Module 1 – SE -Overview of IT Industry

1)What is a Program?
A program is a collection of code designed to perform a specific task when executed by a computer.

2) Explain in your own words what a program is and how it functions.
A program is a written set of instructions that a computer can follow to perform tasks and solve problems automatically.

- How a Program Functions:
    1. Write the Code
    2. Translate to Machine Language(binary)
    3. Execute the Instructions
    4. Get Output

3) What is Programming?
Programming is the skill of writing code that instructs computers to perform specific actions.

4) What are the key steps involved in the programming process?

- Understand the problem
- Plan the solution (algorithm, pseudocode)
- Write the code (implementation)
- Compile or interpret the code
- Test the program
- Debug errors
- Final execution
- Documentation and maintenance

5) Types of Programming Languages?

Low level Languages: Machine Learning, Assembly Language
High-Level Languages: C, C++, Java, Python
Domain-Specific Languages: SQL, HTML

6) Differences Between High-Level and Low-Level Programming Languages:

Feature	High-Level Language	Low-Level Language
Abstraction	Closer to human language	Closer to machine hardware
Ease of Use	Easy to learn and understand	Difficult to understand
Examples	Python, Java, C++, JavaScript	Assembly language, Machine code
Portability	Portable across different systems	Not portable (hardware dependent)
Speed of Execution	Slower compared to low-level	Very fast and efficient

Control over Hardware	Less control	More control over memory and CPU
Compilation	Needs a compiler or interpreter	May use assembler or direct execution

7) World Wide Web & How Internet Works?

World Wide Web (WWW):

- The World Wide Web (WWW) is a system of interlinked web pages and resources that can be accessed over the Internet using a web browser.

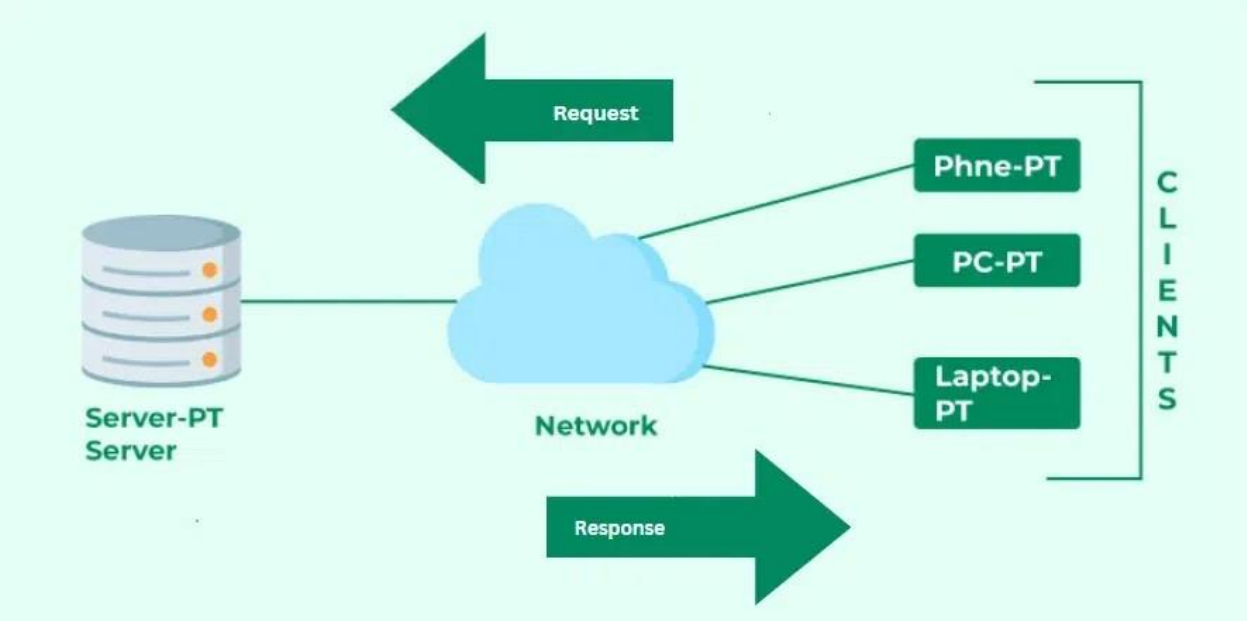
How the Internet Works:

- The Internet is a huge network that connects millions of computers around the world.

Steps :

- You type a website URL in your browser (e.g., www.google.com)
- DNS (Domain Name System) converts it to an IP address (like 142.250.190.78)
- Your device sends a request to that IP address using the Internet
- The web server receives the request, finds the web page
- The web server sends back the data (HTML, CSS, images)
- Your browser receives it and displays the page

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



9) Describe the roles of the client and server in web communication.

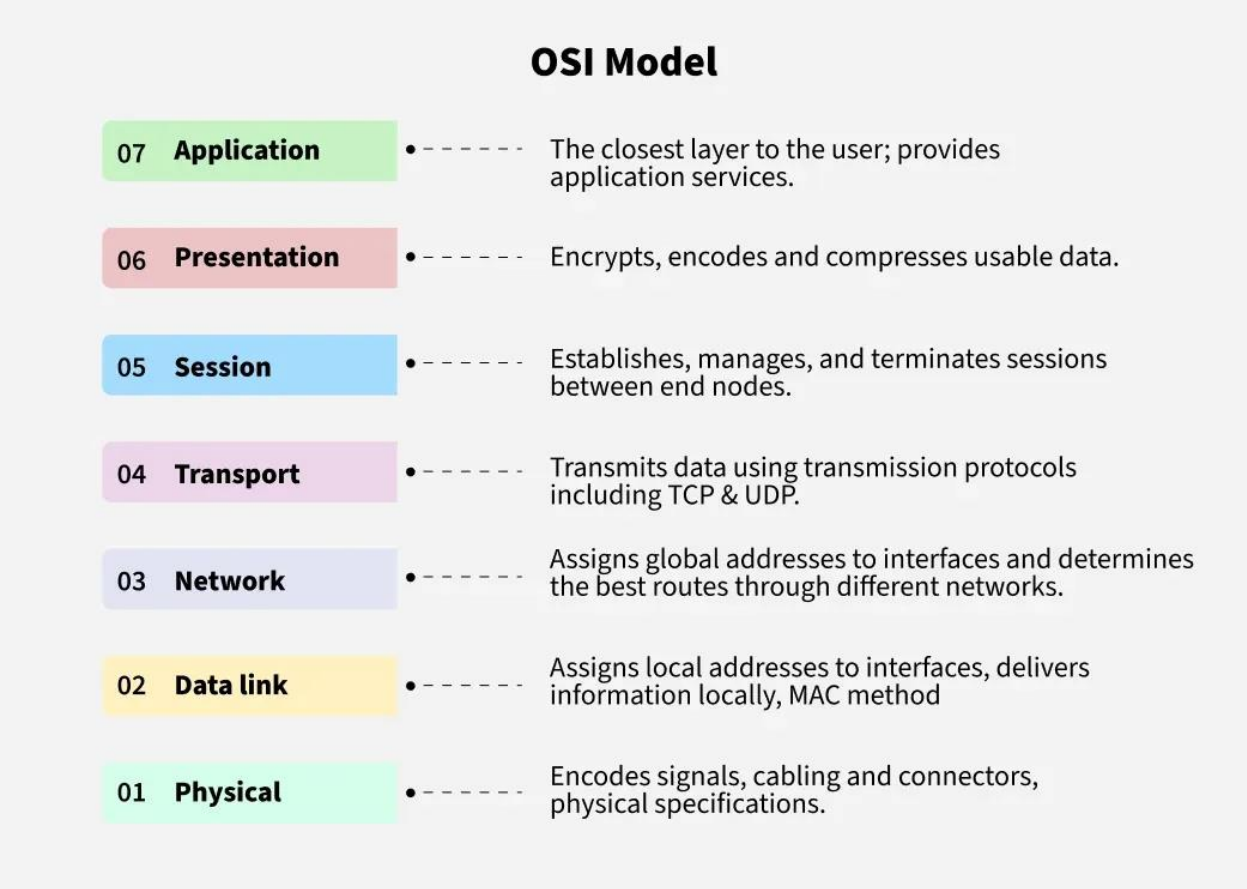
i. Client:

- **Definition:** The client is typically a web browser or application that initiates a request to access resources or services on the internet.
- **Role:**
  - Sends requests to the server (e.g., to view a webpage, submit a form, etc.).
  - Displays content received from the server (like HTML, CSS, images).
  - Often runs frontend code like JavaScript to provide interactivity.

ii. Server:

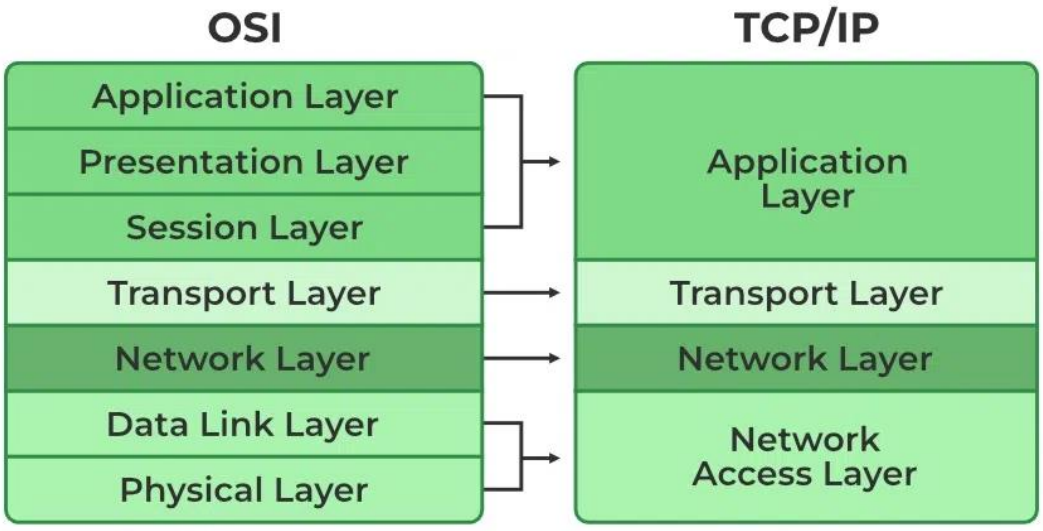
- **Definition:** The server is a powerful computer or system that listens for client requests and sends back the appropriate response.
- **Role:**
  - Processes requests (e.g., retrieving data from a database).
  - Sends responses (e.g., HTML pages, JSON data).
  - Runs backend logic, such as user authentication, data storage, etc.

10) Network Layers on Client and Server.



11) Explain the function of the TCP/IP model and its layers.

The TCP/IP model (Transmission Control Protocol/Internet Protocol) is a set of networking protocols that allow computers to communicate over the internet**.** It defines how data should be packetized, addressed, transmitted, routed, and received



Layer No.	Layer Name	Function
1	**Link Layer**	Handles communication between physical network devices (MAC addressing).
2	**Internet Layer**	Sends packets across multiple networks using IP addressing (like postal system).
3	**Transport Layer**	Ensures reliable data delivery (TCP) or fast, no-guarantee delivery (UDP).
4	**Application Layer**	Interfaces with user applications (e.g., web, email, file transfer).

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

i. Fiber Optic Internet:

- **What it is**: Uses light signals through fiber-optic cables.
- **Where used**: Urban cities, developed countries (USA, Japan, South Korea, India—metros).

Pros:

- Extremely fast speeds (up to 10+ Gbps).
- Very low latency — great for video calls, gaming.
- Highly reliable connection (not affected by weather).

Cons**:**

- Expensive infrastructure to install.
- Not available in rural or remote areas.

ii. Cable Broadband

- **What it is**: Uses coaxial cable TV lines.
- **Where used**: Widely used in North America, Europe, and parts of Asia.

Pros:

- High speeds (up to 1 Gbps).
- Good for streaming, browsing, and downloading.
- Reliable in cities.

Cons:

- Bandwidth shared with neighbors — slower at peak times.
- Slightly older tech than fiber.

iii. Mobile Internet (4G/5G)

- **What it is**: Wireless internet using cellular networks.
- **Where used**: Everywhere — mobile phones, portable hotspots.

Pros:

- Portable and wireless.

- 5G offers very high speed and low latency.
- No wiring needed.

Cons:

- Signal quality depends on location.
- Data plans may be expensive or limited.
- 5G not fully available in many rural areas.

iv. Satellite Internet

- **What it is**: Uses satellite signals to provide internet.
- **Where used**: Remote areas without fiber or mobile towers (Africa, rural India, mountain zones).

Pros:

- Works anywhere on Earth (Starlink, etc.).
- Good for disaster zones or isolated regions.

Cons:

- High latency and slower than fiber.
- Expensive equipment and monthly cost.
- Weather can affect performance.

v. Public Wi-Fi & Hotspots *(Supplemental access)*

- **What it is**: Free or paid internet in cafes, stations, libraries.
- **Where used**: Globally in cities and tourist areas.

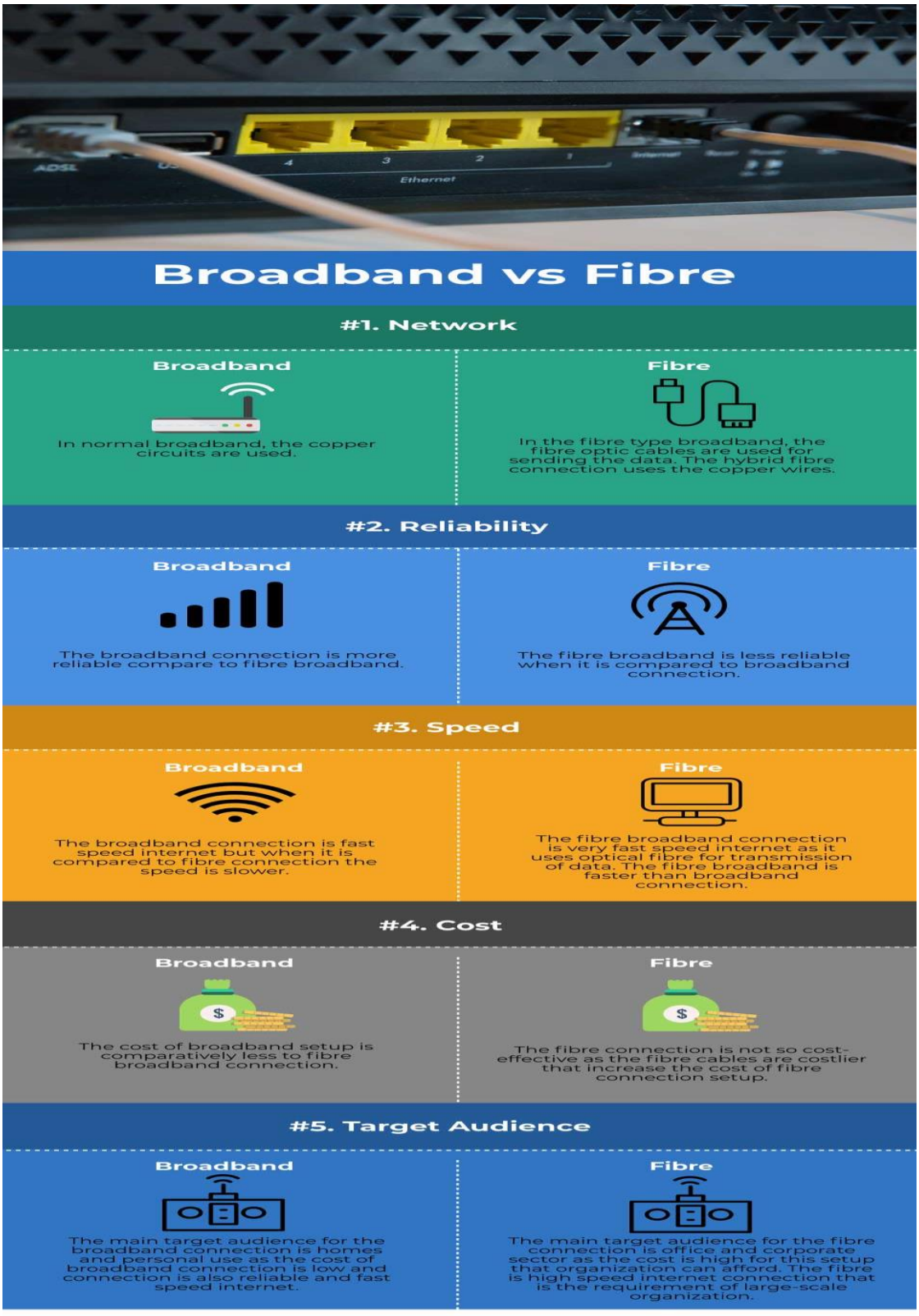
Pros:

- Free or cheap access.
- No data cost to users.

Cons:

- Not secure for personal use.
- Speed may be slow.

13) How does broadband differ from fiber-optic internet?

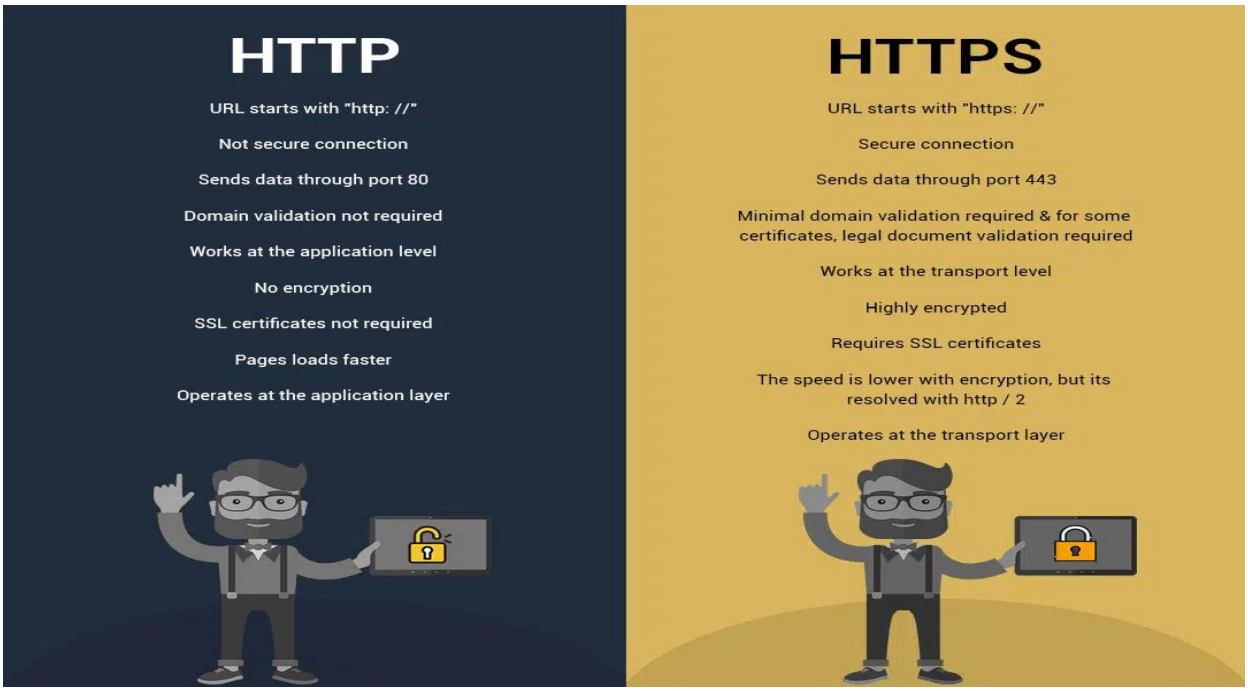


14) Protocols

Protocols are rules and standards that define how data is transmitted and received over a network.

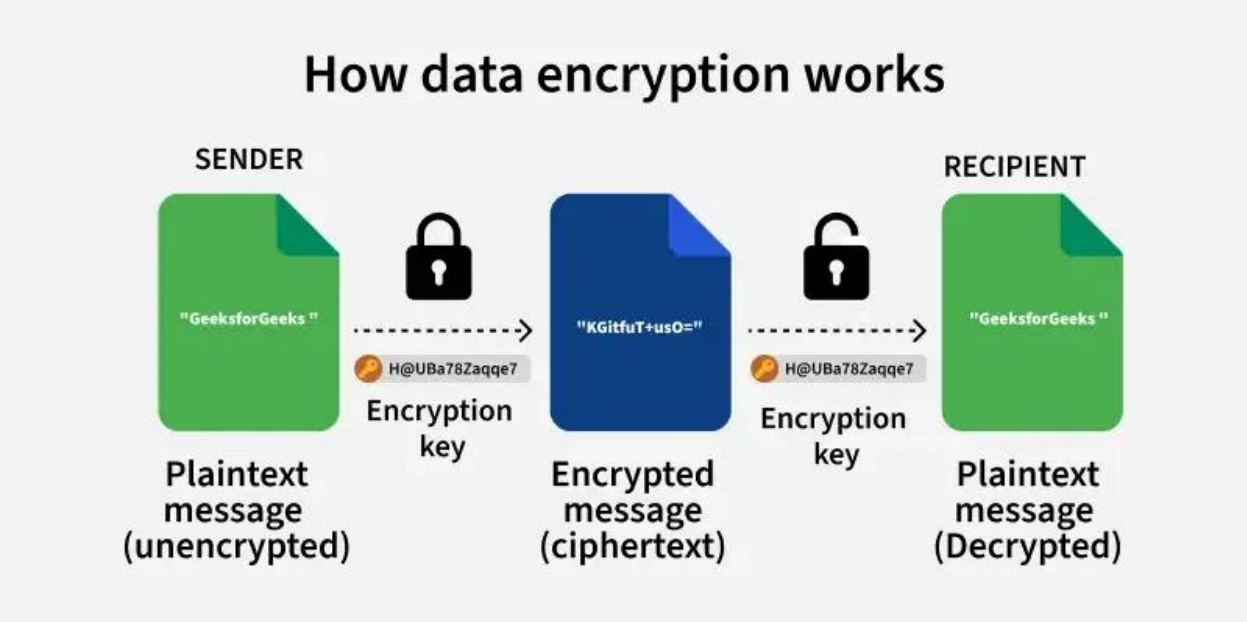
Protocol	Full Form	Purpose
HTTP	Hyper Text Transfer Protocol	Used for web browsing (loading websites).
HTTPS	HTTP Secure	Secure version of HTTP with encryption (SSL/TLS).
FTP	File Transfer Protocol	Used to upload/download files between computers.
SMTP	Simple Mail Transfer Protocol	Used to send emails.
POP3/IMAP	Post Office Protocol / Internet Message Access Protocol	Used to receive and read emails.
TCP	Transmission Control Protocol	Ensures reliable, ordered delivery of data.
UDP	User Datagram Protocol	Fast, no guarantee of delivery (used in games, streaming).
IP	Internet Protocol	Provides addressing and routing for data.
DNS	Domain Name System	Converts domain names (like google.com) into IP addresses.
DHCP	Dynamic Host Configuration Protocol	Automatically assigns IP addresses to devices.

15) What are the differences between HTTP and HTTPS protocols?



16) What is the role of encryption in securing applications.

Encryption is the process of converting data into a secret code to protect it from unauthorized access. It plays a critical role in application security by ensuring that even if data is intercepted or stolen, it cannot be read or used without the proper decryption key.



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

Application	Type	Purpose
Google Chrome	Application	Browsing the web
Microsoft Word	Application	Writing and editing documents
Windows OS	System	Managing system resources
Antivirus	System	Protecting from malware
WhatsApp	Application	Messaging and communication

18) What is the difference between system software and application software?

Feature	System Software	Application Software
Definition	Software that controls hardware and runs the system	Software that helps users perform specific tasks
Main Purpose	To manage system resources and hardware	To allow users to perform activities like writing, browsing, editing
Interaction	Less direct (runs in background)	Direct interaction with the user
Installation	Comes pre-installed with the OS	Installed by the user as needed
Examples	Windows, Linux, macOS, BIOS, Device Drivers	MS Word, Chrome, WhatsApp, VLC Media Player
Runs When?	Starts with system boot	Starts when the user opens the application
Dependence	Needed for running the computer	Depends on system software to run

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

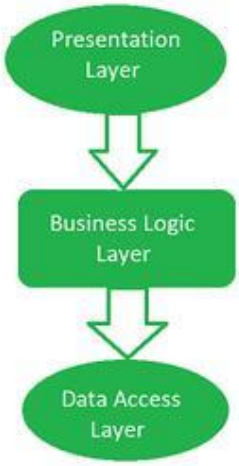
Presentation Layer :	The front-end (HTML/CSS/JS) shown to the user in a browser or mobile app.
Application Layer :	Business logic layer; processes requests, communicates with the database.
Database Layer	: Stores and retrieves data (user info, products, messages, etc.).

20) What is the significance of modularity in software architecture?

Modularity means dividing a software system into separate, independent modules or components, where each module handles a specific function.

Benefit	Explanation
1. Easier Maintenance	Each module can be updated or fixed without affecting the whole system.
2. Code Reusability	Modules can be reused in other projects or features.
3. Better Debugging	Easier to identify and fix issues because each module is isolated.
4. Team Collaboration	Teams can work on different modules at the same time without conflicts.
5. Scalability	New features can be added easily by plugging in new modules.
6. Testability	Modules can be tested individually (unit testing), improving software quality.

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



- **Presentation Layer:** The layer at which users interact with the application and the final data will be visible to the users at this interface. It acts as an interface between the user and the application.
- **Business Logic Layer:** It acts as an intermediate between the Presentation and the Data Access Layer.
- **Data Access Layer:** The layer at which the data is managed.

EX: ZOMATO OR SWIGGY

1. Presentation Layer (UI Layer)

📌 Function:

This is what the user sees and interacts with — the frontend.

📱 Features:

- Displays a list of nearby restaurants
- Shows food items, ratings, prices
- Allows user login/signup
- Accepts delivery location and payment details
- Shows order status and estimated time

💻 Technologies Used:

- HTML, CSS, JavaScript
- React.js / Angular / Flutter
- API calls to fetch data from the backend

2. Business Logic Layer (Middle Layer / Service Layer)

📌 Function:

This is the brain of the application. It processes user input, applies rules, and communicates between UI and database.

□ Responsibilities:

- Validates user login
- Checks restaurant availability
- Applies discount codes
- Calculates total amount, taxes, and delivery charges
- Manages payment transactions
- Notifies restaurant and delivery partner

🛠 Technologies Used:

- Node.js / Java / Python / .NET (backend)
- RESTful APIs
- Authentication and session management

3. Data Access Layer (DAL)

📌 Function:

Handles all database interactions — reading, writing, updating, and deleting data.

🏭 Responsibilities:

- Fetches user data (login info, address)
- Stores order details
- Retrieves menu and prices
- Updates order status (e.g., “Delivered”)
- Stores restaurant and driver ratings

🛢 Technologies Used:

- SQL/NoSQL Databases (MySQL, MongoDB, PostgreSQL)
- ORM Tools (like Sequelize, Hibernate)
- Stored procedures and queries

22) Why are layers important in software architecture?

Layers in software architecture help break down a complex application into manageable, organized parts — each with a clear responsibility.

1. Separation of Concerns

- Each layer has its own responsibility:
    - **Presentation Layer** – UI
    - **Business Logic Layer** – rules/decisions
    - **Data Access Layer** – database communication
- This reduces complexity and makes code easier to read and manage

2. Improved Maintainability

- Changes in one layer (e.g., UI redesign) won't affect others (e.g., database).
- Developers can fix bugs or upgrade components **without breaking the entire system**.

3. Better Reusability

- Business logic can be reused across multiple platforms (web, mobile, APIs).
- Data access functions can be reused for different queries or modules.

4. Security and Control

- Sensitive logic and database access are handled in secure, backend layers.
- UI doesn't directly touch the database, preventing SQL injection or data leaks.

5. Easier Testing

- Each layer can be tested independently (unit testing).
- For example, you can test the logic layer without worrying about the UI.

6. Scalability

- You can scale layers independently. For example:
    - Add more frontend servers for user traffic.
    - Use caching in business logic.
    - Optimize or scale databases separately.

7. Team Collaboration

- Frontend, backend, and database teams can work in parallel.
- Clear boundaries prevent code conflicts.

23) Explain the importance of a development environment in software production.

A **development environment** is a setup where software developers write, test, and debug their code before it's released. It includes tools, frameworks, libraries, hardware, and software systems needed for efficient software creation.

Importance:

1. Safe Space to Build and Test

- Developers can experiment, make mistakes, and test features **without affecting real users or live systems**.
- Bugs can be identified and resolved early.

2. Faster and More Efficient Development

- Equipped with **code editors**, **debuggers**, **version control**, and **automation tools**, which help in writing and managing code efficiently.
- Integrated tools reduce repetitive work and speed up development.

3. Consistency Across Teams

- Everyone in the team uses the **same versions** of programming languages, dependencies, and tools.
- This reduces compatibility issues and unexpected bugs in production.

4. Supports Collaboration

- Developers can **work together**, track changes via **Git**, and merge code through **CI/CD pipelines**.
- Shared environments help in code review and pair programming.

5. Facilitates Testing

- Unit, integration, and system tests can be run in the development environment to ensure **code quality**.
- Simulates real-world conditions using test databases and mock APIs.

6. Helps in Configuration Management

- Environments can be configured to match the production setup closely (e.g., using Docker), which minimizes surprises when deploying the code live.

24) What is the difference between source code and machine code?

Aspect	Source Code	Machine Code
Definition	Human-readable code written by programmers	Binary code (0s and 1s) understood by the computer
Written In	High-level languages (Python, C, Java, etc.)	Low-level binary (machine language)
Readability	Easy to read and understand by humans	Only readable by computers (not human-friendly)
Example	print("Hello")	10110000 01100001 (binary representation)
Needs Compilation?	Yes – must be compiled or interpreted	No – it is directly executed by the CPU
Editable?	Easily editable by developers	Hard to edit manually
Use	For writing and developing programs	For running programs on hardware

25) Why is version control important in software development?

- A **Version Control System (VCS)** is a tool that helps track and manage changes to a project's codebase over time. It allows multiple developers to work on the same project simultaneously without conflicts, maintains a history of all changes, and enables easy rollback to previous versions if needed.

IMPORTANCE:

- **Track Changes Over Time:** VCS allow developers to track every modification made to the codebase. This means you can always go back to previous versions, ensuring no changes are lost.
- **Collaboration:** VCS simplify collaboration between team members. Everyone can work on different parts of the project without worrying about overwriting each other's work.
- **Code History and Audit Trails:** With version control, you can see who made specific changes, when they were made, and why. This audit trail is invaluable for debugging, reviewing, or maintaining code.
- **Backup and Recovery:** Version control systems offer a way to back up your project. If something goes wrong, you can always recover previous versions.
- **Branching and Merging:** You can create branches for different features or bug fixes, allowing multiple developers to work simultaneously without interfering with each other's code. Once the work is done, branches can be merged back into the main project seamlessly.

26) **What are the benefits of using Github for students?**

Subject	GitHub Use
Web Dev	Host your HTML/CSS/JS projects
Python	Share scripts and mini-projects
IoT/AI/ML	Store code, models, and datasets
Group Project	Collaborate via pull requests & issues

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

Category	Examples
System Software	Windows, Linux, Drivers
Application Software	Chrome, Word, WhatsApp
Utility Software	Antivirus, CCleaner, WinRAR

28) What are the differences between open-source and proprietary software?

Feature	Open-Source Software	Proprietary Software
Source Code Access	Freely available to anyone	Kept secret by the company
Modification	Can be modified and redistributed	Cannot be legally modified or shared
Cost	Usually free	Often requires a paid license or subscription
Ownership	Owned by the community or individuals	Owned by a company or individual
Support	Community-based support (forums, contributors)	Official support from the company
Examples	Linux, VLC Media Player, Mozilla Firefox	Windows OS, Microsoft Office, Adobe Photoshop
Security	Transparent code allows public security audits	Security handled by vendor (may lack transparency)

Feature	Open-Source Software	Proprietary Software
Update Frequency	Updated by community regularly	Updated based on company schedule

29) How does GIT improve collaboration in a software development team?

Benefit	How Git Helps
Version Control	Track and manage code changes
Parallel Development	Work independently using branches
Code Quality	Review via pull requests
Collaboration	Team communication through commits & PRs
Risk Reduction	Safe testing and rollback of changes
Remote Work	Supports global collaboration

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

🔹 Types of Application Software and Their Productivity Benefits

1. Word Processing Software

- **Examples**: Microsoft Word, Google Docs, LibreOffice Writer
- **Purpose**: Creating, editing, formatting text documents
- **Productivity Impact**:
    - Speeds up documentation tasks
    - Allows collaboration and comments in real time
    - Reduces errors through spell-check and grammar tools

2. Spreadsheet Software

- **Examples**: Microsoft Excel, Google Sheets
- **Purpose**: Handling numerical data, formulas, charts, and reports
- **Productivity Impact**:
    - Automates calculations and data analysis
    - Helps in budgeting, forecasting, and financial tracking
    - Enables real-time collaboration

3. Presentation Software

- **Examples**: Microsoft PowerPoint, Google Slides, Prezi
- **Purpose**: Designing and delivering presentations

- **Productivity Impact**:
    - Helps communicate ideas clearly and visually
    - Enhances public speaking with organized slides
    - Allows remote team presentations and edits

4. Database Management Software (DBMS)

- **Examples**: Microsoft Access, MySQL, Oracle DB
- **Purpose**: Creating and managing structured data
- **Productivity Impact**:
    - Organizes large volumes of data efficiently
    - Supports faster information retrieval and reporting
    - Reduces manual recordkeeping

5. Graphics and Design Software

- **Examples**: Adobe Photoshop, Canva, CorelDRAW
- **Purpose**: Creating images, logos, banners, and multimedia content
- **Productivity Impact**:
    - Enables rapid creation of professional visuals
    - Boosts marketing and branding efforts
    - Saves time with templates and digital tools

6. Communication Software

- **Examples**: Zoom, Microsoft Teams, Slack, Gmail
- **Purpose**: Facilitating instant messaging, video calls, and emails
- **Productivity Impact**:
    - Enhances remote teamwork and coordination
    - Speeds up feedback and decision-making
    - Keeps teams connected anytime, anywhere

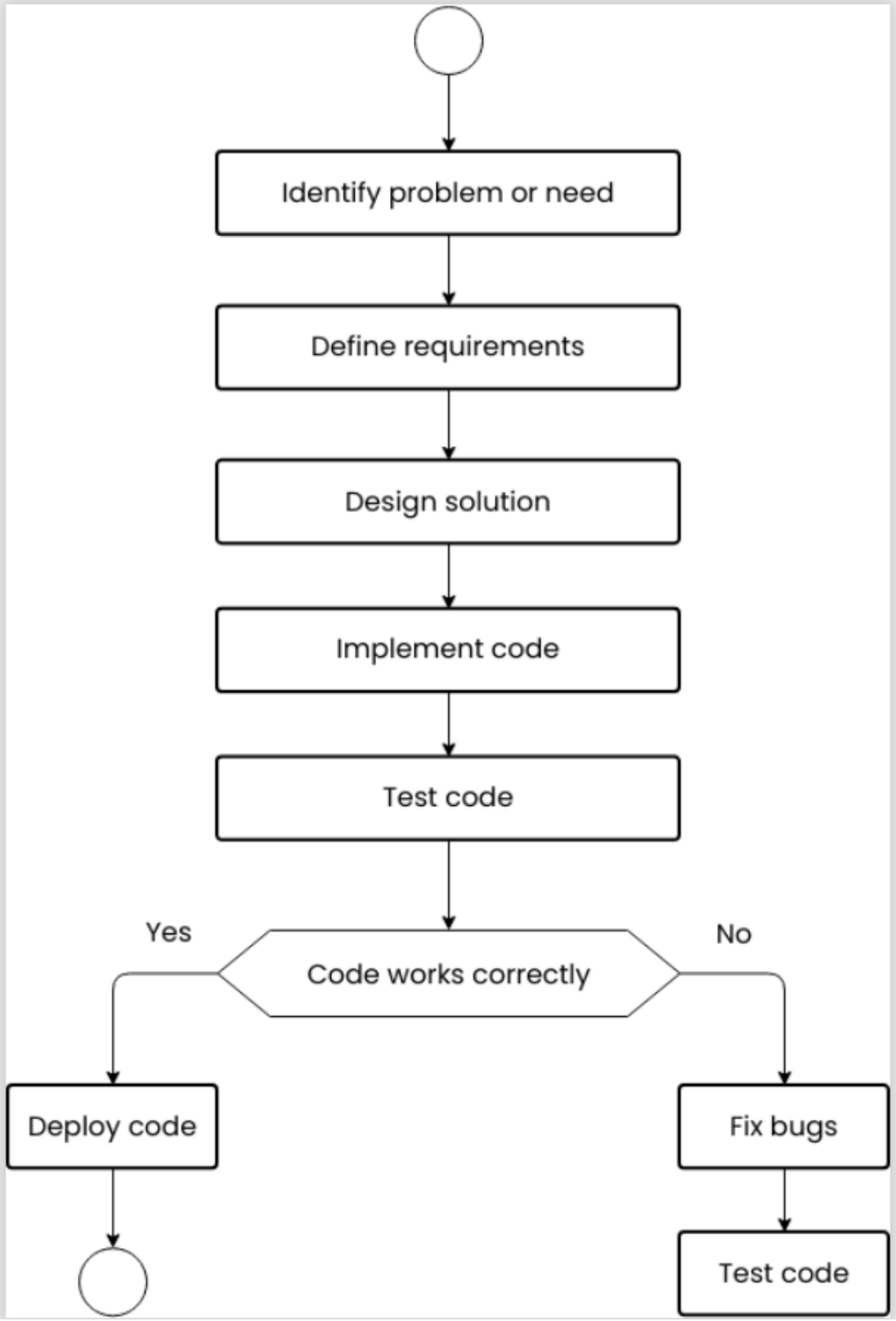
7. Project Management Software

- **Examples**: Trello, Asana, Microsoft Project
- **Purpose**: Planning, tracking, and managing projects
- **Productivity Impact**:
    - Keeps tasks organized with deadlines and priorities
    - Increases accountability and transparency
    - Allows real-time progress tracking

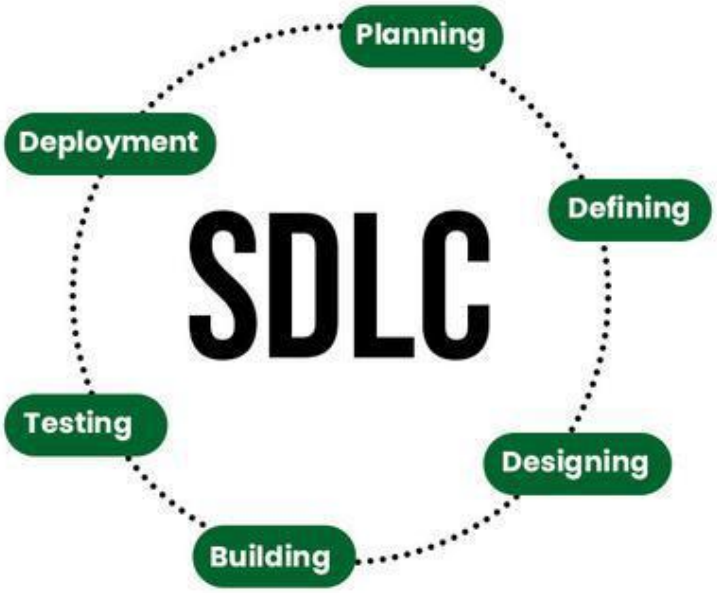
31) What is the role of application software in businesses?

Function	Software Examples	Benefit
Document Handling	MS Word, Google Docs	Faster writing & sharing
Data Management	Oracle DB, MySQL	Secure & structured data
Communication	Slack, Teams, Zoom	Real-time collaboration
Finance	Tally, QuickBooks	Accurate financial tracking
Marketing & CRM	Mailchimp, Salesforce	Improved customer engagement
Decision Making	Power BI, Tableau	Data-driven business strategies

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



33) What are the main stages of the software development process?



Stage	Purpose	Output
Requirement Analysis	Understand what the user needs	Requirement document
Planning	Set goals, timelines, and resources	Project plan
Design	Blueprint for building the software	System design docs, UI layouts
Implementation	Actual software development (coding)	Source code
Testing	Identify and fix defects	Tested, bug-free application
Deployment	Deliver software to users	Live software
Maintenance	Keep the software running smoothly	Updates, patches

34) Write a requirement specification for a simple library management system.

🔷 1. Introduction

1.1 Purpose

The purpose of this document is to outline the requirements for a simple Library Management System (LMS) that allows librarians to manage books, users, and book borrowing efficiently.

1.2 Scope

This system will enable:

- Book entry and cataloging
- Member registration
- Book issuing and return tracking
- Fine calculation for late returns
- Search functionality for books and users

🔷 2. Functional Requirements

2.1 User Management

- Add, update, and delete member profiles
- Assign unique ID to each user
- View borrowing history of each user

2.2 Book Management

- Add, update, delete book records
- Store details: Title, Author, ISBN, Category, Quantity
- Check availability status

2.3 Issue and Return

- Issue book to a user
- Record issue date and due date
- Return book and update records
- Calculate fine if the book is returned late

2.4 Search Functionality

- Search books by title, author, or ISBN
- Search users by name or user ID

2.5 Reports

- Generate reports for:
  - Books currently issued
  - Overdue books
  - Frequent borrowers

🔹 3. Non-Functional Requirements

3.1 Usability

- Simple and user-friendly interface for librarians

3.2 Performance

- Quick response time for book searches and updates

3.3 Reliability

- System should function correctly with minimal downtime

3.4 Security

- Password-protected admin access
- Role-based access control (admin, librarian)

🔹 4. System Requirements

4.1 Hardware Requirements

- A PC or server with at least 4GB RAM, 1GHz CPU, and 100GB storage

4.2 Software Requirements

- Frontend: HTML/CSS/JavaScript
- Backend: PHP/Python/Java
- Database: MySQL or SQLite
- OS: Windows/Linux

🔹 5. Assumptions and Constraints

- Only one library branch is considered
- Internet connectivity is optional (for a local version)
- Each book has a unique ISBN

🔹 6. Future Enhancements (Optional)

- Barcode scanner support
- SMS/email alerts for due dates
- Online book reservation by users

35) Why is the requirement analysis phase critical in software development?

Clear Understanding of Stakeholder Needs: Through effective requirement analysis, developers gain a detailed understanding of the needs, expectations, and goals of all stakeholders, including clients, end-users, and business leaders. This helps ensure that the final product aligns with user needs and business objectives.

Avoiding Scope Creep: Properly defined requirements help prevent "scope creep," which refers to uncontrolled changes or additions to the project's scope. By clearly documenting the requirements at the beginning, it becomes easier to manage any changes or adjustments, ensuring they are made within the project's boundaries.

Improved Communication and Collaboration: Requirement analysis encourages ongoing communication between developers, project managers, and stakeholders. This collaborative approach ensures that all parties are on the same page and can address potential misunderstandings before they become problems.

Reduced Development Costs: When requirements are clear and well-documented from the start, the development team can focus on delivering the right solution without needing frequent revisions. This helps minimize the time spent on rework and cuts down overall development costs.

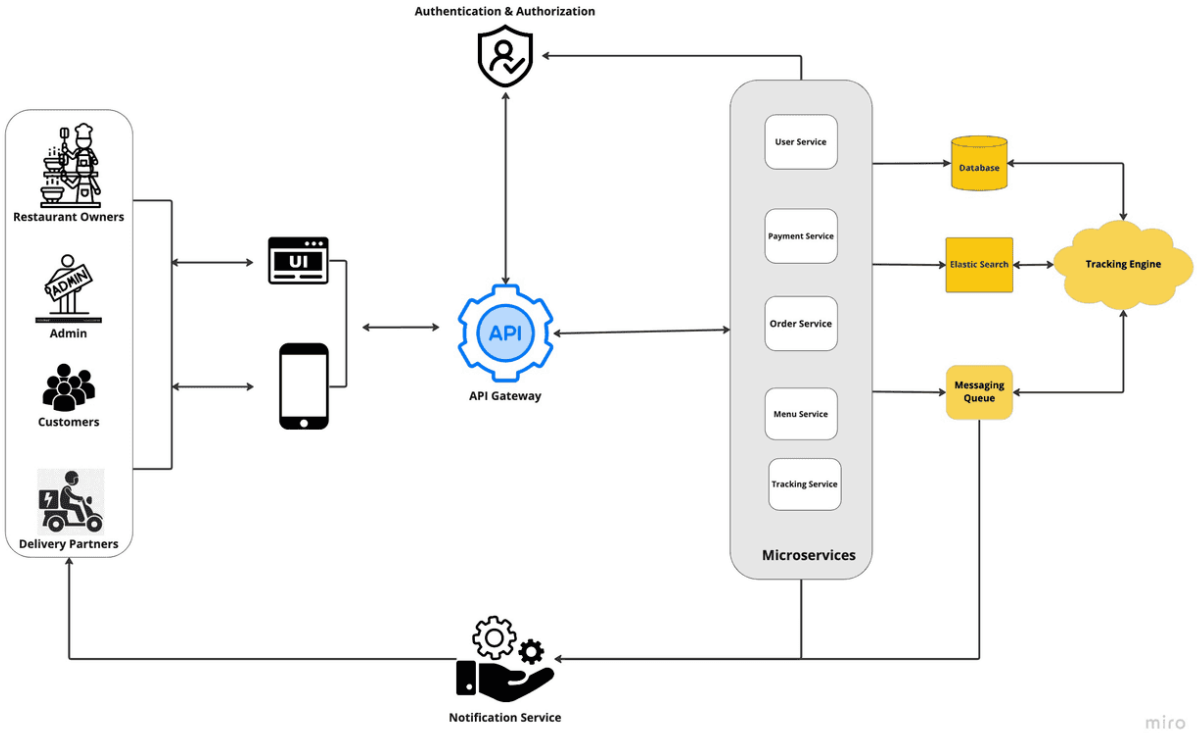
Minimized Risks and Errors: Identifying and documenting requirements early helps to reduce the risks of technical errors and misalignments. A solid requirement analysis acts as a roadmap for the development team, helping them avoid building the wrong features or delivering incomplete solutions.

Better Project Planning: Requirement analysis provides the necessary input for creating detailed project timelines, resource allocation plans, and budget estimates. With clear

requirements in hand, project managers can plan more accurately, ensuring that the project stays on track and within budget.

Improved Quality and User Satisfaction: By gathering functional and non-functional requirements carefully, the final product is more likely to meet both the technical specifications and user expectations. This leads to higher product quality and greater user satisfaction.

36) Design a basic system architecture for a food delivery app.



37) What are the key elements of system design?

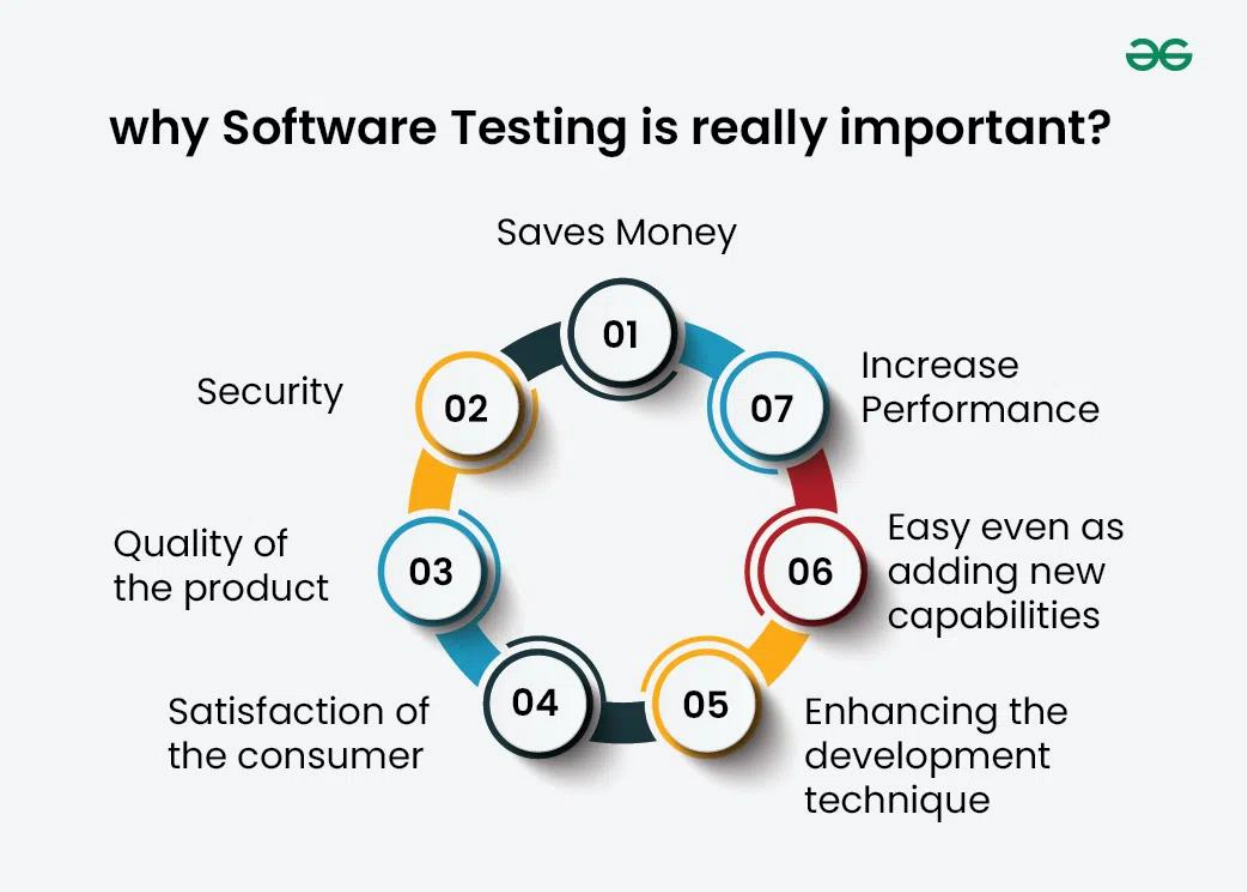
1. **Front-end web or mobile application:** This is the interface customers use to generate menus, browse menus, place orders, and track delivery status.
2. **Backend server or Services:** This component handles requests from the front end, communicates with the database, and coordinates with delivery partners.
3. **Database:** This stores information about menus, orders, customers, and delivery partners.
4. **API Gateway:** This is responsible for request routing, composition, and protocol translation, among other things, between an application and a set of micro services.
5. **Messaging Queue:** An asynchronous communication between systems that allows multiple systems to send and receive messages reliably and efficiently without needing to be constantly connected.
6. **Notification Service:** To send notifications to users, typically through email or push notifications.
7. **Tracking Engine:** This will constantly watch for changes in the DB, update the elastic search index, and notify the messaging queue.

38) Develop test cases for a simple calculator program.

No.	Test Case ID	Test Case Objective	Prerequisite	Steps	Input Data	Expected Result	Actual Result	Remarks / Status
1	TC1	To add two integers and display the result	Calculator is switched on	1. Enter valid integer2. Press +3. Enter second valid integer	135 + 100	235 (Addition. If result > 10 digits, show in exponential form)	235	Pass
2	TC2	To subtract two integers and display the result	Calculator is switched on	1. Enter valid integer2. Press -3. Enter second valid integer	135 - 100	35 (Subtraction. If result > 10 digits, show in exponential form)	35	Pass
3	TC3	To multiply two integers and display the result	Calculator is switched on	1. Enter valid integer2. Press ×3. Enter second valid integer	100 × 400	40000 (Multiplication. If result > 10 digits, show in exponential form)	40000	Pass
4	TC4	To divide two integers and display the result	Calculator is switched on	1. Enter valid integer2. Press /3. Enter second valid integer	100 / 25	4 (Division. If result > 10 digits, show in exponential form)	4	Pass

No.	Test Case ID	Test Case Objective	Prerequisite	Steps	Input Data	Expected Result	Actual Result	Remarks / Status
5	TC5	To clear the screen	Calculator is switched on	Press C	—	Screen should show symbol 0	Symbol 0	Pass
6	TC6	To delete digits one by one using backspace	Calculator is switched on	Press <- (backspace)	—	One digit should be deleted from the right-hand side	One digit deleted	Pass

39) Why is software testing important?



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

Real-World Case: Slack's Outage – January 2021

📌 Background:

Slack is a widely used communication and collaboration software used by businesses and organizations worldwide. On **January 4, 2021**, Slack experienced a **major service outage** that affected millions of users globally for several hours.

🔎 Issue Summary:

- Users could not send messages, join calls, or access channels.
- The outage occurred during the first working day of the new year, causing large-scale disruption in workplaces.
- Slack's status page reported degraded performance in key services like messaging and notifications.

⚙ Type of Maintenance Involved:

Corrective and Emergency Maintenance

1. Corrective Maintenance

- Identified a **network configuration issue** as the root cause.
- Rolled back recent changes that had been deployed during system scaling.

2. Emergency Maintenance

- Bypassed or reset key services to restore uptime.
- Applied real-time fixes to bring back partial and then full functionality.

💡 Actions Taken:

- Engineers isolated the issue related to network traffic routing.
- Reconfigured backend load balancers.
- Restarted message queues and notification services.
- Communicated real-time updates through the Slack status page and Twitter.

✓ Outcome:

- Partial services restored within a few hours.

- Full functionality resumed after ~4 hours of downtime.
- Slack conducted a full **postmortem report** and introduced more **failover systems** to prevent recurrence.

📈 Lessons Learned:

- The incident highlighted the importance of:
  - Robust load testing and capacity planning
  - Clear rollback strategies for deployed changes
  - Real-time monitoring and transparent user communication

📄 Summary Table:

Aspect	Details
Application	Slack
Incident Date	January 4, 2021
Maintenance Type	Corrective + Emergency
Cause	Network configuration issues after scaling infrastructure
Impact	Global service outage (~4 hours)
Resolution	Rolling back configs, restarting services
Lesson	Importance of robust testing and rapid incident response

41) What types of software maintenance are there?

Type	Goal	Example
Corrective	Fix errors/bugs	Repairing a broken login function
Adaptive	Keep software compatible	Updating app for Android 14 support
Perfective	Improve performance or features	Making UI faster or more user-friendly
Preventive	Prevent future issues	Refactoring code to remove technical debt

42) What are the key differences between web and desktop applications?

Feature	Web Application	Desktop Application
Installation	No installation required (runs in browser)	Needs to be installed on each device
Access	Accessible from any device with internet	Only accessible on the installed device
Updates	Centralized updates on the server	Manual updates needed on each device
Internet Connection	Usually requires internet	Often works offline
Platform Dependency	Platform-independent (runs on any OS with a browser)	Platform-dependent (Windows, macOS, etc.)
Performance	Slightly slower (browser-dependent)	Generally faster (runs directly on OS)
Security	Depends on web server and internet security	Depends on local system security
Development Complexity	Easier to deploy; needs responsive design	Needs handling OS compatibility, installation, etc.
Example	Gmail, Google Docs, Facebook	Microsoft Word, Photoshop, VLC Media Player

43) What are the advantages of using web applications over desktop applications?

Advantage	Explanation
🌐 **Accessibility Anywhere**	Can be accessed from any device with a browser and internet connection.
🔄 **No Installation Needed**	No need to install software — just open the URL and start using.
🚀 **Automatic Updates**	Updates happen on the server — users always see the latest version.
💻 **Cross-Platform Compatibility**	Works on all operating systems (Windows, macOS, Linux, etc.) via browsers.
💾 **Saves Storage Space**	Runs in the browser — doesn't take up disk space like desktop apps.
👥 **Easier Collaboration**	Real-time data sharing and updates (e.g., Google Docs, Trello).
🔐 **Centralized Data & Backup**	Data is stored securely on the cloud, with backup and recovery options.
Easier Maintenance	Only the server needs updating — no need to update each user's machine.
📈 **Scalability**	Easier to scale with increasing users — no local hardware dependency.

Ex:

 Google Docs (Web App) allows multiple users to edit a document in real-time from anywhere.

 Microsoft Word (Desktop App) requires installation, and files are usually edited offline.

44) What role does UI/UX design play in application development?

UI (User Interface) Design – Role:

UI focuses on how the **app looks**.

Role	Description
🎨 **Visual Design**	Ensures the interface is aesthetically pleasing (colors, typography, layout).
🧭 **Navigation**	Makes it easy for users to move between different sections.
💡 **Consistency**	Keeps design elements uniform, which reduces confusion.
🎯 **Responsiveness**	Ensures the app works well across various screen sizes and devices.

UX (User Experience) Design – Role:

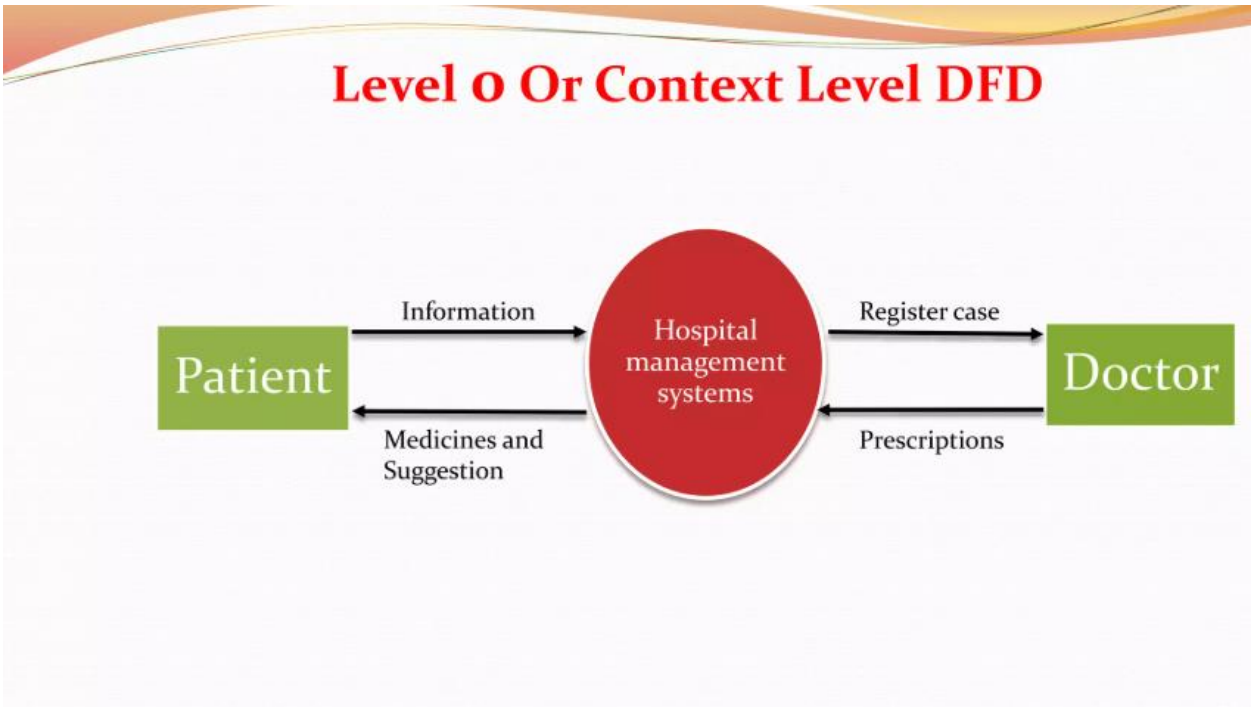
UX focuses on how the **app works** and how users **feel** when using it.

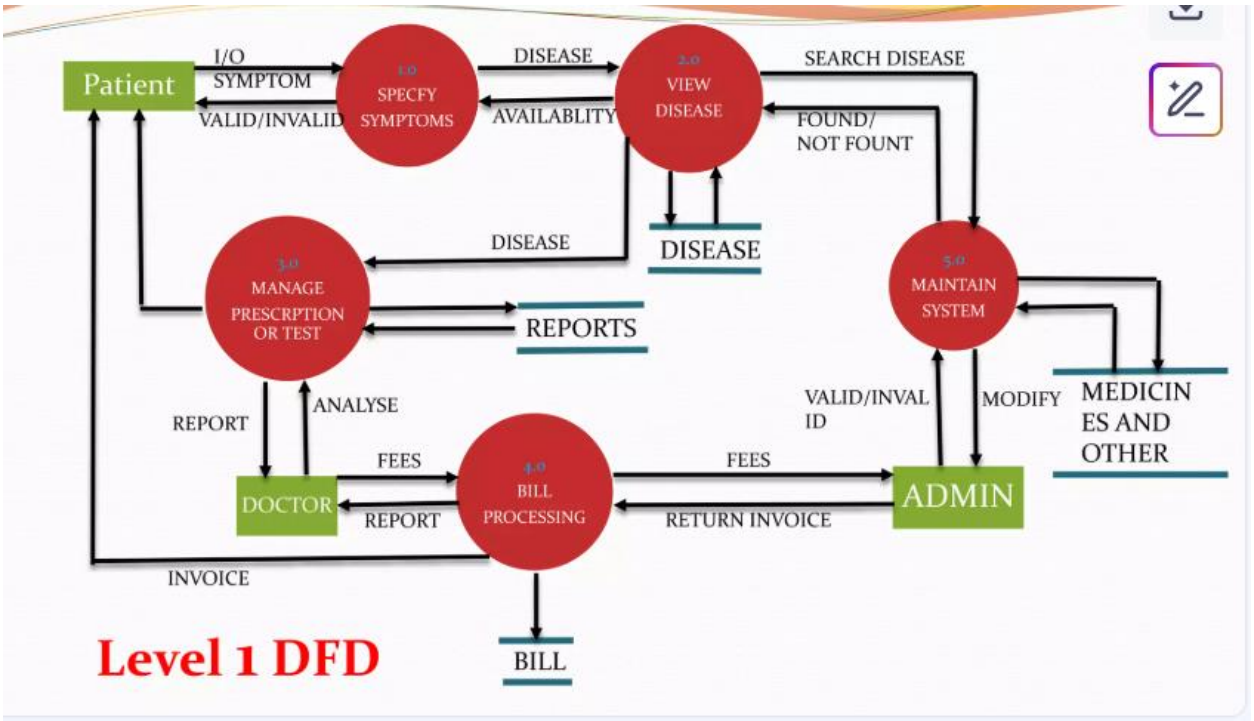
Role	Description
🗺️ **User Flow**	Designs intuitive paths for completing tasks (e.g., signing up, buying a product).
📊 **Usability**	Improves ease of use, efficiency, and learnability.
👩‍🔬 **User Research**	Understands user needs, pain points, and expectations.
🔄 **Feedback Loops**	Builds ways to gather and respond to user feedback.

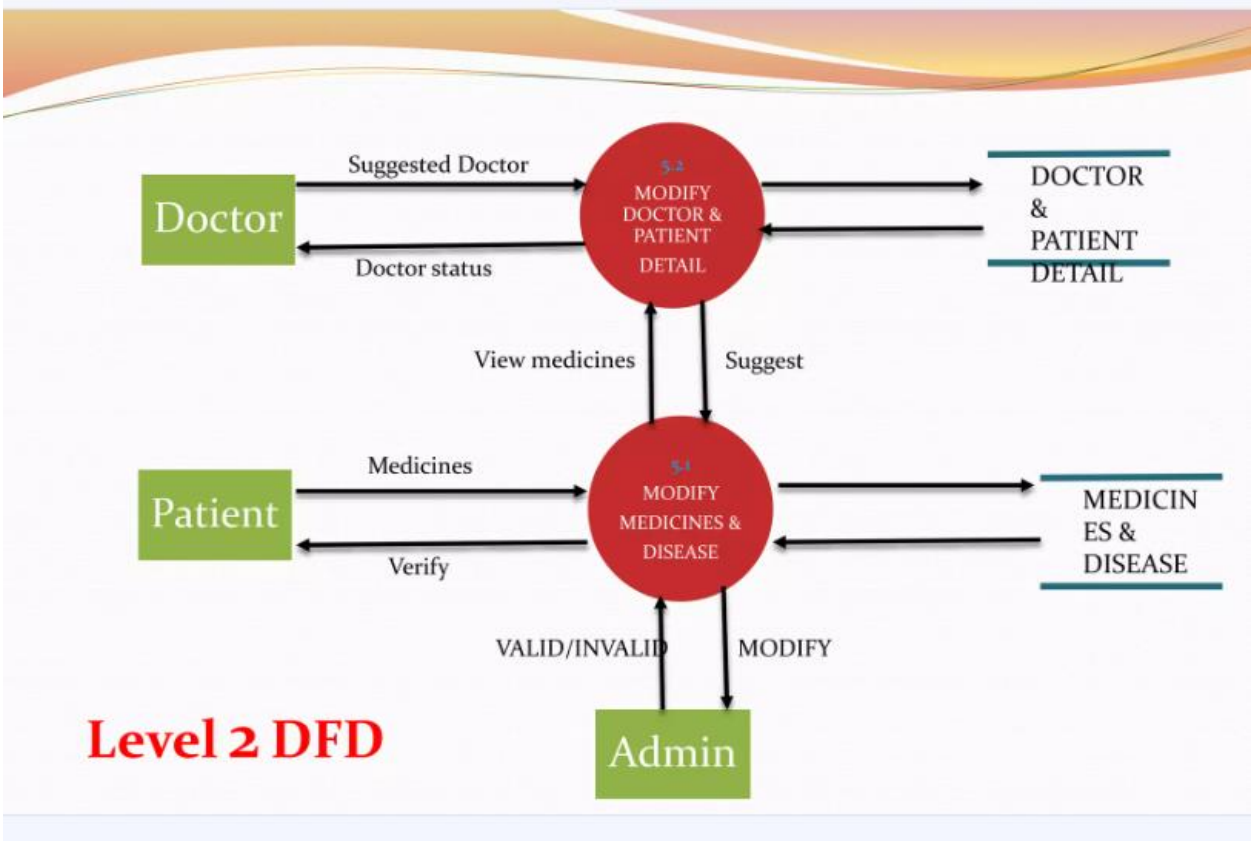
45) What are the differences between native and hybrid mobile apps?

Feature	Native Apps	Hybrid Apps
Definition	Built specifically for one platform (Android or iOS)	Built using web technologies and wrapped in a native shell
Languages Used	Java/Kotlin (Android), Swift/Obj-C (iOS)	HTML, CSS, JavaScript with frameworks like Ionic, React Native
Performance	High performance, faster, smoother animations	Moderate performance, may lag with complex features
Platform Dependency	Platform-specific (needs separate codebases)	One codebase runs on multiple platforms
Access to Device Features	Full access to device APIs (camera, GPS, sensors)	Limited or uses plugins to access device features
Development Cost	Higher (separate teams for each platform)	Lower (single team, shared codebase)
User Experience (UX)	Consistent with OS design; smoother native feel	May feel less native or consistent across platforms
Update & Maintenance	Requires updating each platform's app separately	Easier — update shared codebase once
Examples	WhatsApp, Instagram, Google Maps	Instagram (partially hybrid), Uber (admin panel), Twitter (earlier)

46) Create a DFD for a hospital management system.







47) What is the significance of DFDs in system analysis?

Aspect	Significance
Visual Clarity	DFDs provide a clear and simple graphical view of the system's processes, data stores, and data flows.
Understanding Workflow	Helps analysts and developers understand how data is processed, stored, and moved within a system.
Improves Communication	Bridges the gap between technical and non-technical stakeholders by using intuitive symbols.
Aids System Design	Used as a foundation to design databases, modules, and data processing logic.
Requirement Validation	Ensures that the software will meet user needs by modeling functional requirements.
Identifies Redundancies	Helps spot unnecessary processes or repeated data handling.
Data Security & Control	Reveals sensitive data points and helps plan for secure handling of information.

48) What are the pros and cons of desktop applications compared to web applications?

🖥️ Desktop Applications

✅ **Pros:**

Feature	Explanation
Offline Access	Can work without internet.
Performance	Faster and more responsive due to direct access to system resources.
Rich Features	More access to hardware (printers, storage, graphics, etc.).
Security	Data can be stored locally; reduces exposure to online threats.

❌ **Cons:**

Feature	Explanation
Installation Needed	Must be downloaded and installed on each device.
Platform Dependent	Often OS-specific (e.g., Windows-only).
Update Management	Users must manually install updates unless auto-updater is built in.
Limited Mobility	Data and use are tied to one machine unless synced.

🌐 Web Applications

✅ **Pros:**

Feature	Explanation
Accessible Anywhere	Use from any device with a browser and internet connection.
No Installation	Runs in browser — no setup required.
Easy Updates	Updates are applied on the server — users always get the latest version.
Cross-Platform	Works on Windows, macOS, Linux, mobile, etc.

❌ **Cons:**

Feature	Explanation
Needs Internet	Usually can't work without an internet connection.
Slower Performance	May lag compared to desktop apps, especially with complex tasks.
Security Risks	Exposed to online threats, data stored on the cloud.
Limited Device Access	Cannot access system hardware easily (like USB or camera in some cases).

49) How do flowcharts help in programming and system design?

🔹 In Programming:

1. **Visualizing Logic**
   - Flowcharts show the step-by-step flow of logic in a program.
   - They make it easier to understand conditions, loops, and sequences before writing actual code.
2. **Debugging and Maintenance**
   - Easier to trace logic errors or inefficiencies by looking at the flowchart.
   - Makes maintaining or modifying code simpler when the logic is clearly mapped out.
3. **Communication**
   - Flowcharts help programmers explain code logic to non-technical stakeholders or new team members.
4. **Efficient Planning**
   - Prevents coding errors by allowing pre-coding analysis.
   - Helps ensure all cases (e.g., input conditions) are considered before implementation.

🔹 In System Design:

1. **Modeling System Workflows**
   - Flowcharts describe how data moves through the system, including inputs, processes, and outputs.
   - They highlight dependencies and interactions between components or subsystems.
2. **Identifying Bottlenecks and Redundancies**
   - Visual inspection helps locate inefficiencies or unnecessary steps in a process.
3. **Design Validation**
   - Makes it easier to verify if the system meets its functional requirements before development.
4. **Collaboration and Documentation**
   - Teams (including developers, analysts, and stakeholders) can collaborate better using a shared visual language.
   - Serves as documentation for future reference or audits.