Module 1 – Overview of IT Industry

Theory questions:-

Q1:- what is a program?

- ⇨ A **program** is a set of **instructions written in a programming language** that a computer can follow to perform a specific task or solve a problem.
- ⇨ **In simple words:**

A **program** tells the computer **what to do** and **how to do it**.

Q2:-what is programming?

⇨ **What is Programming?**
Programming is the process of writing a set of **instructions (called code)** to make a computer perform a specific task.

Simple Definition:
Programming means **telling the computer what to do** using a language it understands.

Example Tasks Done by Programming:

- Making websites or apps
- Creating games
- Calculating numbers
- Controlling robots or machines

Key Points:

- Programming is also called **coding**.
- It is done using **programming languages** like C, Python, Java, etc.
- It helps solve problems and automate tasks.

Q3:-Type of programing language

⇨ There is 4 types of programing language

1.procedual programing language

⇨ Procedural programming languages are a type of programming language that follows a set of procedures or routines to perform tasks. In procedural programming, the program is structured around procedures or functions, which are blocks of code that can be called to execute specific tasks. This paradigm

emphasizes a sequence of steps to be followed in order to achieve a desired outcome.

- ⇨ **Examples of Procedural Programming Languages:**

C: A widely used procedural language known for its performance and control over system resources.

Pascal: Designed for teaching programming concepts, it emphasizes structured programming.

Fortran: One of the oldest programming languages, primarily used in scientific and engineering applications.

BASIC: An easy-to-learn language that was popular in the early days of personal computing.

Ada: A language designed for large, long-lived applications, particularly in systems programming and embedded systems.

2.object oriented programing language

- ⇨ Object-oriented programming (OOP) languages are designed around the concept of "objects," which can contain both data and methods that operate on that data. This paradigm allows for a more modular and organized approach to programming, making it easier to manage complex software systems.
- ⇨ Examples of Object-Oriented Programming Languages:

Java: A widely-used OOP language known for its portability across platforms (Write Once, Run Anywhere) and strong community support.

C++: An extension of the C programming language that includes OOP features, allowing for both procedural and object-oriented programming.

Python: A versatile language that supports multiple programming paradigms, including OOP, and is known for its readability and simplicity.

C#: Developed by Microsoft, C# is primarily used for developing Windows applications and supports OOP principles.

Ruby: An OOP language known for its elegant syntax and focus on simplicity and productivity.

3.logical programing language

- ⇨ Logical programming languages are a type of programming language that is based on formal logic. In these languages, programs are expressed in terms of relations, and computation is performed through logical inference. The primary

paradigm of logical programming is to define facts and rules, and then query these facts to derive conclusions.

Examples of Logical Programming Languages:

Prolog: One of the most well-known logical programming languages, Prolog is used for tasks that involve complex data structures and relationships, such as natural language processing and artificial intelligence.

Mercury: A logic programming language that emphasizes efficiency and is designed for real-world applications. It combines features of logic programming with functional programming.

Datalog: A subset of Prolog, Datalog is used primarily in databases and knowledge representation. It focuses on querying and reasoning about data.

Answer Set Programming (ASP): A form of declarative programming oriented towards difficult combinatorial search problems. It is used in artificial intelligence and knowledge representation.

4.functional programing language

- ⇨ Functional programming languages are designed to support and encourage a programming style that treats computation as the evaluation of mathematical functions. This paradigm emphasizes the use of functions as the primary building blocks of programs, promoting immutability and first-class functions.

Examples of Functional Programming Languages:

Haskell: A purely functional programming language known for its strong static typing, lazy evaluation, and emphasis on purity. It is widely used in academia and industry for complex applications.

Lisp: One of the oldest programming languages, Lisp supports functional programming and is known for its unique syntax and powerful macro system. Variants include Common Lisp and Scheme.

Scala: A hybrid language that combines functional and object-oriented programming paradigms. It runs on the Java Virtual Machine (JVM) and is interoperable with Java.

F#: A functional-first language that runs on the .NET platform, F# supports both functional and object-oriented programming and is used for a variety of applications.

Erlang: A functional programming language designed for building concurrent and distributed systems. It is known for its fault tolerance and is used in telecommunications and messaging systems.

Q4:-what are the main difference between higher level and lower level programing languages?

Higher level programming language	**lower level programming lang**
Higher-level programming languages are designed to be more human-readable and easier to use	lower-level languages are closer to machine code and offer more control over hardware
High abstraction and closer to human language	Low abstraction and closer to machine code
Built-in error handling features	Limited error handling, requires manual checks
Python, Java, C++, JavaScriptv	Assembly language, Machine code

Q5:- describe the roles of the client and server in web communication

Client

The client is typically a web browser or application that initiates requests to a server. It is the user-facing side of web communication and is responsible for the following:

User Interface: The client provides the graphical user interface (GUI) through which users interact with the web application. This includes displaying web pages, forms, and other content.

Request Generation: When a user performs an action (e.g., clicking a link, submitting a form), the client generates an HTTP request to the server. This request includes information such as the requested resource (URL), request method (GET, POST, etc.), and any additional data (e.g., form inputs).

Data Presentation: After receiving a response from the server, the client processes and displays the data to the user. This may involve rendering HTML, CSS, and JavaScript to create a dynamic and interactive experience.

Session Management: The client often manages user sessions through cookies or tokens, allowing the server to recognize returning users and maintain state across multiple requests.

Error Handling: The client is responsible for handling errors that may occur during communication, such as displaying error messages when a resource is not found (404) or when there is a server error (500).

Server

- ⇨ The server is a remote machine or service that processes requests from clients and provides the requested resources or data. Its roles include:

- ⇨ **Request Handling**: The server listens for incoming HTTP requests from clients and processes them based on the requested resource and method. It determines how to respond to each request.
- ⇨ **Data Processing**: The server may perform various operations, such as querying a database, processing business logic, or performing calculations, to generate the appropriate response.
- ⇨ **Resource Management**: The server manages and serves various resources, including HTML pages, images, videos, and APIs. It ensures that these resources are accessible to clients.
- ⇨ **Response Generation**: After processing a request, the server generates an HTTP response, which includes a status code (indicating success or failure), headers (providing metadata), and the requested content (e.g., HTML, JSON).
- ⇨ **Security and Authentication**: The server is responsible for implementing security measures, such as authentication and authorization, to protect sensitive data and ensure that only authorized users can access certain resources.
- ⇨ **Session Management**: The server may also manage user sessions, storing session data and ensuring that it is associated with the correct client.
- ⇨

Q6:- explain the function of the TCP/IP model and its layers

The TCP/IP model is designed to enable communication between different devices on a network. It provides a set of protocols that govern how data is transmitted, ensuring reliable and efficient communication.

Layers of the TCP/IP Model

1. **Application Layer**
    - **Function**: This layer provides network services directly to end-user applications. It enables software applications to communicate over the network.
    - **Protocols**: Common protocols include HTTP (for web browsing), FTP (for file transfer), SMTP (for email), and DNS (for domain name resolution).
    - **Example**: When a user accesses a website, the browser uses HTTP to request and display web pages.
2. **Transport Layer**
    - **Function**: The Transport Layer is responsible for end-to-end communication, ensuring data is delivered reliably and in the correct order. It manages error detection and correction, as well as flow control.
    - **Protocols**: The main protocols are TCP (Transmission Control Protocol), which provides reliable communication, and UDP (User Datagram Protocol), which offers faster, connectionless communication.

- **Example**: When sending an email, TCP ensures that all data packets arrive intact and in the correct sequence.

3. **Internet Layer**
    - **Function**: This layer handles the routing of data packets across different networks. It defines the addressing scheme and manages the delivery of packets from the source to the destination.
    - **Protocols**: The primary protocol is IP (Internet Protocol), which includes IPv4 and IPv6. Other protocols include ICMP (Internet Control Message Protocol) for error messages and diagnostics.
    - **Example**: When a user sends a message, the Internet Layer determines the best path for the data packets to reach the recipient.
4. **Network Interface Layer (Link Layer)**
    - **Function**: The Network Interface Layer is responsible for the physical transmission of data over the network medium. It defines how data is formatted for transmission and how devices on the same local network communicate.
    - **Protocols**: This layer includes protocols such as Ethernet, Wi-Fi, and ARP (Address Resolution Protocol).
    - **Example**: When data is sent over a local network, Ethernet frames are used to encapsulate the data for transmission over physical cables.

Q7 explain client server communication

Client-server communication is a model used in network architecture where multiple clients (users or devices) interact with a centralized server to request and receive services or resources. This model is fundamental to many applications and services on the internet, including web browsing, email, and file sharing. Here's an overview of how client-server communication works:

Components of Client-Server Communication

1. **Client**
    - **Definition**: A client is a device or application that initiates requests for services or resources from a server. Clients can be computers, smartphones, tablets, or any device capable of network communication.
    - **Function**: Clients send requests to the server and wait for responses. They typically provide a user interface for users to interact with the server.

2. **Server**

    - **Definition**: A server is a powerful computer or application that provides resources, services, or data to clients over a network. Servers can host websites, manage databases, or provide file storage.
    - **Function**: Servers listen for incoming requests from clients, process those requests, and send back the appropriate responses. They can handle multiple client requests simultaneously.

Communication Process

1. **Request Initiation**:
    - The client initiates communication by sending a request to the server. This request typically includes information about the desired service or resource, such as a web page or a file.
2. **Request Transmission**:
    - The request is transmitted over the network using a specific protocol (e.g., HTTP for web requests). The request may include headers that provide additional context, such as the type of data being requested.
3. **Request Processing**:
    - Upon receiving the request, the server processes it. This may involve querying a database, performing calculations, or accessing files. The server determines the appropriate response based on the request.
4. **Response Generation**:
    - After processing the request, the server generates a response. This response may include the requested data, an acknowledgment of an action taken, or an error message if something went wrong.
5. **Response Transmission**:
    - The server sends the response back to the client over the network. Like the request, the response is transmitted using the same protocol.
6. **Response Handling**:
    - The client receives the response and processes it. This may involve displaying data to the user, updating the user interface, or taking further actions based on the response.

Types of Client-Server Communication

1. **Synchronous Communication**:
    - In synchronous communication, the client waits for the server to respond before continuing its operations. This is common in applications where immediate feedback is required, such as web browsing.
2. **Asynchronous Communication**:
    - In asynchronous communication, the client can continue its operations without waiting for the server's response. This is often used in applications that can handle multiple tasks simultaneously, such as email clients or messaging apps.

Q8 types of internet connections

- ⇨ There are 6 types of connection
- ⇨ There are several types of internet connections, each with its own characteristics, advantages, and disadvantages. Here's an overview of the most common types of internet connections:

1. Dial-Up

- ⇨ **Description**: Dial-up connections use a standard telephone line to connect to the internet. A modem converts digital data from a computer into analog signals for transmission over the phone line.
- ⇨ **Speed**: Typically up to 56 Kbps.
- ⇨ **Advantages**: Widely available, low cost.
- ⇨ **Disadvantages**: Slow speeds, cannot use the phone and internet simultaneously, and limited bandwidth.

2. DSL (Digital Subscriber Line)

- ⇨ **Description**: DSL uses existing telephone lines but provides faster speeds than dial-up by using higher frequency bands for data transmission.
- ⇨ **Speed**: Ranges from 256 Kbps to over 100 Mbps, depending on the type of DSL (e.g., ADSL, VDSL).
- ⇨ **Advantages**: Always-on connection, faster than dial-up, and does not interfere with phone use.
- ⇨ **Disadvantages**: Speed decreases with distance from the service provider's central office.

3. Cable

- ⇨ **Description**: Cable internet uses coaxial cables, the same infrastructure used for cable television, to provide internet access.
- ⇨ **Speed**: Typically ranges from 10 Mbps to 1 Gbps.
- ⇨ **Advantages**: High speeds, widely available in urban areas, and supports multiple users.

- ⇨ **Disadvantages**: Speeds can slow down during peak usage times due to shared bandwidth.

4. Fiber Optic

- ⇨ **Description**: Fiber optic internet uses light signals transmitted through fiber optic cables to provide high-speed internet access.
- ⇨ **Speed**: Ranges from 100 Mbps to 10 Gbps or more.
- ⇨ **Advantages**: Extremely high speeds, low latency, and high reliability.
- ⇨ **Disadvantages**: Limited availability in some areas and higher installation costs.

5. Satellite

- ⇨ **Description**: Satellite internet connects users to the internet via satellites orbiting the Earth. A satellite dish is required for the connection.
- ⇨ **Speed**: Typically ranges from 12 Mbps to 100 Mbps.
- ⇨ **Advantages**: Available in remote and rural areas where other types of connections may not be feasible.
- ⇨ **Disadvantages**: Higher latency, potential weather interference, and data caps.

6. Wireless (Wi-Fi)

- ⇨ **Description**: Wireless internet connections use radio waves to transmit data between a router and devices within a certain range. This can be provided through various technologies, including DSL, cable, or fiber.
- ⇨ **Speed**: Varies based on the underlying connection type (e.g., DSL, cable).
- ⇨ **Advantages**: Convenient, allows multiple devices to connect without physical cables.
- ⇨ **Disadvantages**: Signal strength can be affected by distance and obstacles, and security risks if not properly secured.

7. Mobile (Cellular)

- ⇨ **Description**: Mobile internet connections use cellular networks (3G, 4G, 5G) to provide internet access to mobile devices and hotspots.
- ⇨ **Speed**: Ranges from a few Mbps (3G) to over 1 Gbps (5G).
- ⇨ **Advantages**: Highly portable, available almost anywhere there is cellular coverage.
- ⇨ **Disadvantages**: Data caps, potential for slower speeds in congested areas, and battery consumption on mobile devices.

8. Fixed Wireless

- ⇨ **Description**: Fixed wireless internet uses radio signals to provide internet access to a fixed location, typically using antennas mounted on buildings or towers.
- ⇨ **Speed**: Ranges from 1 Mbps to 100 Mbps or more.
- ⇨ **Advantages**: Useful in rural areas where wired connections are not available.

⇨ **Disadvantages**: Line of sight is often required, and speeds can be affected by weather conditions.

Q9 how dose broadband different from fiber optic internet

Key Differences Between Broadband and Fiber Optic Internet

- **Transmission Method**:
    - Broadband can utilize various technologies such as DSL, cable, and satellite, which often rely on electrical signals or radio waves.
    - Fiber optic internet uses light signals transmitted through thin glass or plastic fibers, allowing for faster data transfer.
- **Speed and Performance**:
    - Fiber optic internet typically offers significantly higher speeds and lower latency than traditional broadband options.
    - While broadband can provide high-speed internet, its performance may vary based on the technology used and network congestion.
- **Reliability**:
    - Fiber optic connections are generally more reliable, with less susceptibility to interference from environmental factors compared to other broadband types.
    - Traditional broadband methods, such as DSL and cable, may experience slower speeds during peak usage times.
- **Bandwidth Capacity**:
    - Fiber optic internet has a much higher bandwidth capacity, making it ideal for data-intensive activities like streaming, gaming, and large file transfers.
    - Other broadband types may have limitations on bandwidth, affecting performance when multiple users are online simultaneously.
- **Cost and Availability**:
    - Fiber optic internet can be more expensive to install and maintain due to the advanced technology and infrastructure required.
    - Broadband options like DSL and cable are often more widely available, especially in rural areas, but may not provide the same level of performance.

Q10 what are the difference between http and https protocols?

HTTP (Hypertext Transfer Protocol) and HTTPS (Hypertext Transfer Protocol Secure) are both protocols used for transmitting data over the internet, but they have key differences, particularly in terms of security. Here's a detailed comparison:

Key Differences Between HTTP and HTTPS

1. **Security**:
    - **HTTP**: Data transmitted over HTTP is not encrypted, making it vulnerable to interception and eavesdropping by malicious actors. This means that any data exchanged between the client and server can be read by anyone who can access the network.
    - **HTTPS**: HTTPS uses encryption to secure the data transmitted between the client and server. It employs protocols such as SSL (Secure Sockets Layer) or TLS (Transport Layer Security) to encrypt the data, ensuring that it cannot be easily intercepted or tampered with.
2. **Port Number**:
    - **HTTP**: The default port for HTTP is port 80.
    - **HTTPS**: The default port for HTTPS is port 443.
3. **URL Prefix**:
    - **HTTP**: URLs that use HTTP begin with "http://".
    - **HTTPS**: URLs that use HTTPS begin with "https://", indicating that the connection is secure.
4. **Data Integrity**:
    - **HTTP**: Since HTTP does not provide encryption, there is no guarantee that the data has not been altered during transmission. This makes it susceptible to man-in-the-middle attacks.
    - **HTTPS**: HTTPS ensures data integrity by using checksums and encryption, which helps verify that the data sent and received has not been altered.
5. **Authentication**:
    - **HTTP**: There is no built-in mechanism for verifying the identity of the server. This can lead to phishing attacks, where users may unknowingly connect to malicious servers.
    - **HTTPS**: HTTPS uses digital certificates issued by trusted Certificate Authorities (CAs) to authenticate the identity of the server. This helps users verify that they are communicating with the intended server.
6. **SEO and Browser Support**:

- **HTTP**: Websites using HTTP may be viewed as less secure by modern web browsers, which can lead to warnings for users.
- **HTTPS**: Search engines like Google give preference to HTTPS websites in their rankings, and browsers often display a padlock icon in the address bar to indicate a secure connection. This can enhance user trust and improve SEO.

http	**https**
HTTP stands for HyperText Transfer Protocol	HTTPS stands for HyperText Transfer Protocol Secure
http work on application layer	https work on transport layer
http speed is faster than https	https speed slow compare to http
http are not more secure compare to https	https more secure compare to http

Q11 :- What is the role of encryption in securing applications?

What is Encryption?

Encryption is the process of converting readable data (plaintext) into unreadable data (ciphertext) using a specific algorithm and a secret key. Only someone with the correct key can decrypt and access the original data.

Roles of Encryption in Application Security:

1. **Data Protection:**
    - Keeps sensitive information (passwords, credit card numbers, etc.) safe from unauthorized access.
    - Prevents data leaks even if a system is compromised.

   Secure Communication:
    - Ensures secure data transfer between client and server (e.g., HTTPS uses SSL/TLS encryption).
    - Protects against eavesdropping and man-in-the-middle (MITM) attacks.

   User Privacy:
    - Maintains confidentiality of user information by making data unreadable to attackers.

- Builds trust with users, especially in finance, healthcare, and messaging apps.

Data Integrity:

- Helps ensure that data has not been altered during transmission or storage.
- Some encryption algorithms also include hashing or digital signatures for this.

2. **Regulatory Compliance:**
    - Many laws (like GDPR, HIPAA, PCI-DSS) require encryption to protect user and customer data.
    - Helps avoid legal issues and heavy fines.

Types of Encryption Used:

- **Symmetric Encryption:** Same key for encryption and decryption (e.g., AES).
- **Asymmetric Encryption:** Public key for encryption, private key for decryption (e.g., RSA).
- **End-to-End Encryption:** Only sender and receiver can read the data (used in WhatsApp, Signal).

Q12 :- : What is the difference between system software and application software?

System software and application software are two fundamental categories of software that serve different purposes in a computer system. Here's a detailed comparison of the two:

System Software

1. **Definition**: System software is designed to manage and control the hardware components of a computer system. It provides a platform for running application software and facilitates communication between the hardware and software.
2. **Examples**:
    - **Operating Systems**: Windows, macOS, Linux, and Android are examples of operating systems that manage hardware resources and provide a user interface.

- **Device Drivers**: Software that allows the operating system to communicate with hardware devices, such as printers, graphics cards, and network adapters.
- **Utility Programs**: Tools that perform maintenance tasks, such as disk management, antivirus software, and backup tools.

3. **Functionality**:
    - Manages hardware resources (CPU, memory, storage, etc.).
    - Provides a user interface for users to interact with the computer.
    - Facilitates the execution of application software.
    - Ensures system security and stability.
4. **User Interaction**: System software typically runs in the background and is not directly interacted with by users, although users may interact with the operating system interface.

Application Software

1. **Definition**: Application software is designed to perform specific tasks or applications for the user. It is built to help users accomplish particular functions, such as word processing, data analysis, or graphic design.
2. **Examples**:
    - **Productivity Software**: Microsoft Office (Word, Excel, PowerPoint), Google Workspace (Docs, Sheets, Slides).
    - **Web Browsers**: Chrome, Firefox, Safari.
    - **Media Players**: VLC, Windows Media Player.
    - **Graphics Software**: Adobe Photoshop, CorelDRAW.
3. **Functionality**:
    - Performs specific tasks or solves particular problems for users.
    - Utilizes the capabilities provided by the system software to function.
    - Can be standalone applications or part of a suite of applications.
4. **User Interaction**: Application software is designed for direct interaction with users, providing graphical user interfaces (GUIs) and features tailored to user needs.

System software	Application software
System software is desigen to manage and control the hardware and basic oprations	Application software is design to help users porfomens specific task and activity
Operates the computer and provides a platform for applications	Helps users perform activities like writing, browsing, etc.
System software is usually pre instal in the system	Application software is instald by user
System software is run in background and minimal user intarect	Application software is directali use by user
For exampal os(windows,linux) ,drivers	Ms world, googal chrom

Q13:- What is the significance of modularity in software architecture?

- ⇨ Modularity is a key principle in software architecture that refers to dividing a software system into smaller, independent, and manageable components called modules.
- ⇨ Each module performs a specific function and can be developed, tested, and maintained separately

Significance of Modularity in Software Architecture

Modularity means designing a software system as a collection of separate, independent components (modules), each handling a specific piece of functionality.

Key Significance of Modularity:

- ⇨ **Easier Maintenance:**
  - Each module can be updated or fixed independently without affecting the entire system.
- ⇨ **Reusability:**
  - Common modules (e.g., login, payment) can be reused across different applications.
- ⇨ **Simplified Testing:**
  - Modules can be tested individually (unit testing), making debugging easier and faster.
- ⇨ **Team Collaboration:**
  - Multiple developers or teams can work on different modules at the same time without conflict.
- ⇨ **Scalability:**
  - You can scale specific parts of the system (e.g., only the database or API layer) without redesigning everything.
- ⇨ **Better Code Organization:**
  - Clear structure makes the code easier to read, understand, and manage.
- ⇨ **Improved Security:**

- Modules with sensitive logic can be isolated and protected more effectively.

Q14 :- Why are layers important in software architecture?

- ⇨ in software architecture Layers are used to organize and separate different concerns or responsibilities of a software system
- ⇨ Each layer handles a specific function and interacts only with the layer directly above or below it
- ⇨ This layered approach is fundamental to building scalable, maintainable, and well-structured applications

Importance of Layers in Software Architecture:

- ⇨ **Separation of Concerns:**
  - Each layer has a clear role (e.g., UI layer handles visuals, business layer processes logic).
  - Reduces complexity and makes the system easier to understand.
- ⇨ **Easier Maintenance and Updates:**
  - You can change or upgrade one layer (like the UI) without affecting others.
- ⇨ **Reusability:**
  - Common functionality (like data access logic) can be reused across different parts of the application.
- ⇨ **Better Debugging and Testing:**
  - Since each layer is independent, issues can be traced and fixed more easily.
- ⇨ **Team Efficiency:**
  - Different teams can work on different layers simultaneously (e.g., frontend and backend developers).

Q15:- Explain the importance of a development environment in software production.

- ⇨ A development environment is a set of tools, software, and settings that developers use to write, test, and debug their code
- ⇨ It typically includes a code editor, compiler or interpreter, debugger, version control system, and other supporting tools.
- ⇨ That's why development environment importance in software production
- ⇨ A development environment is vital in software production for several reasons:

1. Consistency and Standardization

- ⇨ Ensures uniformity across teams, reducing discrepancies and easing onboarding for new developers.

2. Enhanced Productivity

- ⇨ Integrates tools and automation, allowing developers to focus on coding rather than tool management.

3. Testing and Debugging

- ⇨ Mimics production settings for effective testing and provides debugging tools to identify issues quickly.

4. Version Control and Collaboration

- ⇨ Integrates with version control systems, facilitating collaboration and management of code changes.

5. Configuration Management

- ⇨ Simplifies environment configuration, ensuring consistent application performance across different setups.

6. Security and Compliance

- ⇨ Controls access to sensitive data and helps identify vulnerabilities early in the development process.

7. Support for CI/CD

- ⇨ Integrates with CI/CD pipelines for automated testing and deployment, providing immediate feedback on code quality.

8. Cross-Platform Support

- ⇨ Supports multiple languages and platforms, enabling testing across various devices and operating systems.
- ⇨

Q16:- What is the difference between source code and machine code?

- ⇨ Source code and machine code are two fundamental concepts in computer programming and software development. They represent different stages in the process of creating executable programs. Here's a detailed comparison of the two:

Source Code

1. Definition: Source code is the human-readable set of instructions written in a programming language. It is the original code that developers write to create software applications.

2. Characteristics:

- ⇨ **Human-Readable**: Source code is written in a high-level programming language (e.g., Python, Java, C++) that is designed to be easily understood by humans.

- ⇨ **Syntax and Semantics**: It follows specific syntax rules and semantics defined by the programming language, allowing developers to express algorithms and logic clearly.
- ⇨ **Editable**: Source code can be modified, maintained, and updated by programmers to add features, fix bugs, or improve performance.

3. **Compilation/Interpretation**: Source code must be translated into machine code through a process called compilation (for compiled languages) or interpretation (for interpreted languages) before it can be executed by a computer.

Machine Code

1. **Definition**: Machine code is the low-level binary code that is directly executed by a computer's central processing unit (CPU). It consists of a series of binary instructions that the hardware can understand and execute.
2. **Characteristics**:
    - **Machine-Readable**: Machine code is not human-readable; it consists of binary digits (0s and 1s) that represent instructions and data.
    - **Architecture-Specific**: Machine code is specific to a particular CPU architecture (e.g., x86, ARM), meaning that code compiled for one type of processor may not run on another without modification.
    - **Execution**: Machine code is the final output of the compilation or interpretation process and is what the CPU executes to perform tasks.

3. Performance: Machine code is typically faster to execute than source code because it is already in a form that the CPU can directly process without further translation.

Source code	**Machine code**
Source code is the original code written by programmers in a human-friendly language	Machin code is final code that the computer understand and executes
Source code write in higher level language	Machin code is write in lower level language
Source code is easily understood and edits by human	Machin code is not understood without any special tool
Source code Can be easily changed and improved	Machin code is Difficult to modify directly
print("Hello, World!") in Python	01001000 01100101 01101100 01101100 01101111 (binary format)

Q17 :- Why is version control important in software development?

Version control is a critical aspect of software development that helps manage changes to code and track the history of a project. Its importance can be summarized through several key points:

1. Change Tracking

- **History of Changes**: Version control systems (VCS) maintain a complete history of changes made to the codebase, allowing developers to review, compare, and revert to previous versions if necessary.

2. Collaboration

- **Team Coordination**: VCS enables multiple developers to work on the same project simultaneously without conflicts. Changes can be merged efficiently, facilitating collaboration among team members.

3. Branching and Merging

- **Feature Development**: Developers can create branches to work on new features or fixes independently. Once completed, these branches can be merged back into the main codebase, ensuring a clean integration of changes.

4. Backup and Recovery

- **Data Safety**: Version control acts as a backup system, protecting against data loss. If a mistake is made or data is corrupted, developers can easily restore previous versions of the code.

5. Accountability and Audit Trails

- **Tracking Contributions**: VCS records who made specific changes and when, providing accountability and an audit trail for the development process. This is useful for understanding the evolution of the code and for compliance purposes.

6. Continuous Integration and Deployment (CI/CD)

- **Automated Workflows**: Version control integrates with CI/CD pipelines, enabling automated testing and deployment processes that enhance the speed and reliability of software releases.

7. Experimentation

- **Safe Testing**: Developers can experiment with new ideas in isolated branches without affecting the main codebase, allowing for innovation while maintaining stability.

8. Improved Code Quality

- **Code Reviews**: VCS facilitates code reviews by allowing team members to review changes before they are merged, leading to higher quality code and better adherence to coding standards.

Q18:- What are the benefits of using Github for students?

⇨ GitHub is a widely used platform for version control and collaboration, particularly in software development. For students, using GitHub offers numerous benefits that can enhance their learning experience and prepare them for future careers. Here are some key advantages:

1. Version Control

⇨ **Track Changes**: GitHub allows students to track changes made to their code over time, making it easy to revert to previous versions if needed.
⇨ **Branching and Merging**: Students can create branches to experiment with new features or ideas without affecting the main codebase. This encourages experimentation and learning.

2. Collaboration

⇨ **Team Projects**: GitHub facilitates collaboration on group projects, allowing multiple students to work on the same codebase simultaneously. They can review each other's code, suggest changes, and merge contributions.
⇨ **Pull Requests**: Students can submit pull requests to propose changes, enabling peer review and discussion before integrating new code into the main project.

3. Portfolio Development

⇨ **Showcase Work**: Students can use GitHub to showcase their projects and contributions, creating a portfolio that demonstrates their skills to potential employers.
⇨ **Public Repositories**: By making repositories public, students can share their work with the community, gaining visibility and recognition for their efforts.

4. Learning and Skill Development

- ⇨ **Hands-On Experience**: Using GitHub provides practical experience with version control systems, which are essential skills in the software development industry.
- ⇨ **Access to Open Source Projects**: Students can contribute to open source projects, gaining exposure to real-world coding practices and collaborating with experienced developers.

5. Community and Networking

- ⇨ **Engagement with Developers**: GitHub has a large community of developers, allowing students to connect with others, seek help, and learn from experienced programmers.
- ⇨ **Networking Opportunities**: Engaging in open source projects and collaborating with others can lead to networking opportunities and potential job offers.

6. Documentation and Learning Resources

- ⇨ **Project Documentation**: Students can learn the importance of documentation by writing README files and comments in their code, which helps others understand their projects.
- ⇨ **Learning Resources**: GitHub hosts numerous repositories with tutorials, sample projects, and educational resources that students can use to enhance their learning.

7. Integration with Other Tools

- ⇨ **Continuous Integration/Continuous Deployment (CI/CD)**: GitHub integrates with various CI/CD tools, allowing students to automate testing and deployment processes, which is valuable for modern software development practices.
- ⇨ **Collaboration with Other Platforms**: GitHub can be integrated with other tools like Slack, Trello, and JIRA, enhancing project management and communication.

8. Free Access for Students

- ⇨ **GitHub Education Program**: GitHub offers free access to its Pro features for students through the GitHub Education program, providing additional tools and resources for learning and development.

Q19:- : What are the differences between open-source and proprietary software?

⇨ Open-source software and proprietary software are two distinct categories of software that differ in terms of licensing, accessibility, and development practices. Here's a detailed comparison of the two:

Open-Source Software

Definition:

Open-source software is software whose source code is made available to the public. Users can view, modify, and distribute the code freely.

Licensing:

⇨ Open-source software is released under licenses that comply with the Open Source Definition, allowing users to use, modify, and share the software.
⇨ Common open-source licenses include the GNU General Public License (GPL), MIT License, and Apache License.

Accessibility:

⇨ The source code is accessible to anyone, enabling users to study how the software works and make improvements or customizations.
⇨ Users can contribute to the development of the software, report bugs, and suggest features.

Cost:

⇨ Open-source software is often available for free, although some projects may offer paid support or additional features.

Community Development:

⇨ Open-source projects are typically developed collaboratively by a community of developers, which can lead to rapid innovation and diverse contributions.
⇨ Users can participate in the development process, fostering a sense of community and shared ownership.

Examples:

⇨ Popular open-source software includes the Linux operating system, Apache web server, Mozilla Firefox, and the LibreOffice suite.

Proprietary Software

Definition:

Proprietary software is software that is owned by an individual or a company. The source code is not made available to the public, and users must adhere to specific licensing agreements.

Licensing:

⇨ Proprietary software is distributed under licenses that restrict access to the source code and limit how the software can be used, modified, or shared.

⇨ Users typically pay for a license to use the software, and the terms of use are defined by the software vendor.

Accessibility:

⇨ The source code is not accessible to users, meaning they cannot modify or customize the software.
⇨ Users must rely on the vendor for updates, bug fixes, and support.

Cost:

⇨ Proprietary software usually requires a purchase or subscription fee, which can vary based on the software's features and intended use.

Vendor Control:

⇨ Development and updates are controlled by the software vendor, which can lead to slower innovation compared to open-source projects.
⇨ Users have limited influence over the direction of the software and its features.

Examples:

⇨ Common proprietary software includes Microsoft Windows, Adobe Photoshop, and commercial software applications like Microsoft Office.
⇨

Open-source software	**Proprietary software**
It is freely available for view modify and shar	It is not available for user
Open source code is usually free	Proprietary software is offer paid and licensed
Open licenses (e.g., GNU, MIT)	Restricted licenses from the developer
Can be customized by anyone	Customization is limited or not allowed
Linux, Mozilla Firefox, LibreOffice	Windows, Microsoft Office, Adobe Photoshop

Q**20 :- How does GIT improve collaboration in a software development team?**

Git is a distributed version control system that significantly enhances collaboration within software development teams. Here are several ways in which Git improves collaboration:

1. Version Control

- **Track Changes**: Git allows team members to track changes made to the codebase over time. Each change is recorded with a unique identifier (commit), making it easy to review the history of modifications.
- **Revert Changes**: If a mistake is made, Git enables users to revert to previous versions of the code, minimizing the risk of losing work or introducing errors.

2. Branching and Merging

- **Feature Branches**: Developers can create separate branches for new features, bug fixes, or experiments. This allows them to work on different aspects of the project simultaneously without interfering with the main codebase (often referred to as the "main" or "master" branch).
- **Merging Changes**: Once a feature is complete, developers can merge their branches back into the main branch. Git provides tools to handle merge conflicts, ensuring that changes from different team members can be integrated smoothly.

3. Collaboration and Code Review

- **Pull Requests**: Git platforms like GitHub, GitLab, and Bitbucket allow developers to submit pull requests when they want to merge their changes. This initiates a review process where team members can comment on the code, suggest improvements, and discuss changes before they are integrated.
- **Code Quality**: Code reviews foster collaboration and knowledge sharing, helping to maintain high code quality and adherence to coding standards.

4. Distributed Development

- **Local Repositories**: Each team member has a complete copy of the repository on their local machine, allowing them to work offline and commit changes without needing a constant internet connection.
- **Synchronization**: Developers can push their changes to a shared remote repository when they are ready, and pull updates from others to keep their local copies in sync.

5. Conflict Resolution

- **Handling Conflicts**: When multiple developers make changes to the same part of the code, Git helps identify conflicts during the merging process. Developers can resolve these conflicts collaboratively, ensuring that the final code reflects the contributions of all team members.

6. History and Accountability

- **Commit History**: Git maintains a detailed history of all changes, including who made each change and when. This accountability helps teams understand the evolution of the codebase and facilitates troubleshooting.

- **Blame Feature**: Git provides a "blame" feature that allows developers to see who last modified a specific le of code, making it easier to identify the source of issues or to ask questions about specific changes.

7. Integration with CI/CD

- **Continuous Integration/Continuous Deployment (CI/CD)**: Git integrates seamlessly with CI/CD tools, allowing teams to automate testing and deployment processes. This ensures that code changes are continuously tested and deployed, improving collaboration and reducing integration issues.

8. Documentation and Communication

- **Commit Messages**: Developers can write descriptive commit messages that explain the purpose of changes, providing context for other team members.
- **Wiki and Issue Tracking**: Many Git platforms offer built-in wikis and issue tracking systems, enabling teams to document processes, track bugs, and manage project tasks collaboratively.

Q21 :-what is the role of application software in business?

⇨ Application software helps businesses perform specific task efficiently such as managing data, automating process and improving communication.
⇨ Application software plays a crucial role in business by providing tools that enhance efficiency and productivity. Here are the key roles:

1. Enhancing Productivity

⇨ Automates repetitive tasks and streamlines workflows, allowing employees to focus on strategic activities.

2. Facilitating Communication

⇨ Improves internal and external communication through email, messaging, and CRM systems, fostering teamwork and customer engagement.

3. Data Management and Analysis

⇨ Organizes and stores data, enabling easy access and analysis for informed decision-making.

4. Financial Management

⇨ Manages finances with accounting software, tracks expenses, and assists in budgeting and forecasting.

5. Customer Relationship Management

⇨ CRM systems help track customer interactions and improve satisfaction through personalized service.

6. Sales and Marketing Support

⇨ Streamlines sales processes and enhances marketing strategies through automation tools.

7. Human Resource Management

⇨ HR software simplifies recruitment, onboarding, payroll, and performance management.

8. Project Management

⇨ Helps plan, execute, and monitor projects, ensuring deadlines and resource utilization.

9. Compliance and Security

⇨ Assists in regulatory compliance and protects sensitive data from cyber threats.

Q22 :- What are the main stages of the software development process?

The software development process typically consists of several key stages, each contributing to the successful creation and deployment of software applications. The main stages are:

1. Planning

- **Requirements Gathering**: Identify and document the needs and expectations of stakeholders.
- **Feasibility Study**: Assess the technical, operational, and financial feasibility of the project.

2. Design

- **System Architecture**: Define the overall structure of the software, including components and their interactions.
- **User Interface Design**: Create wireframes and prototypes for the user interface, focusing on user experience.

3. Implementation (Coding)

- **Development**: Write the actual code based on the design specifications, using appropriate programming languages and tools.
- **Version Control**: Use version control systems to manage code changes and collaborate with team members.

4. Testing

- **Unit Testing**: Test individual components for functionality and correctness.

- **Integration Testing**: Ensure that different components work together as intended.
- **System Testing**: Validate the complete system against requirements to ensure it meets expectations.
- **User Acceptance Testing (UAT)**: Involve end-users to verify that the software meets their needs and is ready for deployment.

5. Deployment

- **Release**: Deploy the software to a production environment where users can access it.
- **Training**: Provide training and documentation to users to facilitate adoption.

6. Maintenance

- **Bug Fixes**: Address any issues or bugs that arise after deployment.
- **Updates and Enhancements**: Implement updates to improve functionality, security, and performance based on user feedback and changing requirements.

7. Evaluation

- **Performance Review**: Assess the software's performance and user satisfaction.
- **Lessons Learned**: Document insights and experiences from the project to improve future development processes

Q23 :- Why is the requirement analysis phase critical in software development?

The requirement analysis phase is a crucial step in the software development lifecycle (SDLC) that involves gathering, analyzing, and documenting the needs and expectations of stakeholders regarding a software project. This phase is critical for several reasons:

1. Understanding Stakeholder Needs

- **Clarification of Requirements**: Requirement analysis helps in understanding what stakeholders (clients, users, and other parties) truly need from the software. This ensures that the development team has a clear and accurate understanding of the project goals.
- **Identifying Expectations**: By engaging with stakeholders, the team can identify their expectations, preferences, and any specific constraints that must be considered during development.

2. Defining Scope

- **Scope Management**: Clearly defined requirements help in establishing the scope of the project. This prevents scope creep, where additional features or changes are introduced without proper evaluation, leading to delays and increased costs.
- **Prioritization**: Requirement analysis allows teams to prioritize features based on stakeholder needs, ensuring that the most critical functionalities are developed first.

3. Reducing Risks

- **Identifying Risks Early**: By thoroughly analyzing requirements, potential risks and challenges can be identified early in the project. This allows the team to develop mitigation strategies before they become significant issues.
- **Avoiding Miscommunication**: Clear documentation of requirements helps prevent misunderstandings between stakeholders and the development team, reducing the risk of delivering a product that does not meet expectations.

4. Facilitating Design and Development

- **Guiding Design Decisions**: Well-defined requirements serve as a foundation for the design phase. They guide architects and developers in making informed decisions about system architecture, technology stack, and user interface design.
- **Establishing Acceptance Criteria**: Requirements provide the basis for acceptance criteria, which are used to evaluate whether the final product meets the agreed-upon specifications.

5. Improving Communication

- **Documentation**: The requirement analysis phase results in comprehensive documentation that can be shared with all stakeholders. This documentation serves as a reference point throughout the project, improving communication and alignment.
- **Stakeholder Involvement**: Engaging stakeholders during this phase fosters collaboration and ensures that their voices are heard, leading to a more user-centered product.

6. Cost and Time Efficiency

- **Reducing Rework**: By identifying and addressing requirements early, the likelihood of costly rework later in the development process is minimized. Changes made during later stages of development are often more expensive and time-consuming.

- **Efficient Resource Allocation**: A clear understanding of requirements allows for better planning and allocation of resources, ensuring that the project stays on schedule and within budget.

7. Enhancing Quality

- **Quality Assurance**: Well-defined requirements contribute to higher quality software by ensuring that the final product aligns with user needs and expectations. This leads to greater user satisfaction and fewer defects.
- **Testing and Validation**: Requirements serve as a basis for developing test cases, ensuring that the software is thoroughly tested against the specified criteria.

Q24 :- What is the role of software analysis in the development process?

- ⇨ Software analysis plays a keys role in understanding and definding the functional and non-functional requirement of a system.
- ⇨ It involves studying the problem, identifying user needs ,and specifying what the software should do.
- ⇨ This phase helps in planning the system accurately , choosing the right approach and avoiding errors later in development
- ⇨ It ensures the final software meets the business goals and user expectations

1. Foundation for Success

- Establishes a clear understanding of stakeholder needs and defines project scope, preventing scope creep.

2. Risk Mitigation

- Identifies potential issues early, reducing the risk of costly changes later and assessing feasibility within constraints.

3. Improved Communication

- Engages stakeholders, fostering collaboration and producing clear documentation for reference throughout development.

4. Enhanced Quality

- Focuses on user needs, leading to software that meets expectations and provides a positive experience, while also aiding in test case creation.

5. Cost Efficiency

- Reduces rework and allows for more accurate cost estimation, saving time and resources.

6. Facilitating Change Management

- Enables better management of changes and prioritization of features based on business value.

Q25 :- What are the key elements of system design?

The key elements of system design encompass various aspects that ensure the system is effective, efficient, and meets user requirements. Here are the main elements:

1. Architecture Design

- **System Architecture**: Defines the overall structure of the system, including components, modules, and their interactions.
- **Design Patterns**: Utilizes established design patterns to solve common problems and improve system organization.

2. User Interface Design

- **Usability**: Focuses on creating an intuitive and user-friendly interface that enhances user experience.
- **Accessibility**: Ensures the system is accessible to all users, including those with disabilities.

3. Data Design

- **Data Modeling**: Involves creating data models (e.g., ER diagrams) to represent data structures and relationships.
- **Database Design**: Defines how data will be stored, accessed, and managed, including schema design and normalization.

4. Component Design

- **Modularity**: Breaks down the system into smaller, manageable components or modules that can be developed and tested independently.
- **Interface Specifications**: Defines how components will interact with each other, including APIs and communication protocols.

5. Security Design

- **Authentication and Authorization**: Implements measures to ensure that only authorized users can access the system and its data.
- **Data Protection**: Incorporates security measures to protect data from unauthorized access and breaches.

6. Performance Design

- **Scalability**: Ensures the system can handle increased loads and can be scaled up or down as needed.
- **Efficiency**: Focuses on optimizing resource usage (CPU, memory, bandwidth) to enhance system performance.

7. Integration Design

- **Interoperability**: Ensures the system can integrate with other systems and services, facilitating data exchange and functionality.
- **Middleware**: Utilizes middleware solutions to enable communication between different components or systems.

8. Testing and Validation

- **Test Design**: Plans for testing strategies, including unit tests, integration tests, and system tests to ensure the system meets requirements.
- **Validation**: Ensures that the system fulfills its intended purpose and meets user needs through user acceptance testing.

9. Documentation

- **Design Documentation**: Provides detailed documentation of the design decisions, architecture, and components for future reference and maintenance.
- **User Documentation**: Creates user manuals and guides to help users understand and effectively use the system.

Q26:- Why is software testing important?

⇨ **Why is Software Testing Important?**
⇨ **Software Testing** is important because it ensures that the software **works correctly**, is **safe to use**, and meets the **user's needs** before it is released.

Simple Definition:

⇨ **Software Testing** is the process of checking a program for **errors, bugs, or issues** to make sure it behaves as expected.
⇨

Reasons Why Testing is Important:

Reason	Explanation
Finds Bugs Early	Detects errors before the software reaches users.
Ensures Security	Helps protect data from hackers and security breaches.
Improves Quality	Ensures the software meets high standards and works smoothly.

Reason	Explanation
Increases Customer Trust	Satisfied users are more likely to trust and use the product.
Saves Time & Cost	Fixing bugs early is cheaper and faster than fixing them after release.
Ensures Functionality	Confirms that the software performs all its intended tasks correctly.
Supports Maintenance	Makes future updates and improvements easier and safer.

Q27:- What types of software maintenance are there?

1. Corrective Maintenance

- **Purpose**: Involves fixing defects or bugs in the software that are identified after deployment.
- **Activities**: Debugging, patching, and resolving issues that affect functionality or performance.

2. Adaptive Maintenance

- **Purpose**: Adjusts the software to accommodate changes in the environment, such as new operating systems, hardware, or third-party services.
- **Activities**: Modifying the software to ensure compatibility with new technologies or regulations.

3. Perfective Maintenance

- **Purpose**: Enhances the software by adding new features or improving existing functionality based on user feedback or changing requirements.
- **Activities**: Implementing enhancements, optimizing performance, and refining user interfaces.

4. Preventive Maintenance

- **Purpose**: Aims to prevent future issues by making improvements or updates to the software before problems arise.
- **Activities**: Refactoring code, updating documentation, and performing regular system checks to ensure reliability.

5. Emergency Maintenance

- **Purpose**: Addresses critical issues that require immediate attention to restore functionality or security.
- **Activities**: Rapidly deploying fixes for severe bugs or vulnerabilities that could lead to system failure or data breaches.

6. Scheduled Maintenance

- **Purpose**: Involves planned updates and maintenance activities that are performed at regular intervals.
- **Activities**: Routine checks, updates, and optimizations to ensure the software remains efficient and secure.

Q28:- What are the key differences between web and desktop applications?

Web application	**Desktop application**
Runs in a web browser; no installation needed	Requires installation on the user's device
Accessible from any device with internet	Accessible only on the installed device
Mostly requires an internet connection	Can work offline (depends on the application)
May be slower due to browser limitations	Generally faster with direct access to system resources
Depends on browser and web server security	Depends on local system security

Q29:- What are the advantages of using web applications over desktop applications?

- ⇨ Web applications are accessible from any device, require no installation, work across platforms, and are easy to update.
- ⇨ They support centralized data storage, real-time collaboration, and are often more cost-effective and scalable than desktop applications.
- ⇨ Web applications offer several advantages over traditional desktop applications, making them increasingly popular for both users and developers. Here are the key benefits:

1. Accessibility

- ⇨ **Platform Independence**: Web applications can be accessed from any device with a web browser, regardless of the operating system (Windows, macOS, Linux, etc.).
- ⇨ **Remote Access**: Users can access web applications from anywhere with an internet connection, facilitating remote work and collaboration.

2. No Installation Required

- ⇨ **Ease of Use**: Users do not need to install software on their devices, reducing setup time and complexity.
- ⇨ **Automatic Updates**: Updates and new features are deployed on the server side, ensuring all users have access to the latest version without manual installation.

3. Cost-Effectiveness

- ⇨ **Lower Development Costs**: Developing a single web application can be more cost-effective than creating multiple versions for different operating systems.
- ⇨ **Reduced Maintenance Costs**: Centralized maintenance and updates reduce the overall cost of managing the application.

4. Scalability

- ⇨ **Easier Scaling**: Web applications can be scaled more easily to accommodate increased user loads by upgrading server resources or optimizing the application.
- ⇨ **Cloud Integration**: Many web applications leverage cloud services, allowing for flexible scaling and resource management.

5. Collaboration and Sharing

- ⇨ **Real-Time Collaboration**: Web applications often support real-time collaboration features, enabling multiple users to work on the same document or project simultaneously.
- ⇨ **Data Sharing**: Users can easily share data and resources without the need for complex file transfer processes.

6. Cross-Device Compatibility

- ⇨ **Responsive Design**: Web applications can be designed to work seamlessly across various devices, including desktops, tablets, and smartphones.
- ⇨ **Consistent User Experience**: Users can expect a similar experience regardless of the device they are using.

7. Security

- ⇨ **Centralized Security Management**: Security measures can be implemented and managed centrally on the server, reducing vulnerabilities associated with individual devices.
- ⇨ **Regular Updates**: Web applications can be updated frequently to address security vulnerabilities and improve protection against threats.

8. Integration Capabilities

- ⇨ **API Integration**: Web applications can easily integrate with other web services and APIs, enhancing functionality and enabling data exchange between systems.
- ⇨ **Third-Party Services**: They can leverage third-party services for payment processing, analytics, and more, streamlining development.
- ⇨

Q30:- What role does UI/UX design play in application development?

- ⇨ UI/UX design plays a key role in making applications user-friendly, visually appealing, and easy to navigate.
- ⇨ It enhances user satisfaction, improves usability, and helps retain users by providing a smooth and engaging experience.
- ⇨ UI (User Interface) and UX (User Experience) design are essential in application development for several key reasons:

1. User -Centric Approach

- ⇨ Understanding Needs: Focuses on researching user preferences and pain points to create relevant solutions.
- ⇨ Empathy: Develops personas and user journeys to enhance intuitive design.

2. Improving Usability

- ⇨ Intuitive Navigation: Ensures easy navigation and task completion for users.
- ⇨ Clear Interaction: Provides feedback and guidance for user interactions.

3. Enhancing Aesthetics

- ⇨ Visual Appeal: Creates attractive interfaces that enhance user experience.
- ⇨ Brand Consistency: Maintains brand identity through consistent design elements.

4. Increasing Engagement

- ⇨ User Retention: Encourages users to return, boosting engagement and loyalty.
- ⇨ Emotional Connection: Fosters a connection that leads to recommendations.

5. Facilitating Accessibility

- ⇨ Inclusive Design: Ensures usability for individuals with disabilities.
- ⇨ Responsive Design: Adapts interfaces for various devices and screen sizes.

6. Reducing Development Costs

- ⇨ Early Prototyping: Identifies issues early through wireframes and prototypes.
- ⇨ Minimizing Rework: Reduces significant changes during development.

7. Guiding Development

- ⇨ Design Specifications: Provides guidelines for developers to align with user experience.
- ⇨ Collaboration: Works closely with developers to address technical constraints.

8. Measuring Success

- ⇨ User Testing: Gathers feedback to improve usability and satisfaction.
- ⇨ Analytics: Tracks user behavior to inform future design enhancements.

Q31:- What are the differences between native and hybrid mobile apps?

Native mobile apps	**Hybrid mobile app**
Native mobile app is built for a specific platform	Run on a multiple platform using a single codebase
High performance and responsiveness	May have low performance than native apps
User platform specific language (swift,kotlin)	Uses web technology (html,css,javascipt)
Take more time and cost for each platform	Faster development using one codebase
Full access to device features	Limited or dependent on plugins

Q32:- What is the significance of DFDs in system analysis?

- ⇨ DFDs (Data Flow Diagrams) are important in system analysis because they visually represent how data moves through a system.
- ⇨ They help analysts understand processes, data sources, storage, and the flow of information, making it easier to design or improve a system effectively
- ⇨ Data Flow Diagrams (DFDs) are significant in system analysis for several reasons:

1. Visual Representation

- ⇨ **Clarity**: DFDs provide a clear visual of system processes, making data flow easier to understand.
- ⇨ **Simplification**: They break down complex processes into manageable components.

2. Identification of Data Flows

- ⇨ **Data Movement**: Illustrate how data is input, processed, stored, and output.
- ⇨ **Sources and Destinations**: Highlight where data originates and where it is sent.

3. Facilitating Communication

- ⇨ **Stakeholder Engagement**: Serve as a communication tool among analysts, developers, and stakeholders.
- ⇨ **Feedback Mechanism**: Provide a basis for discussions and feedback.

4. Identifying System Requirements

- ⇨ **Requirement Gathering**: Help document system requirements by visualizing necessary data flows.
- ⇨ **Scope Definition**: Define the system's scope by outlining included processes.

5. Analyzing Efficiency

- ⇨ **Bottleneck Identification**: Reveal inefficiencies in data flow for optimization.
- ⇨ **Redundancy Detection**: Identify redundant processes to streamline operations.

6. Supporting System Design

- ⇨ **Foundation for Design**: Guide developers in creating system architecture.
- ⇨ **Integration Planning**: Assist in planning component integration.

7. Documentation and Maintenance

- ⇨ **System Documentation**: Serve as part of system documentation for future reference.
- ⇨ **Change Management**: Help assess the impact of changes on data flows.

Q33:- What are the pros and cons of desktop applications compared to web applications?

Aspect	Desktop application	Web application
Pros	-High performance -Work offline -Better access to system resources	-Accessible anywhere -No installation needed -Easy to update and maintain
Cons	- Requires installation - Platform dependent	- Needs internet connection

	- Manual updates needed	- May have slower performance - Limited access to device features

Q34:- How do flowcharts help in programming and system design?

- ⇨ Flowcharts help in programming and system design by visually representing the flow of logic or processes.
- ⇨ They make it easier to understand, plan, debug, and communicate how a system or program works step by step.

1. Visual Representation

- ⇨ **Clarity**: Provide a clear visual of processes, making logic easier to understand.
- ⇨ **Simplification**: Break down complex processes into manageable steps.

2. Planning and Design

- ⇨ **Process Mapping**: Help visualize the structure and flow before coding.
- ⇨ **Identifying Requirements**: Assist in determining necessary components.

3. Debugging and Troubleshooting

- ⇨ **Error Identification**: Help pinpoint logical errors and bottlenecks.
- ⇨ **Simplified Troubleshooting**: Make tracing execution flow easier.

4. Communication Tool

- ⇨ **Stakeholder Engagement**: Facilitate understanding among developers and non-technical members.
- ⇨ **Documentation**: Serve as visual documentation for reference.

5. Standardization

- ⇨ **Consistent Approach**: Promote a standardized process design for collaboration.
- ⇨ **Best Practices**: Incorporate best practices for efficient programming.

6. Facilitating Code Development

- ⇨ **Guiding Implementation**: Serve as a guide during coding to follow intended logic.
- ⇨ **Modular Design**: Aid in designing modular components for easier maintenance.

7. Training and Onboarding

- ⇨ **Learning Tool**: Help new team members quickly understand processes.
- ⇨ **Knowledge Transfer**: Provide a visual representation for easy sharing.

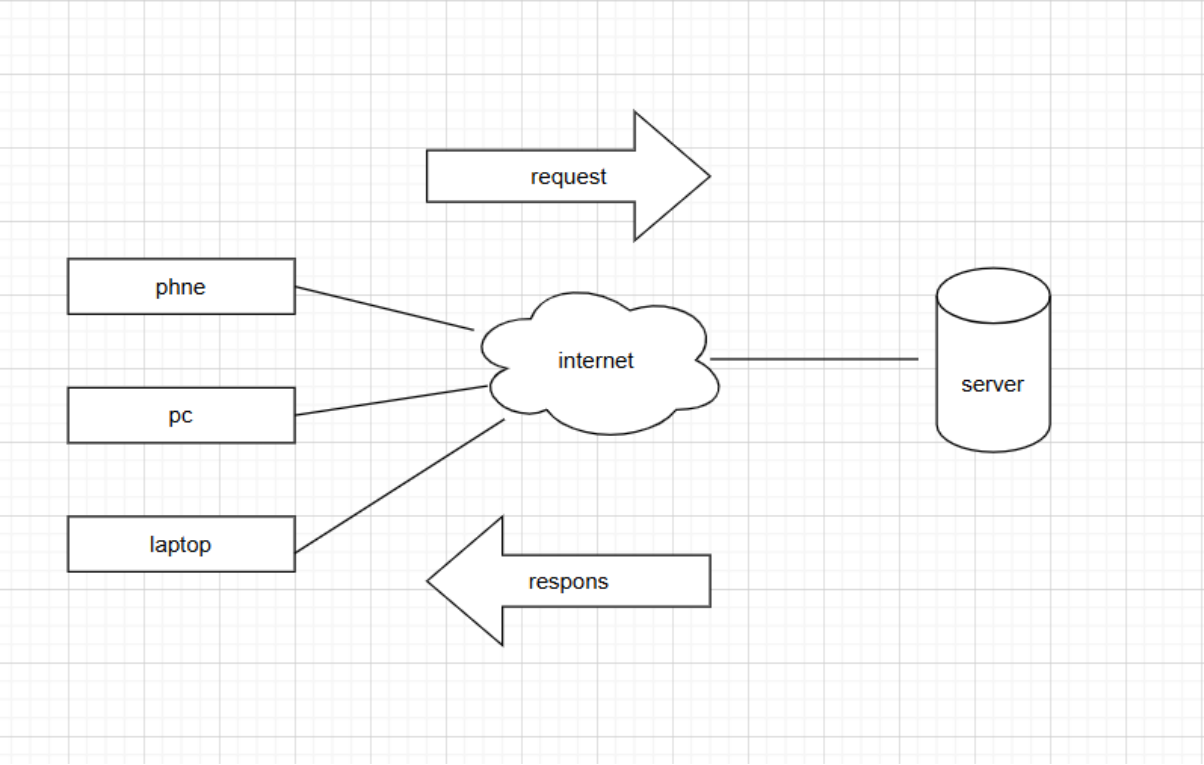
Lab questions:-

Q1 :- Write a simple "Hello World" program in two different programming languages of your choice. Compare the structure and syntax.

file:///D:\sneha\name.cpp

c	Python
printf("...") with format specifiers	print("...") is simple
#include <stdio.h> for input/output	No need for imports for print
Must compile first (gcc)	Run directly with python
Ends lines with ;	

Q2:- Research and create a diagram of how data is transmitted from a client to a server over the internet.



Q3:- Research different types of internet connections (e.g., broadband, fiber, satellite) and list their pros and cons.

Type	Description	Pros	Cons
Dial-up	Uses telephone lines to connect to the internet.	- Very low cost - Available in rural areas	- Very slow speeds (max ~56 Kbps) - Blocks phone line when used
Dsl(digital subscriber line)	Uses telephone lines but allows internet and phone simultaneously.	- Affordable - Faster than dial-up	- Slower than modern broadband - Speed
Cable	Uses coaxial TV cables to provide internet.	- High speeds - Better	- Shared bandwidth may reduce speed
Fiber optic	Transmits data as light through glass or plastic fibers.	- Very high speeds (up to 1 Gbps or more) - Reliable and low latency	- Expensive installation - Limited availability in rural areas
Satellite	Connects to the internet via orbiting satellites.	- Available in remote or rural areas	- High latency - Slower speeds - Affected by weather
Mobile data	Uses cellular networks to access internet through SIM cards or dongles.	- Portable - No cables required - Fast with 5G	- Limited data plans - Signal strength varies
Hotspot	Internet shared via devices like phones or routers using wireless technology.	Convenient - Easy setup	- Limited range - Speed depends on source connection

Q6:- Identify and explain three common application security vulnerabilities. Suggest possible solutions

1:- sql injection

What is it?
SQL Injection occurs when an attacker inserts malicious SQL code into an input field (like login forms or search boxes) to access or manipulate the database.

2:-cross side scripting

What is it?
XSS allows attackers to inject malicious JavaScript into web pages viewed by other users. This script can steal cookies, redirect pages, or deface content.

3:- Broken Authentication

What is it?
This happens when an application's login system is poorly implemented — allowing attackers to guess passwords, steal sessions, or bypass authentication.

Q7:- Identify and classify 5 applications you use daily as either system software or application software.

1. Google Chrome (Web Browser):- Link: https://www.google.com/chrome/

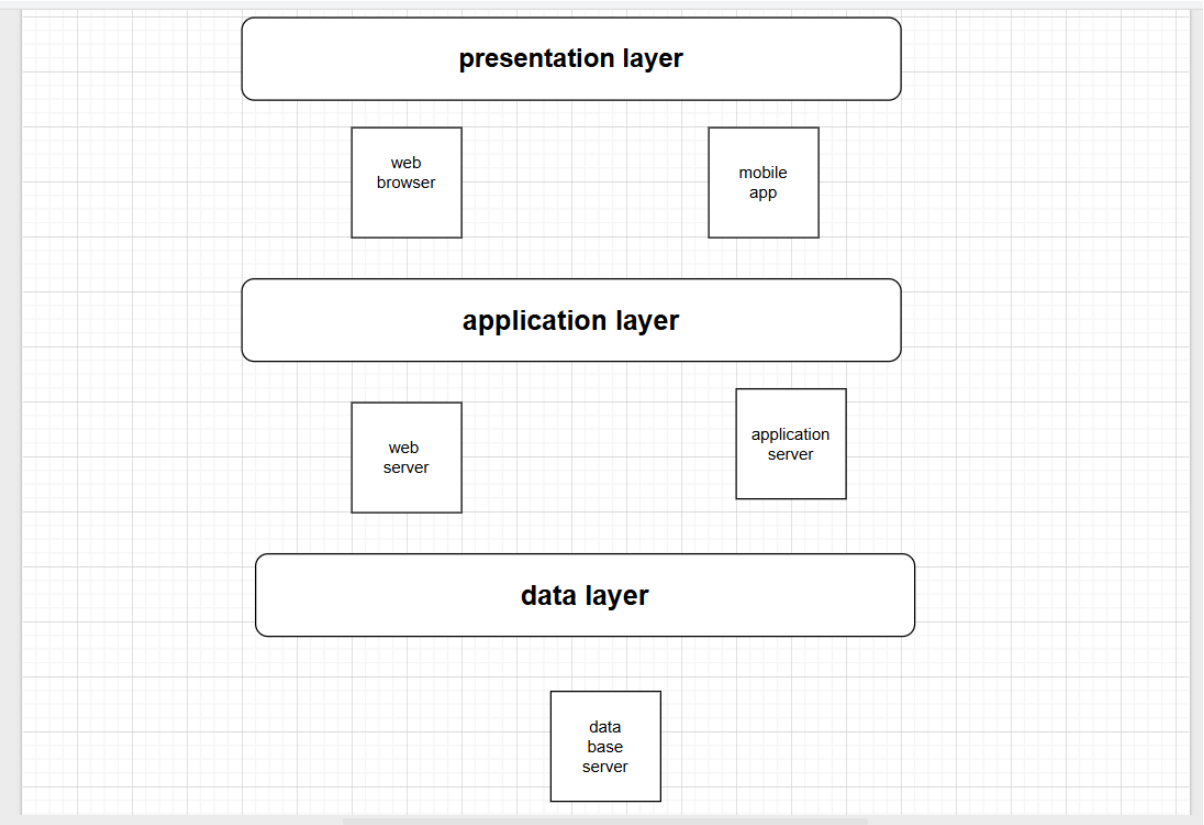
2. Microsoft Word (Part of Microsoft Office) :- Link: https://www.microsoft.com/en-us/microsoft-365/word

3. Windows Operating System (Windows 10/11):- https://www.microsoft.com/en-in/software-download/windows11

4. **WhatsApp Desktop/Web :- WhatsApp Web: https://web.whatsapp.com/**

5.**instagram :- https://www.instagram.com/**

Q8:- Design a basic three-tier software architecture diagram for a web application.



Q9:- Create a case study on the functionality of the presentation, business logic, and data access layers of a given software system.

⇨ **Introduction**

This case study examines the functionality of the Presentation, Business Logic, and Data Access Layers in an e-commerce software system.

1. Presentation Layer

- **Functionality**: Manages the user interface (UI) using HTML, CSS, and JavaScript. Handles user input, displays data, and performs client-side validation.
- **Example**: When a user searches for a product, the UI dynamically displays results without reloading the page.

2. Business Logic Layer

- **Functionality**: Processes requests from the Presentation Layer, enforces business rules, transforms data, and manages security.
- **Example**: When adding a product to the cart, it checks availability and applies discounts.

3. Data Access Layer

- **Functionality**: Interacts with the database to perform CRUD operations, manages connections, and handles errors.
- **Example**: When an order is placed, it saves the order details and retrieves updated inventory status.

Interaction Flow

1. User searches for a product (Presentation Layer).
2. Request sent to Business Logic Layer.
3. Business Logic Layer processes the request and queries the Data Access Layer.
4. Data Access Layer retrieves data from the database.
5. Business Logic Layer formats the data and sends it back.
6. Presentation Layer updates the UI with search results.

Q10:- Explore different types of software environments (development, testing, production). Set up a basic environment in a virtual machine.

Types of Software Environments

1. **Development Environment**
   - **Purpose**: This is where developers write and test their code. It includes tools and libraries necessary for coding.
   - **Characteristics**:
     - Local setup on developers' machines.
     - May include integrated development environments (IDEs), version control systems, and debugging tools.
     - Frequent changes and iterations.
2. **Testing Environment**
   - **Purpose**: This environment is used to test the software for bugs and issues before it is released. It mimics the production environment as closely as possible.
   - **Characteristics**:
     - Isolated from the development environment to ensure stability.
     - Includes automated testing tools and frameworks.
     - May involve different types of testing (unit, integration, system, acceptance).
3. **Production Environment**
   - **Purpose**: This is the live environment where the software is deployed and used by end-users.
   - **Characteristics**:
     - Highly stable and secure.
     - Performance is critical, and downtime should be minimized.
     - Changes are carefully managed and typically involve a deployment process.

Q11:- Create a student account on Github and collaborate on a small project with a classmate.

⇨ Creating a student account on GitHub and collaborating on a small project with a classmate involves several steps. Here's a detailed guide to help you through the process:

Step 1: Create a Student Account on GitHub

1. **Sign Up for GitHub**:
   - Go to GitHub.
   - Click on the "Sign up" button in the top right corner.
2. **Fill in Your Details**:
   - Enter your email address, create a password, and choose a username.
   - GitHub offers free accounts, which are suitable for students.
3. **Verify Your Account**:
   - Follow the prompts to verify your email address and complete the sign-up process.
4. **Set Up Your Profile**:
   - After logging in, you can set up your profile by adding a profile picture, bio, and any other relevant information.

Step 2: Collaborate on a Small Project

1. **Create a New Repository**:
   - Click on the "+" icon in the top right corner and select "New repository."
   - Name your repository (e.g., **student-collaboration-project**).
   - Optionally, add a description and choose to make it public or private.
   - Check the box to initialize with a README file.
   - Click "Create repository."
2. **Invite Your Classmate to Collaborate**:
   - Go to the repository page.
   - Click on the "Settings" tab.
   - In the left sidebar, click on "Manage access."
   - Click on "Invite teams or people."
   - Enter your classmate's GitHub username or email address and click "Invite."
3. **Clone the Repository Locally**:
   - Both you and your classmate should clone the repository to your local machines:

bash
RunCopy code
git clone https://github.com/your_username/student-collaboration-project.git

4. **Make Changes to the Project**:
   - Navigate to the cloned repository:

bash
RunCopy code
cd student-collaboration-project

   - Create or edit files as needed. For example, you could create a simple Python script:

python2 lines
Click to close
# hello.py
print("Hello from our collaborative project!")

5. **Commit and Push Changes**:
    - After making changes, check the status:

bash
RunCopy code
git status

    - Add the changes to the staging area:

bash
RunCopy code
git add hello.py

    - Commit the changes:

bash
RunCopy code
git commit -m "Add hello.py for our collaborative project"

    - Push the changes to GitHub:

bash
RunCopy code
git push origin master

6. **Collaborate and Review Changes**:
    - Your classmate can pull the latest changes, make their own modifications, and push them back to the repository.
    - Use GitHub's "Pull Requests" feature to review each other's changes:
        - After pushing changes, your classmate can create a pull request from the repository page.
        - You can review the changes, leave comments, and merge them into the main branch.

Q12:- Create a list of software you use regularly and classify them into the following categories: system, application, and utility software.

Here’s a list of software commonly used, classified into the categories of system software, application software, and utility software:

System Software

1. **Operating Systems**:
    - Windows 10/11
    - macOS
    - Linux (e.g., Ubuntu, Fedora)
2. **Device Drivers**:
    - Graphics drivers (e.g., NVIDIA, AMD)
    - Printer drivers
3. **Firmware**:

- BIOS/UEFI firmware for motherboards

Application Software

1. **Office Productivity**:
    - Microsoft Office (Word, Excel, PowerPoint)
    - Google Workspace (Docs, Sheets, Slides)
2. **Web Browsers**:
    - Google Chrome
    - Mozilla Firefox
    - Microsoft Edge
3. **Communication Tools**:
    - Slack
    - Microsoft Teams
    - Zoom
4. **Development Tools**:
    - Visual Studio Code
    - IntelliJ IDEA
    - GitHub Desktop
5. **Media Players**:
    - VLC Media Player
    - Windows Media Player
6. **Graphics and Design**:
    - Adobe Photoshop
    - GIMP
    - Canva

Utility Software

1. **File Management**:
    - WinRAR (file compression)
    - 7-Zip (file compression)
2. **Antivirus and Security**:
    - Norton Antivirus
    - Malwarebytes
3. **Backup and Recovery**:
    - Acronis True Image
    - Windows Backup and Restore
4. **System Monitoring**:
    - CCleaner (system optimization)
    - HWMonitor (hardware monitoring)
5. **Disk Management**:
    - Disk Cleanup (Windows)
    - GParted (Linux)

Q13:- Write a report on the various types of application software and how they improve productivity.

Introduction

Application software refers to programs designed to perform specific tasks for users. Unlike system software, which manages hardware and provides a platform for applications, application software directly assists users in accomplishing their goals. This report explores various types of application software and discusses how they enhance productivity in different domains.

Types of Application Software

1. Word Processing Software

Examples: Microsoft Word, Google Docs, LibreOffice Writer

Word processing software allows users to create, edit, format, and print text documents. Features such as spell check, grammar check, templates, and collaboration tools enable users to produce professional documents efficiently. This software improves productivity by streamlining the writing process and facilitating easy revisions and sharing.

2. Spreadsheet Software

Examples: Microsoft Excel, Google Sheets, LibreOffice Calc

Spreadsheet software is used for data organization, analysis, and visualization. Users can perform calculations, create graphs, and analyze data trends using formulas and functions. This type of software enhances productivity by enabling quick data manipulation and decision-making based on real-time analysis.

3. Presentation Software

Examples: Microsoft PowerPoint, Google Slides, Prezi

Presentation software helps users create visual presentations to communicate ideas effectively. With features like templates, animations, and multimedia integration, users can engage their audience and convey information clearly. This software improves productivity by reducing the time spent on designing presentations and enhancing the overall impact of the message.

4. Database Management Software

Examples: Microsoft Access, MySQL, Oracle Database

Database management software allows users to create, manage, and manipulate databases. It provides tools for data entry, querying, and reporting. By organizing large amounts of data efficiently, this software improves productivity by enabling quick access to information and facilitating data-driven decision-making.

5. Project Management Software

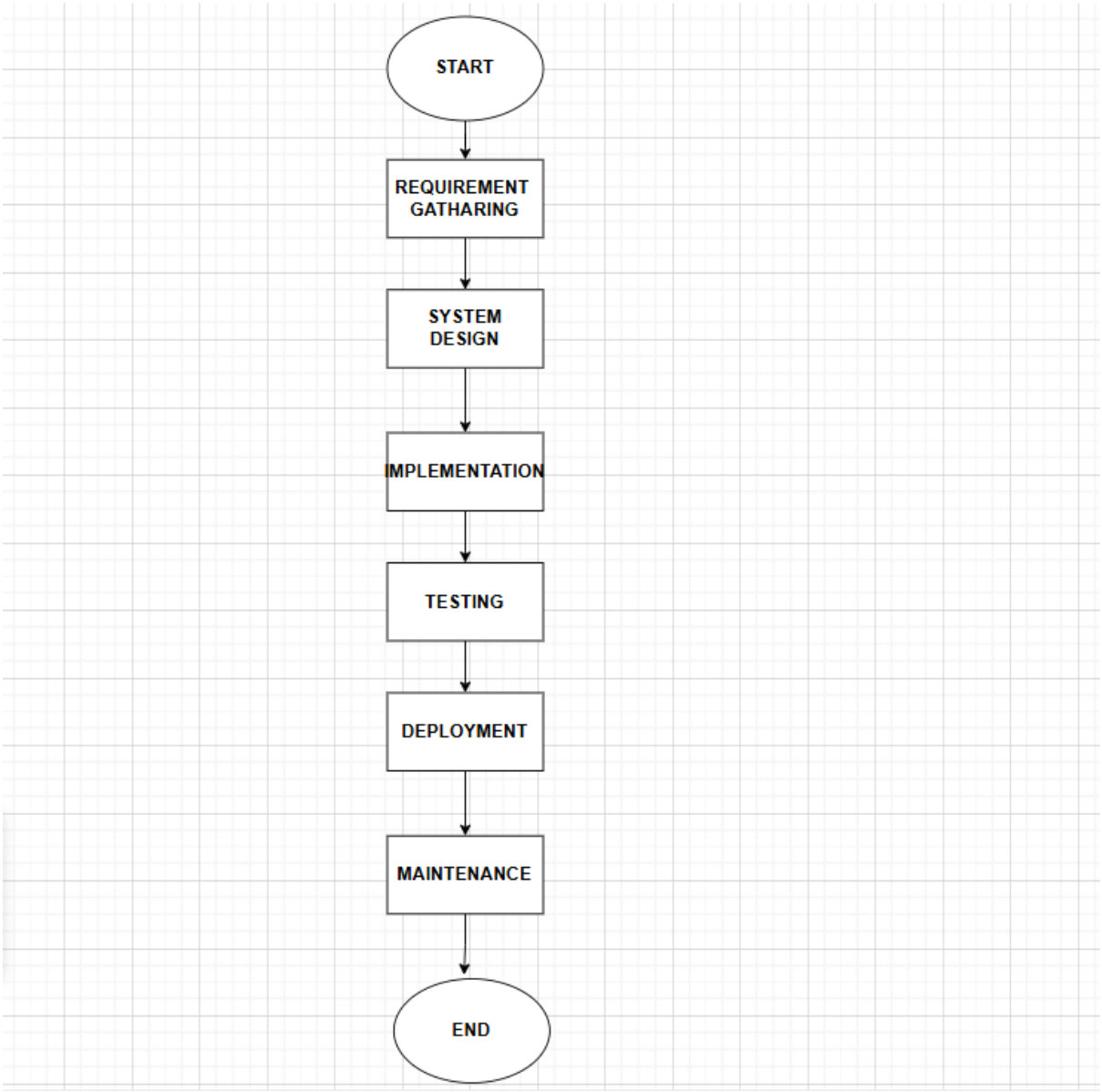
Examples: Trello, Asana, Microsoft Project
Project management software assists teams in planning, executing, and monitoring projects. Features such as task assignment, progress tracking, and collaboration tools help streamline workflows and improve communication. This software enhances productivity by ensuring that projects are completed on time and within budget.

6. Communication Software
Examples: Slack, Microsoft Teams, Zoom
Communication software facilitates real-time communication and collaboration among team members. Features like instant messaging, video conferencing, and file sharing improve connectivity and teamwork. This type of software boosts productivity by reducing communication barriers and enabling quick decision-making.

7. Graphic Design Software
Examples: Adobe Photoshop, Canva, CorelDRAW
Graphic design software allows users to create and edit visual content, including images, logos, and marketing materials. With user-friendly interfaces and powerful design tools, this software enhances creativity and efficiency. It improves productivity by enabling designers to produce high-quality visuals quickly.

8. Accounting Software
Examples: QuickBooks, FreshBooks, Xero
Accounting software automates financial tasks such as invoicing, expense tracking, and payroll management. By simplifying complex financial processes, this software saves time and reduces errors. It improves productivity by providing businesses with accurate financial insights and facilitating compliance with regulations.

Q17:- Create a flowchart representing the Software Development Life Cycle (SDLC).



Q18:- Write a requirement specification for a simple library management system.

1. Purpose

To automate the library operations such as managing books, issuing and returning them, tracking users, and generating reports.

2. Scope

This system will:

- Maintain records of books and students
- Allow students to search, issue, and return books
- Enable librarians to manage inventory and users
- Automatically calculate fines for late returns
- Generate reports on transactions and inventory

3. Users & Roles

User	Role & Access
Student	Login, search books, issue/return books
Librarian	Manage books and student records, handle issues
Admin *(optional)*	System setup and configuration

4. Key Functional Requirements

- **User Login** (with validation)
- **Book Management** (add, edit, delete, search)
- **Student Registration** (add, update profile)
- **Issue Book** (check availability, max limit)
- **Return Book** (auto fine if overdue)
- **Search Books** (by title, author, subject)
- **View Issued Books**
- **Report Generation** (daily/weekly/monthly)

5. Non-Functional Requirements

- **Performance**: Fast access to search and issue books
- **Security**: Passwords hashed, role-based access
- **Usability**: Simple, clean, and intuitive UI
- **Scalability**: Support up to 1,000 users and 10,000 books
- **Reliability**: Accurate data and stable system

6. System Requirements (Tech Stack)

Component	Technology
Frontend	HTML/CSS/JS or Java Swing
Backend	PHP / Python / Java
Database	MySQL / SQLite
OS	Windows / Linux

7. Glossary

Term	Description
Book ID	Unique identifier for each book
Fine	Late return penalty

Term	Description
Transaction	Record of issue or return operation

Q19:- Perform a functional analysis for an online shopping system.

1. Purpose

To automate the library operations such as managing books, issuing and returning them, tracking users, and generating reports.

2. Scope

This system will:

- Maintain records of books and students
- Allow students to search, issue, and return books
- Enable librarians to manage inventory and users
- Automatically calculate fines for late returns
- Generate reports on transactions and inventory

3. Users & Roles

User	Role & Access
Student	Login, search books, issue/return books
Librarian	Manage books and student records, handle issues
Admin *(optional)*	System setup and configuration

4. Key Functional Requirements

- **User Login** (with validation)
- **Book Management** (add, edit, delete, search)
- **Student Registration** (add, update profile)
- **Issue Book** (check availability, max limit)
- **Return Book** (auto fine if overdue)
- **Search Books** (by title, author, subject)
- **View Issued Books**
- **Report Generation** (daily/weekly/monthly)

5. Non-Functional Requirements

- **Performance**: Fast access to search and issue books
- **Security**: Passwords hashed, role-based access
- **Usability**: Simple, clean, and intuitive UI
- **Scalability**: Support up to 1,000 users and 10,000 books
- **Reliability**: Accurate data and stable system

6. System Requirements (Tech Stack)

Component	Technology
Frontend	HTML/CSS/JS or Java Swing
Backend	PHP / Python / Java
Database	MySQL / SQLite
OS	Windows / Linux

7. Glossary

Term	Description
Book ID	Unique identifier for each book
Fine	Late return penalty
Transaction	Record of issue or return operation

20:- Perform a functional analysis for an online shopping system.

Here is a **Functional Analysis** for an **Online Shopping System** — broken down clearly into **functional areas, use cases, and responsibilities**.

Functional Analysis – Online Shopping System

🔹 1. User Roles

Role	Description
Customer	Browses, purchases, and tracks orders
Admin	Manages products, users, and orders
Seller *(optional)*	Adds and manages own products
Delivery Staff *(optional)*	Updates delivery status

🔹 2. Core Functionalities

A. User Management

- Register/Login/Logout
- Edit Profile & Address Book

- Password recovery

B. Product Browsing

- View products by category, brand, etc.
- Product search and filters (price, rating, etc.)
- View product details

C. Shopping Cart

- Add/remove products to/from cart
- Update quantity
- View total price

D. Order Management

- Place order (checkout)
- Choose delivery address and payment method
- Order confirmation & summary

E. Payment Integration

- Support for online payments (UPI, cards, wallets)
- Cash on delivery (COD)
- Invoice generation

F. Order Tracking

- View order status: pending, shipped, delivered
- Cancel or return orders

G. Admin Panel

- Add/update/delete products
- Manage user accounts
- Manage orders and refunds
- View reports (sales, inventory)

🔹 3. Functional Use Case Table

Use Case	Actor	Description
Register/Login	Customer	Access system features
Browse Products	Customer	View/search/filter items
Add to Cart	Customer	Temporarily save items for checkout
Place Order	Customer	Confirm purchase and payment

Use Case	Actor	Description
Manage Inventory	Admin	Add/update/delete products
Process Orders	Admin	Update order status and handle issues
View Reports	Admin	Monitor sales and inventory data
Track Orders	Customer	View delivery status

🔹 **4. System Outputs**

- Product listings and search results
- Order confirmations and emails
- Invoices and reports
- Notifications (order status, delivery updates)

🔹 **5. Integration Points**

- **Payment Gateways** (Paytm, Razorpay, Stripe, etc.)
- **Email/SMS** for notifications
- **Shipping APIs** (e.g., Shiprocket, Delhivery)

Q22 :- Develop test cases for a simple calculator program.

⇨ Here are some test cases for a simple calculator program that performs basic arithmetic operations: addition, subtraction, multiplication, and division. Each test case includes the operation, input values, expected output, and a brief description.

Test Cases for Simple Calculator

Test Case ID	Operation	Input Values	Expected Output	Description
TC1	Addition	2, 3	5	Test addition of two positive integers.
TC2	Addition	-1, 1	0	Test addition of a negative and a posit
TC3	Addition	0, 0	0	Test addition of two zeros.
TC4	Subtraction	5, 3	2	Test subtraction of two positive intege
TC5	Subtraction	3, 5	-2	Test subtraction resulting in a negativ
TC6	Subtraction	0, 0	0	Test subtraction of two zeros.
TC7	Multiplication	4, 5	20	Test multiplication of two positive inte

Test Case ID	Operation	Input Values	Expected Output	Description
TC8	Multiplication	-2, 3	-6	Test multiplication of a negative and a
TC9	Multiplication	0, 5	0	Test multiplication with zero.
TC10	Division	6, 3	2	Test division of two positive integers.
TC11	Division	5, 0	Error/Infinity	Test division by zero (should handle e
TC12	Division	0, 5	0	Test division of zero by a positive integ
TC13	Division	-6, 2	-3	Test division of a negative integer by a
TC14	Mixed	10, 5, 2	10	Test mixed operations: (10 - 5) + 2.
TC15	Mixed	3, 2, 1	6	Test mixed operations: (3 + 2) * 1.

Q23:- : Document a real-world case where a software application required critical maintenance.

Case Study: AWS Kinesis Outage – Critical Software Maintenance
Date: November 25, 2020
Organization: Amazon Web Services (AWS)
Service Affected: AWS Kinesis Data Streams (and dependent services)

What Happened?

- A critical **AWS service called Kinesis** (used for real-time data processing) failed due to an **operating system configuration limit** being unknowingly exceeded.
- This affected other services like **CloudWatch**, **Lambda**, **Auto Scaling**, and **DynamoDB** across **US-East-1 region**.
- Many popular applications and platforms such as **Adobe, Roku, 1Password, and Coinbase** reported outages or degraded performance.

Problem Details

- A hidden configuration in the **OS agent** limited the number of threads a process could spawn.
- An update caused the number of threads needed by the Kinesis front-end servers to exceed this limit.
- As a result, **Kinesis API requests started failing**, triggering a cascade of failures in dependent services.

Critical Maintenance Actions Taken

1. **Emergency diagnostics** to identify the OS-level constraint causing failures.

2. **Code patching** and configuration tuning to increase thread limits.
3. **Service-by-service restart** to recover dependent systems like CloudWatch and Lambda.
4. **Postmortem report** published to increase transparency and accountability.

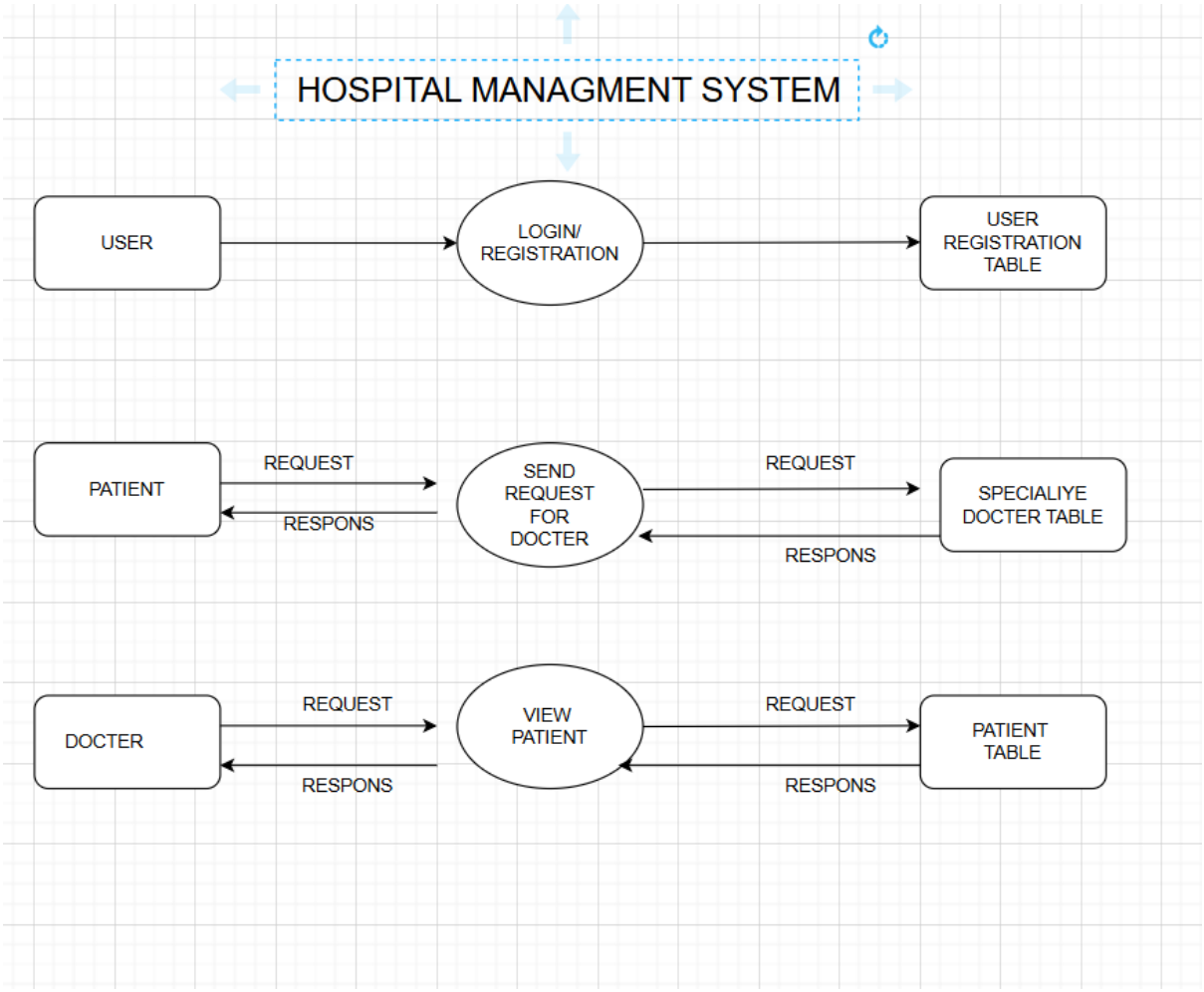
Impact of Maintenance

Area	Impact
Downtime	~5–7 hours of service disruption
Affected Services	23+ AWS services
Customer Impact	Thousands of businesses worldwide
Data Loss	No data loss reported
Resolution	Configuration fix and system reboot

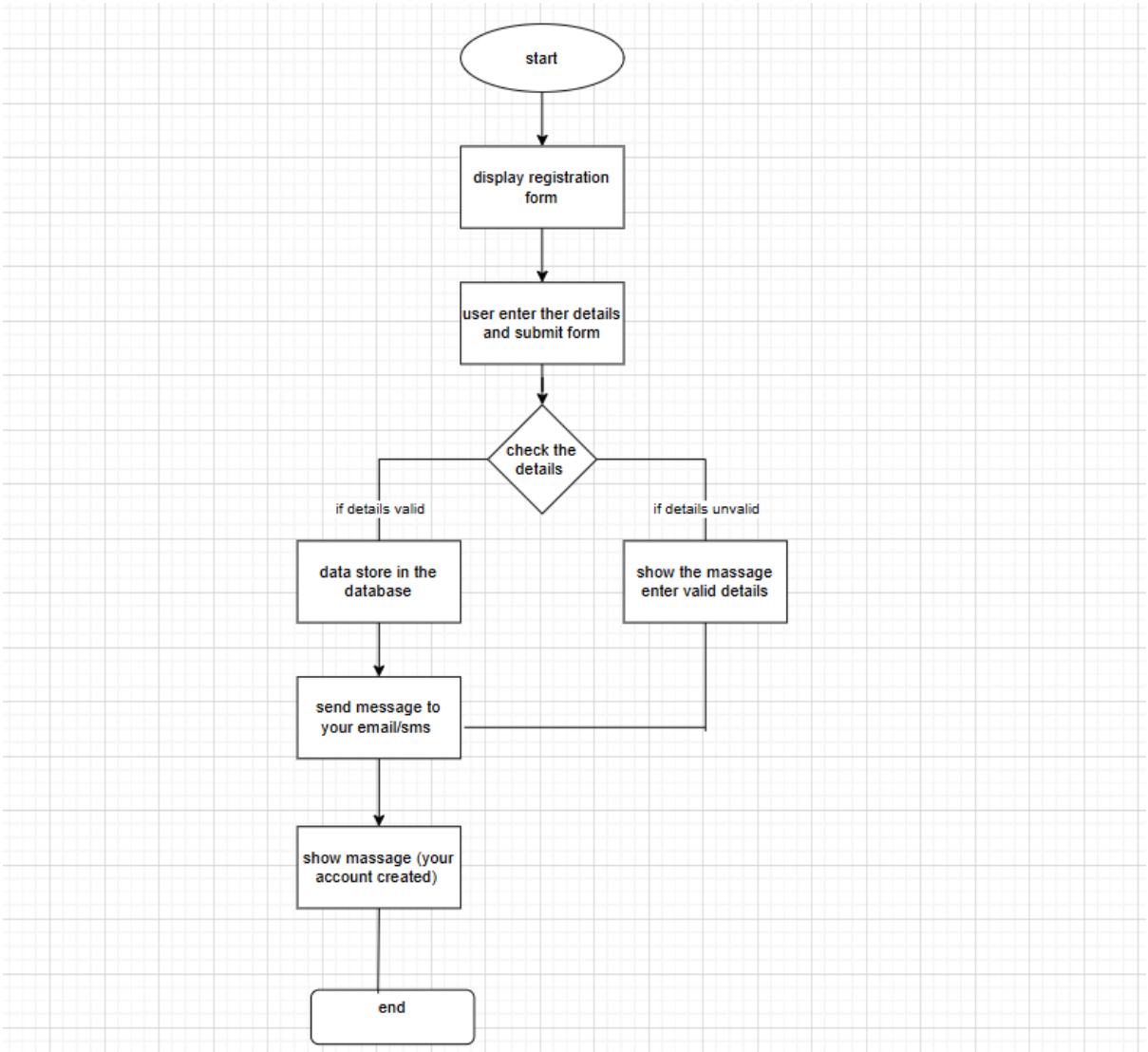
Lessons Learned

- Importance of testing for **invisible OS-level limits**.
- Need for **graceful fallback mechanisms** across services.
- Monitoring must detect not just **failures**, but also **capacity pressure**.

Q24:- Create a DFD for a hospital management system.



Q25:- Draw a flowchart representing the logic of a basic online registration system.


start
display registration form
user enter ther details and submit form
check the details
if details valid
if details unvalid
data store in the database
show the massage enter valid details
send message to your email/sms
show massage (your account created)
end