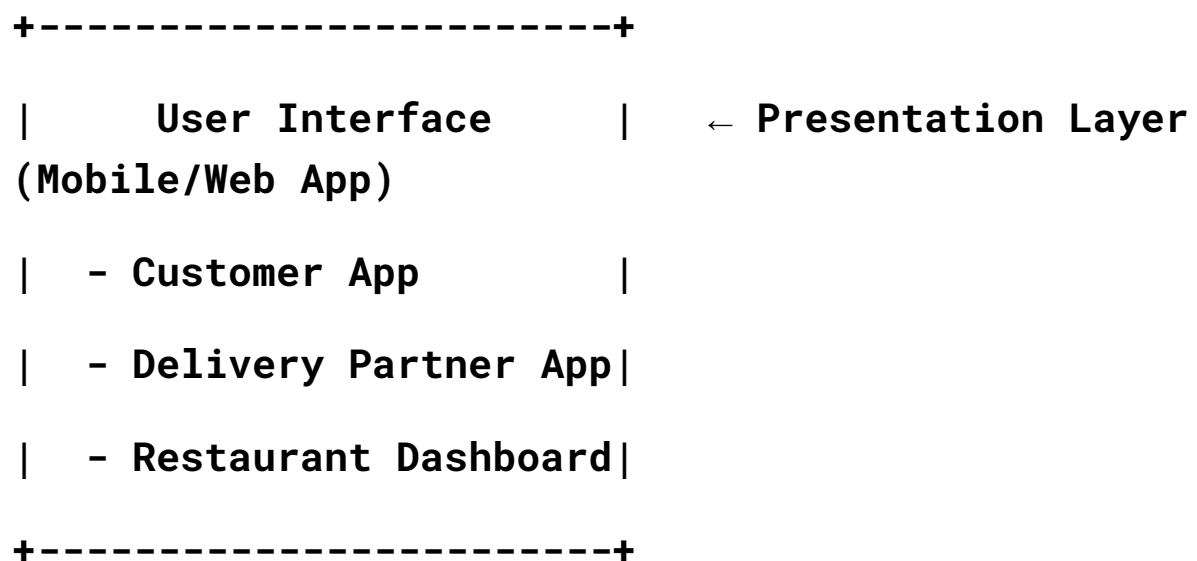


CopyEdit

Login/Register → Browse → Add to Cart → Checkout → Payment → Order Confirmation → D

**21. Design a basic system architecture for a food delivery app.**

**Ans**



```
| - Delivery Tracking |
+-----+
```



```
+-----+
|      Data Layer      | ← Database (Storage)
| - Users Table        |
| - Orders Table       |
| - Restaurants Table  |
| - Menu Items Table   |
| - Payments Table     |
+-----+
```

## . Key Components

### Frontend (Presentation Layer)

- **Customer App:** Browse restaurants, order food, track delivery.
- **Restaurant Panel:** Manage menu, accept/reject orders.

- **Delivery Partner App:** Get assigned orders, update status.

#### **Backend (Application Layer)**

- **Order Service:** Handles order placement and updates.
- **Payment Service:** Integrates with payment gateways.
- **Location Service:** Tracks delivery agents in real time.
- **Notification Service:** Sends SMS/Email/Push alerts.
- **Authentication Service:** Manages user login and security.

#### **Database (Data Layer)**

- **Stores all user, restaurant, order, and payment data.**
- **Could use MySQL, MongoDB, or PostgreSQL.**

---

### **3. Additional Services**

Service	Use
API Gateway	Manages all API requests
Cloud Storage	Stores images (e.g., food photos)
Caching (Redis)	For fast access to frequently used data
Analytics	Tracks user behavior and sales

---

#### ✓ 4. External Integrations

- Payment Gateway (Razorpay, Stripe, PayPal)
- SMS Gateway (Twilio)
- Maps API (Google Maps for location & tracking)

22. Develop test cases for a simple calculator program.

**Simple Test Cases for Calculator**

Test No.	Operation	Input	Expected Result
1	Addition	$5 + 3$	8
2	Subtraction	$10 - 4$	6
3	Multiplication	$6 \times 2$	12
4	Division	$12 \div 3$	4
5	Division by Zero	$7 \div 0$	Error / Not allowed
6	Negative numbers	$-5 + 3$	-2
7	Decimal numbers	$2.5 + 1.5$	4.0
8	Zero addition	$0 + 9$	9
9	Zero multiplication	$0 \times 8$	0

10      Invalid input "a" +    Error message  
2

**23.Document a real-world case where a software application required criticalmaintenance.**

**Ans.**

## **Case Study: Critical Maintenance in WhatsApp (October 2021 Outage)**

---

### **Background**

On October 4, 2021, WhatsApp (along with Facebook and Instagram) experienced a global outage that lasted for over 6 hours. The issue was caused by a critical configuration change on Facebook's backbone routers that coordinate network traffic between its data centers.

---

### **What Happened?**

- A routine maintenance update went wrong and disrupted communication between Facebook's servers.
  - As a result, WhatsApp servers became unreachable, affecting billions of users globally.
  - Not only messaging but also voice and video calls were down.
- 

### **Impact**

- WhatsApp, Facebook, and Instagram were completely unavailable for over 6 hours.
  - Businesses and users relying on WhatsApp for communication faced major disruptions.
  - Facebook lost an estimated \$100 million+ in revenue due to the downtime.
  - Many users switched temporarily to other apps like Telegram and Signal.
- 

#### ✓ Critical Maintenance Actions Taken

1. Network Engineers Investigated the root cause of the configuration issue.
  2. Manual reset and reconnection of data center communication paths was required.
  3. Security systems had to be bypassed to physically access and fix the servers.
  4. Once resolved, services were gradually restored to prevent overload.
- 

#### ✓ Lessons Learned

- A small network configuration mistake can have a massive global impact.
- There is a need for better rollback mechanisms and testing before deployment.

- Critical systems should have redundancy and better monitoring.
- 

#### ✓ Conclusion

This case shows how even a large and stable software system like WhatsApp can require critical maintenance due to unexpected failures. Quick action and technical expertise were essential to restore services and reduce the impact on millions of users.

## 24.Create a DFD for a hospital management system.

**Ans**

### Explanation of Main Processes

Process No.	Process Name	Description
1	Register Patient	Registers patient details into the system.
2	Manage Appointments	Schedules and stores doctor-patient appointments.
3	Manage Treatments	Doctors update diagnosis and treatment records.
4	Billing & Payment	Generates and stores billing details for patients.

---

#### ✓ Data Stores

- Patient Database: Stores patient information.



- Appointment DB: Stores appointment schedules.
  - Treatment Records: Stores diagnosis, prescriptions, test results.
  - Bill DB: Stores payment and billing history.
- 

#### ✓ External Entities

- Patient: Registers, books appointments, pays bills.
- Doctor: Views and updates appointments and treatments.

## 25. Build a simple desktop calculator application using a GUI library.

**Ans**

### Tools Used

- **Language:** Python
  - **GUI Library:** Tkinter (comes with Python)
- 

#### ✓ Steps (Short)

1. **Import Tkinter**  
`import tkinter as tk`

**2. Create main window**

```
window = tk.Tk()
```

**3. Add Entry box** for input/output

**4. Create buttons** for 0-9, +, -, ×, ÷, =, C

**5. Define functions** for:

- Button press
- Clear
- Evaluate expression using `eval()`

**6. Use `grid()`** to arrange buttons

**7. Run the GUI**

```
window.mainloop()
```

## **26. Build a simple desktop calculator application using a GUI library.**

### **Tools Used**

- Language: Python
- GUI Library: Tkinter (comes with Python)

---

### Steps (Short)

**1. Import Tkinter**

```
import tkinter as tk
```

2. Create main window

```
window = tk.Tk()
```

3. Add Entry box for input/output

4. Create buttons for 0-9, +, -, ×, ÷, =, C

5. **Define functions** for:

- Button press
- Clear
- Evaluate expression using eval()

6. **Use grid()** to arrange buttons

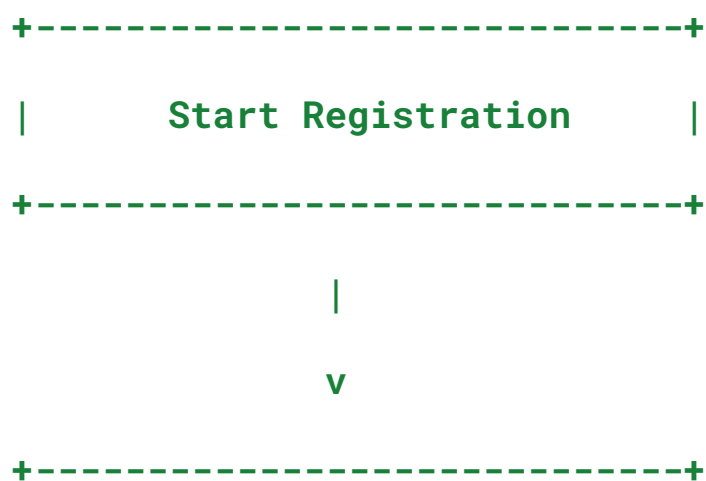
7. **Run the GUI**

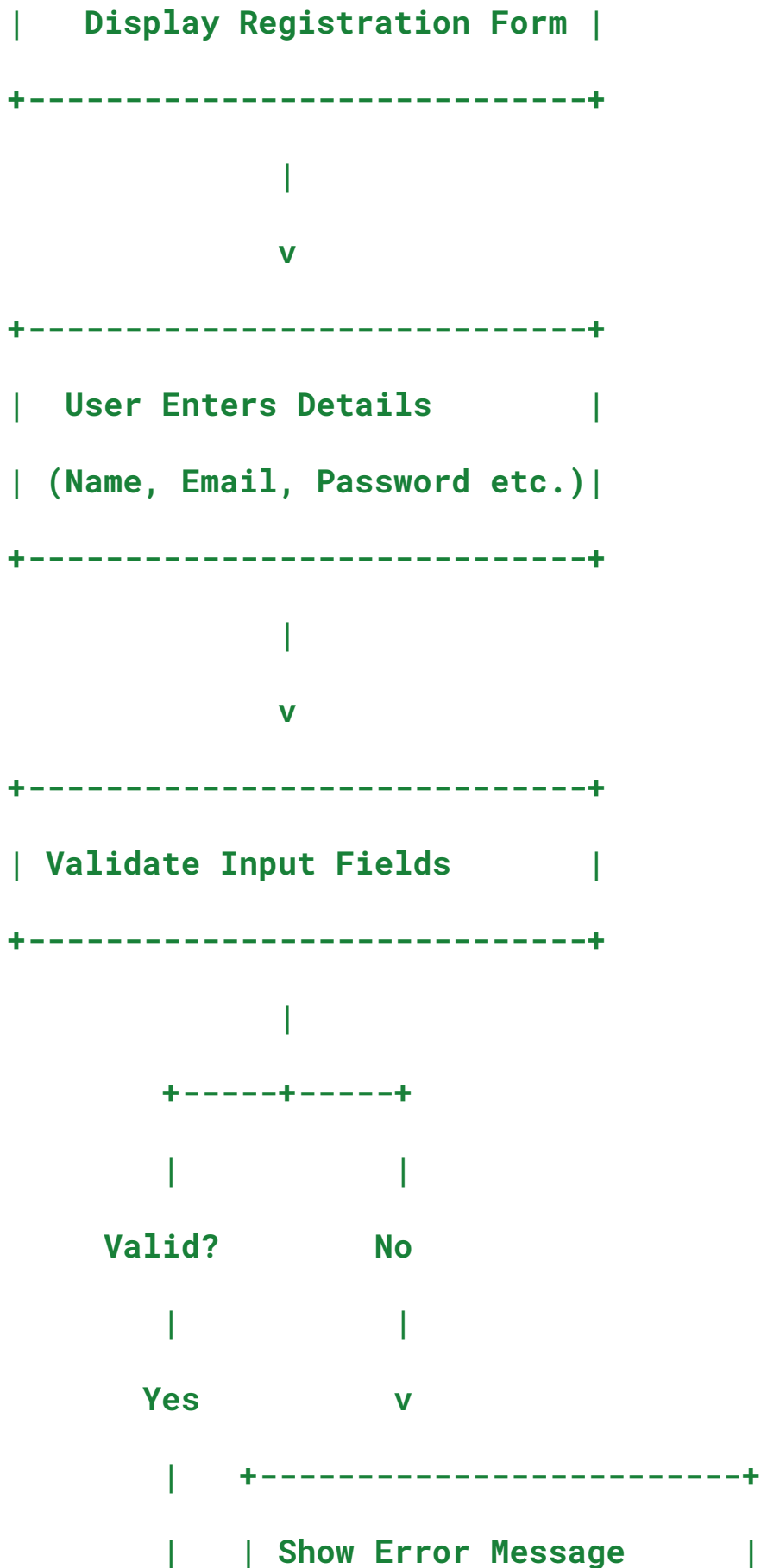
```
window.mainloop()
```

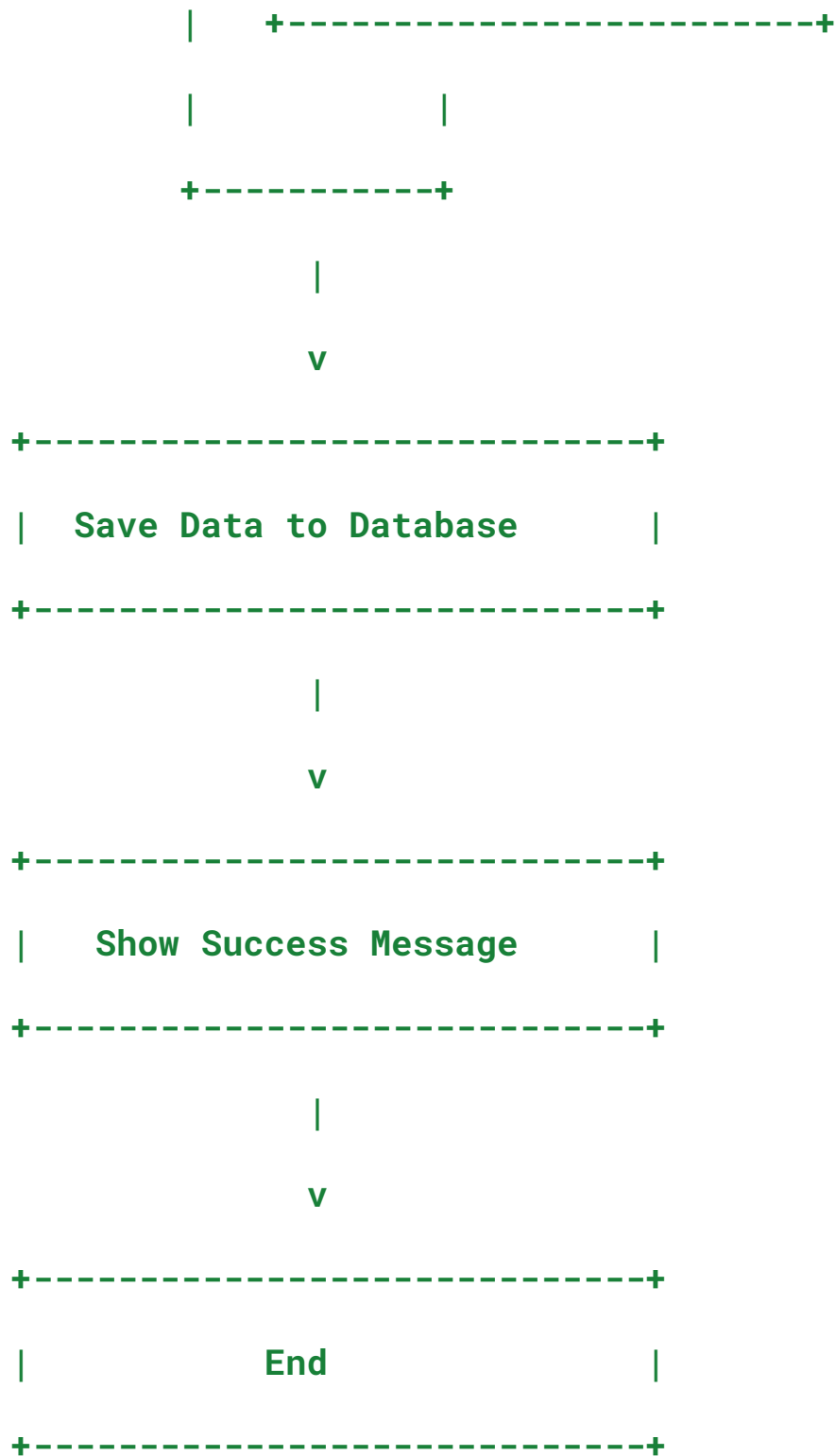
**27. Draw a flowchart representing the logic of a basic online registration system.**

**Ans**

**Flowchart logic basic step**







## Lab Assignment Task: 1

1. Create a simple HTML webpage that includes: A header (), footer (), main section (), and aside section (). A paragraph with some basic text. A list (both ordered and unordered). A link that opens in a new tab.

Ans

HTML CODE

```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <title>Simple Webpage</title>

</head>

<body>


    <header>

        <h1>Welcome to My Simple Webpage</h1>

    </header>


    <main>

        <h2>Main Content</h2>

        <p>This is a paragraph of basic text explaining the purpose of the webpage.</p>
```

### <h3>Ordered List</h3>

<ol>

<li>First item</li>

<li>Second item</li>

<li>Third item</li>

</ol>

### <h3>Unordered List</h3>

<ul>

<li>Apple</li>

<li>Banana</li>

<li>Cherry</li>

</ul>

<p>

Visit <a href="https://www.google.com" target="\_blank">Google</a> in a new tab.

</p>

</main>

```
<aside>

    <h3>Side Notes</h3>

    <p>This is some additional information in the
aside section.</p>

</aside>


<footer>

    <p>© 2025 My Simple Webpage</p>

</footer>


</body>

</html>
```

**2.Lab Assignment Task: Create a contact form with the following fields: Full name (text input) Email (email input) Phone number (tel input) Subject (dropdown menu) Message (textarea) Submit button.**

**Ans**

**HTML CODE CONTACT FROM**

```
<!DOCTYPE html>

<html lang="en">
```



```
<head>

    <meta charset="UTF-8">

    <title>Contact Form</title>

</head>

<body>


    <h2>Contact Us</h2>


    <form action="#" method="post">

        <!-- Full Name -->

        <label for="fullname">Full Name:</label><br>

        <input type="text" id="fullname"
name="fullname" required><br><br>


        <!-- Email -->

        <label for="email">Email:</label><br>

        <input type="email" id="email" name="email"
required><br><br>


        <!-- Phone Number -->

        <label for="phone">Phone Number:</label><br>
```

```
<input type="tel" id="phone" name="phone"
required><br><br>
```

```
<!-- Subject Dropdown -->

<label for="subject">Subject:</label><br>

<select id="subject" name="subject" required>

    <option value="">--Select a
Subject--</option>

    <option value="general">General
Inquiry</option>

    <option value="support">Technical
Support</option>

    <option
value="feedback">Feedback</option>

</select><br><br>
```

```
<!-- Message -->
```

```
<label for="me
```

border and some basic styling using inline CSS. Use  
colspan or rowspan to merge cells where applicable**Lab**

### **Assignment Task:3**

**Create a product catalog table that includes  
the following columns: Product Name Product  
Image (use placeholder image URLs) Price**

**Description   Availability (in stock, out of stock)   Additional Requirements:   Use thead for the table header.**

**Ans**

```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <title>Product Catalog</title>

</head>

<body>

    <h2>Product Catalog</h2>


    <table style="border: 1px solid black; border-collapse: collapse; width: 100%;">

        <thead>

            <tr style="background-color: #f2f2f2;">

                <th style="border: 1px solid black; padding: 8px;">Product Name</th>

                <th style="border: 1px solid black; padding: 8px;">Product Image</th>

                <th style="border: 1px solid black; padding: 8px;">Price</th>
```

```

        <th style="border: 1px solid black; padding:
8px;">Description</th>

        <th style="border: 1px solid black; padding:
8px;">Availability</th>

    </tr>

</thead>

<tbody>

    <tr>

        <td style="border: 1px solid black; padding:
8px;">Wireless Mouse</td>

        <td style="border: 1px solid black; padding:
8px;">

        </td>

        <td style="border: 1px solid black; padding:
8px;">$15</td>

        <td style="border: 1px solid black; padding:
8px;" rowspan="2">

            Compact wireless mouse with ergonomic
design.

        </td>

        <td style="border: 1px solid black; padding:
8px;">In Stock</td>

    </tr>

    <tr>

```

```

        <td style="border: 1px solid black; padding:
8px;">Wireless Keyboard</td>

        <td style="border: 1px solid black; padding:
8px;">

        </td>

        <td style="border: 1px solid black; padding:
8px;">$25</td>

        <td style="border: 1px solid black; padding:
8px;">Out of Stock</td>

    </tr>

    <tr>

        <td style="border: 1px solid black; padding:
8px;" colspan="2">USB Flash Drive (64GB)</td>

        <td style="border: 1px solid black; padding:
8px;">$10</td>

        <td style="border: 1px solid black; padding:
8px;">Fast data transfer USB 3.0.</td>

        <td style="border: 1px solid black; padding:
8px;">In Stock</td>

    </tr>

</tbody>

</table>

</body>

</html>

```

