Assignment 1 THEORY EXE.	Module 1 – Overview of IT Industry

1. What is Program?

THEORY EXERCISE: Explain in your own words what a program is and how it functions.

- **ANS :**

A program is a set of instructions written in some different languages to tell a computer device that what it have to compute.
HOW IT FUNCTIONS:

As we know a computer knows the Binary Language. But we can't explain big calculations to the computer in Binary language. So some specific developers developed some programming languages (like JAVA, PYTHON, PHP, C, C++, C# etc.) that can be understandable by both HUMAN and COMPUTERS.

There are some main functions of these programming languages:

- Input
- Processing
- Output
- Manage (Storage, Execution)

2. What is Programming?

THEORY EXERCISE: What are the key steps involved in the programming process?

- **ANS :**

There are few steps in writing the program and run and compilation.

The first step is to choose the programming language, in which you would like to solve the problem (EX; JAVA, PYTHON, C, C++, C#).

Then according to the chosen language you need to choose the platform or a complier on which you can execute your program code (EX; V.S. CODE, IDLE(for python), INTELLIJ IDEA (for java), TURBO C & C++).

After that write the code in the chosen programming language. Implement the plan & algorithms step-by-step.

Compile the code to identify syntax errors. Debug these errors. You can also use debugging tools given by the platform or compiler you use. And test your program again.

So, these are the main steps in programming. You can also document the code and improve as your requirement.

3. Types of Programming Languages

THEORY EXERCISE: What are the main differences between high-level and low-level programming languages?

- **ANS:**

The main difference is that the High-Level languages are Human friendly. They are closer to human language, making them easier to read, write, and understand and the Low-Level languages are Machine friendly. They are closer to the computer's hardware and more difficult for humans to read and write.

Example of High-Level languages: Java, Python, PhP, C, C++, C#
Example of Low-Level languages: Binary language

4. World Wide Web & How Internet Works

THEORY EXERCISE: Describe the roles of the client and server in web communication.

- **ANS:**

The main difference is that the High-Level languages are Human friendly. They are closer to human language, making them easier to read, write, and understand and the Low-Level languages are Machine friendly. They are closer to the computer's hardware and more difficult for humans to read and write.

Example of High-Level languages: Java, Python, PhP, C, C++, C#
Example of Low-Level languages: Binary language

5. Network Layers on Client and Server

THEORY EXERCISE: Explain the function of the TCP/IP model and its layers.

- **ANS:**

The TCP/IP model is Transmission Control Protocol/Internet Protocol. It is a conceptual framework used to enable communication between computers over a network.

--> There mainly 4 layers of TCP/IP model

*Application Layer

*Transport Layer

*Internet Layer

*Network Access Layer

6. Client and Servers

THEORY EXERCISE: Explain Client Server Communication

- **ANS:**

Client-server communication is a model where two entities, a client and a server, interact over a network. The client makes requests, and the server processes and responds to these requests. This architecture underpins many internet-based applications, such as web browsing, email, and file sharing.

7. Types of Internet Connections

THEORY EXERCISE: How does broadband differ from fiber-optic internet?

- **ANS:**

Broadband and fiber-optic internet are two types of internet connection technologies.

Broadband is a generic term that refers to high-speed internet connections using various technologies, including DSL (Digital Subscriber Line), cable, satellite, and fiber. Speeds range from 1 Mbps to 500 Mbps, depending on the technology used

Uses electric signals to transmit data over copper or coaxial cables. Higher latency compared to fiber, especially in satellite and DSL connections. Generally more affordable upfront, with lower installation and monthly costs.

Fiber-optic internet is a specific type of broadband connection that uses fiber-optic cables made of thin strands of glass or plastic to transmit data as light signals. Offers significantly faster speeds, often up to 1 Gbps (Gigabit per second) or higher. Transmits data as light signals through fiber-optic cables. Offers low latency, making it ideal for online gaming, video conferencing, and other latency-sensitive activities. Higher installation and subscription costs due to advanced technology and infrastructure requirements.

8. Protocols

THEORY EXERCISE: What are the differences between HTTP and HTTPS protocols?

- **ANS:**

HTTP is Hyper Text Transfer Protocol. It is a protocol for transmitting hypertext over the internet. In this there is no encryption (Data transmitted is in plain text) . Risky for sensitive information such as login credentials, payment details, or personal data. It’s URL is **‘ http:// ’** . (EXAMPLE; http://google.com)

HTTPs is Hyper Text Transfer Protocol Secure. An extension of HTTP that uses encryption to secure data transmission. It uses encryption in communication. Data is encrypted and cannot be read by unauthorized parties, so it safe to use in sharing personal data in logging like processes. It’s URL is **‘ https:// ’.**
(EXAMPLE; https://google.com)

9. Application Security

THEORY EXERCISE: What is the role of encryption in securing applications?

- **ANS:**

Encryption plays a critical role in securing applications by converting data into a coded format to protect it from unauthorized access. In a communication, when a user sends data to the other user, it converts the data into miscellaneous codes. It ensures that even if data is intercepted or accessed by malicious actors, it cannot be understood or misused without the appropriate decryption key. Encryption is essential for maintaining the confidentiality, integrity, and authenticity of data in applications.

10. Software Applications and Its Types

THEORY EXERCISE: What is the difference between system software and application software?

- **ANS:**

System software and application software are two types of computer software.

System Software: Software designed to manage and control computer hardware and provide a platform for running application software. It acts as an interface between hardware and user-level software. In simple words Operating Systems are the System software. Operating systems (e.g., Windows, macOS, Linux). Also some Utility programs like antivirus software are also the examples of System Software.

Application Software: These are the software developed to help users perform specific tasks or activities. It runs on top of system software and is tailored to user needs. Tools like Microsoft Word, Excel Google Docs etc. are the examples of Application software. Also web browsers like Firefox, Tor Browser, Google Chrome etc. are the Application Software.

11. Software Architecture

THEORY EXERCISE: What is the significance of modularity in software architecture?

- **ANS:**

Here is why modularity is significant in software architecture:

1. Improved Maintainability (Maintenance after development)
2. Scalability and Flexibility
3. Cost Efficiency
4. Reduced Risk and Higher the Security

12. Layers in Software Architecture

THEORY EXERCISE: Why are layers important in software architecture?

- **ANS:**

There are mainly 5 layers in Software Architecture;

1. Presentation Layer (User Interface Layer)
2. Business Logic Layer (Application Layer)
3. Domain layer
4. Infrastructure layer
5. Database layer

Importance of Layers in Software Architecture;

Layered architecture is a design approach that organizes software systems into distinct layers, each with a specific role and responsibility. Layers improve structure, maintainability, and scalability, making them an essential element of modern software design. Here are the main point that explains why layers are important in Software Architecture;

1. Separation of Concerns: Each layer focuses on a specific aspect of the application, such as data management, business logic, or user interface.
2. Maintainability: Changes in one layer do not affect other layers if the architecture is properly designed.
3. Scalability
4. Reusability etc...

13. Software Environments

THEORY EXERCISE: Explain the importance of a development environment in software production.

- **ANS:**

A development environment is a setup of tools, frameworks, and configurations used by developers to write, test, and debug software during its production. It serves as the foundation for building reliable, scalable, and maintainable software applications. Here's why it is crucial in software production:

1. Facilitates Code Development
2. Standardization
3. Supports Testing
4. Reduces Deployment Risks
5. Collaboration and Team Productivity
6. Encourages Innovation and Experimentation
7. Facilitates Debugging and Monitoring
8. Supports Continuous Integration/Continuous Deployment
9. Security

14. Source Code

THEORY EXERCISE: What is the difference between source code and machine code?

- **ANS:**

--> Definition

Source Code : Human-readable code written by programmers in high-level programming languages (e.g., Python, C).

Machine Code : Binary instructions directly executed by a computer's CPU.

--> Language

Source Code : Written in high-level or assembly languages.

Machine Code : Written in binary (0s and 1s), understood by the computer's processor.

--> Purpose

Source Code : Designed for humans to write, read, and understand software programs.

Machine Code : Designed for machines to execute tasks at the hardware level.

--> Conversion Process

Source Code : Translated into machine code using compilers, interpreters, or assemblers.

Machine Code : Produced as the output of the compilation or assembly process.

--> EXAMPLE

Source Code : print("Hello, World!") (Python)

Machine Code : 10101010 11011011 00001111 (Binary instructions for a CPU).

15. Github and Introductions

THEORY EXERCISE: Why is version control important in software development?

- **ANS:**

Version control is a system that records changes to a file or set of files over time, enabling developers to track, manage, and collaborate on software projects efficiently. It is an essential part of modern software development for several reasons.

Version control is important for keeping track of changes to code, files, and other digital assets.

With version control, contributors can easily identify and access the most recent draft, reducing the risk of mistakenly working on an outdated version. It enables higher velocity of data teams while reducing the cost of errors.

16. Student Account in Github

THEORY EXERCISE: What are the benefits of using Github for students?

- **ANS:**

GitHub is a web-based platform that allows developers to store, manage, and share their code. It's a collaboration tool that helps developers work together on projects.

GitHub can help students learn to code, build projects, and collaborate with others.

Benefits of GitHub for students:

1. Free tools and services
2. Collaboration
3. Version control (Students can save different versions of their projects for future reference.)
4. Showcase projects

17. Types of Software

THEORY EXERCISE: What are the differences between open-source and proprietary software?

- **ANS:**

--> Definition

Open-Source Software : Software whose source code is freely available for modification, redistribution, and use.

Proprietary Software: Software with a closed source code that is owned and controlled by its creator or vendor.

--> Cost

Open-Source Software : free or available at a minimal cost

Proprietary Software: Usually requires purchasing a license or subscription to use.

--> Security

Open-Source Software : Security depends on the community's vigilance in finding and fixing vulnerabilities.

Proprietary Software: Security updates and patches are managed by the vendor, but users must wait for official fixes.

--> Examples

Open-Source Software : Linux, Apache, Mozilla Firefox, LibreOffice, MySQL

Proprietary Software: Microsoft Windows, Adobe Photoshop, Microsoft Office, Oracle Database

18. GIT and GITHUB Training

THEORY EXERCISE: How does GIT improve collaboration in a software development team?

- **ANS:**

Git improves collaboration in software development teams by allowing developers to work on different parts of a project simultaneously without overwriting each other's changes. Git also allows developers to review and discuss changes before merging them into the main branch.

19. Application Software

THEORY EXERCISE: What is the role of application software in businesses?

- **ANS:**

Application Software helps businesses work more efficiently by automating and optimizing processes, and by improving communication and data management. The points given belove are the factors in which application software helps for business:

1. Communication
2. Data management
3. Process management
4. Customer relationship management

A big point as an example is ERP model :

Enterprise resource planning (ERP): Helps businesses manage core business processes like finance, human resources, and supply chain

20. Software Development Process

THEORY EXERCISE: What are the main stages of the software development process?

- **ANS:**

The main stages of the software development process are:

1. Planning: Create a basic plan for the application, based on business requirements
2. Requirements analysis: Gather and analyze the requirements for the application
3. Design: Create the product architecture
4. Coding: Write the code for the application
5. Testing: Test the application for functionality
6. Deployment: Make the application available for use
7. Maintenance: Continue to support and improve the application

21. Software Requirement

THEORY EXERCISE: Why is the requirement analysis phase critical in software development?

- **ANS:**

The requirement analysis phase is critical in software developmen because it ensures a clear understanding of what the client or user needs, providing a solid foundation for the entire project. By defining goals and expectations early, it prevents misunderstandings, saves time and money by avoiding costly changes later, and ensures the final product meets user requirements. This phase also serves as a roadmap for the design, coding, and testing stages, improving the overall quality and effectiveness of the software.

22. Software Analysis

THEORY EXERCISE: What is the role of software analysis in the development process?

- **ANS:**

Software analysis plays a key role in the development process by helping the team understand the problem they need to solve and plan how to build the solution. It involves studying user needs, system requirements, and any constraints to create a clear roadmap for the project. This step ensures that the team focuses on the right features, avoids mistakes, and builds software that meets user expectations, saving time, effort, and costs in the long run.

23. System Design

THEORY EXERCISE: What are the key elements of system design?

- **ANS:**

The key elements of system design are:

1. System Architecture: Overall structure and component interaction.
2. Data Design: Organizing and structuring data.
3. User Interface Design: Ensuring user-friendliness.
4. Component Design: Dividing the system into modules.
5. Security Design: Protecting data and operations.
6. Scalability and Performance: Handling growth and maintaining efficiency.
7. Integration Design: Connecting with external systems.
8. Reliability and Fault Tolerance: Ensuring system stability.
9. Technology Stack: Tools and platforms used.

10. Compliance and Standards: Meeting legal and industry requirements.

24. Software Testing

THEORY EXERCISE: : Why is software testing important?

- **ANS:**

Software testing is important because it ensures the software is reliable, functional, and meets user expectations. It helps identify and fix bugs, errors, or security vulnerabilities before the product is released, reducing the risk of failures and costly fixes later. Testing als improves performance, usability, and compatibility across different devices and environments, ensuring a high-quality product. Ultimately, it builds user trust, saves time and money, and ensures the software works as intended.

25. Maintenance

THEORY EXERCISE: : What types of software maintenance are there?

- **ANS:**

There are mainly 4 types of Software maintenance;

1. **Corrective Maintenance:** Fixing bugs, errors, and defects in the software identified after deployment.
2. **Adaptive Maintenance:** Updating software to work with changes in the environment, such as new operating systems, hardware, or regulations.
3. **Perfective Maintenance:** Improving performance, usability, or functionality to enhance the software’s efficiency and meet user demands.
4. **Preventive Maintenance:** Anticipating potential issues and making updates to prevent future problems and ensure long-term reliability.

26. Development

THEORY EXERCISE: What are the key differences between web and desktop applications?

- **ANS:**

Web applications are accessed through a browser and do not require installation, making them platform-independent and easy to access from any device with an internet connection. They rely on an active internet connection to function and are updated automatically on the server side, which ensures all users have access to the latest version. On the other hand, desktop applications are installed on specific operating systems and run directly from the device. They can work offline and often offer better performance since they utilize local system resources. However, updates must be manually downloaded and installed, and the software is generally platform-dependent, meaning it may only work on certain operating systems. Security for web apps is critical on the server side, while desktop apps rely on the security of the local device. Overall, web apps offer convenience and accessibility, while desktop apps provide better performance and offline capabilities.

27. Web Application

THEORY EXERCISE: What are the advantages of using web applications over desktop applications?

- **ANS:**

The Points given below are the main advantages of Web applications:

1. Scalability: Web apps can easily handle an increasing number of users by scaling server resources, ensuring reliability as usage grows.
2. Accessibility: Web applications can be accessed from anywhere with an internet connection, making them highly convenient for remote work or collaboration.
3. Platform Independence and No installation Required. Also automatic update available.

4.Cost-Effectiveness: Web apps often have lower upfront costs since they don’t require extensive hardware or software installations on use devices.
5. Collaboration Features: Web apps often include real-time collaboration tools, enabling users to work together seamlessly from different locations.

28. Designing

THEORY EXERCISE: What role does UI/UX design play in application development?

- **ANS:**

UI/UX design plays a important role in application development by ensuring the app is visually appealing, user-friendly, and provides a seamless experience.

UI/UX design bridges the gap between user expectations and the application’s functionality, ensuring the product is not only functional but also delightful to use.

29. Mobile Application

THEORY EXERCISE: What are the differences between native and hybrid mobile apps?

- **ANS:**

Native apps are developed specifically for a single platform, such as iOS or Android, using platform-specific programming languages like Swift or Kotlin. They offer high performance and a superior user experience because they are optimized for the platform and follow its design guidelines

Native apps also provide full access to device hardware and features like the camera, GPS, and sensors. However, they are more expensive and time-consuming to develop since separate codebases are required for each platform.

H**ybrid apps** are built using web technologies like HTML, CSS, and JavaScript and are wrapped in a native container, enabling them to run on multiple platforms using a single codebase. While they are faster and cheaper to develop, they typically offer slightly lower performance and less refined UI/UX compared to native apps. Hybrid apps have limited access to device hardware, though plugins can help bridge these gaps.

- Native apps are ideal for performance-intensive and platform-specific functionalities, while hybrid apps are better suited for cost-effective solutions requiring cross-platform compatibility.

30. DFD (Data Flow Diagram)

THEORY EXERCISE: What is the significance of DFDs in system analysis?

- **ANS:**

A D**ata Flow Diagram** (DFD) is a significant modeling technique for analyzing and constructing information processes.

DFDs are important in system analysis because they help you visualize how data moves through a system. They can help you identify problems, improve processes, and protect data.

DFD literally means an illustration that explains the course or movement of information in a process.

DFD illustrates this flow of information in a process based on the inputs and outputs.

31`.Desktop Application

THEORY EXERCISE: What are the pros and cons of desktop applications compared to webapplications?

- **ANS:**

ON THE NEXT PAGE>>>>

--> Pros of Desktop Applications:

1. Offline Access: Work without an internet connection.
2. Performance: Faster and more resource-efficient as they utilize local hardware.
3. Customization: Deeper integration with the operating system allows more advanced features.

--> Pros of Web Applications:

1. Accessibility: Access from any device with an internet connection.
2. Platform Independence: Works across multiple operating systems via browsers.
3. Automatic Updates: Always run the latest version without manual updates.

--> Cons of Desktop Applications:

1. Platform Dependency: Need separate versions for different operating systems.
2. Installation: Requires downloading and installing updates manually.
3. Limited Accessibility: Can only be used on the device where installed.

--> Cons of Web Applications:

1. Internet Dependency: Requires a stable internet connection to function.
2. Performance: Slower than desktop apps for resource-intensive tasks.
3. Limited Features: Restricted access to hardware and system-level functionalities.

32. Flow Chart

THEORY EXERCISE: How do flowcharts help in programming and system design?

- **ANS:**
  ON THE NEXT PAGE>>>>

How Flowcharts Help in Programming and System Design:

1. Simplify Processes: Break down complex logic into simple, visual steps.
2. Enhance Communication: Provide a universal visual language for better collaboration.
3. Detect Errors: Help identify issues or inefficiencies in the system.
4. Aid Planning: Serve as a blueprint for algorithms and workflows.
5. Support Documentation: Act as a reference for future maintenance.
6. Facilitate Problem-Solving: Visualize paths and outcomes for debugging and optimization.

Assignment 1 LAB EXE.

Module 1 – Overview of IT Industry

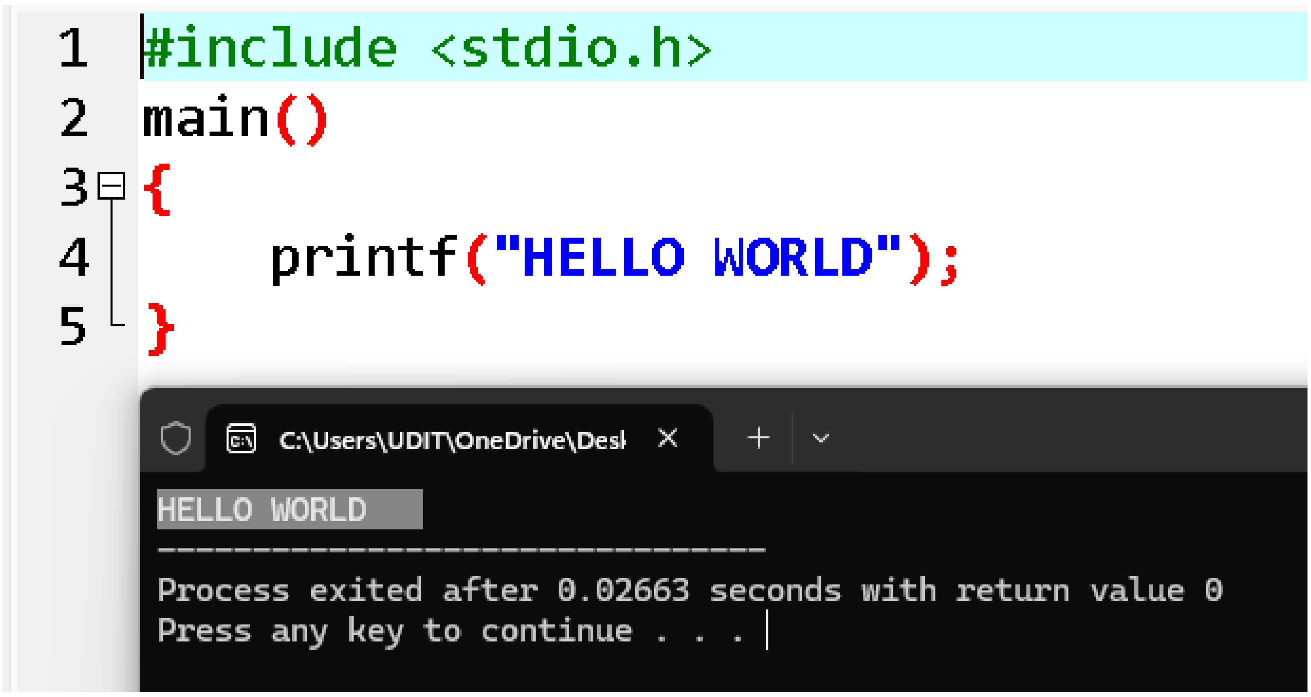
1. What is Program?

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

Here is the first code in C language for printing "Hello World" :

```
#include <stdio.h>
main()
{
   printf("HELLO WORLD");
}
```

Compilation of This code;



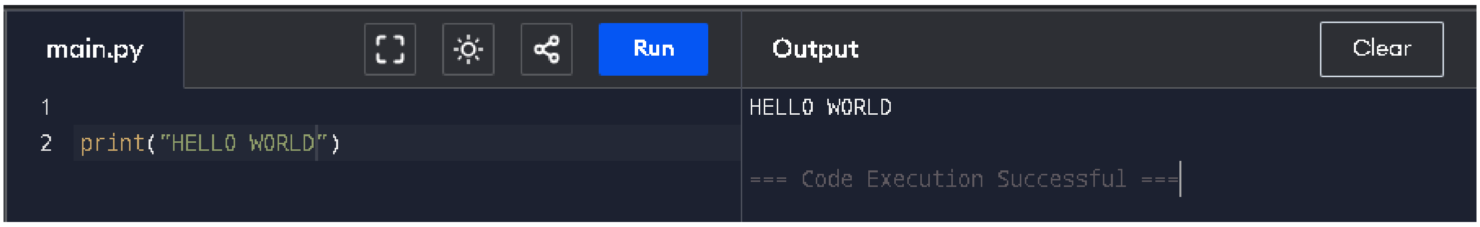
```
#include <stdio.h>
main()
{
    printf("HELLO WORLD");
}
```

```
HELLO WORLD
--------------------------------
Process exited after 0.02663 seconds with return value 0
Press any key to continue . . .
```

Here is the second code in python for printing “Hello World” :

print(“HELLO WORLD”)

Compilation of This code;



```
print("HELLO WORLD")
```

```
HELLO WORLD

=== Code Execution Successful ===
```

Comparison of the structure and syntax of these languages ;

- **Syntax :**

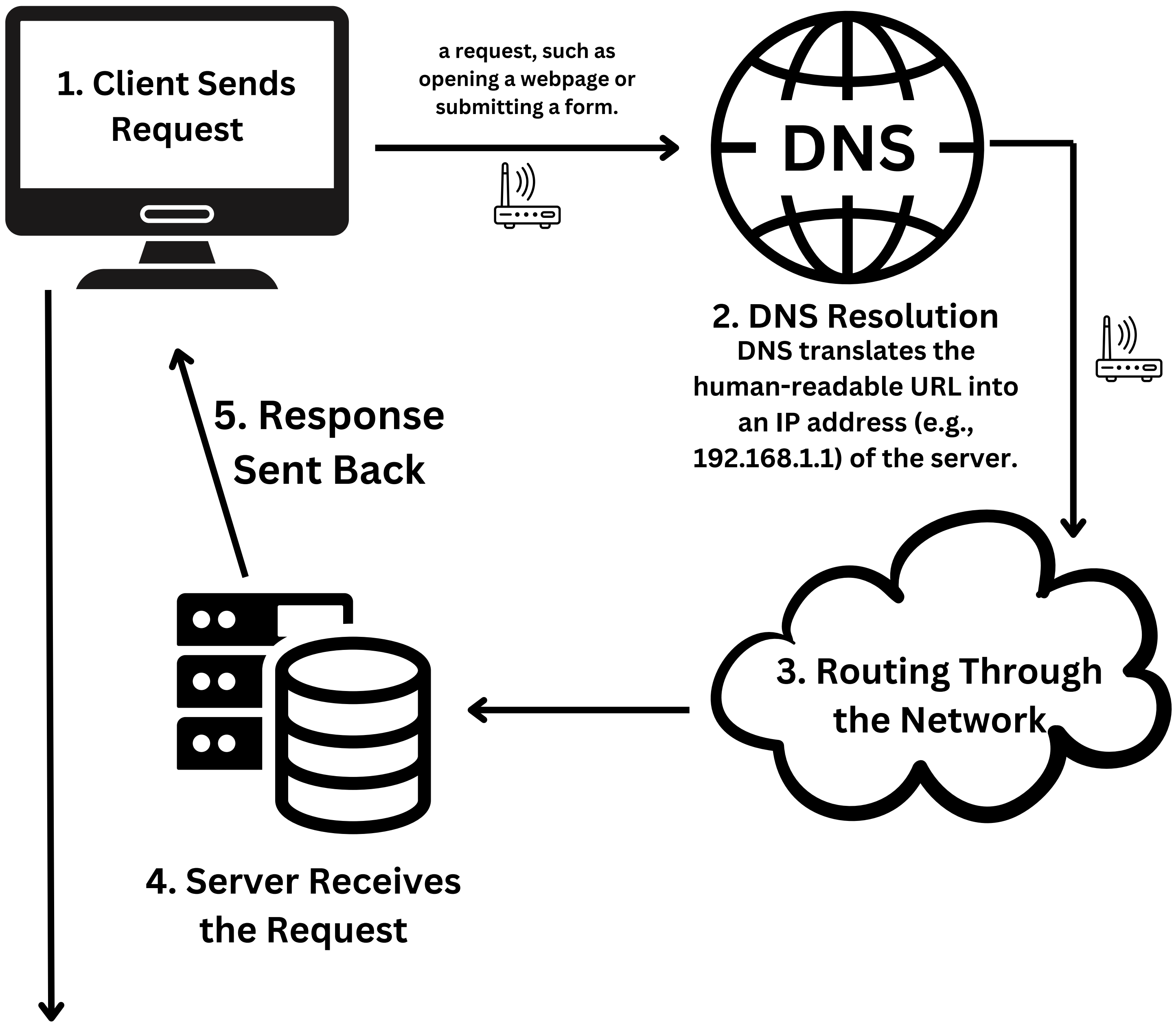
In Syntax of C language, it requires detailed syntax, including semicolons, braces and type declaration. But, in Python, simpler syntax, no semicolons, uses indentation instead of braces.

- **Structure :**

In C language, it uses function as the basic unit of execution, with main() as the entry point.
In Python, it executes line by line, no mandatory main() function.

4. World Wide Web & How Internet Works

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


1. Client Sends Request
a request, such as opening a webpage or submitting a form.
DNS
2. DNS Resolution
DNS translates the human-readable URL into an IP address (e.g., 192.168.1.1) of the server.
3. Routing Through the Network
4. Server Receives the Request
5. Response Sent Back

6. Client Displays the Data

- **The client receives the packets, reassembles them, and displays the content, such as rendering a webpage.**

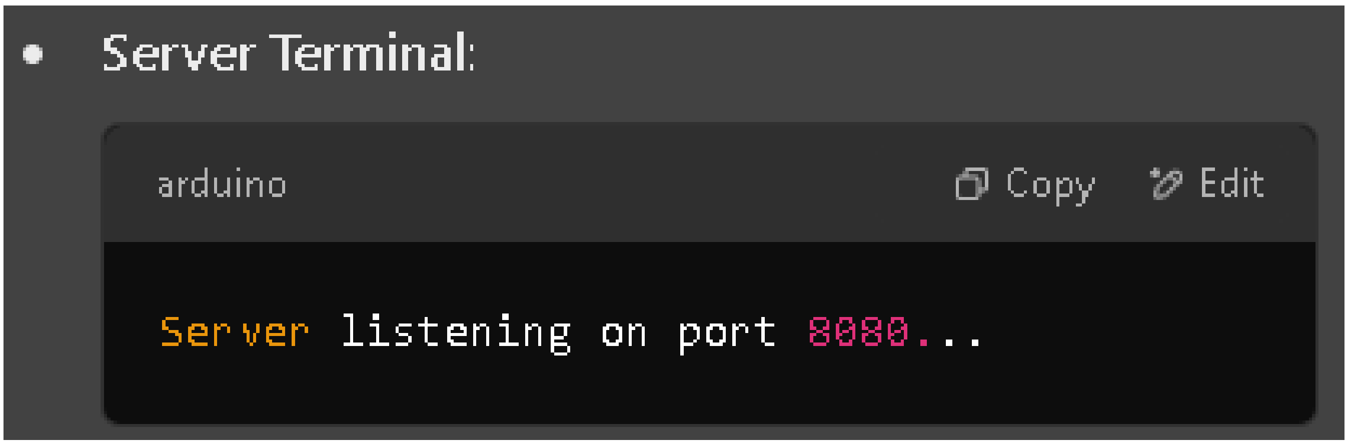
5. Network Layers on Client and Server

LAB EXERCISE: Design a simple HTTP client-server communication in any language

SERVER CODE

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <arpa/inet.h>

#define PORT 8080

int main() {
   int server_fd, new_socket;
   struct sockaddr_in address;
   int addrlen = sizeof(address);
   char response[] =
      "HTTP/1.1 200 OK\r\n"
      "Content-Type: text/html\r\n"
      "Content-Length: 46\r\n"
      "\r\n"
      "<html><body><h1>Hello, World!</h1></body></html>";

   // Create socket
   if ((server_fd = socket(AF_INET, SOCK_STREAM, 0)) == 0) {
      perror("Socket failed");
      exit(EXIT_FAILURE);
   }

   // Bind the socket to the network address and port
   address.sin_family = AF_INET;
   address.sin_addr.s_addr = INADDR_ANY;
   address.sin_port = htons(PORT);

   if (bind(server_fd, (struct sockaddr *)&address,
sizeof(address)) < 0) {
      perror("Bind failed");
      exit(EXIT_FAILURE);
   }
// Listen for incoming connections
if (listen(server_fd, 3) < 0) {
perror("Listen failed");
exit(EXIT_FAILURE);
}

printf("Server listening on port %d...\n", PORT);

// Accept incoming connection
if ((new_socket = accept(server_fd, (struct sockaddr
*)&address, (socklen_t*)&addrlen)) < 0) {
perror("Accept failed");
exit(EXIT_FAILURE);
}

// Send HTTP response
write(new_socket, response, strlen(response));
close(new_socket);
close(server_fd);

return 0;
}
```

OUTPUT


- Server Terminal:

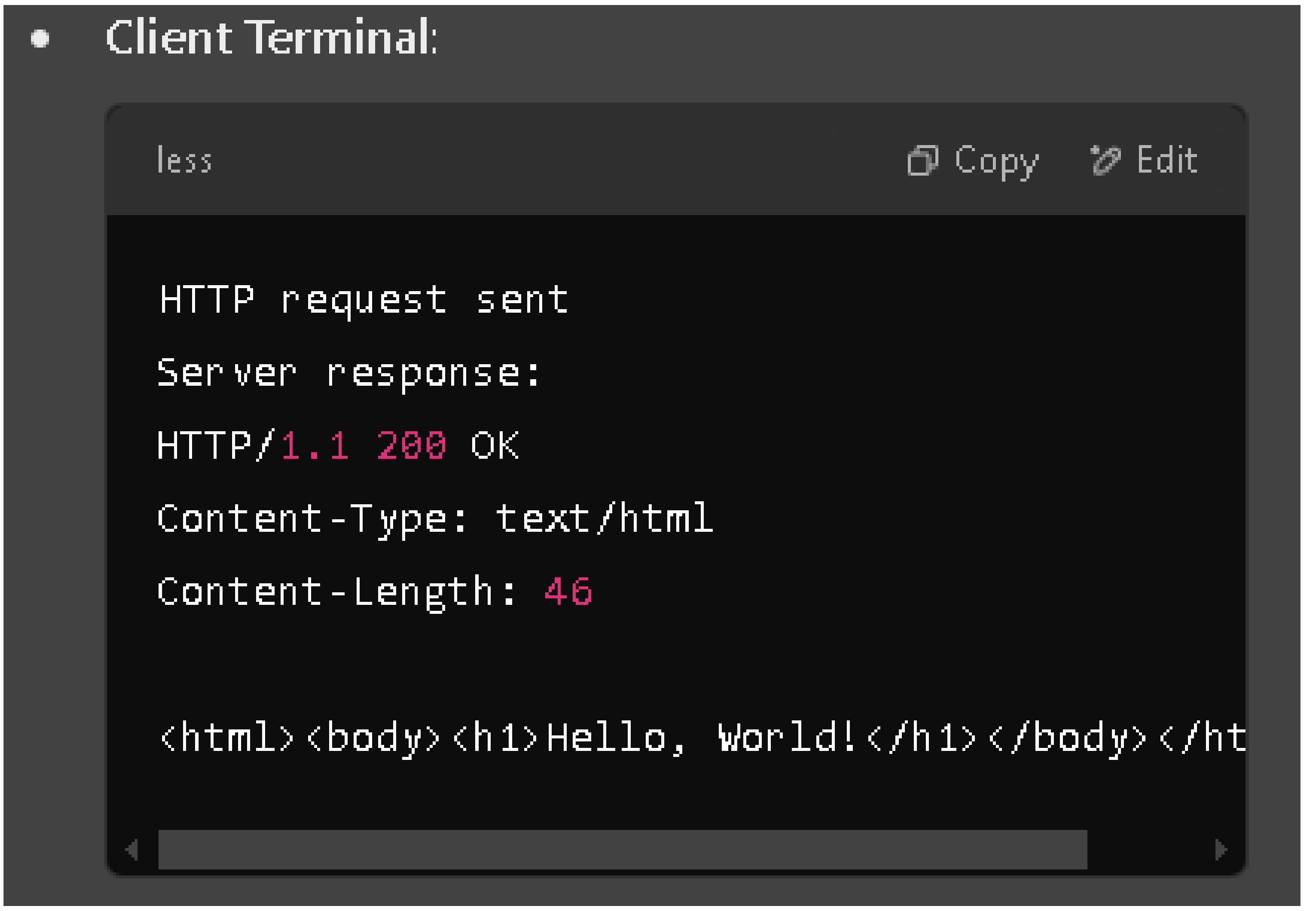
```
Server listening on port 8080...
```

CLIENT CODE

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <arpa/inet.h>

#define PORT 8080

int main() {
   int sock = 0;
   struct sockaddr_in serv_addr;
   char request[] = "GET / HTTP/1.1\r\nHost: localhost\r\n\r\n";
   char buffer[1024] = {0};

   // Create socket
   if ((sock = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
      printf("\n Socket creation error \n");
      return -1;
   }

   serv_addr.sin_family = AF_INET;
   serv_addr.sin_port = htons(PORT);

   // Convert IPv4 and IPv6 addresses from text to binary form
   if (inet_pton(AF_INET, "127.0.0.1", &serv_addr.sin_addr) <= 0) {
      printf("\nInvalid address/ Address not supported \n");
      return -1;
   }

   // Connect to the server
   if (connect(sock, (struct sockaddr *)&serv_addr, sizeof(serv_addr))
< 0) {
      printf("\nConnection Failed \n");
      return -1;
   }

   // Send HTTP request
   send(sock, request, strlen(request), 0);
   printf("HTTP request sent\n");
```

```c
// Read server response
read(sock, buffer, sizeof(buffer));
printf("Server response:\n%s\n", buffer);

close(sock);
return 0;
}
```

OUTPUT



```less
HTTP request sent
Server response:
HTTP/1.1 200 OK
Content-Type: text/html
Content-Length: 46

<html><body><h1>Hello, World!</h1></body></ht
```

7. Types of Internet Connections

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

1. **BROADBAND :**

- Uses existing telephone lines (DSL) or cable TV lines (Cable) for internet connectivity.
- Common in urban and suburban areas.

Pros:

- Widely available in cities and towns.
- Affordable compared to other options.
- Allows simultaneous use of the phone line and the internet

Cons:

- Speeds can decrease during peak usage hours (Cable).
- Limited speed and range for DSL based on distance from the provider's central office.
- Susceptible to signal degradation over old or poor-quality lines.

2. **Fiber-Optic Internet :**

- Transmits data as light through glass or plastic fibers.
- Offers very high-speed internet access.

Pros:

- Ultra-fast speeds (up to 1 Gbps or more).
- Extremely reliable with low latency.

Cons:

- Expensive to install and maintain.
- Limited availability in rural or underdeveloped areas.
- Installation can be time-consuming.

---

3. Satellite Internet :

- Relies on satellites orbiting the Earth to provide internet access.
- Common in remote or rural areas where other types are unavailable.

Pros:

- Accessible in areas where other types of internet are not available.
- Can provide basic internet services globally.

Cons:

- High latency due to the long distance data must travel to and from satellites.
- Limited speeds compared to broadband and fiber.
- Expensive monthly plans and equipment setup.

---

4. Mobile Internet (4G/5G) :

- Uses cellular networks to provide internet access via SIM cards or mobile hotspots.

Pros:

- Highly portable—can be accessed anywhere within the coverage area.
- Increasingly faster speeds with 5G technology.
- No reliance on physical lines or installations.

Cons:

- Dependent on cellular coverage, which may be poor in remote areas.
- Data plans may have usage limits or high costs for unlimited plans.
- Speeds can vary based on network congestion.

8. Protocols

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

1. HTTP Requests

a. GET Request

Fetch data from a server using an HTTP GET request.

```
curl http://example.com
```

This retrieves the HTML content of the website.

b. POST Request

Send data to a server using an HTTP POST request.

```
curl -X POST -d "key1=value1&key2=value2"
http://example.com/api
```

- -X POST: Specifies the HTTP method.
- -d: Sends the data as the body of the POST request.

c. Headers

Include custom headers in an HTTP request.

```
curl -H "Content-Type: application/json" -H
"Authorization: Bearer TOKEN"
http://example.com/api
```

d. Save Response to a File

Download content and save it locally.

```
curl -o output.html http://example.com
```

(-o: Specifies the output file name.)

e. Follow Redirects

Automatically follow HTTP redirects.

```
curl -L http://example.com
```

---

2. FTP Requests

a. List Directory Contents

Retrieve the list of files and directories from an FTP server.

```
curl ftp://ftp.example.com/ --user udit:pass12word
```

b. Download a File

Download a specific file from an FTP server.

c. Upload a File

Upload a file to an FTP server.

```
curl -T file.txt ftp://ftp.example.com/ --user
udit:pass12word
```

(-T: Specifies the file to upload.)

d. Anonymous FTP Access

Access an FTP server without providing credentials.

```
curl ftp://ftp.example.com/
```

9. Application Security

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

1. Injection Attacks

Attackers inject malicious input (e.g., SQL, command, or NoSQL injection) to manipulate application behavior or access data.

Solution:

Use parameterized queries or prepared statements.

Sanitize and validate user inputs.

Regularly update databases and code libraries.

2. Cross-Site Scripting (XSS)

Attackers inject malicious scripts into web pages to steal data or impersonate users.

Solution:

Encode user inputs before rendering.

Implement Content Security Policy (CSP).

Use input sanitization tools like DOMPurify.

3. Broken Authentication

Weak authentication allows unauthorized access (e.g., weak passwords or token exposure).

Solution:

Enforce strong passwords and use MFA.

Secure session tokens in HTTP-only cookies.

Regularly test and improve authentication mechanisms.

10. Software Applications and Its Types

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

1. Operating System (e.g., Windows, macOS, Linux)
Role: Manages hardware and software resources, enabling other software to function.

2. Antivirus Software (e.g., Windows Defender)
Role: Protects the system from malware and security threats, ensuring safe operation.

3. Web Browser (e.g., Google Chrome, Firefox)
Role: Enables users to browse the internet and access web-based content.

4. Messaging App (e.g., WhatsApp, Microsoft Teams)
Role: Facilitates communication via text, voice, and video.

5. Media Player (e.g., VLC, Spotify)
Role: Plays audio and video files or streams multimedia content.

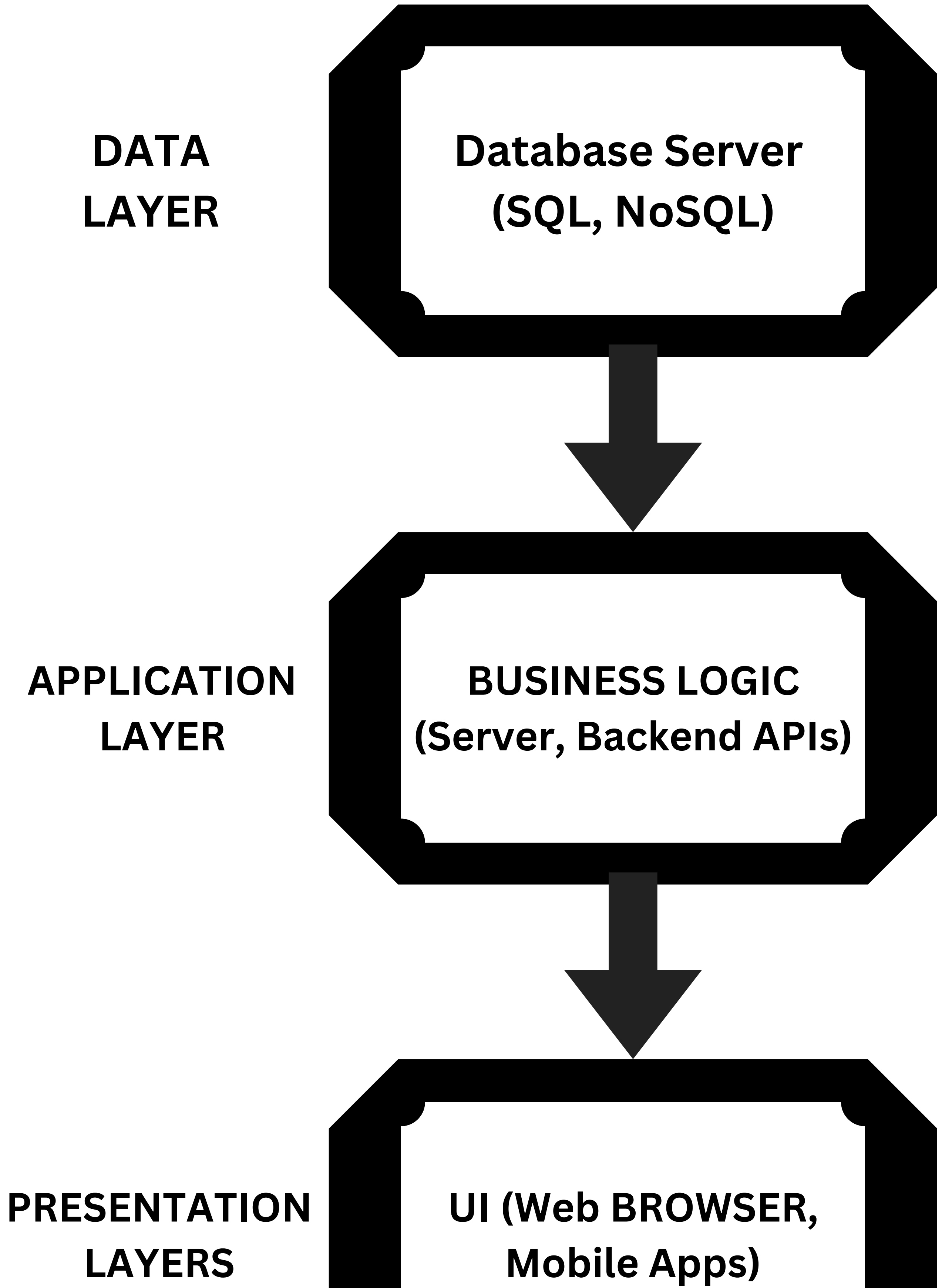
Here application no. 1 & 2 are System Software and application no. 3, 4 & 5 are Application Software.

11. Software Architecture

LAB EXERCISE: Design a basic three-tier software architecture diagram for a web application

Here is a basic three-tier software architecture diagram for a web application:

1. Presentation Layer: The user interface, such as a web browser or mobile app, where users interact with the application.
2. Application Layer: The business logic layer, which processes user requests, executes the core logic, and communicates with the data layer.
3. Data Layer: The database server that stores, retrieves, and manages the application's data.


DATA LAYER
Database Server (SQL, NoSQL)
APPLICATION LAYER
BUSINESS LOGIC (Server, Backend APIs)
PRESENTATION LAYERS
UI (Web BROWSER, Mobile Apps)

12. Layers in Software Architecture

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

Case Study: Functionality of the Presentation, Business Logic, and Data Access Layers in an E-commerce Application

Introduction

E-commerce applications are prime examples of software systems using a three-tier architecture, which separates the software into distinct layers: the presentation layer, business logic layer, and data access layer. This case study focuses on an online bookstore named "House OF Books" to illustrate the functionality of these layers.

1. Presentation Layer

Purpose: The presentation layer is the user interface of the application that allows customers to interact with the system.

Functionality:

- **Components:**
  - A responsive web application built with HTML, CSS, and JavaScript.
  - Mobile application built using Flutter.
- **Features:**
  - Displays book categories, search results, and detailed product information.
  - Provides forms for user registration, login, and feedback.
  - Facilitates the addition of books to the shopping cart.
  - Offers payment options and order tracking interfaces.

Example Scenario: A customer visits the website and searches for "Science Fiction Books." The presentation layer captures the search query and sends it to the business logic layer for processing.

2. Business Logic Layer
Purpose: The business logic layer processes user requests, enforces rules, and manages workflows.
Functionality:

- **Components:**
    - Server-side application written in Python (Django framework).
    - RESTful APIs for communication between the front end and back end.
- **Features:**
    - Processes user actions such as search queries, login attempts, and payment transactions.
    - Enforces application rules, such as ensuring users are logged in before they can place an order.
    - Handles notifications for order updates and promotional offers.

Example Scenario: When the search query for "Science Fiction Books" is received, the business logic layer validates the input, formulates a database query, and fetches results from the data layer. It then applies filters like availability and price range before sending the data to the presentation layer.

3. Data Access Layer
Purpose: The data access layer is responsible for storing, retrieving, and managing data in the database.
Functionality:

- **Components:**
    - Relational Database Management System (RDBMS) like PostgreSQL.
    - Object-Relational Mapping (ORM) tool for seamless database interaction.

- **Features:**
  - Stores user details, product information, orders, and payment records.
  - Manages relationships between data entities (e.g., books, authors, and reviews).
  - Ensures data integrity and security through access control mechanisms.

Example Scenario: The data layer processes the query generated by the business logic layer to retrieve a list of science fiction books. It sends the results back in the required format.

13. Software Environments

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

Steps to Set Up a Basic Environment:

1. Install Virtual Machine Software
2. Create a New Virtual Machine
3. Load the Operating System
4. Install Necessary Tools

- For Development Environment
- For Testing Environment
- For Production Environment:

5. Configure Networking
6. Create Snapshots
7. Test the Environment

14. Source Code

LAB EXERCISE: Write and upload your first source code file to Github.

https://github.com/udit1818/demo/blob/main/simplecalculater.py

15. Github and Introductions

LAB EXERCISE: Create a Github repository and document how to commit and push code changes

https://github.com/udit1818/demo

16. Student Account in Github

LAB EXERCISE: Create a student account on Github and collaborate on a small project with aclassmate.

17. Types of Software

LAB EXERCISE: 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 hardware and provides essential services for other software. Examples include operating systems and device drivers.

- **Operating System:**
    - Windows 10/11
    - macOS
    - Linux (Ubuntu)

- **Device Drivers:**
    - NVIDIA Graphics Driver
    - Printer Drivers

2. Application Software
Application software is designed for specific user tasks, such as word processing or web browsing.

- **Web Browsers:**
    - Google Chrome
    - Mozilla Firefox
- **Office Suite:**
    - Microsoft Word
    - Excel
- **Communication Tools:**
    - Zoom
    - Microsoft Teams
- **Entertainment:**
    - Spotify
    - VLC Media Player

18. GIT and GITHUB Training

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

19. Application Software

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

- **Report on the Various Types of Application Software and Their Role in Improving Productivity**

Introduction

Application software is designed to help users perform specific tasks, ranging from document creation to complex data analysis. This report explores various types of application software and their contributions to enhancing productivity in personal, educational, and professional settings.

Types of Application Software

1.Word Processing Software

- Examples: Microsoft Word, Google Docs.
- Purpose: Enables users to create, edit, and format text documents.
- Productivity Impact: Simplifies document creation and editing, supports collaboration through cloud integration, and improves efficiency with tools like spell check and templates.

2.Spreadsheet Software

- Examples: Microsoft Excel, Google Sheets.
- Purpose: Assists in organizing, analyzing, and visualizing numerical data.
- Productivity Impact: Automates calculations, supports data analysis with charts and pivot tables, and enables team collaboration through cloud-based sharing.

3.Presentation Software

- Examples: Microsoft PowerPoint, Canva.
- Purpose: Facilitates the creation of visual presentations.
- Productivity Impact: Enhances communication through engaging visuals, streamlines content structuring with templates, and allows for multimedia integration.

4.**Web Browsers**

- Examples: Google Chrome, Mozilla Firefox.
- Purpose: Provides access to the internet and web-based resources.
- Productivity Impact: Allows quick information retrieval, supports cloud-based application use, and facilitates research and online collaboration

5.Graphics and Multimedia Software

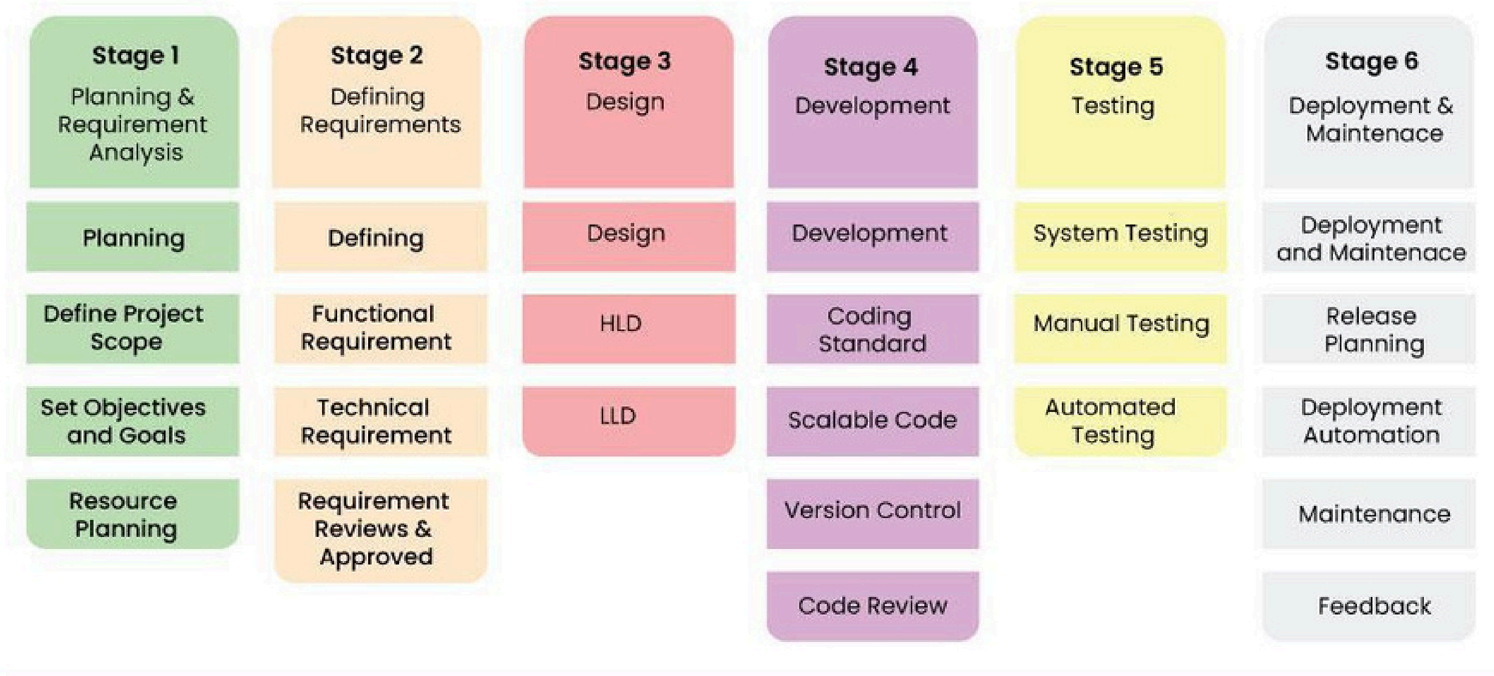
- Examples: Adobe Photoshop, Final Cut Pro.
- Purpose: Supports the creation and editing of images, videos, and other media.
- Productivity Impact: Enhances creativity, supports efficient editing workflows, and allows professionals to produce high-quality visuals and media.

6.Database Management Software (DBMS)

- Examples: MySQL, Microsoft Access.
- Purpose: Manages and organizes large volumes of data.
- Productivity Impact: Improves data retrieval and management, supports efficient decision-making through queries and reports, and enhances data security.

19. Software Development Process

LAB EXERCISE: Create a flowchart representing the Software Development Life Cycle (SDLC)



20. Software Requirement

LAB EXERCISE: Write a requirement specification for a simple library management system.

The Library Management System (LMS) is a software application designed to manage the operations of a library. It simplifies tasks like managing book records, issuing books, and maintaining user details. This system aims to enhance efficiency and user experience for librarians, staff, and library members.

1. **User Management:**
    - Allow library staff to register new users (students, faculty, etc.).
    - Maintain user profiles, including personal details and borrowing history.
    - Enable account deactivation/reactivation.
2. **Book Management:**
    - Add, update, and remove book records.
    - Categorize books by genre, author, and availability.
    - Search functionality for books by title, author, or ISBN.
3. **Borrowing and Returning:**
    - Issue books to users with automated due date calculations.
    - Allow users to return books and update their borrowing history.
    - Notify users about overdue books via email or SMS.
4. **Fine Calculation:**
    - Calculate fines for overdue books based on predefined rules.
    - Provide an interface for users to view and clear fines.
5. **Reports:**
    - Generate reports on library usage, such as most borrowed books and active users.
    - Track inventory changes and fine collections.

21. Software Analysis

LAB EXERCISE: Perform a functional analysis for an online shopping system.

An online shopping system facilitates the buying and selling of products through a web-based platform. This functional analysis identifies the core functionalities required to deliver an efficient and user-friendly experience for customers and administrators.

1.User Management

- **Registration and Login:**
  - Enable users to create accounts with personal details (name, email, phone number, etc.).
  - Provide secure login and password recovery mechanisms.
- **User Profiles:**
  - Allow users to view and update their personal information.
  - Store order history and payment preferences for future use

2. Product Management

- **Product Catalog:**
  - Display products categorized by type, brand, and price range.
- **Search and Filter:**
  - Provide a search bar with filtering options for price, rating, brand, etc.
- **Product Details:**
  - Show detailed information, including product images, descriptions, reviews, and availability.

1.

3.Shopping Cart and Wishlist

- **Cart Management:**
  - Enable users to add, update, or remove products from their cart.
  - Display a summary of selected items with quantity and total price.
- **Wishlist:**
  - Allow users to save products for future purchase.

4.Order Management

- **Order Placement:**
  - Facilitate checkout with shipping details, payment options, and order confirmation.
- **Order Tracking:**
  - Allow users to track the status of their orders in real time.
- **Returns and Refunds:**
  - Support return requests and manage refund processing.

5.Payment Management

- **Payment Methods:**
  - Support multiple payment options, including credit/debit cards, net banking, UPI, and wallets.
- **Secure Transactions:**
  - Ensure secure payment processing with encryption and fraud detection.
- **Invoices:**
  - Generate downloadable receipts for completed transactions.

6. Review and Feedback

- Allow users to leave ratings and reviews for purchased products.
- Provide a feedback mechanism for customer support and platform improvements.

7.Administrative Features

- **Inventory Management:**
    - Allow admins to add, update, and remove products.
    - Track stock levels and receive low-stock alerts.
- **Order Management:**
    - Monitor and process customer orders and returns.
- **User Management:**
    - Manage customer accounts and resolve account-related issues.

8.Notification System

- **Customer Notifications:**
    - Send order confirmations, shipping updates, and promotional offers via email or SMS.
- **Admin Notifications:**
    - Alert admins about low inventory, pending orders, or technical issues.

22. System Design

LAB EXERCISE: Design a basic system architecture for a food delivery app

Basic System Architecture for a Food Delivery App

1. **Frontend Interfaces:**
    - Customer App: Browse restaurants, place orders, and track deliveries.
    - Restaurant App: Manage menus, confirm orders, and update preparation status.
    - Delivery App: Accept delivery tasks, navigate, and update delivery progress.
2. **Backend System:**
    - Business Logic Layer: Handles order processing, delivery assignments, and workflows.
    - API Layer: Facilitates communication between apps and the server.
    - Notification Service: Sends updates like order confirmation and delivery status.
3. **Database:**
    - Stores data for customers, restaurants, orders, and delivery personnel.
4. **Third-Party Integrations:**
    - Payment Gateway: For secure transactions.
    - Map Services: Real-time navigation and delivery tracking.
    - Notification Services: For SMS, emails, and push notifications.

System Flow: Customers place orders via the app, restaurants confirm and prepare them, delivery personnel picks up the order, and it is tracked and delivered. All tasks are managed and coordinated via the backend system.

23. Software Testing

LAB EXERCISE: Develop test cases for a simple calculator program.

Test Cases for a Simple Calculator

1. **Addition Tests**
    - Add two positive numbers: 5 + 3 → 8
    - Add a positive and a negative number: 7 + (-4) → 3

1. **Subtraction Tests**
    - Subtract two numbers: 10 - 4 → 6
    - Subtract a larger number from a smaller one: 3 - 5 → -2

1. **Multiplication Tests**
    - Multiply two positive numbers: 4 * 2 → 8
    - Multiply by zero: 7 * 0 → 0

1. **Division Tests**
    - Divide two positive numbers: 8 / 2 → 4
    - Divide by zero: 5 / 0 → Error/Infinity

1. **Edge Cases**
    - Input invalid operator: 5 & 3 → Error
    - Input non-numeric values: "a" + 2 → Error

24. Maintenance

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

Real-World Case: Boeing 737 MAX Software Maintenance

Background:
The Boeing 737 MAX aircraft experienced two fatal crashes in 2018 and 2019, leading to a global grounding of the fleet. Investigations revealed that the Maneuvering Characteristics Augmentation System (MCAS) software, designed to adjust the plane's pitch automatically, was a major contributor to the crashes.

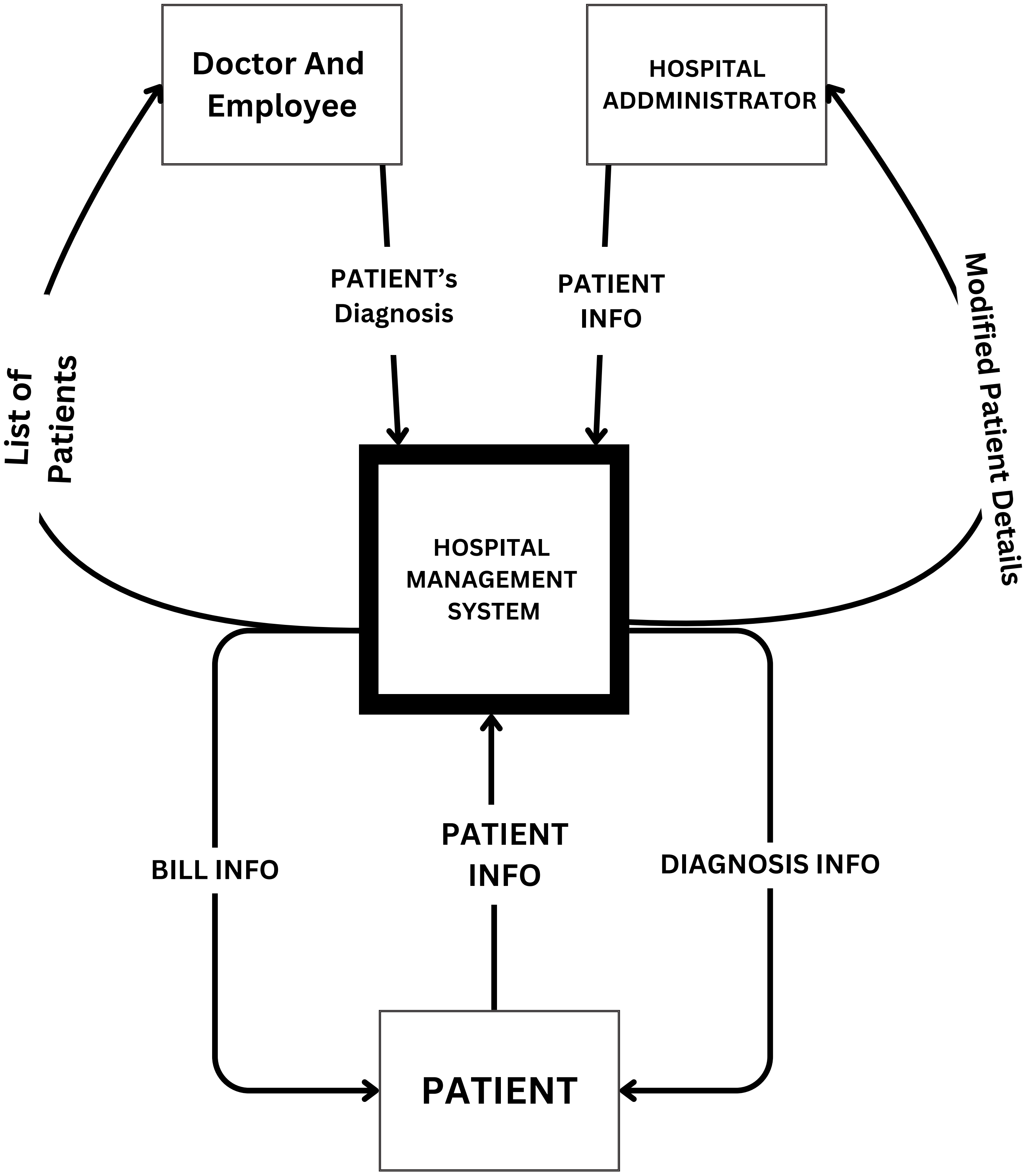
Critical Maintenance:
Boeing had to perform urgent software updates to:

- Fix the MCAS logic to prevent erroneous activation.
- Enhance redundancy by incorporating inputs from multiple sensors.
- Provide better pilot training materials on system behavior.

Outcome:
After rigorous testing and regulatory approvals, the updated software helped restore confidence, allowing the aircraft to safely return to service.

30. DFD (Data Flow Diagram)

LAB EXERCISE: Create a DFD for a hospital management system.


Doctor And Employee
HOSPITAL ADDMINISTRATOR
List of Patients
PATIENT's Diagnosis
PATIENT INFO
Modified Patient Details
HOSPITAL MANAGEMENT SYSTEM
BILL INFO
PATIENT INFO
DIAGNOSIS INFO
PATIENT

Real-World Case: Boeing 737 MAX Software Maintenance

Background:

The Boeing 737 MAX aircraft experienced two fatal crashes in 2018 and 2019, leading to a global grounding of the fleet. Investigations revealed that the Maneuvering Characteristics Augmentation System (MCAS) software, designed to adjust the plane's pitch automatically, was a major contributor to the crashes.

Critical Maintenance:

Boeing had to perform urgent software updates to:

- Fix the MCAS logic to prevent erroneous activation.
- Enhance redundancy by incorporating inputs from multiple sensors.
- Provide better pilot training materials on system behavior.

Outcome:

After rigorous testing and regulatory approvals, the updated software helped restore confidence, allowing the aircraft to safely return to service.

31. Desktop Application

LAB EXERCISE: Build a simple desktop calculator application using a GUI library

Real-World Case: Boeing 737 MAX Software Maintenance

Background:

The Boeing 737 MAX aircraft experienced two fatal crashes in 2018 and 2019, leading to a global grounding of the fleet. Investigations revealed that the Maneuvering Characteristics Augmentation System (MCAS) software, designed to adjust the plane's pitch automatically, was a major contributor to the crashes.

Critical Maintenance:

Boeing had to perform urgent software updates to:

- Fix the MCAS logic to prevent erroneous activation.
- Enhance redundancy by incorporating inputs from multiple sensors.
- Provide better pilot training materials on system behavior.

Outcome:

After rigorous testing and regulatory approvals, the updated software helped restore confidence, allowing the aircraft to safely return to service.

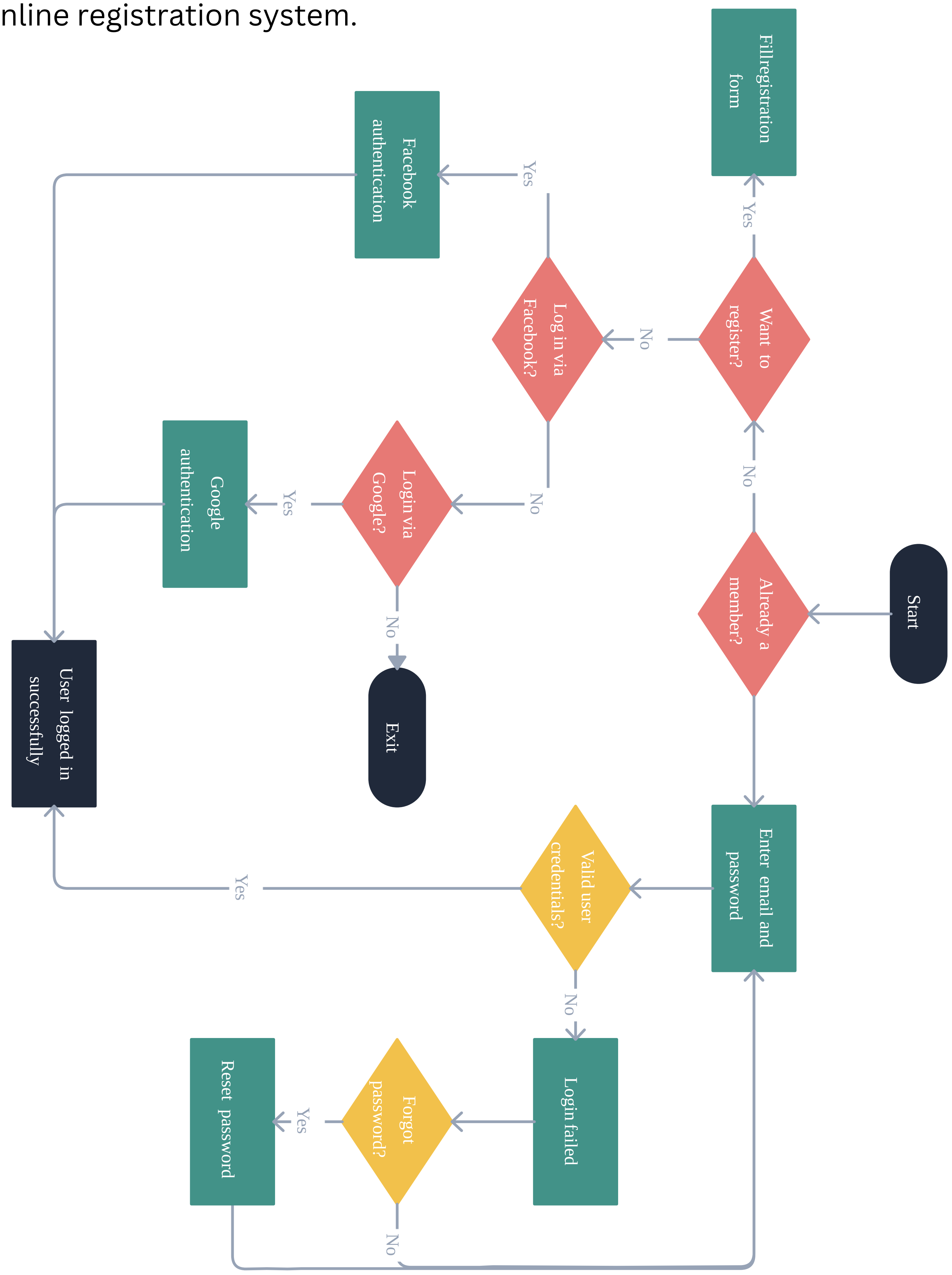
31. Desktop Application

LAB EXERCISE: Build a simple desktop calculator application using a GUI library

https://github.com/udit1818/demo/blob/main/simpleguicalculator.py

32. Flow Chart

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



THE END