LAB EXERCISE (PRACTICAL)

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

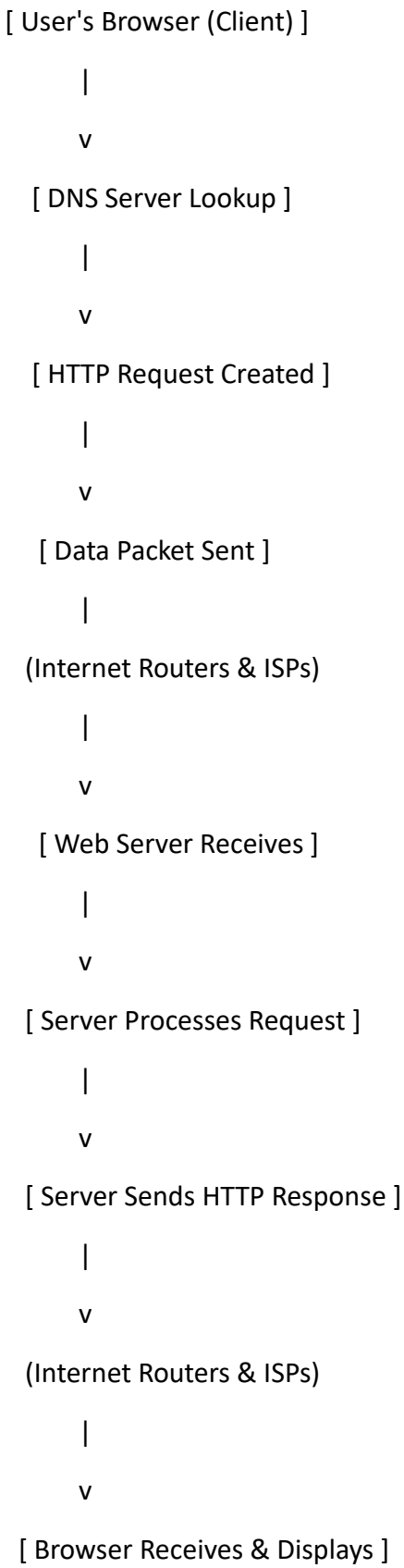
Ans: HTML5

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Hello world</title>
</head>
<body>
   <h1>hello world</h1>
</body>
</html>
```

PYTHON:

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

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


[User's Browser (Client)]
|
v
[DNS Server Lookup]
|
v
[HTTP Request Created]
|
v
[Data Packet Sent]
|
(Internet Routers & ISPs)
|
v
[Web Server Receives]
|
v
[Server Processes Request]
|
v
[Server Sends HTTP Response]
|
v
(Internet Routers & ISPs)
|
v
[Browser Receives & Displays]

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

1. Broadband (DSL / Cable)

Description:

- Uses telephone lines (DSL) or cable TV lines (Cable) for high-speed internet.

Pros:

- Widely available
- Always-on connection
- Suitable for streaming, browsing, video calls

Cons:

- Speed depends on distance from ISP center (DSL)
- Slower than fiber
- Shared bandwidth (Cable) may slow down during peak hours

---

2. Fiber Optic Internet

Description:

- Uses light signals through glass or plastic cables to provide ultra-fast speeds.

Pros:

- Very high speed (up to 1 Gbps or more)
- Stable and reliable connection
- Excellent for gaming, HD/4K streaming, cloud usage

Cons:

- Expensive to install
- Limited availability in rural areas

---

3 . Satellite Internet

Description:

- Data is transmitted to and from a satellite orbiting Earth.

Pros:

- Available in remote and rural areas
- Doesn’t require telephone or cable lines

Cons:

- High latency (delay in communication)
- Slower speeds compared to fiber or cable
- Weather can affect performance
- Expensive data plans

---

4. Mobile Data (4G/5G Networks)

Description:

- Wireless internet provided by cellular networks to phones and hotspots.

Pros:

- Portable – use it anywhere with mobile coverage
- 5G offers very high speeds and low latency
- Great for mobile users

Cons:

- Limited data plans (expensive if overused)
- Coverage varies by location
- Battery consumption on mobile devices

---

5. Dial-Up Internet (Very Outdated)

Description:

- Connects via telephone lines using a modem; extremely slow.

Pros:

- Very low cost
- Still available in some rural areas

Cons:

- Extremely slow (max 56 kbps)

- Ties up the phone line
- Cannot support modern internet needs

(4) : Design a simple HTTP client-server communication in any language

ANS: # server.py

```
from http.server import BaseHTTPRequestHandler, HTTPServer

class SimpleHandler(BaseHTTPRequestHandler):
    def do_GET(self):
        self.send_response(200)  # Status code: OK
        self.send_header("Content-type", "text/html")
        self.end_headers()
        self.wfile.write(b"<h1>Hello from the server!</h1>")

if __name__ == "__main__":
    server_address = ("localhost", 8000)  # Host and port
    httpd = HTTPServer(server_address, SimpleHandler)
    print("Server running at http://localhost:8000/")
    httpd.serve_forever()

# client.py
import requests

response = requests.get("http://localhost:8000")
print("Server Response:")
print(response.text)
```

(5) Identify and explain three common application security vulnerabilities. Suggestpossible solutions.

ANS:

1. SQL Injection (SQLi)

Description:

An attacker injects malicious SQL code into input fields to manipulate or access the database.

Example:

sql

CopyEdit

SELECT * FROM users WHERE username = 'admin' OR '1'='1';

Solution:

- Use Prepared Statements or Parameterized Queries to separate SQL logic from user input.
- Input validation: Allow only expected data formats.
- Use ORM (Object-Relational Mapping) libraries like SQLAlchemy (Python), Hibernate (Java), etc.

---

2. Cross-Site Scripting (XSS)

Description:

Attackers inject malicious scripts into web pages viewed by other users, stealing cookies or session data.

Example:

html

CopyEdit

<script>alert('Hacked!');</script>

Solution:

- Escape user input before displaying it in the browser (<, >, &, etc.).
- Use Content Security Policy (CSP) headers to restrict script sources.
- Sanitize inputs using libraries like DOMPurify (JavaScript) or Bleach (Python).

3. Broken Authentication

Description:

Weak login mechanisms or session handling allow attackers to impersonate users.

Example:

- Guessable passwords.
- Session IDs in URLs.

Solution:

- Use multi-factor authentication (MFA).
- Secure session handling (use cookies with HttpOnly and Secure flags).
- Limit login attempts and implement CAPTCHA after multiple failures.
- Use frameworks' built-in auth modules instead of custom authentication code.

(6) Identify and classify 5 applications you use daily as either system software orapplication software.

ANS:

Here are **5 applications** you might use daily, classified as either **system software** or **application software**:

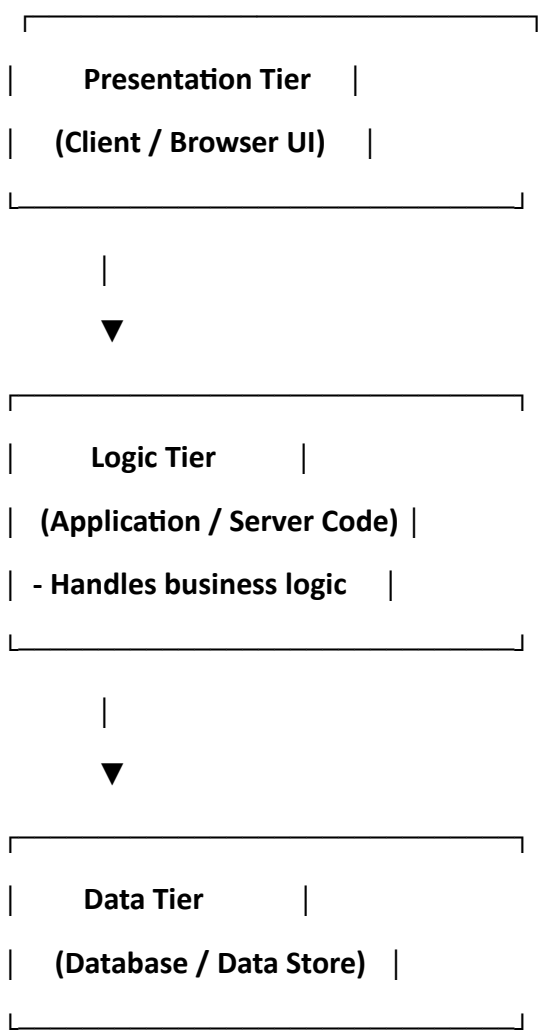
Application	**Type**	**Classification**	**Reason**
Google Chrome / Browser	Application Software	✓ Application Software	Used to browse the internet; directly interacts with the user.
Microsoft Word	Application Software	✓ Application Software	Used for word processing; user-level task.
Windows 10 / 11 (OS)	System Software	✓ System Software	Manages hardware and software; platform for other apps to run.
Antivirus (e.g., Avast)	System Software	✓ System Software	Protects system resources and files; works in background.
WhatsApp (Desktop/Mobile)	Application Software	✓ Application Software	User-level messaging and communication tool.

Definitions:

- **System Software**: Software that controls and manages hardware and system operations (e.g., Operating Systems, device drivers).
- **Application Software**: Software that helps users perform specific tasks (e.g., browsers, games, word processors).

(7) Design a basic three-tier software architecture diagram for a web application.

ANS:



Explanation of Each Tier:

1. **Presentation Tier (Front-End)**
    - Role: Interacts with the user (UI/UX)

- Examples: HTML, CSS, JavaScript, React, Angular
- Tools: Web browsers (Chrome, Firefox)

2. **Logic Tier (Middle Layer / Back-End)**
    - Role: Processes input, handles logic, connects frontend with data
    - Examples: Node.js, Express.js, Django, Spring Boot
    - Function: API handling, data processing, authentication
3. **Data Tier (Database Layer)**
    - Role: Stores and retrieves data
    - Examples: MySQL, PostgreSQL, MongoDB, SQLite
    - Operations: CRUD (Create, Read, Update, Delete

(8) : Create a case study on the functionality of the presentation, business logic, and dataaccess layers of a given software system.

ANS:

Case Study: Plant Care Assistant Web Application

Overview:

The Plant Care Assistant is a web-based application that helps users manage and care for their indoor and outdoor plants. It includes features like plant care reminders, image-based disease detection, and care suggestions based on weather and environment.

Three-Tier Architecture Functionality

Presentation Layer (User Interface)

Function:

- Displays the interface to the user via a web browser.
- Collects user input (e.g., plant details, uploaded images).
- Displays outputs like care instructions, reminders, and disease analysis.

Technologies Used:

- HTML5, CSS3, JavaScript (ES6+)
- Frontend frameworks: Bootstrap for design, or React (optional)

Example Functions:

- User registers and logs in
- Uploads a photo of their plant
- Views a dashboard with plant status and care tips

Business Logic Layer (Application Server)

Function:

- Processes user inputs and determines responses
- Contains logic for disease detection, scheduling reminders, and filtering recommendations
- Handles authentication, validation, and session control

Technologies Used:

- Node.js + Express.js
- Plant disease detection via Python ML model (optional integration)
- APIs to fetch real-time weather data

Example Logic:

- If a user uploads a plant image, the system sends it to the model and returns the disease status.
- Sends weekly watering reminders based on plant type and weather conditions.

Data Access Layer (Database)

Function:

- Stores and retrieves data from the database
- Maintains data integrity and relationships
- Supports secure access to user records, plant data, and history

Technologies Used:

- SQLite3 (or MySQL/PostgreSQL for production)
- ORM (optional): Sequelize (Node.js)

Example Data Stored:

- User profiles and login credentials
- Plant information (type, care schedule)
- Uploaded image references and analysis results

Workflow Example

User Action: A registered user uploads a photo of their infected plant.

1. Presentation Layer:
    - The image is uploaded through a web form.
    - A progress bar is shown.
2. Business Logic Layer:
    - Receives the image, validates file type.
    - Sends it to a Python-based image analysis model.
    - Receives the diagnosis (e.g., "Powdery Mildew").
3. Data Access Layer:
    - Saves image details and diagnosis result to the database.
    - Links it to the user's profile.
4. Presentation Layer:
    - Shows the result to the user with treatment tips.

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

ANS:

Exploring Software Environments

In software development, different **environments** are used to ensure stability, quality, and performance before delivering software to end users. Here are the three main types:

1. Development Environment

- **Purpose**: Where developers write and test code.
- **Characteristics**:

    - Frequent changes and updates
    - Debugging and code testing
    - Includes IDEs, code editors, local servers
- **Tools**: VS Code, Git, Node.js, Docker, etc.

2. Testing/Staging Environment

- **Purpose**: Used by QA/testers to verify features and catch bugs.
- **Characteristics**:
    - Mirrors production setup
    - Runs automated/manual tests
    - Isolated from developers
- **Tools**: Selenium, Postman, JUnit, CI/CD pipelines (GitHub Actions, Jenkins)

3. Production Environment

- **Purpose**: Final environment where the live application runs for users.
- **Characteristics**:
    - Highly stable and secure
    - Monitored and backed up regularly
    - Little to no direct developer access
- **Tools**: Web servers (Apache, Nginx), Cloud (AWS, Azure), Monitoring (New Relic)

Part 2: Set Up a Basic Environment in a Virtual Machine (VM)

Here's a simple example of how to **set up a basic development environment** in a VM using VirtualBox and Ubuntu.

🔧 **Tools Needed:**

- **VirtualBox** (VM Manager)
- **Ubuntu ISO** (Linux OS)
- **Optional**: Vagrant (to simplify setup)

Steps:

Step 1: Install VirtualBox

- Download from: https://www.virtualbox.org/
- Install it on your Windows/Linux/Mac system.

Step 2: Download Ubuntu ISO

- Get the latest version from: https://ubuntu.com/download/desktop

Step 3: Create Virtual Machine

- Open VirtualBox → Click "New"
- Name: DevEnv
- Type: Linux
- Version: Ubuntu (64-bit)
- Memory: 2048 MB (min)
- Create a virtual hard disk (10–20 GB)

Step 4: Install Ubuntu on the VM

- Mount the ISO file as startup disk
- Boot the VM and install Ubuntu as usual

Step 5: Set Up Development Tools

Inside Ubuntu terminal:

bash

CopyEdit

sudo apt update

sudo apt install build-essential git curl vim

sudo apt install nodejs npm

Optional:

bash

CopyEdit

sudo snap install --classic code # VS Code

(10) Write and upload your firstsource code file to Github.

ANS:

Step-by-Step Instructions

1. Create a GitHub Account (if you don't have one)

- Go to: https://github.com
- Sign up and confirm your email.

---

2. Create a New Repository

1. Go to your GitHub profile.
2. Click on **"New"** to create a new repository.
3. Fill in:
    - **Repository name**: first-code
    - **Description**: My first code on GitHub
    - Choose: **Public** or **Private**
    - Check: "Initialize this repository with a README"
4. Click **Create repository**

---

3. Write Your First Source Code

Let's create a simple C program:

hello.c

c

CopyEdit

```
#include <stdio.h>


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

---

4. Upload the File to GitHub

Option A: Using GitHub Web Interface (Easy Way)

1. Go to your repository first-code.

2. Click **"Add file" > "Upload files"**
3. Drag and drop or browse to select your hello.c file.
4. Click **"Commit changes"**

---

5. (Optional) Use Git Command Line (Advanced Way)

🔧 **Setup Git (one-time)**

bash

CopyEdit

git config --global user.name "Your Name"

git config --global user.email "your@email.com"

⬇Clone Your Repo

bash

CopyEdit

git clone https://github.com/your-username/first-code.git

cd first-code

Add Your Code

Save hello.c in this folder.

Push to GitHub

bash

CopyEdit

git add hello.c

git commit -m "Add hello.c - my first program"

git push origin main

Done!

Your hello.c file is now **live on GitHub** 🎉
To check it:

- Go to: https://github.com/your-username/first-code

(11) : : Create a student account on Github and collaborate on a small project with aclassmate.

ANS: Step 1: Create a GitHub Student Account

A. Sign Up for GitHub (if you don't already have an account)

- **Go to: https://github.com**
- **Click Sign Up, enter your email, username, and password.**

B. Apply for the GitHub Student Developer Pack

1. **Visit: https://education.github.com/pack**
2. **Click Get Student Benefits**
3. **Fill out the form:**
    - **Use your school/university email ID (if you have one)**
    - **Upload a photo of your student ID card**
    - **Enter your school name**
4. **Submit and wait for approval (usually 1–2 days)**

Benefits: Free access to tools like Replit, Canva Pro, Namecheap domain, etc.

Step 2: Create a GitHub Repository for the Project

1. **Log in to GitHub**
2. **Click + (top right) → New Repository**
3. **Set up:**
    - **Repository name: student-collab-project**
    - **Description: A simple collaborative project**
    - **Public or Private**
    - **Initialize with a README**
4. **Click Create Repository**

Step 3: Invite Your Classmate as a Collaborator

1. **Go to your new repository**
2. **Click Settings → Collaborators and Teams**
3. **Click "Add People"**

4. **Enter your classmate's GitHub username and Invite**

They'll get an email/invite to accept.

Step 4: Start Collaborating

Option A: Work Using Git and Local Code Editor

Example Project: hello.c

c

CopyEdit

```
#include <stdio.h>
int main() {
   printf("Hello from GitHub Collaboration!\n");
   return 0;
}
```

Git Workflow

For both you and your classmate:

1. **Clone the repo:**

bash

CopyEdit

```
git clone https://github.com/your-username/student-collab-project.git
cd student-collab-project
```

2. **Make changes (add files, edit code)**
3. **Stage and commit:**

bash

CopyEdit

```
git add .
git commit -m "Added hello.c"
```

4. **Push changes:**

bash

CopyEdit

git push origin main

Optional: Use GitHub Projects or Issues

- **Go to Projects tab in the repo**
- **Create a To-Do / In Progress / Done board**
- **Assign tasks using Issues**

(12) Create a student account on Github and collaborate on a small project with aclassmate.

ANS: Part 1: Create a Student Account on GitHub

1. **Go to GitHub's website:**
   https://github.com
2. **Sign Up:**
   - **Click Sign up at the top right.**
   - **Fill in your email, username, and password.**
   - **Verify your account by solving a puzzle or confirming your email.**
3. **Apply for GitHub Student Pack (optional but useful):**
   - **Visit: https://education.github.com/pack**
   - **Click Get Student Benefits.**
   - **Verify your student status using:**
     - **Your school email address (like you@college.edu), or**
     - **Upload a student ID card or fee receipt.**
   - **Approval may take a few days.**

Part 2: Create a Repository for Your Project

1. **Log in to GitHub.**
2. **Click the + icon (top right) → New repository.**
3. **Fill details:**
   - **Repository name: e.g., simple-project**

- **Description: e.g., "A small student project"**
- **Set to Public or Private**
- **Check Initialize this repository with a README**

4. **Click Create repository.**

Part 3: Collaborate with Your Classmate

1. **Go to your new repository.**
2. **Click Settings → Collaborators.**
3. **Click Add people.**
4. **Enter your classmate's GitHub username or email.**
5. **Click Add. They'll get an invite to join.**

Part 4: Clone the Repository and Work Together

You and your classmate should:

1. **Clone the repo to your computer:**

bash

CopyEdit

git clone https://github.com/your-username/simple-project.git

2. **Make changes (add code, files, etc.)**
3. **Push changes:**

bash

CopyEdit

git add .

git commit -m "Added first code file"

git push origin main

Sample Project Idea

"Student Contact Book"
A simple web project using HTML, CSS, and JavaScript to store and display student contact details.

Files:

- **index.html**
- **style.css**
- **script.js**

(13) Create a list of software you use regularly and classify them into the followingcategories: system, application, and utility software.

ANS: 1. System Software

(Manages computer hardware and system resources)

Software Name	Purpose
Windows 10/11	Operating System (OS)
Linux (Ubuntu)	Operating System (OS)
macOS	Operating System (OS)
BIOS/UEFI	Hardware initialization
Device Drivers	Enable communication with devices

---

2. Application Software

(Used by the user to perform specific tasks)

Software Name	Purpose
Google Chrome	Web browsing
Microsoft Word	Word processing
Zoom / Google Meet	Video conferencing
VS Code / Notepad++	Code and text editing
Adobe Photoshop	Image editing
Spotify / VLC Media Player	Music/Video playback
WhatsApp / Telegram	Messaging & communication

Software Name	Purpose
Tally / MS Excel	Accounting and data analysis
BGMI / Free Fire	Gaming

---

3. Utility Software

(Helps in maintaining, analyzing, or optimizing a system)

Software Name	Purpose
Antivirus (Avast, Quick Heal)	Protect from viruses
WinRAR / 7-Zip	File compression
Disk Cleanup	Remove temporary/unnecessary files
Task Manager	Monitor running apps & performance
Backup & Restore	Data recovery
CCleaner	System optimization & cleaning

(14) Follow a GIT tutorial to practice cloning, branching, and merging repositories

ANS: 1. Cloning a Repository

Cloning is copying a remote GitHub repository to my local machine.

Steps I followed:

bash

CopyEdit

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

This command created a local copy of the project.

- **2. Creating and Switching to a New Branch**

Branching helps in working on new features or changes without affecting the main code.

Steps I followed:

bash

CopyEdit

git checkout -b feature-branch

This created a new branch called feature-branch and switched to it.

---

3. Making Changes and Committing

I made a small change in a file (like index.html) and saved it.

bash

CopyEdit

git add .

git commit -m "Added a new feature in feature-branch"

---

4. Merging Branch into Main

After testing my changes, I switched back to the main branch and merged the feature branch.

bash

CopyEdit

git checkout main

git merge feature-branch

- **5. Pushing Changes to GitHub**

To update the remote repository:

bash

CopyEdit

git push origin main

(15) : Write a report on the various types of application software and how they improveproductivity

ANS: Report: Types of Application Software and Their Role in Improving Productivity

1. Introduction

Application software refers to programs designed to perform specific tasks for users. Unlike system software, which manages hardware and system resources, application software directly helps users complete tasks such as writing documents, editing images, or communicating online. In today's digital world, application software significantly boosts productivity across all industries.

---

2. Types of Application Software

a. Word Processing Software

- **Examples: Microsoft Word, Google Docs, LibreOffice Writer**
- **Use: Creating and editing text documents**
- **Productivity Benefit: Enables quick formatting, spell check, templates, and easy sharing of documents.**

---

b. Spreadsheet Software

- **Examples: Microsoft Excel, Google Sheets, LibreOffice Calc**
- **Use: Data organization, calculation, and analysis**
- **Productivity Benefit: Automates complex calculations, supports data visualization, and helps in budgeting and forecasting.**

---

c. Presentation Software

- **Examples: Microsoft PowerPoint, Google Slides, Canva**
- **Use: Creating visual slides for meetings, lectures, or proposals**
- **Productivity Benefit: Helps in presenting ideas clearly and engagingly, saving preparation time.**

---

d. Database Management Software

- **Examples: Microsoft Access, MySQL, Oracle DB**
- **Use: Storing and managing large amounts of data**
- **Productivity Benefit: Ensures efficient data retrieval, storage, and backup for businesses.**

---

e. Multimedia Software

- **Examples: Adobe Photoshop, VLC Media Player, Audacity**
- **Use: Image, video, and audio editing or playback**
- **Productivity Benefit: Assists in content creation, marketing, and entertainment, which are vital for digital industries.**

---

f. Communication Software

- **Examples: Zoom, Microsoft Teams, WhatsApp Web, Slack**
- **Use: Messaging, calling, and virtual meetings**
- **Productivity Benefit: Enables real-time collaboration, remote work, and team communication from anywhere.**

---

g. Web Browsers

- **Examples: Google Chrome, Mozilla Firefox, Microsoft Edge**
- **Use: Accessing internet-based services**
- **Productivity Benefit: Provides quick access to online tools, research, and cloud-based applications.**

---

h. Educational Software

- **Examples: Duolingo, Khan Academy, Google Classroom**
- **Use: E-learning and virtual classrooms**
- **Productivity Benefit: Supports self-paced learning and helps students and teachers manage education online.**

---

i. Project Management Software

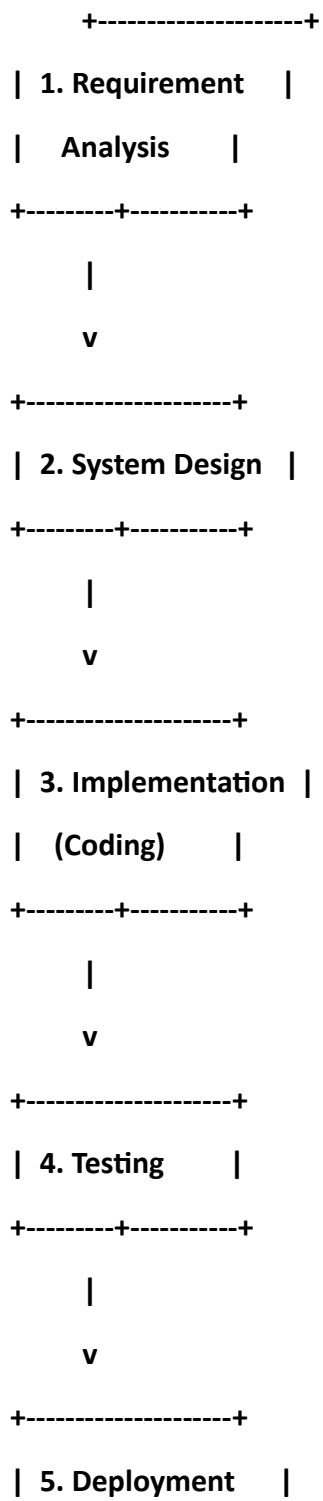
- **Examples: Trello, Asana, Microsoft Project**
- **Use: Task tracking, planning, and team collaboration**
- **Productivity Benefit: Enhances time management, accountability, and team productivity.**

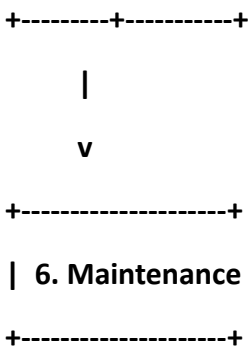
3. Conclusion

Application software has revolutionized the way individuals and organizations function. From creating documents to managing complex projects and communicating with teams, these tools significantly enhance productivity, save time, and reduce manual effort. As technology advances, the role of application software will continue to grow, making work faster, easier, and more efficient.

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

ANS:





Description of Each Phase:

1. **Requirement Analysis: Gathering functional and non-functional requirements from stakeholders.**
2. **System Design: Creating architecture, user interfaces, database design, etc.**
3. **Implementation (Coding): Writing code based on the design documents.**
4. **Testing: Verifying that the software works as expected (unit, integration, system testing).**
5. **Deployment: Releasing the product to users.**
6. **Maintenance: Fixing bugs and updating the software post-deployment.**

(17) Write a requirementspecification for a simple library managementsystem.

ANS: Software Requirements Specification (SRS)

Project Title: Simple Library Management System
Prepared by: *Your Name*
Date: *[Insert Date]*

---

1. Introduction

1.1 Purpose

The purpose of this document is to outline the requirements for the development of a Library Management System (LMS) to manage book records, user information, and borrowing/returning processes efficiently.

1.2 Scope

The Library Management System will allow:

- **Admins to manage books and users.**

- **Students to search, borrow, and return books.**
- **The system to keep track of due dates and availability.**

1.3 Intended Users

- **Librarian (Admin)**
- **Students/Users**

1.4 Technologies Used

- **Frontend: HTML, CSS, JavaScript**
- **Backend: Node.js / PHP / Python (any one)**
- **Database: SQLite / MySQL**

---

2. Functional Requirements

2.1 User Authentication

- **Login and logout functionality for users and admins.**
- **Only admins can add/remove books or users.**

2.2 Book Management

- **Add new books with details: title, author, ISBN, category.**
- **Edit/delete existing book records.**
- **Search for books by title, author, or ISBN.**
- **View book availability status.**

2.3 User Management

- **Add/edit/delete student user accounts (admin only).**
- **View user borrowing history.**

2.4 Book Borrowing and Return

- **Allow users to borrow available books.**
- **Record issue date and calculate due date.**
- **Update book status on return.**
- **Fine calculation if returned late (optional).**

2.5 Search and Filter

- **Search functionality for books by keyword.**

- **Filter by category or author.**

---

3. Non-Functional Requirements

3.1 Performance

- **The system should be responsive and load within 2 seconds for major operations.**

3.2 Security

- **Passwords should be securely stored (hashed).**
- **Admin functionalities should be protected via roles.**

3.3 Usability

- **Easy-to-use interface for both students and admin.**
- **Help section or user manual included.**

---

4. System Requirements

4.1 Hardware Requirements

- **Minimum 4GB RAM and 500MB disk space.**
- **Internet browser (for web version).**

4.2 Software Requirements

- **Operating System: Windows/Linux**
- **Web Server: XAMPP/WAMP (for PHP) or Node.js runtime**
- **Database: MySQL or SQLite**

---

5. Assumptions and Constraints

- **The system is used only within one institution.**
- **Users must return books before borrowing new ones.**
- **Each user can borrow a maximum of 3 books.**

(18) Perform a functional analysis for an online shopping system

ANS: Functional Analysis – Online Shopping System

---

1. User Functions

1.1 User Registration & Login

- **Register a new user with email, password, address.**
- **Login with credentials.**
- **Forgot password/reset password.**
- **Profile management (edit name, address, password, etc.).**

1.2 Product Browsing

- **View product categories (e.g., electronics, clothing).**
- **Search products by name, brand, or tags.**
- **Filter/sort by price, popularity, rating.**

1.3 Product Details View

- **View product name, image, price, description.**
- **See available stock and reviews.**

1.4 Shopping Cart

- **Add products to cart.**
- **Update quantity or remove items from cart.**
- **View total price before checkout.**

1.5 Checkout & Payment

- **Enter delivery address or select saved one.**
- **Choose payment method (e.g., credit card, UPI, cash on delivery).**
- **Confirm and place order.**

1.6 Order Tracking

- **View order status: Placed → Packed → Shipped → Delivered.**
- **Cancel an order (if not shipped).**
- **Download invoice or receipt.**

1.7 Feedback & Rating

- **Rate and review products.**
- **Report product issues or raise complaints.**

---

2. Admin Functions

2.1 Product Management

- **Add/edit/delete product listings.**
- **Update product stock, price, category.**

2.2 Order Management

- **View all user orders.**
- **Update order status (processed, shipped, delivered).**
- **Handle returns/refunds.**

2.3 User Management

- **View registered users.**
- **Block/unblock users.**

2.4 Report Generation

- **Sales reports (daily/weekly/monthly).**
- **Inventory reports.**
- **User activity reports.**

---

3. Common/Shared Functions

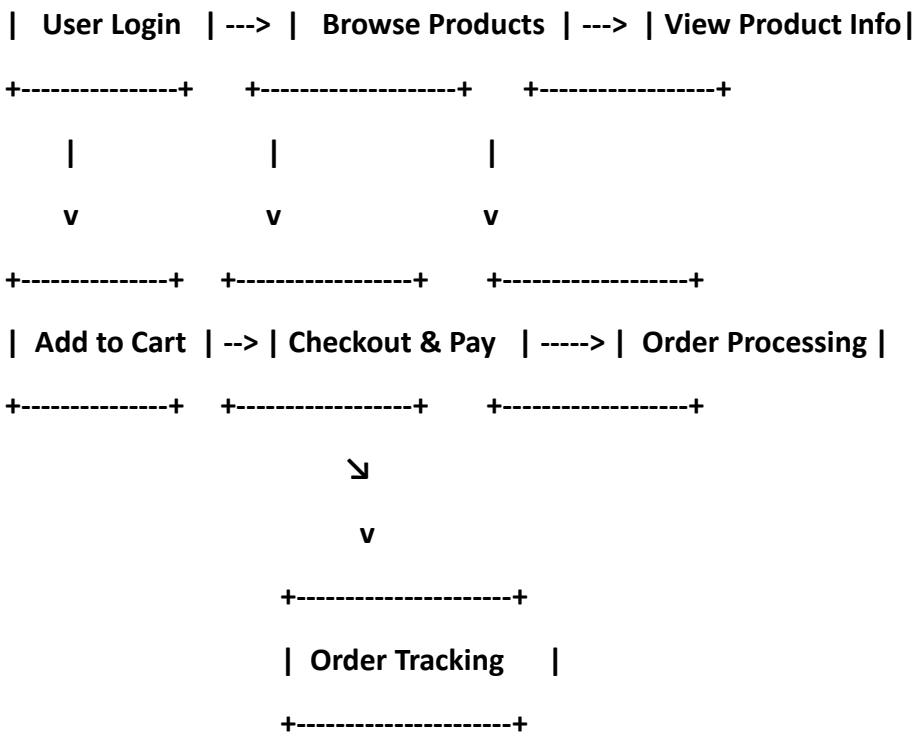
- **Authentication & Authorization**
  Role-based access (admin vs. user).
- **Search System**
  Universal search across categories and products.
- **Notifications**
  Email/SMS alerts for order updates or offers.
- **Security**
  Secure payment gateway integration, data encryption, input validation.

Functional Diagram (Text Version)

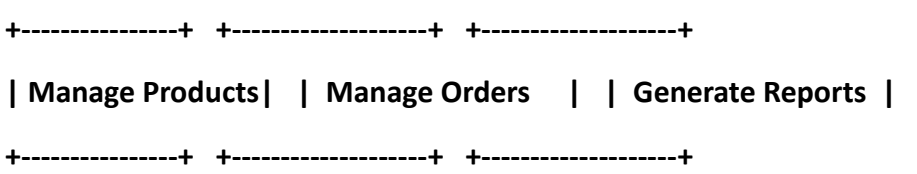
sql

CopyEdit

+----------------+ +--------------------+ +------------------+



ADMIN PANEL:



(19) Design a basic system architecture for a food delivery app

ANS: Food Delivery App – System Architecture

1. Presentation Layer (Frontend)

Interfaces users interact with:

- **Customer App**
  - **Browse restaurants**
  - **View menu, place orders, track delivery**
  - **Payments and reviews**
- **Delivery Partner App**
  - **View assigned orders**

    - **Navigation and delivery updates**
- **Restaurant Dashboard**
    - **Manage menu, view orders**
    - **Update availability and delivery status**
- **Admin Web Panel**
    - **Monitor system, manage users, generate reports**

---

2. Application Layer (Backend/Business Logic)

Handles core functionalities:

- **User Management Service**
    - **Login, signup, profile, role handling**
- **Restaurant Management**
    - **Menu items, pricing, availability**
- **Order Management System**
    - **Order placement, status updates, notifications**
- **Delivery Management System**
    - **Assign delivery agents, route optimization, live tracking**
- **Payment Gateway Integration**
    - **Online payments via UPI, cards, wallets**
- **Rating & Review System**
    - **Feedback on food and delivery**

---

3. Data Layer (Database & Storage)

Stores all persistent data:

- **Relational Database (e.g., MySQL/PostgreSQL)**
    - **Users, restaurants, orders, reviews**
- **NoSQL Database (e.g., MongoDB, Redis)**
    - **Caching user sessions, search history**
- **File Storage (e.g., AWS S3 or Cloudinary)**

- Images of food, menus, profile photos

---

4. External Services / APIs

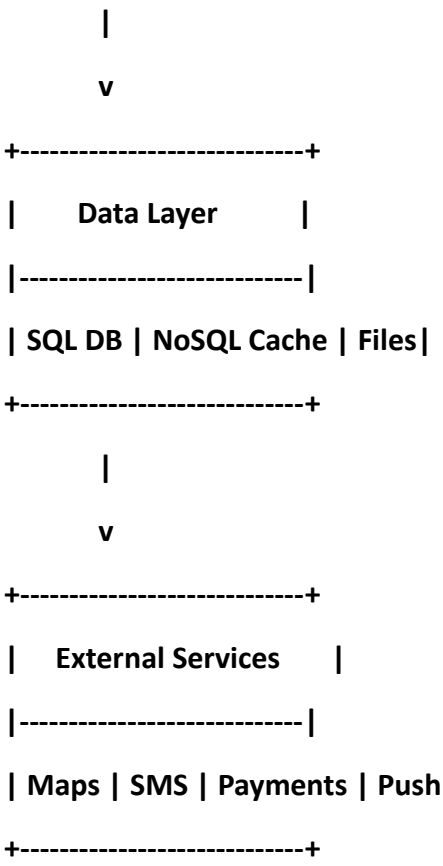
- **Maps & Location APIs (e.g., Google Maps)**
  - **Live location tracking and delivery route mapping**
- **SMS/Email Notification Services**
  - **OTPs, order confirmations, promotions**
- **Push Notification Service**
  - **Firebase or OneSignal for real-time updates**
- **Analytics & Logging**
  - **User behavior, error logging, app usage statistics**

Diagram (Text View)

sql

CopyEdit

```
+-----------------------------+
|    User Interface Layer     |
|-----------------------------|
| Customer | Delivery | Admin |
|   App    |  App     | Panel |
+-----------------------------+
          |
          v
+-----------------------------+
|    Application Layer        |
|-----------------------------|
| Auth  | Orders | Payments   |
| Menu  | Delivery| Ratings   |
+-----------------------------+
```



(20) Develop test cases for a simple calculator program

ANS: Test Case Table – Simple Calculator

Test Case ID	Test Case Description	Inputs	Expected Output	Status
TC001	Test Addition of two positive numbers	5 + 3	8	Pass/Fail
TC002	Test Addition of negative and positive	-5 + 10	5	Pass/Fail
TC003	Test Subtraction of two numbers	10 - 4	6	Pass/Fail
TC004	Test Subtraction resulting in negative	3 - 7	-4	Pass/Fail
TC005	Test Multiplication of two numbers	6 * 7	42	Pass/Fail

Test Case ID	Test Case Description	Inputs	Expected Output	Status
TC006	Test Multiplication with zero	9 * 0	0	Pass/Fail
TC007	Test Division of two numbers	12 / 4	3	Pass/Fail
TC008	Test Division by 0 (should handle error)	5 / 0	Error/Exception	Pass/Fail
TC009	Test Decimal Addition	2.5 + 1.75	4.25	Pass/Fail
TC010	Test Invalid Input (e.g., string)	"abc" + 5	Error/Invalid Input Msg	Pass/Fail
TC011	Test Clear/Reset Function	Press 'C' after input	All fields cleared	Pass/Fail
TC012	Test Chained Operation (if supported)	2 + 3 * 4	14 or Error (if not supported)	Pass/Fail
TC013	Test Negative Division	-12 / 3	-4	Pass/

(21) Document a real-world case where a software application required criticalmaintenance.

ANS: Study: Critical Maintenance of WhatsApp During Global Outage (2022)

Application: WhatsApp (by Meta)

Incident Date: October 25, 2022

Issue: Global Service Outage

Downtime Duration: ~2 hours

Impact Area: Worldwide (including India, Europe, and the US)

Affected Users: Over 2 billion users

Issue Summary

On October 25, 2022, WhatsApp experienced a major global outage that prevented users from sending or receiving messages, accessing group chats, or using media attachments. Both mobile and web platforms were down.

Symptoms Observed

- **Messages stuck on "Sending"**
- **Group chats not updating**
- **No connection to WhatsApp servers**
- **Web client unable to sync**

Impact

- **Communication disruption for businesses and individuals**
- **Failure of customer support services relying on WhatsApp Business**
- **Negative user sentiment and social media outrage**

Cause of the Issue (As Per Meta)

Meta did not disclose a detailed technical root cause but indicated it was due to a server-side configuration change that triggered the outage—requiring urgent rollback and re-deployment.

Critical Maintenance Actions Taken

1. **Immediate Incident Response Activated**
   Meta's engineering team detected service degradation and started rollback procedures.
2. **Rollback of Recent Configuration**
   A faulty backend configuration (possibly affecting routing or authentication) was reversed.
3. **Server Cluster Synchronization**
   Backend services were resynced and restarted to restore user connections.
4. **Monitoring and Validation**
   Continuous monitoring was performed to ensure global restoration.

Outcome

- **Services began restoring within 90–120 minutes.**
- **Full functionality resumed globally.**
- **WhatsApp acknowledged the issue via Twitter/X and assured users.**

Lessons Learned

- **Even large-scale, mature applications like WhatsApp can suffer from deployment issues.**
- **Critical maintenance must include:**
  - **Strong rollback mechanisms**
  - **Real-time monitoring tools**
  - **Communication channels with users**

Conclusion

This case shows that critical maintenance is essential, especially for globally used platforms. Minor backend misconfigurations can escalate into major disruptions, underlining the need for robust DevOps practices.

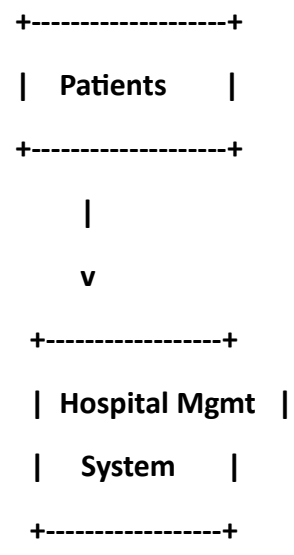
(22) Create a DFD for a hospital managementsystem.

ANS: Level 0 DFD (Context Level)

This is the highest-level overview showing the system as a single process with external entities.

pgsql

CopyEdit



```
        ^     |     ^
        |     v     |
  +--------------+  +-----------------+  +-----------------+
  | Doctors      |  | Admin/Staff     |  | Billing Dept    |
  +--------------+  +-----------------+  +-----------------+
```

External Entities:

- **Patients: Provide personal info, book appointments, view prescriptions.**
- **Doctors: Access patient records, update treatments.**
- **Admin/Staff: Manage appointments, update patient data.**
- **Billing Dept: Generates bills based on services.**

🔹 **Level 1 DFD**

Breaks down the system into sub-processes.

csharp

CopyEdit

[1.0] Patient Registration <----- Patients

|

v

[2.0] Appointment Scheduling <--- Admin/Staff

|

v

[3.0] Medical Treatment <------ Doctors

|

v

[4.0] Pharmacy/Reports

|

v

[5.0] Billing System <-------- Billing Dept

🗃️ Data Stores:

- **D1: Patient Database – Stores patient details, reports**
- **D2: Appointment Records**
- **D3: Medical Records**
- **D4: Billing Records**

📘 **Process Descriptions:**

Process No.	Name	Description
1.0	Patient Registration	Collects patient data and stores it in Patient DB
2.0	Appointment Scheduling	Schedules appointment with available doctor
3.0	Medical Treatment	Doctor diagnoses and updates treatment records
4.0	Pharmacy/Reports	Generates prescriptions, test results
5.0	Billing System	Calculates charges, creates and prints bill

(23) Build a simple desktop calculator application using a GUI library.

ANS: Python Code: GUI Calculator (Tkinter)

python

CopyEdit

```
import tkinter as tk

def press(key):
    entry.insert(tk.END, key)

def clear():
    entry.delete(0, tk.END)

def calculate():
    try:
```

```
            result = eval(entry.get())
            entry.delete(0, tk.END)
            entry.insert(tk.END, str(result))
        except Exception:
            entry.delete(0, tk.END)
            entry.insert(tk.END, "Error")

# Create main window
root = tk.Tk()
root.title("Simple Calculator")
root.geometry("300x400")
root.resizable(False, False)

# Input field
entry = tk.Entry(root, font=("Arial", 20), bd=10, relief=tk.RIDGE, justify='right')
entry.grid(row=0, column=0, columnspan=4, padx=10, pady=20)

# Button labels
buttons = [
    '7', '8', '9', '/',
    '4', '5', '6', '*',
    '1', '2', '3', '-',
    '0', '.', 'C', '+'
]

# Create buttons dynamically
row, col = 1, 0
for button in buttons:
    action = lambda x=button: press(x) if x not in ['C'] else clear()
```

```
    tk.Button(root, text=button, width=5, height=2, font=("Arial", 18),
              command=action).grid(row=row, column=col, padx=5, pady=5)
    col += 1
    if col > 3:
        col = 0
        row += 1

# Equal button
tk.Button(root, text='=', width=22, height=2, font=("Arial", 18),
          command=calculate).grid(row=row, column=0, columnspan=4, padx=5, pady=10)

# Start the GUI
root.mainloop()
```

💡 **Features:**

- **Clean interface with number and operation buttons.**
- **Evaluates input using Python's eval() (safe here, but for real apps use a parser).**
- **Handles basic error cases (e.g., divide by zero or invalid input).**

📓 **Requirements:**

- **Python installed (3.x)**
- **No external libraries needed — uses built-in tkinter.**

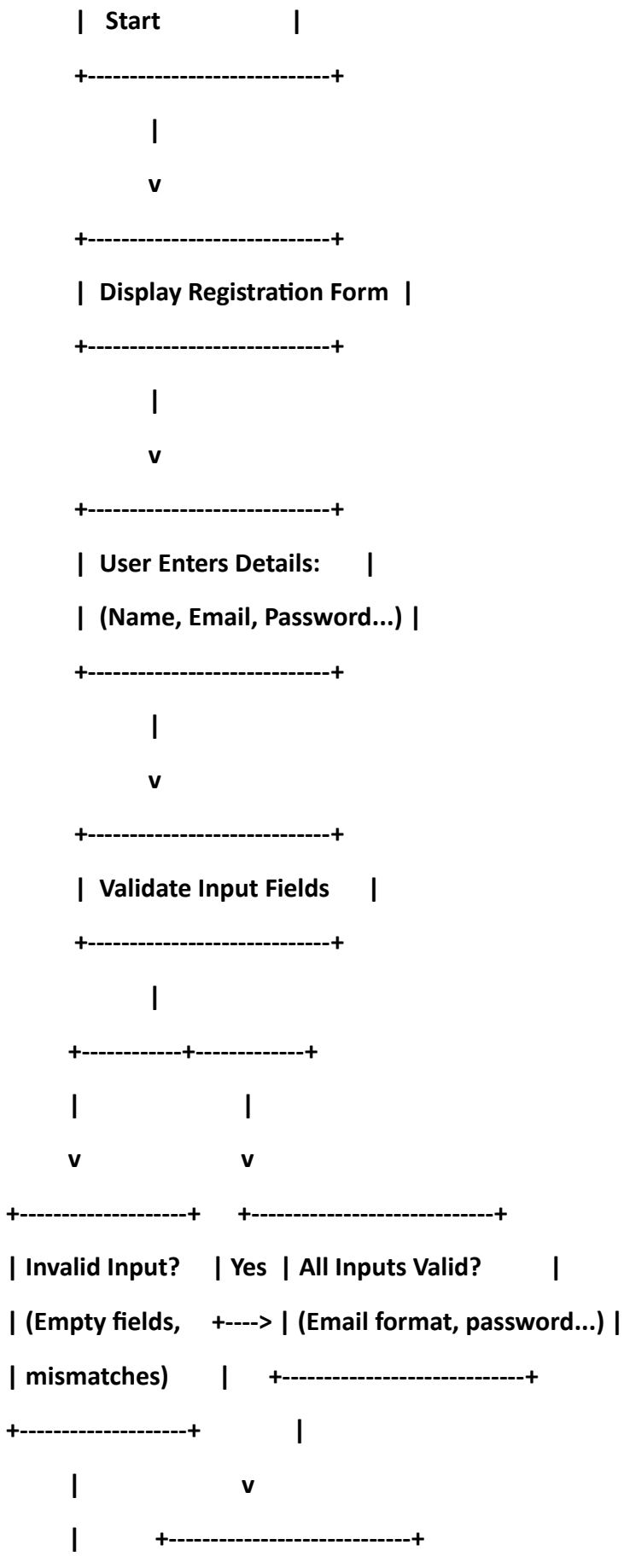
(24) Draw a flowchart representing the logic of a basic online registration system.

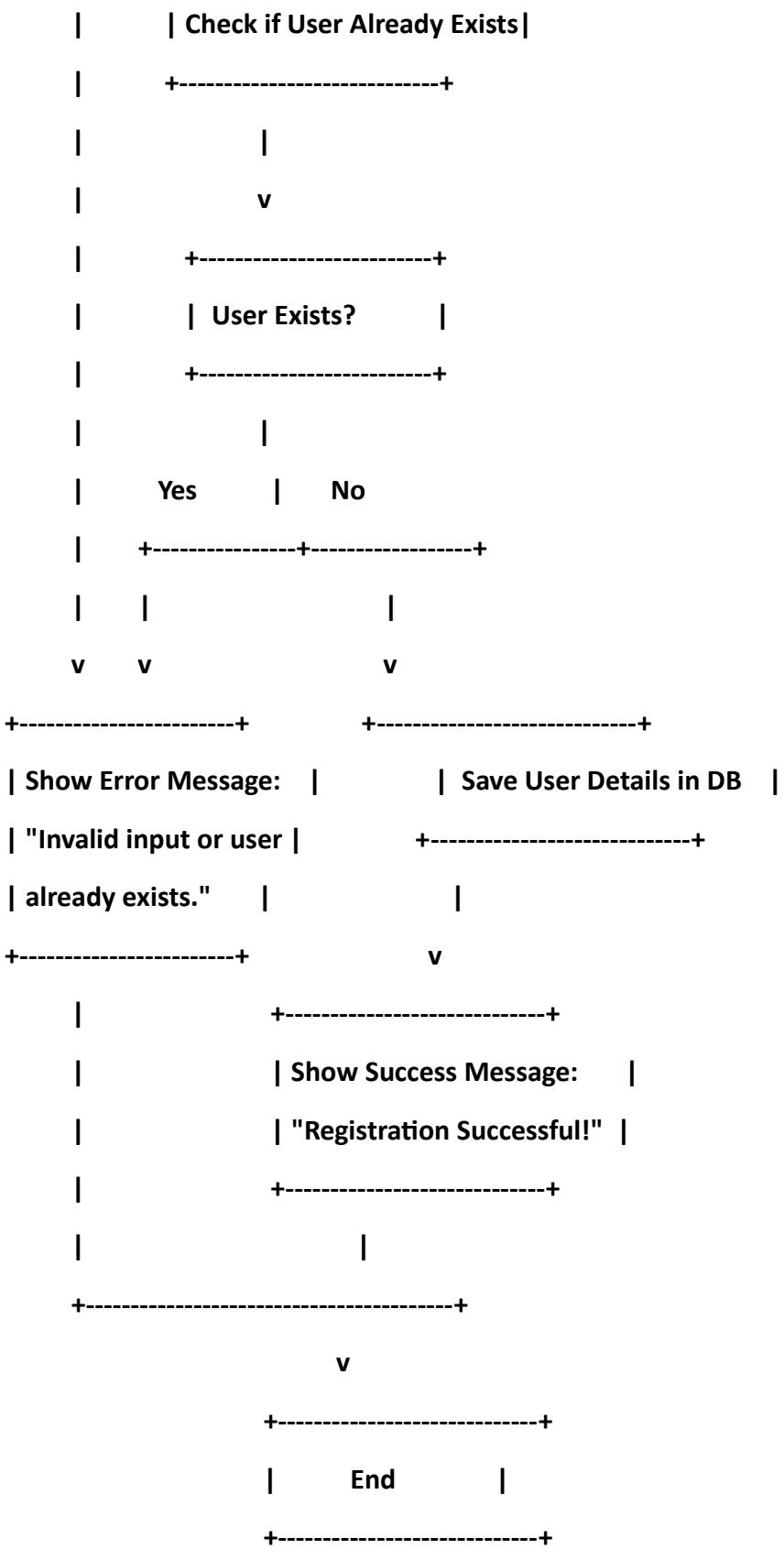
ANS: Flowchart: Online Registration System Logic

sql

CopyEdit

```
        +----------------------------+
```


Start
Display Registration Form
User Enters Details:
(Name, Email, Password...)
Validate Input Fields
Invalid Input?
(Empty fields,
mismatches)
Yes
All Inputs Valid?
(Email format, password...)



✅ **Key Features Represented**

- **Form display and user input**
- **Input validation**
- **Duplicate account check**
- **Data saving (e.g., to a database)**
- **User feedback (success or error)**