MODULE 1 – OVERVIEW OF IT INDUSTRY

What is a Program?

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

1. C

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

Explanation:

- Uses #include <stdio.h> for input/output functions like printf.
- main() is the entry point of the program.
- printf() prints text; \n adds a newline.
- return 0; indicates successful execution.

2. C++

```
#include <iostream>  // Header for cout using
namespace std; // Avoids writing std:: each time

int main() {     cout << "Hello,
World!" << endl;     return 0; //
Exit status
}
```

Explanation:

- Uses #include <iostream> for input/output streams.
- cout is an output stream object; << is the stream insertion operator.
- endl adds a newline and flushes the output buffer.
- using namespace std; allows using standard library names without std:: prefix.

Compare the structure and syntax

Feature	C (printf)	C++ (cout)
Header File	<stdio.h>	<iostream>
Printing Function	printf("Hello, World!\n");	cout << "Hello, World!" << endl;
Namespace	Not applicable	Often uses using namespace std;
Style	Procedural	Can be procedural or object-oriented
Output Method	C-style function call	Stream-based operator overloading
Standard	C Standard Library	C++ Standard Library (STL)

World Wide Web & How Internet Works

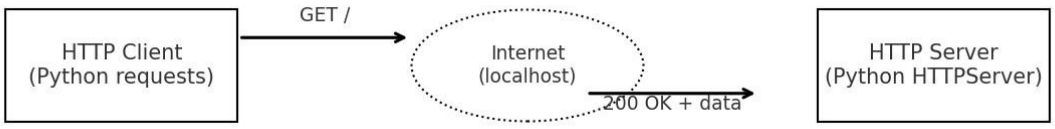
Research and create a diagram of how data is transmitted from a client to a server over the internet.

Explanation:

1. Client sends a request (e.g., a web page request).
2. The request travels through the Internet (routers, cables, etc.).
3. Server receives the request and processes it.
4. Server sends a response back through the Internet to the Client.

Network Layers on Client and Server

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



Here's the HTTP client-server communication diagram for the Python example.

1. The client sending a GET / request to the server through the Internet (in our case localhost).
2. The server responding with 200 OK and the message data.

Types of Internet Connections

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

1. Broadband (DSL / Cable Internet)

Description: Uses existing telephone lines (DSL) or coaxial cables (Cable) to deliver highspeed internet.

Pros:

- Widely available in cities and towns.
- Generally affordable.
- Stable connection for daily use (streaming, browsing, video calls).

Cons:

- Slower upload speeds compared to fiber.
- Performance may drop during peak hours.
- Speed depends on distance from the provider's network hub (DSL) or number of users on the network (Cable).

2. Fiber-Optic Internet

Description: Uses fiber-optic cables to transmit data as light signals, offering very high speeds.

Pros:

- Extremely fast speeds (both upload and download).
- Very low latency — excellent for gaming, video conferencing, and cloud work.
- Reliable even during high-traffic periods.

Cons:

- Limited availability (mainly in cities and developed areas).
- Higher installation costs.
- Can take longer to set up.

3. Satellite Internet

Description: Sends data to and from satellites orbiting Earth, accessible almost anywhere.

Pros:

- Available in remote and rural areas where no other connection is possible.
- Doesn't require cables or physical network infrastructure locally.

Cons:

- High latency (signal travels to space and back).
- More expensive than broadband or fiber for similar speeds.
- Affected by weather conditions.
- Lower data caps in some plans.

Protocols

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

1. Simulate an HTTP GET Request bash

curl https://example.com

- This will fetch the HTML source of the webpage.
- https://example.com can be replaced with any URL.

2. Simulate an HTTP POST Request

If you want to send data (e.g., to an API):

bash

curl -X POST https://httpbin.org/post -d "name=John&age=25"

- -X POST → method type.
- -d → send data in the body.

3. Download a File via HTTP bash

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

- -O saves the file with the same name as on the server.

4. Simulate an FTP Request (Anonymous Login) bash

curl ftp://speedtest.tele2.net/1MB.zip -O

- Downloads 1MB.zip from the FTP server.

5. Simulate an FTP Request with Username & Password bash

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

- -u sets FTP credentials.
- -O saves the file locally.

6. Upload a File via FTP bash

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

- -T → specify the file to upload.

Application Security

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

1. SQL Injection

- What happens? Hacker puts bad code in a form to control your database.
- Risk: Data stolen or deleted.
- Fix: Always clean/check user input and use *prepared queries*.

2. XSS (Cross-Site Scripting)

- What happens? Hacker puts bad JavaScript in your site so it runs on visitors' browsers.
- Risk: Steals cookies, sends users to fake sites.
- Fix: Escape HTML output and block untrusted scripts.

3. CSRF (Cross-Site Request Forgery)

- What happens? Hacker tricks you into clicking a link that does something on a site where you're already logged in.
- Risk: Your account does things you didn't want.
- Fix: Use *CSRF tokens* and ask for password again for important actions.

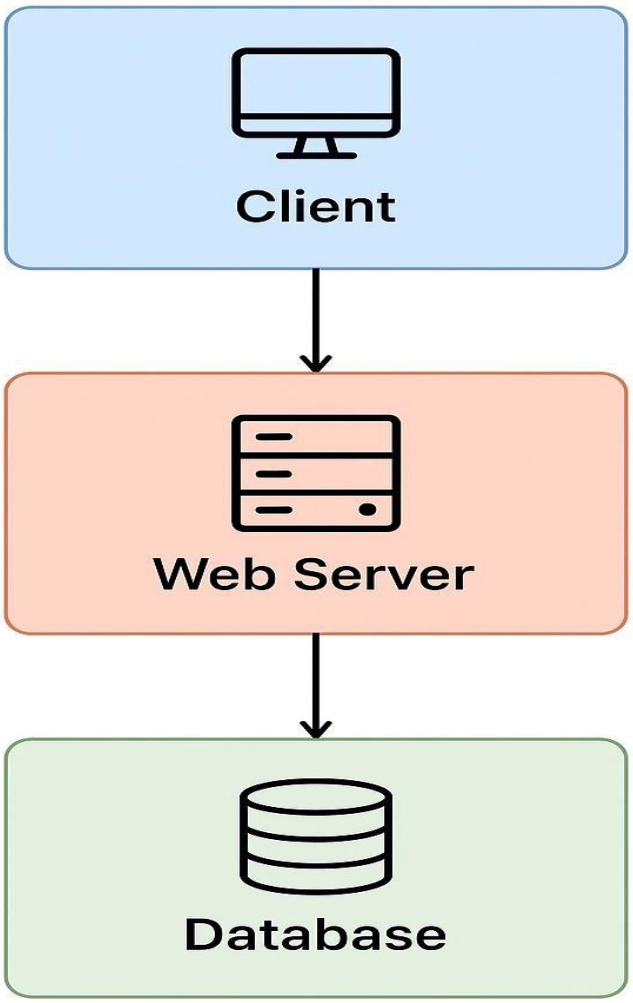
Software Applications and Its Types

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

Application	Type	Why?
Google Chrome	Application software	Used for browsing the internet.
Microsoft Word	Application software	Used for creating and editing documents.
WhatsApp	Application software	Messaging and calling app for communication.
Windows 10 / 11	System software	Operating system that runs your computer.
Antivirus (e.g., Avast)	System software	Protects the system from viruses and malware.

Software Architecture

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



Layers in Software Architecture

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

1.Case Study: Online Banking System 1. Presentation Layer

- **Function: User interface where customers log in, check balances, and transfer money.**
- **Example:** Mobile banking app or website.

2. Business Logic Layer

- **Function:** Processes transactions, checks rules (e.g., sufficient balance, transfer limits).
- **Example:** Validates transfer request, calculates fees, updates account balance.

3. Data Access Layer

- **Function:** Reads/writes data to the bank's database.
- **Example:** Retrieves account balance, updates transaction history.

Workflow:

1. User requests a fund transfer in the app (Presentation).
2. System checks balance and processes the request (Business Logic).
3. Transaction is stored in the database (Data Access).

Software Environments

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

1. Development Environment

- Where developers write and test code.
- Has debugging tools, text editors, and runtime environments.
- Usually runs on individual machines or isolated containers.
- Goal: Fast iteration, no risk of affecting users.

2. Testing / QA Environment

- Used for running automated and manual tests.
- Mirrors the production environment more closely.
- May include CI/CD pipelines.
- Goal: Validate code quality, catch bugs before deployment.

3. Production Environment

- Live environment used by end-users.
- Highly secured, optimized, and monitored.
- Changes go through controlled deployments.
- Goal: Deliver a stable and reliable user experience.

Set up a basic environment in a virtual machine

Quick Set-Up in a Virtual Machine (Ubuntu)

1. Create & Launch VM
    - Use VirtualBox (or similar).
    - Install Ubuntu (20.04 LTS recommended).
2. Install Basic Tools

    bash

    sudo apt update && sudo apt install -y git curl
3. Configure Node.js (for a dev web app)

bash curl -fsSL https://deb.nodesource.com/setup_18.x | sudo -E

bash -

sudo apt install -y nodejs

4. Create & Run a Simple Server

bash echo 'console.log("Hello Dev!");' >

app.js

node app.js

- Visit the server via the VM's IP and port.

Source Code

Write and upload your first source code file to Github.

1. Make a file

Create a file hello.py with:

python

print("Hello, GitHub!")

2. Open Git Bash in the file's folder

bash

git init

git add hello.py git commit -

m "first commit"

3. Create a repo on GitHub (click + → New repository).

4. Connect and upload

bash git remote add origin https://github.com/YOUR_USERNAME/REPO_NAME.git

git branch -M main

git push -u origin main

Github and Introductions

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

Quick Steps to Set Up & Push Code to GitHub 1. Create a Repository on GitHub

1. Log in to GitHub.
2. Click the **"+" icon** (top-right) → **New repository**.
3. Enter a name, optionally add a description, and click **Create repository** (no need to initialize with README) Mediumkickstartcoding.online.

2. Initialize and Commit Locally bash mkdir my-repo && cd

my-repo # create and enter project folder

git init # start local Git repo

echo "# My Repo" > README.md # create README

git add README.md # stage changes git

commit -m "Initial commit" # commit changes

MediumMedium

3. Link to GitHub & Push bash git remote add origin

https://github.com/USERNAME/my-repo.git git push -u origin

main

(If your default branch is master, use that instead.) Mediumkickstartcoding.onlineGitHub

Student Account in Github

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

https://github.com/Danishchaudhari/student_project

Types of Software

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

1. System Software (Runs the computer)
   - Windows 10/11 – Operating System
   - Linux Ubuntu – Operating System
   - macOS – Operating System

2. Application Software (For specific tasks)

- Google Chrome – Web browser
- Microsoft Word – Document editing
- WhatsApp – Messaging
- Spotify – Music streaming
- Zoom – Video conferencing

3. Utility Software (Maintenance & optimization tools)

- WinRAR – File compression

- CCleaner – System cleaning
- Windows Defender – Security
- 7-Zip – File archiving
- Backup and Restore – Data backup tool

GIT and GITHUB Training

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

Git Practice Tutorial

What You'll Learn:

- How to **clone** a GitHub repository
- How to create a **branch**
- How to **merge** changes back into the main branch

Step 1: Clone a Repository

Cloning means copying a project from GitHub to your computer.

Command:

```
clone https://github.com/your-username/your-repo.git
```

Replace `your-username` and `your-repo` with your GitHub info.

Then go into the project folder:

```
your-repo
```

Step 2: Create a Branch

Creating a branch lets you work on a new feature without changing the main code.

Command:

```
checkout -b my-feature
```

Now, make a change. For example:

```
"Hello from my-feature branch!" > feature.txt
```

Then save the change:

```
feature.txt
commit -m "Add feature.txt in my-feature branch"
```

Step 3: Switch to the Main Branch

Go back to the main branch:

```
checkout main
```

Step 4: Merge the Branch

Merge your changes from `my-feature` into `main`:

```
merge my-feature
```

You should see a success message if there are no conflicts.

Step 5: Push to GitHub

To upload your changes to GitHub:

```
push origin main
```

Application Software

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

Report: Types of Application Software and Their Impact on Productivity Introduction

Application software is a class of computer programs designed to perform specific tasks for users. Unlike system software, which runs the computer itself, application software enables users to complete productive, creative, or communication-related tasks. In today's digital world, application software plays a critical role in enhancing individual and organizational productivity.

Types of Application Software 1. Word Processing Software

- **Examples:** Microsoft Word, Google Docs, LibreOffice Writer
- **Purpose:** Creating, editing, formatting, and printing text-based documents.

- **Productivity Benefit:** Speeds up document creation, supports collaboration, enables efficient text formatting, and automates repetitive tasks with templates and macros.

2. Spreadsheet Software

- **Examples:** Microsoft Excel, Google Sheets, LibreOffice Calc
- **Purpose:** Organizing data, performing calculations, and analyzing numerical information.
- **Productivity Benefit:** Automates complex calculations, creates dynamic reports and charts, and supports data-driven decision-making.

3. Presentation Software

- **Examples:** Microsoft PowerPoint, Google Slides, Keynote
- **Purpose:** Designing and delivering visual presentations.
- **Productivity Benefit:** Enhances communication through visuals, simplifies the sharing of ideas, and supports remote collaboration and audience engagement.

4. Database Management Software

- **Examples:** Microsoft Access, MySQL Workbench, Oracle Database
- **Purpose:** Storing, retrieving, and managing large volumes of data.
- **Productivity Benefit:** Improves data organization, ensures data accuracy, and supports efficient information retrieval and reporting.

5. Project Management Software

- **Examples:** Trello, Microsoft Project, Asana, Monday.com
- **Purpose:** Planning, tracking, and managing projects and tasks.
- **Productivity Benefit:** Streamlines project workflows, enables team coordination, sets deadlines, and monitors progress in real time.

6. Communication Software

- **Examples:** Microsoft Teams, Slack, Zoom, Skype
- **Purpose:** Facilitating messaging, video conferencing, and collaboration.

- **Productivity Benefit:** Reduces communication delays, enables remote work, and integrates with other productivity tools.

7. Graphics and Multimedia Software

- **Examples:** Adobe Photoshop, Canva, Final Cut Pro
- **Purpose:** Creating and editing images, videos, and other media.
- **Productivity Benefit:** Supports creative projects, improves marketing content, and simplifies media editing and design.

8. Web Browsers

- **Examples:** Google Chrome, Mozilla Firefox, Safari
- **Purpose:** Accessing and interacting with content on the internet.
- **Productivity Benefit:** Provides instant access to online resources, cloud apps, and research tools.

9. Accounting and Finance Software

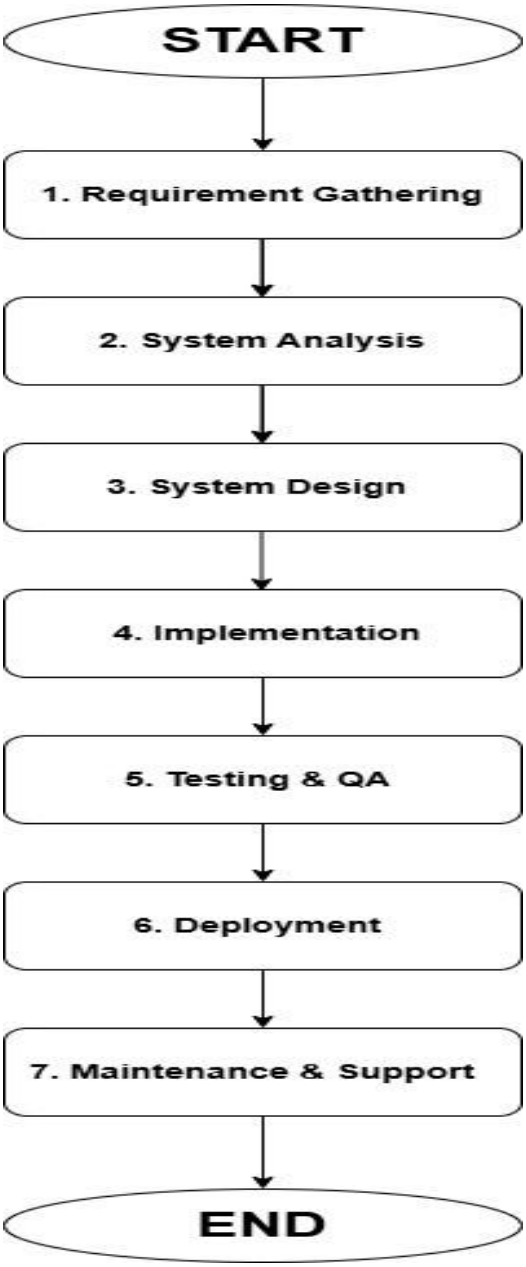
- **Examples:** QuickBooks, FreshBooks, SAP
- **Purpose:** Managing financial transactions, invoicing, and payroll.
- **Productivity Benefit:** Reduces manual errors, automates financial reporting, and ensures regulatory compliance.

10. Educational and e-Learning Software

- **Examples:** Moodle, Duolingo, Khan Academy
- **Purpose:** Delivering training and educational content.
- **Productivity Benefit:** Enhances self-paced learning, facilitates skill development, and supports corporate training initiatives.

Software Development Process

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



Software Requirement

Write a requirement specification for a simple library management system.

Project: Library Management System 1. Purpose

This system will help manage library activities like adding books, registering members, issuing and returning books, and tracking due dates.

2. Users

- **Admin (Librarian)** – Can manage books and members.
- **Member (User)** – Can view available books and borrow them.

3. Features / Functional Requirements

1. **Add Books** – Admin can add, update, or delete books.
2. **Register Members** – Admin can add or remove members.
3. **Search Books** – Users can search for books by name, author, or ID.
4. **Issue Books** – Admin can issue books to members.
5. **Return Books** – Admin can return books and check if they are late.
6. **Fine System** – If a book is returned late, a fine is shown.
7. **Reports** – Admin can see which books are issued, returned, or overdue.

4. Non-Functional Requirements

1. Easy-to-use interface
2. Secure login system
3. Fast and reliable
4. Works on Windows (or any other OS you choose)

5. Tools (Example)

- Front-end: HTML/Python/Java GUI
- Back-end: MySQL or SQLite database

6. Future Ideas

- Send reminders for return dates
- Members can reserve books online

Software Analysis

Perform a functional analysis for an online shopping system.

Functional Analysis – Online Shopping System 1. User Functions • Account Management

- Sign up, login, and logout
- Update profile information

- **Product Search & Browsing**
  - Search by name, category, or filter by price/brand

- View product details (images, description, price)

- **Shopping Cart Management** ○ Add/remove products from the cart
  - Update quantity
- **Checkout & Payment** ○ Select shipping address & payment method
  - Apply discount codes
- **Order Tracking**
  - View order history and order status

2. Admin Functions • **Product Management**

- Add, update, or delete products

- **Order Management**
  - View, confirm, and ship orders
- **User Management**
  - View or block customer accounts
- **Reports** ○ Sales reports, inventory reports

3. System Functions • **Authentication & Authorization** ○ Verify user credentials

- Restrict admin functions to admins only

- **Database Management**
  - Store user data, product details, orders, and payment info securely
- **Notification System**
  - Email/SMS alerts for order confirmation and shipping updates

System Design

Design a basic system architecture for a food delivery app.

1. Presentation Layer (Frontend)

- Mobile App / Web App
  - Customer app (browse restaurants, order food, track delivery) ○ Delivery partner app (accept delivery, navigation)
  - Restaurant app (manage menu, accept orders)

2. Business Logic Layer (Backend Server)

- Order Management – Process new orders, assign delivery
- Menu Management – Update and manage food items and prices
- Payment Processing – Secure online payment integration
- Delivery Tracking – Live location tracking of delivery partners
- Notification System – Push notifications, SMS updates

3. Data Access Layer (Database)

- Customer Data – Profiles, addresses, preferences
- Restaurant Data – Menus, ratings, opening hours
- Order Data – Order history, current orders, delivery details
- Payment Data – Transaction records

4. External Services

- Payment Gateway – Razorpay, PayPal, Stripe
- Map API – Google Maps for delivery tracking
- SMS/Email API – Order confirmation, delivery updates

Software Testing

Develop test cases for a simple calculator program.

Test Case ID	Description	Input	Expected Output	Remarks
TC01	Addition of two positive numbers	5 + 3	8	Pass if result is correct
TC02	Addition of positive and negative number	5 + (-3)	2	Handles negative numbers
TC03	Subtraction of two numbers	10 - 4	6	Pass if result is correct
TC04	Multiplication of two numbers	7 × 3	21	Pass if result is correct
TC05	Division of two numbers	8 ÷ 2	4	Pass if result is correct
TC06	Division by zero	5 ÷ 0	Error / Infinity	Should handle division by zero gracefully

TC07	Decimal number addition	2.5 + 1.2	3.7	Pass if decimal support works
TC08	Multiplication with zero	9 × 0	0	Pass if correct
TC09	Negative × Negative	-4 × -5	20	Pass if sign handling correct
TC10	Clear/Reset function	Input: any value → Press clear	0	Pass if display resets

Maintenance

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

Critical Maintenance Case – Microsoft Windows 10 Security Patch (2021)

Background:

- In 2021, Microsoft discovered a serious security vulnerability in Windows 10 called PrintNightmare. This bug allowed attackers to gain control of a system through the Windows Print Spooler service.

Reason for Maintenance:

- The vulnerability could let hackers install programs, view/change/delete data, or create new user accounts with full rights.

Type of Maintenance:

- Actions Taken:

- Microsoft released an emergency security patch outside their regular update schedule.
- Users were advised to install the patch immediately or disable the Print Spooler service until updated.

✦ Outcome:

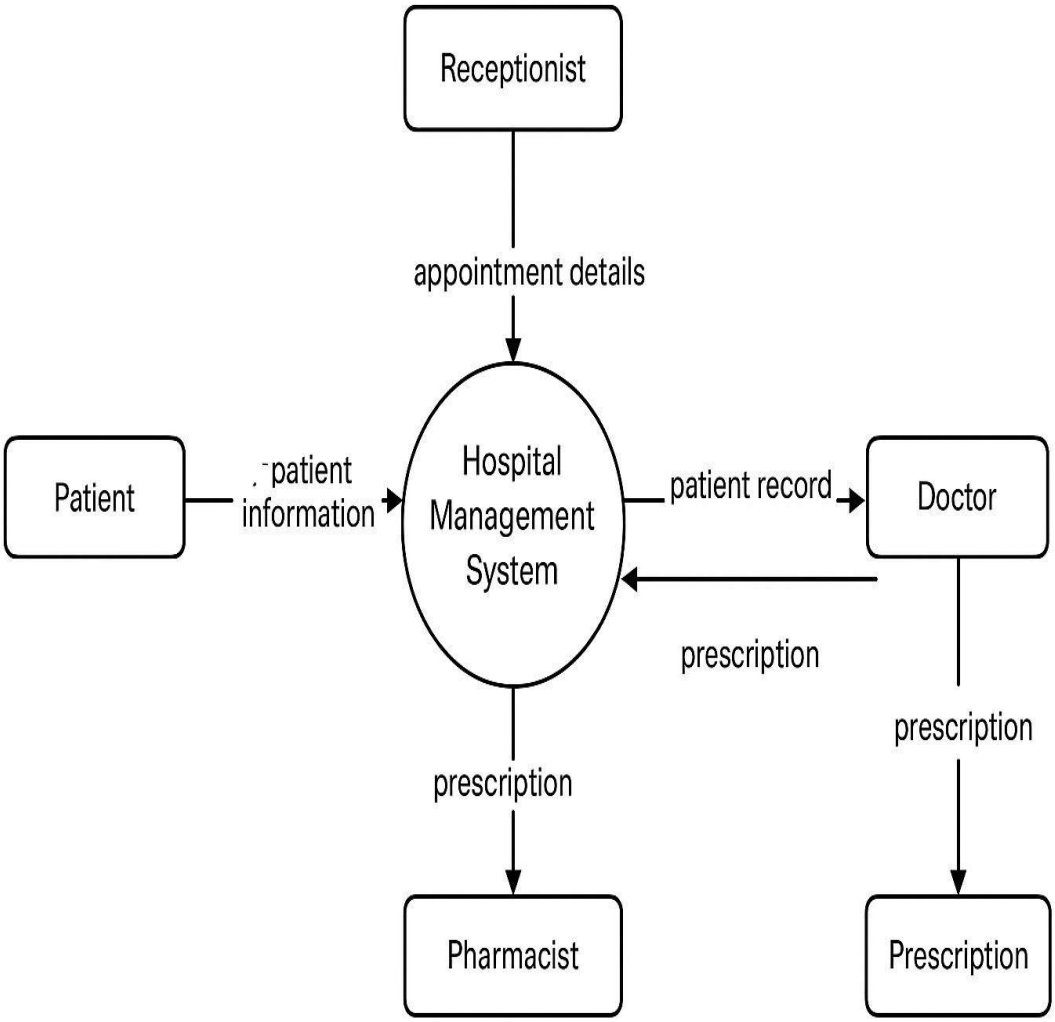
- Millions of devices were protected from potential attacks.

✦ Lesson Learned:

- Critical vulnerabilities must be patched immediately to prevent large-scale cyber threats.

DFD (Data Flow Diagram)

Create a DFD for a hospital management system.



Desktop Application

Build a simple desktop calculator application using a GUI library.

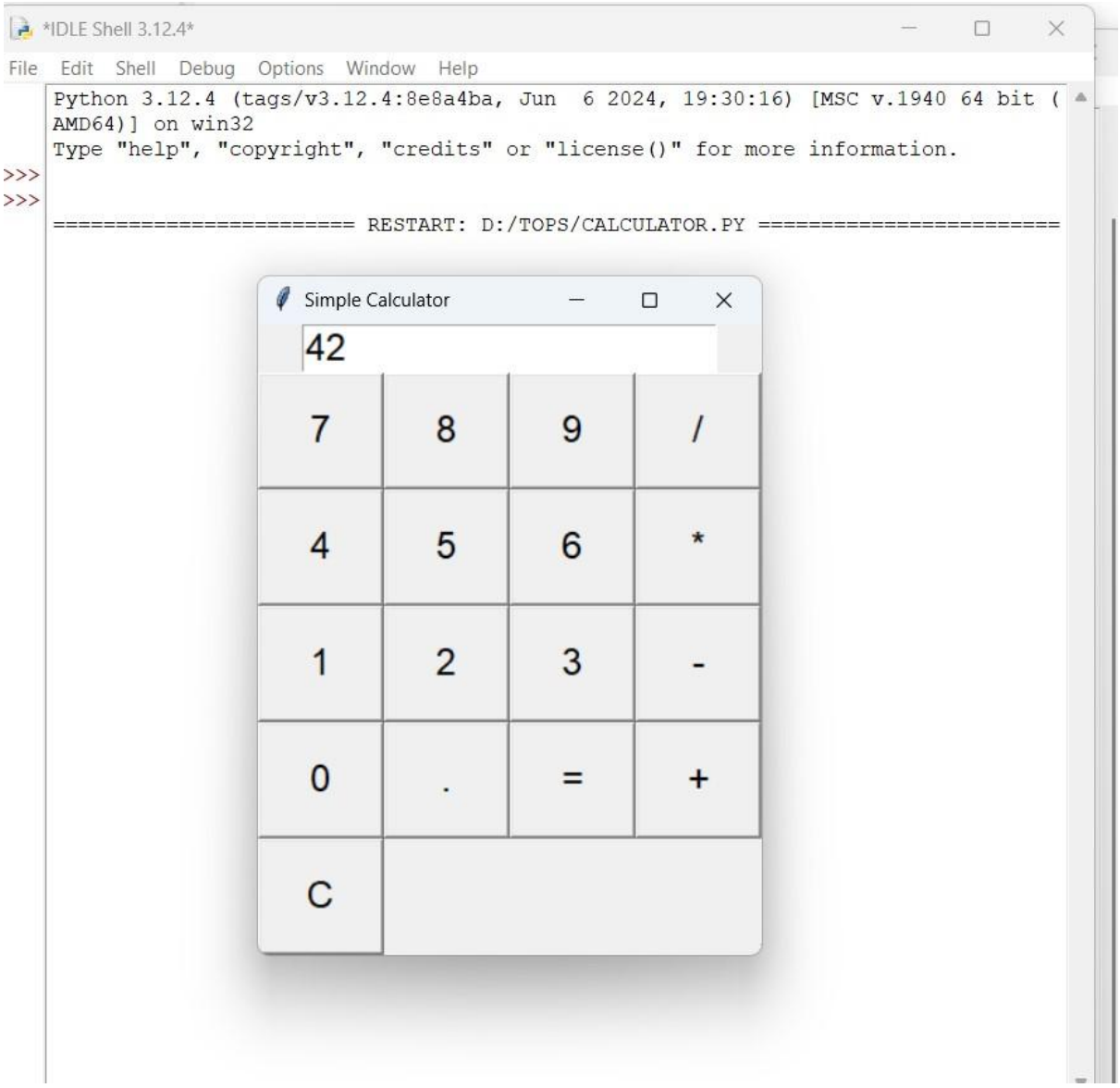
Code:

```
import tkinter as tk
```

```
def click(button_text):
if button_text == "=":
       try:
          result = str(eval(entry.get()))
entry.delete(0, tk.END)
entry.insert(tk.END, result)        except:
          entry.delete(0, tk.END)
entry.insert(tk.END, "Error")     elif
button_text == "C":
entry.delete(0, tk.END)
   else:
       entry.insert(tk.END, button_text)

# Window setup
root = tk.Tk() root.title("Simple
Calculator")

# Entry field
entry = tk.Entry(root, width=20, font=('Arial', 18))
entry.grid(row=0, column=0, columnspan=4)

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

```
row, col = 1, 0 for

button in buttons:

    action = lambda x=button: click(x)

    tk.Button(root, text=button, width=5, height=2, font=('Arial', 18),
command=action).grid(row=row, column=col)

    col += 1

if col > 3:

col = 0

row += 1


root.mainloop()
```

Screenshot:


IDLE Shell 3.12.4
File Edit Shell Debug Options Window Help
Python 3.12.4 (tags/v3.12.4:8e8a4ba, Jun 6 2024, 19:30:16) [MSC v.1940 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
>>>
========== RESTART: D:/TOPS/CALCULATOR.PY ==========
Simple Calculator
42
7 8 9 /
4 5 6 *
1 2 3 -
0 . = +
C

Flow Chart

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