

## **MODULE: 1**

### **SE – Overview Of IT Industry**

#### **1. What is a Program?**

- A **computer program** is nothing but a set of instructions that are used to execute particular tasks to get particular results. It is required for programmers to learn basic concepts of mathematics to write programs. For different types of tasks, we have to write different programs. The computer program is generated by programmers or software developers. The code is then processed and executed to provide the output of the program.

#### **2. Explain in your own words what a program is and how it functions.**

- A **program** is a set of instructions written in a specific programming language that tells a computer what to do. These instructions guide the computer to perform tasks like calculations, data processing, or interacting with users. For example, a web browser is a program that allows you to access websites, while a calculator app performs arithmetic operations.
- ~ These instructions are written in a programming language, which can be understood by both humans and computers, though the computer ultimately processes them as binary code (0s and 1s).

#### **3. What is Programming?**

- **Programming** is the process of writing instructions, or code, that a computer can follow to complete tasks or solve problems. It's a collaboration between humans and computers, where humans create instructions in a language that computers can understand. Programmers use programming languages (like Python, Java, or C++) to write the instructions in a way that a computer can understand.

#### **4. What are the key steps involved in the programming process?**

- Key Steps in the Programming Process:
  - ~ Problem Analysis: Understand the problem and requirements.
  - ~ Design: Plan the solution using algorithms, flowcharts, or pseudocode.
  - ~ Coding: Write the program in a chosen programming language.
  - ~ Testing: Test the program for correctness with various inputs.
  - ~ Debugging: Identify and fix errors in the code.
  - ~ Documentation: Write explanations for the code and usage.
  - ~ Deployment: Deliver or implement the program.
  - ~ Maintenance: Update and improve the program as needed.

#### **5. Types of Programming Languages:- What are the main differences between high-level and low-level programming languages?**

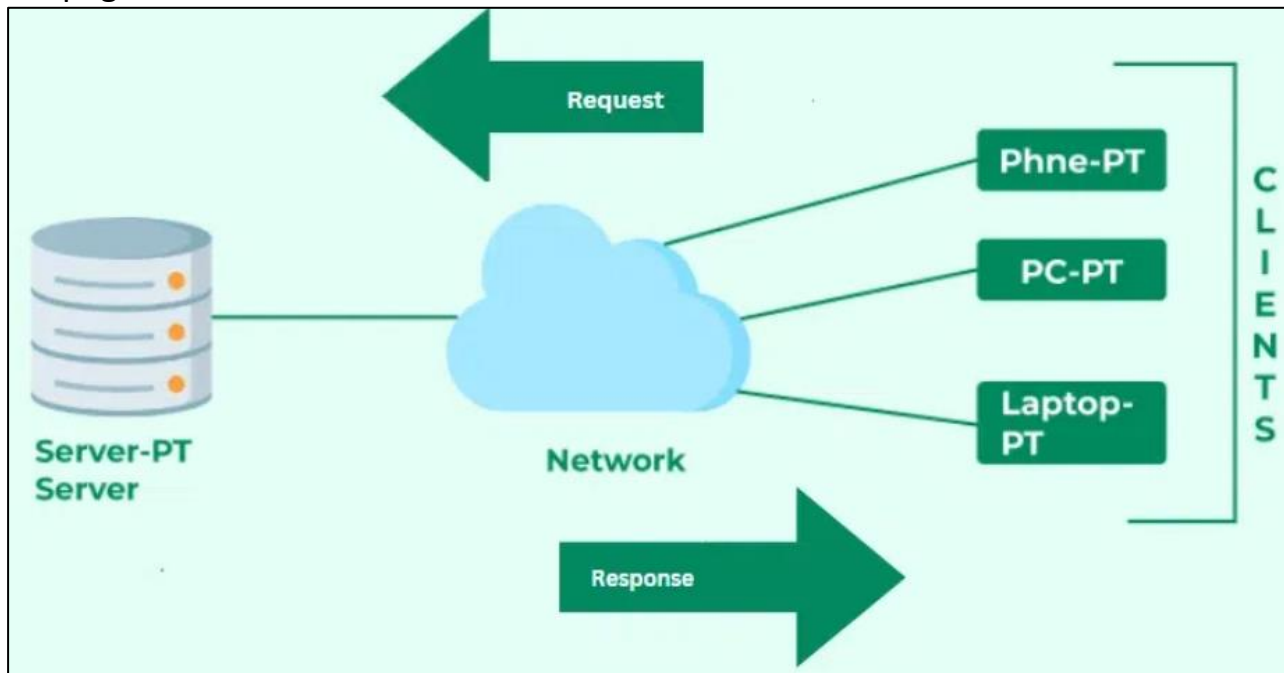
- Both **High level language** and **low level language** are the programming language's types. The main difference between high level language and low level language is that, Programmers can easily understand or interpret or compile the high level language in comparison of machine. On the other hand, Machine can easily understand the low level language in comparison of human beings. Examples of high level languages are C, C++, Java, Python, etc.

## 6. World Wide Web & How Internet Works

- The **World Wide Web (WWW)** is a collection of resources and web pages that are accessed through the internet. The internet is a global network of interconnected devices and computers that allows the WWW to function. The WWW operates on top of the internet using protocols like HTTP and HTTPS.
- The **Internet** is a global network of interconnected computers that communicate using standardized protocols. Data travels in packets via routers and switches. Key steps include:
  - ~ Request: A user sends a request (e.g., accessing a website) via a device.
  - ~ Routing: The request travels through networks to a server.
  - ~ Response: The server processes the request and sends data back.
  - ~ Display: The user's browser displays the received content.

## 7. Describe the roles of the client and server in web communication.

- **Client:** The client is a device (like a computer, smartphone, or browser) that initiates a request to access data or services. It sends HTTP/HTTPS requests to servers, asking for web pages, files, or other resources. Example: A browser requesting a website.
- **Servers:** The server is a computer or system that hosts resources and responds to client requests. It processes the request, retrieves the required data, and sends it back to the client in the form of an HTTP/HTTPS response. Example: A web server delivering a webpage.



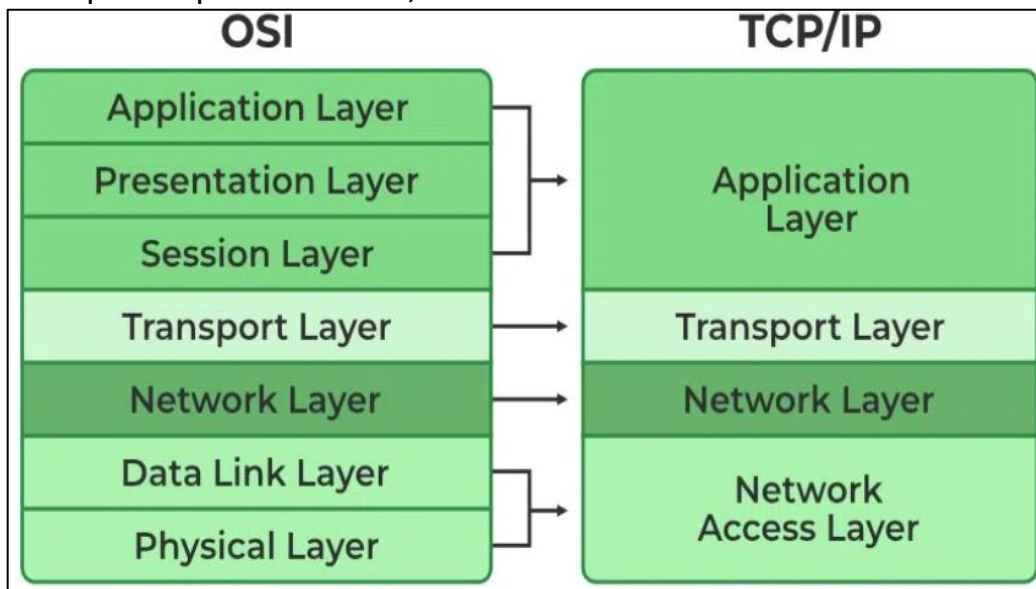
## 8. Network Layers on Client and Server:- Explain the function of the TCP/IP model and its layers.

- The **TCP/IP model** is a fundamental framework for computer networking. It stands for **Transmission Control Protocol/Internet Protocol**, which are the core protocols of the Internet. This model defines how data is transmitted over networks, ensuring reliable communication between devices. It consists of four layers: the Link Layer, the Internet Layer, the Transport Layer, and the Application Layer. Each layer has specific functions that help

manage different aspects of network communication, making it essential for understanding and working with modern networks.

➤ **Layers of the TCP/IP Model:**

- Application Layer:- Interfaces directly with user applications and manages data communication. Protocols:
  - ~ HTTP/HTTPS: Communication between web browsers and servers.
  - ~ SSH: Securely connects to remote devices.
  - ~ NTP: Synchronizes device clocks.
- Transport Layer:- Ensures end-to-end communication and reliable delivery. Protocols:
  - ~ TCP: Reliable, connection-oriented data transfer.
  - ~ UDP: Unreliable, faster, connectionless transfer for smaller data packets.
- Internet or Network Layer:- Handles logical transmission of data across networks. Routes packets using unique IP addresses. Protocols:
  - ~ IP: Routes packets (IPv4 and IPv6 versions).
  - ~ ICMP: Reports network issues.
  - ~ ARP: Resolves IP addresses to hardware (MAC) addresses.
- Network Access Layer:- Manages communication between devices at the physical and data link levels. Identifies the protocol (e.g., TCP/IP) and handles error prevention and framing.
  - ~ Protocols: HTTP, FTP, SMTP.
  - ~ Examples of protocols: PPP, Ethernet IEEE 802.2.



**9. Client and Servers:- Explain Client Server Communication**

- Client-server communication is a model where a client (requester) interacts with a server (provider) to exchange data and perform tasks over a network.
- **Client-Server Communication Explained**
    - ~ Client Sends Request: The client sends a request to the server using a communication protocol (like HTTP or HTTPS). For example, when you type a URL in your browser, it sends a request to the server hosting that webpage.

- ~ Server Processes Request: The server receives the client's request, processes it, and retrieves or generates the requested data. For instance, the server might fetch a webpage, retrieve a database record, or perform a calculation.
- ~ Server Sends Response: After processing, the server sends a response back to the client. This could be a webpage (HTML), an image, a video, or other data, depending on the request.
- ~ Client Uses Response: The client receives the response and displays or processes the data for the user. For example, a browser renders a webpage, or an app displays the requested information.

#### 10. Protocols:- What are the differences between HTTP and HTTPS protocols?

| HTTP   | HTTPS  |
|--|--|
| HTTP stands for Hyper Text Transfer Protocol.  | HTTPS for Hyper Text Transfer Protocol Secure.   |
| HTTP, URL begins with "http://".   | HTTPS, URL starts with "https://".   |
| HTTP uses port number 80 for communication.  | HTTPS uses 443 port number for communication.  |
| Data is transmitted in plain text, making it vulnerable to interception and attacks. | Encrypts data using SSL/TLS, ensuring secure communication between the client and server.                  |
| In HTTP Data is transfer in plaintext.   | In HTTPS Data transfer in ciphertext.  |
| HTTP Does not require SSL/TLS or Certificates  | HTTPS Requires SSL/TLS implementation with Certificates.   |
| HTTP works at Application Layer.   | HTTPS works at Transport Layer.  |
| HTTP faster than HTTPS   | HTTPS slower than HTTP   |
| HTTP does not use data hashtags to secure data.                                      | While HTTPS will have the data before sending it and return it to its original state on the receiver side. |
| Considered less trustworthy and may negatively affect SEO rankings.                  | Builds trust with users and improves SEO rankings.   |
| Suitable for non-sensitive information.  | Essential for sensitive transactions, such as online banking and e-commerce.                               |

#### 11. Types of Internet Connections:- How does broadband differ from fiber-optic internet?

| BROADBAND   | FIBER-OPTIC  |
|---|--|
| Broadband uses copper circuits for data transmission.         | Fibre uses fibre optic cables, and hybrid fibre combines both copper wires and fibre cables. |
| Broadband is less reliable than fibre.                        | Fibre offers better reliability with fewer disruptions.                                      |
| Broadband offers fast speeds but is slower compared to fibre. | Fibre delivers much faster speeds due to the use of optical fibre for data transmission.     |

Broadband is cheaper to set up.

Fibre is more expensive due to the higher cost of fibre cables.

## 12. Application Security:- What is the role of encryption in securing applications?

- **Encryption** is used to protect data from being stolen, changed, or compromised and works by scrambling data into a secret code that can only be unlocked with a unique digital key.
- **Role in securing applications by:**
  - ~ Data Protection: Converts sensitive data into unreadable formats to prevent unauthorized access.
  - ~ Confidentiality: Ensures only intended recipients can decrypt and understand the information.
  - ~ Integrity: Prevents data tampering by verifying its authenticity during transmission or storage.
  - ~ Authentication: Confirms the identities of users and systems exchanging encrypted data.
  - ~ Compliance: Helps meet regulatory standards for data security in various industries.

## 13. What is software? Explain types of software.

- **Software** is a computer program that provides a set of instructions to execute a user's commands and tell the computer what to do. For example like MS-Word etc.
- **Types of Software:-** It is a collection of data that is given to the computer to complete a particular task.
  - ~ System Software:- system software basically controls a computer's internal functioning and also controls hardware devices such as monitors, printers, and storage devices, etc. Types of System Software Operating System, Language Processor, Device Driver
  - ~ Application Software:- It is designed to perform a specific task for end-users. It is a product or a program that is designed only to fulfill end-users' requirements. It includes word processors, database management, etc. Types of Application Software General Purpose Software, Customize Software, Utility Software

## 14. What is software engineering?

- **Software engineering** is a discipline that involves the application of engineering principles to the design, development, testing, deployment, and maintenance of software systems. It aims to produce high-quality software in a systematic, efficient, and scalable manner by following structured methodologies and best practices. Software engineers use a variety of tools, techniques, and frameworks to manage complex software development projects.

## 15. Software Applications and Its Types:- What is the difference between system software and application software?

| SYSTEM SOFTWARE                 | APPLICATION SOFTWARE   |
|---------------------------------|--|
| Written in low-level languages. | Designed for specific tasks or purposes.<br>Written in high-level languages. |
| General-purpose software.       | Specific-purpose software.   |

|  |  |
|--|--|
| Essential for system operation; the system stops without it.                 | The system can operate without it.                         |
| Runs continuously from system startup to shut down.                          | Runs as per user requests.                                 |
| Example: Operating systems (e.g., Windows, Linux).                           | Example: Photoshop, VLC Player, MS Word.                   |
| More complex programming compared to application software.                   | Programming is simpler compared to system software.        |
| Operates in the background, integrating hardware and software.               | Runs in the front end, interacting directly with the user. |
| No direct user interaction; acts as an interface between hardware and users. | Acts as an intermediary between the user and the system.   |
| Runs independently.  | Dependent on system software for functioning.              |

#### **16. Software Architecture:- What is the significance of modularity in software architecture?**

- Enhanced Maintainability: The fact that modularity enhances software maintenance is one of its key benefits. By separating different functions in a software system into different units - modules - it becomes possible to change or correct a flaw in a module without affecting the other parts of the system. One can work on one module without affecting the others, and debugging and maintenance become simplified and easy.
- ~ Reusability: Modular items can be applied in various areas or parts of a single software undertaking and different software projects. Besides, this reuse saves time and effort while fostering uniformity and reducing the probability of errors. When using a well-tested module, its reliability is taken along.
- ~ Scalability: This gives a quick change or growth of a software system. Additional modules could be added to enable more features or expand on existing features and functions. Scalability is important for software systems as they must be flexible enough to adapt to changing requirements.
- ~ Collaboration: Multiple developers often develop different modules in big software development projects. Modularity enables parallel development in that teams can focus on specific modules without interference by other modules. This teamwork can greatly increase productivity.
- ~ Testing and quality assurance: Modular components allow an issue to be identified and rectified because it can be isolated. # The thorough testing of each module enhances the general quality and reliability of the software.
- ~ Debugging and troubleshooting: For example, it is easy to identify the source of a problem that emerges within a modular system. Identifying the problem's culprit module is helpful for the developers in dealing with the problem in this sense.
- ~ Flexibility: In addition, modularity promotes the flexibility of software systems. One may replace an obsolete, incompatible item with another device that works separately without breaking down the whole system.

### **17.Layers in Software Architecture:- Why are layers important in software architecture?**

- Layers are important in software architecture because they organize the system into distinct, logical parts, each responsible for specific tasks. This layered approach brings several key benefits:
- ~ Separation of Concerns:- Each layer handles a specific aspect of the application (e.g., presentation, business logic, data access), making the system easier to understand and manage.
- ~ Maintainability:- Changes in one layer (e.g., updating the user interface) can be made without affecting other layers, simplifying updates and bug fixes.
- ~ Reusability:- Layers like the data access layer or business logic layer can be reused across different applications or projects.
- ~ Scalability:- Individual layers can be optimized or scaled independently to handle increased demand, improving overall system performance.
- ~ Flexibility:- Layers allow for easy swapping or upgrading of technologies. For example, you can replace the database in the data layer without altering the business or presentation layers.
- ~ Testability:- Each layer can be tested independently, improving the reliability and quality of the application.
- ~ Security:- Sensitive operations can be isolated in specific layers, enforcing better control and minimizing vulnerabilities.
- ~ Interoperability:- Well-defined interfaces between layers ensure smooth communication and integration, even if the layers are built using different technologies.

### **18.Software Environments:- Explain the importance of a development environment in software production.**

- A development environment is essential in software production because it provides the tools, resources, and configurations needed to streamline the software development process. Its importance lies in:
- ~ Efficiency:- Combines all necessary tools like code editors, compilers, and debuggers in one place, saving time and effort.
- ~ Collaboration:- Facilitates teamwork through version control systems (e.g., Git) and shared platforms, allowing multiple developers to work on the same project simultaneously.
- ~ Quality Assurance:- Enables testing and debugging during development, ensuring software features and functionality are reliable before launch.
- ~ Automation:- Speeds up workflows with Continuous Integration/Continuous Deployment (CI/CD) pipelines and automated testing.
- ~ Error Reduction:- Helps identify and resolve bugs early in the process, reducing costly errors in later stages.
- ~ Project Management:- Supports task tracking, code reviews, and deployment processes, improving overall project organization and outcomes.
- ~ Replication of Real-World Scenarios:- Provides environments (e.g., staging, production) that mimic real-world conditions for testing and deployment readiness.

### 19.Source Code:- What is the difference between source code and machine code?

| SOURCE CODE  |   | MACHINE CODE   |
|--|---|--|
| Written by humans/programmers in high-level languages.         |   | Generated by compilers/translators, in binary or low-level form. |
| High-level or assembly language.                               |   | Low-level, machine-readable language.                            |
| Human-readable and understandable.                             | Machine-readable, not human-understandable. |  |
| Can be modified easily.  |   | Cannot be modified.  |
| Contains comments for better understanding.                    |   | Does not contain comments.                                       |
| Fewer statements compared to object code.                      |   | More statements due to binary instructions.                      |
| Less close to the machine.                                     |   | Closer to the machine.   |
| Lower performance as it requires translation.                  |   | Higher performance due to direct execution.                      |
| Not system-specific.   |   | System-specific.   |
| Input to compilers/translators (e.g., assembler, interpreter). |   | Output from compilers/translators.                               |
| Ex: <code>print("Hello, World!")</code> (Python).              |   | Ex: 01100110 10011010 (binary code).                             |

### 20.Github and Introductions:- Why is version control important in software development?

#### ➤ Importance:-

- Collaboration: Developers often work from different locations and contribute to various functionalities/features of a project.
- Tracking Changes: Version control tracks modifications to the source code, including who made the changes and what was changed.
- Branching: Developers create separate branches for their changes, and only after review are they merged into the main codebase.
- Organization: Keeps the source code organized and ensures that the latest version is always available.
- Error Recovery: If a new update causes issues, version control allows teams to roll back to a stable version, ensuring continuous development.
- Improved Productivity: Helps developers focus on their tasks while minimizing conflicts and streamlining the development process.

### 21.Student Account in Github:- What are the benefits of using Github for students?

#### ➤ Benefits of Using GitHub for Students:-

- GitHub Education Portal: Access to tools, resources, and benefits in one place.
- Campus Experts: Connect with experts for guidance.
- Free Tools: Access free industry tools via the Student Developer Pack.
- GitHub Classroom: View assignments and track due dates.
- GitHub Copilot: Free subscription for verified students.
- GitHub Codespaces: Free access to GitHub Codespaces with 180 core hours per month.
- Community Engagement: Access to Campus TV, student-created repositories, and curated events.



## 22.Types of Software:- What are the differences between open-source and proprietary software?

| OPEN-SOURCE  | PROPRIETARY   |
|--|---|
| Publicly available for modification and redistribution.        | Source code is private and owned by the company.        |
| Users can modify and improve the software.                     | Only the company can modify the software.               |
| Free of charge.  | Users must purchase a license.                          |
| No need for an authenticated license.                          | Requires a valid, authenticated license to use.         |
| Managed by an open-source community.                           | Managed by a closed team or organization.               |
| Highly flexible and encourages innovation.                     | Limited flexibility and innovation due to restrictions. |
| Community-driven, with faster bug fixes and improved security. | Vendor-controlled, fixes may be slower.                 |
| Limited intellectual property protections.                     | Full intellectual property protections.                 |
| Typically non-profit or community-driven.                      | Usually developed by for-profit organizations.          |
| Linux, Firefox, Android, VLC, GIMP.                            | Windows, macOS, Microsoft Office, Adobe Flash.          |

## 23.GIT and GITHUB Training:- How does GIT improve collaboration in a software development team?

- Git significantly improves collaboration in software development by enabling parallel development, branching, and structured workflows.
- Streamlined Collaboration: Git allows developers to work on different features in separate branches, reducing the risk of overwriting each other's changes. Pull requests enable code review, discussion, and collaboration before merging code, which improves quality.
- Improved Workflow with Branching Models: Git's branching supports structured workflows like GitFlow and GitHub Flow, which organize the development process, ensuring clear processes for feature development, testing, and releases. This leads to more efficient development, fewer conflicts, and better collaboration.

## 24.Application Software:- What is the role of application software in business?

- Application software helps businesses work more efficiently, make better decisions, and serve customers better. Here are some roles that application software plays in business:
- ~ Streamlining Operations: Automates repetitive tasks (e.g., payroll, inventory management) to save time and reduce errors.
- ~ Improving Productivity: Tools like word processors, spreadsheets, and project management software help employees work more efficiently.
- ~ Data Management: Helps businesses store, organize, analyze, and retrieve data for better decision-making (e.g., CRM, ERP systems).

- ~ Enhancing Communication: Facilitates internal communication and collaboration through email, messaging apps, and video conferencing.
- ~ Customer Engagement: Supports customer service, marketing, and sales functions (e.g., e-commerce platforms, customer support tools).
- ~ Scalability: Allows businesses to grow by adding new features or functionalities through adaptable software systems.

## 25. Software Development Process:- What are the main stages of the software development process? / What is SDLC? Explain each phase of SDLC.

➤ **Software Development Life Cycle(SDLC)**, is a project management model that describes the process of creating and maintaining software.

• **SDLC includes the following phases:**

- Requirements gathering and analysis:- This phase involves gathering information about the software requirements from stakeholders, such as customers, end-users, and business analysts.
  - ~ Project plan, timeline, resource allocation
  - ~ Requirement specification document
- Design:- In this phase, the software design is created, which includes the overall architecture of the software, data structures, and interfaces. It has two steps:
  - ~ High-level design (HLD): It gives the architecture of software products.
  - ~ Low-level design (LLD): It describes how each and every feature in the product should work and every component.
  - ~ System architecture, design diagrams
- Implementation or coding:- The design is then implemented in code, usually in several iterations, and this phase is also called as Development.
  - ~ In front-end; In Middleware; In the back-end
  - ~ Source code, executables
- Testing:- The software is thoroughly tested to ensure that it meets the requirements and works correctly. Test cases, bug reports, test results
- Deployment:- After successful testing, The software is deployed to a production environment and made available to end-users. Production environment setup, user access
- Maintenance:- This phase includes ongoing support, bug fixes, and updates to the software. Bug fixes, updates, performance monitoring



## 26. Software Requirement:- Why is the requirement analysis phase critical in software development?

- The requirement analysis phase is critical in software development because it ensures that the development team understands the needs of stakeholders and what the software should

achieve. This phase helps to avoid misunderstandings and costly changes later in the development process.

- **Importance of Requirement Analysis 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.

## **27. Software Analysis:- What is the role of software analysis in the development process?**

- Software analysis plays a crucial role in the development process by ensuring the software meets user needs, is well-structured, and is built efficiently. Here’s how it contributes:
- ~ Requirement Gathering: Identifies and documents the functional and non-functional requirements of the software to ensure it aligns with user and business needs.
- ~ Feasibility Study: Assesses the technical, financial, and operational feasibility of the software project to determine if it is achievable within constraints.
- ~ Problem Understanding: Analyzes the existing system or problem domain to understand the challenges and define the scope of the new system.
- ~ System Modeling: Creates models like Data Flow Diagrams (DFDs), Entity-Relationship Diagrams (ERDs), and Use Case Diagrams to visualize system processes and relationships.

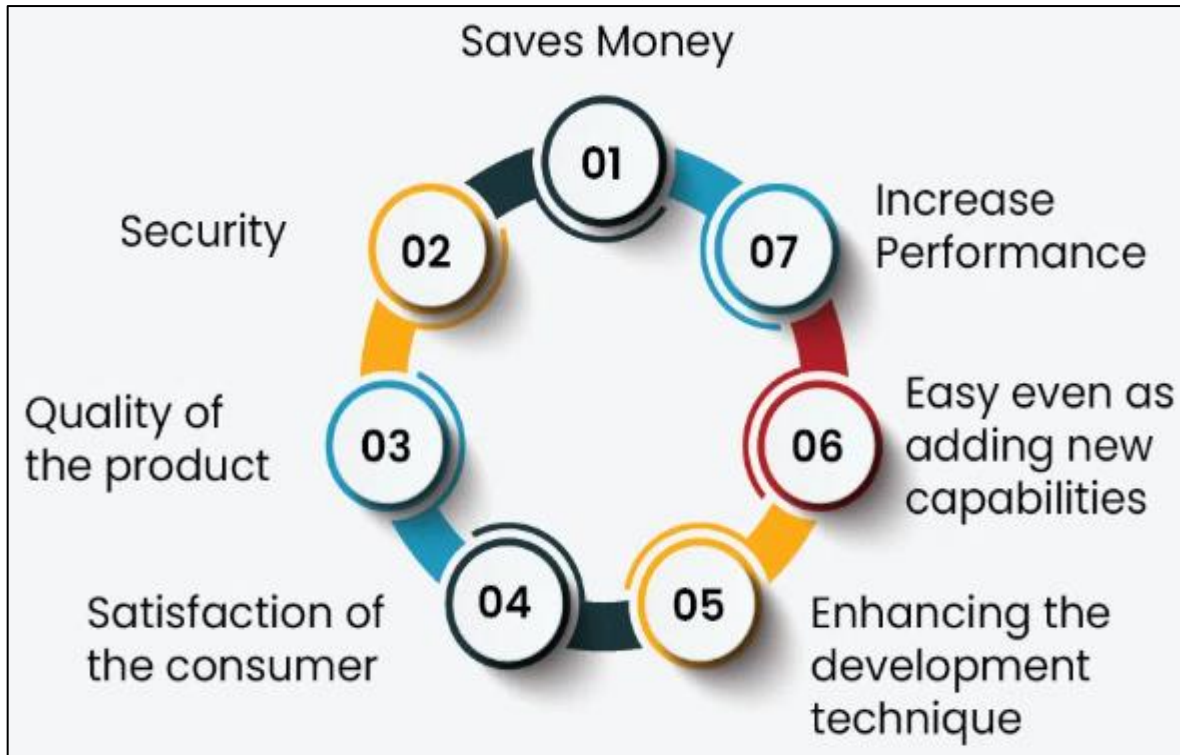
- ~ Risk Identification: Identifies potential risks, such as technical challenges or resource limitations, and proposes mitigation strategies.
- ~ Improves Communication: Acts as a bridge between stakeholders, developers, and designers by clearly defining requirements and system behavior.
- ~ Foundation for Design: Provides a blueprint for system design by detailing what the system should do and how it should interact with users and other systems.
- ~ Ensures Quality: Reduces errors and ambiguities early in the development process, leading to a more robust and reliable software product.
- ~ Cost and Time Efficiency: Helps prevent costly redesigns and delays by addressing issues and ambiguities before development begins.
- ~ Documentation: Creates a detailed record of requirements, analysis, and decisions that guide development and assist in future maintenance.

## **28. System Design:- What are the key elements of system design?**

- System design is the art of creating a blueprint for a system that meets specified requirements, solves user problems, and handles future growth. A well-designed system is scalable, reliable, maintainable, and secure, making it an essential aspect of software engineering. But what exactly are the key components of system design, and what should you keep in mind while designing a system? Let's explore.
- **Key Components of System Design**
  - ~ Architecture:- Defines the system's structure and behavior; choose patterns like monolithic or microservice and serverless, each with its own pros and cons. Selecting the right architecture depends on the specific requirements of your application.
  - ~ Database Design:- Ensures efficient, scalable data storage using SQL or NoSQL, with proper indexing and relationships.
  - ~ APIs and Communication:- APIs (Application Programming Interfaces) define how different components of your system interact with each other and with external systems. The choice of communication protocol (e.g., REST, GraphQL, gRPC) depends on factors like performance, flexibility, and simplicity.
  - ~ Caching:- Speeds up performance by storing frequently accessed data in memory. By reducing the need to repeatedly fetch data from slower storage layers, caching helps improve response times and reduce load on the database.
  - ~ Load Balancing:- Distributes traffic across servers to improve reliability and performance. Common strategies include round-robin, least connections, and weighted load balancing, depending on the system's specific needs.
  - ~ Security:- Protects data and prevents vulnerabilities with encryption, authentication, and regular audits.
  - ~ Scalability and Performance:- Scalability is the ability of a system to handle increasing loads by adding resources. Horizontal scaling (adding more servers) and vertical scaling (upgrading existing servers) are two common approaches.
  - ~ Redundancy and Fault Tolerance:- Redundancy involves duplicating critical components or functions to provide backup in case of failure. Fault tolerance is the system's ability to continue operating despite failures.

- ~ Monitoring and Logging:- Monitoring and logging are essential for understanding the health and performance of your system. Tools like Prometheus, Grafana, and ELK (Elasticsearch, Logstash, Kibana) help track metrics, detect anomalies, and diagnose issues. A well-designed monitoring system can alert you to problems before they affect users.
- ~ User Experience (UX):- While system design is often focused on the backend, it's essential to consider how design decisions affect the user experience. Fast, responsive, and intuitive interfaces contribute to user satisfaction. Ensure that your design supports smooth interactions, even under heavy loads.

## 29. Software Testing:- Why is software testing important?

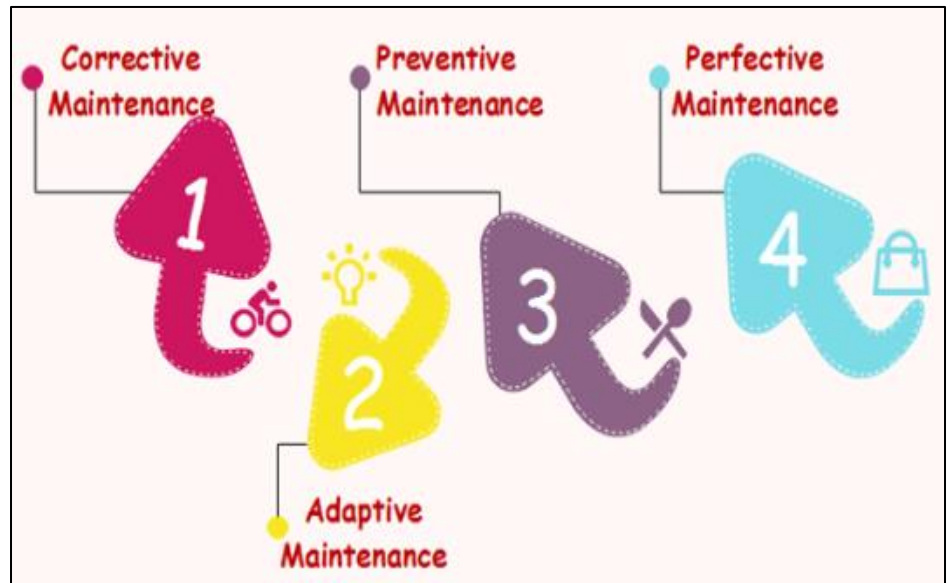


- ~ Saves Money:- Early bug detection reduces the cost of fixing issues. Testing minimizes expensive post-release patches.
- ~ Enhances Security:- Ensures user data is safe and secure from vulnerabilities. Builds trust in the product by delivering reliable and secure software.
- ~ Improves Product Quality:- Ensures the product meets requirements and functions as intended. Tests compatibility across devices and operating systems.
- ~ Increases Customer Satisfaction:- Provides a seamless and reliable user experience. Builds customer trust and long-term loyalty.
- ~ Boosts Development Efficiency:- Helps identify and fix issues faster, improving the development process. Encourages collaboration between developers and testers.
- ~ Supports Feature Addition:- Simplifies adding new features without breaking existing functionality. Makes the software adaptable and competitive in the market.
- ~ Validates Performance:- Ensures the software operates efficiently under various conditions. Avoids a negative impact on reputation due to poor performance.

### 30.Maintenance:- What types of software maintenance are there?

➤ Software maintenance is a part of the Software Development Life Cycle. Its primary goal is to modify and update software application after delivery to correct errors and to improve performance. Software is a model of the real world. When the real world changes, the software require alteration wherever possible.

~ It is an inclusive activity that includes error corrections, enhancement of capabilities, deletion of obsolete capabilities, and optimization.



#### • Several Types of Software Maintenance

- ~ Corrective Maintenance: This involves fixing errors and bugs in the software system. It correct any remaining errors regardless of where they may cause specifications, design, coding, testing, and documentation, etc.
- ~ Adaptive Maintenance: This involves modifying the software system to adapt it to changes in the environment, such as changes in hardware or software, government policies, and business rules.
- ~ Preventive Maintenance: This involves taking measures to prevent future problems, such as optimization, updating documentation, reviewing and testing the system, and implementing preventive measures such as backups.
- ~ Perfective Maintenance: This involves improving functionality, performance, and reliability, and restructuring the software system to improve changeability. It defines improving processing efficiency or performance or restricting the software to enhance changeability. This may contain enhancement of existing system functionality, improvement in computational efficiency, etc.

### 31.Development:- What are the key differences between web and desktop applications?

| WEB APP   | DESKTOP APP   |
|---|---|
| They require installation on the computer to run.                     | They are accessible through web browsers and do not require installation.                               |
| Generally, desktop apps do not require an internet connection to run. | Web apps cannot run without an internet connection.   |
| They are accessible only in the machine they are installed in.        | They are accessible from anywhere and through any device with an internet connection and a web browser. |
| They take space on the hard drive of the local computer.              | They take up space on the remote server.  |

|   |   |
|---|---|
| Deployment and updating are to be done individually on each computer.                 | Deployment and updating are done only on the server.  |
| They have strict hardware requirements for proper functionality.                      | Web apps are hardware-independent and just require a web browser and internet connection to function. |
| As they are confined to a device and single or limited users, they are highly secure. | As web apps are accessible to all through the internet, they are less secure than desktop apps.       |
| Generally, they are faster than web applications.                                     | Generally, they are slower than desktop applications.   |

### **32. Web Application:- What are the advantages of using web applications over desktop applications?**

- **Web applications offer several advantages over desktop applications, including:**
  - ~ **Accessibility:** Web applications are accessible from any device with an internet connection and a browser, allowing users to access the application from different locations and platforms (Windows, Mac, Linux, mobile devices).
  - ~ **No Installation Required:** Since web apps run in a browser, there's no need for users to install software. This eliminates the need for system resources to be taken up by software installations.
  - ~ **Automatic Updates:** Updates and bug fixes are applied automatically on the server-side, meaning users always have access to the latest version of the application without needing to manually update their software.
  - ~ **Cross-Platform Compatibility:** Web applications can run on various operating systems (Windows, macOS, Linux, Android, iOS) without requiring separate versions for each platform, unlike desktop applications that may need to be developed for each OS.
  - ~ **Centralized Data Storage:** Data in web applications is usually stored on centralized servers, making it easier to manage, back up, and access from multiple devices. This reduces the risk of losing data due to local system failures.
  - ~ **Cost-Effectiveness:** Web apps reduce the need for developers to create multiple versions for different platforms, potentially lowering development costs. It also reduces the need for end-users to maintain the app on their devices.
  - ~ **Collaboration:** Many web applications are built with collaboration in mind, allowing multiple users to access and interact with the same data simultaneously, regardless of their physical location.
  - ~ **Security:** With web applications, security updates and patches can be deployed faster and more consistently across all users. Data is often stored on secure servers with advanced protections, while desktop applications might rely on individual user settings for security.
  - ~ **Scalability:** Web applications can be easily scaled by upgrading server resources, whereas desktop applications often require more complex handling for scalability.
  - ~ **Lower System Requirements:** Web apps typically have lower system resource requirements since the heavy lifting is done on the server side, leaving the user's device to handle the display and input/output.



### 33.Designing:- What role does UI/UX design play in application development?

- ~ Better User Retention: Good UI/UX design keep user engaged and encourages them to return.
- ~ Effective Interaction: Ensures easy, enjoyable, and efficient interactions between users and the app.
- ~ Turning Customers to Loyal Ones: UX design creates a positive experience, turning users into repeat customers.
- ~ Saves Resources in the Long Run: A dedicated UI/UX team ensures sustainable development and prevents constant redesigns.
- ~ Communicates Mission and Service: UI and design communicate your brand's message and value to users effectively.
- ~ Scaling with Mobile UI/UX: Mobile UI/UX design targets more users and optimizes the experience for smaller screens.
- ~ UI/UX is More Than Design: UI/UX involves research, user needs analysis, and improving the overall user experience, not just visual design.

### 34.Mobile Application:- What are the differences between native and hybrid mobile apps?

| CHARACTERISTICS        | WEB APP  | HYBRID APP  | NATIVE APP  |
|------------------------|--|---|---|
| Usage                  | Users can access directly from a browser   | Users have to install the app on their device of choice             | Users have to install the app on their device of choice   |
| Internal working       | Client code in the browser communicates with remote server-side code and databases | Client code and browser code wrapped in a native shell or container | Client code written in technology and language specific to the device or platform it will be installed on |
| Native device features | Not accessible   | Accessible  | Accessible  |
| User experience        | Inconsistent and dependent on the browser being used                               | Consistent and engaging   | Consistent and engaging   |
| Access                 | Limited by browser and network connectivity  | One-step access with offline features                               | One-step access with offline features   |
| Performance            | Slower and less responsive   | Faster, but may consume more battery power                          | Performance can be optimized to device  |
| Development            | Cost-efficient, faster time to market  | Cost-efficient, faster time to market                               | Expensive, slower time to market  |
|                        | Learn more about Web Apps  | Learn more about Hybrid Apps  | Learn more about Native Apps  |



### **35.Desktop Application:- What are the pros and cons of desktop applications compared to web applications?**

#### **➤ Desktop Applications**

- Pros:

- ~ Performance: Generally faster and more responsive, especially for resource-intensive tasks.
- ~ Offline Access: Work without an internet connection.
- ~ Hardware Integration: Can leverage device-specific hardware (e.g., GPU, storage).
- ~ Security: Data is stored locally, potentially reducing security risks.
- ~ User Interface: Can offer a more customized and tailored user experience.

- Cons:

- ~ Platform Limitations: Often restricted to a specific operating system.
- ~ Installation and Updates: Requires manual installation and updates, which can be inconvenient.
- ~ Compatibility: May not be compatible with newer operating systems or hardware.
- ~ Cost: Can be expensive to develop and maintain.

#### **➤ Web Applications**

- Pros:

- ~ Cross-Platform Compatibility: Accessible from any device with a web browser.
- ~ Ease of Deployment: No installation required; updates are automatic.
- ~ Accessibility: Accessible from anywhere with an internet connection.
- ~ Collaboration: Easy to share and collaborate on data.
- ~ Cost-Effectiveness: Often cheaper to develop and maintain.

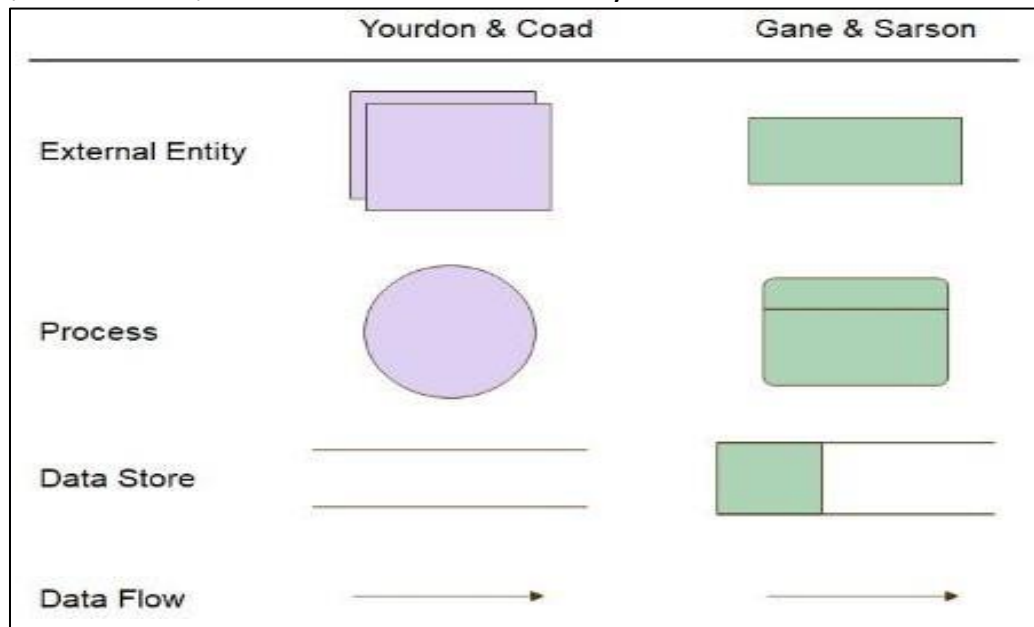
- Cons:

- ~ Performance: Can be slower and less responsive than desktop applications, especially with slow internet connections.
- ~ Offline Access: Limited or no functionality without an internet connection.
- ~ Security: Data is stored on servers, potentially increasing security risks.
- ~ User Interface: May be less customizable due to browser limitations.
- ~ Internet Dependency: Requires a stable internet connection for optimal performance.

### 36.What is DFD? Create a DFD diagram on Flipkart.

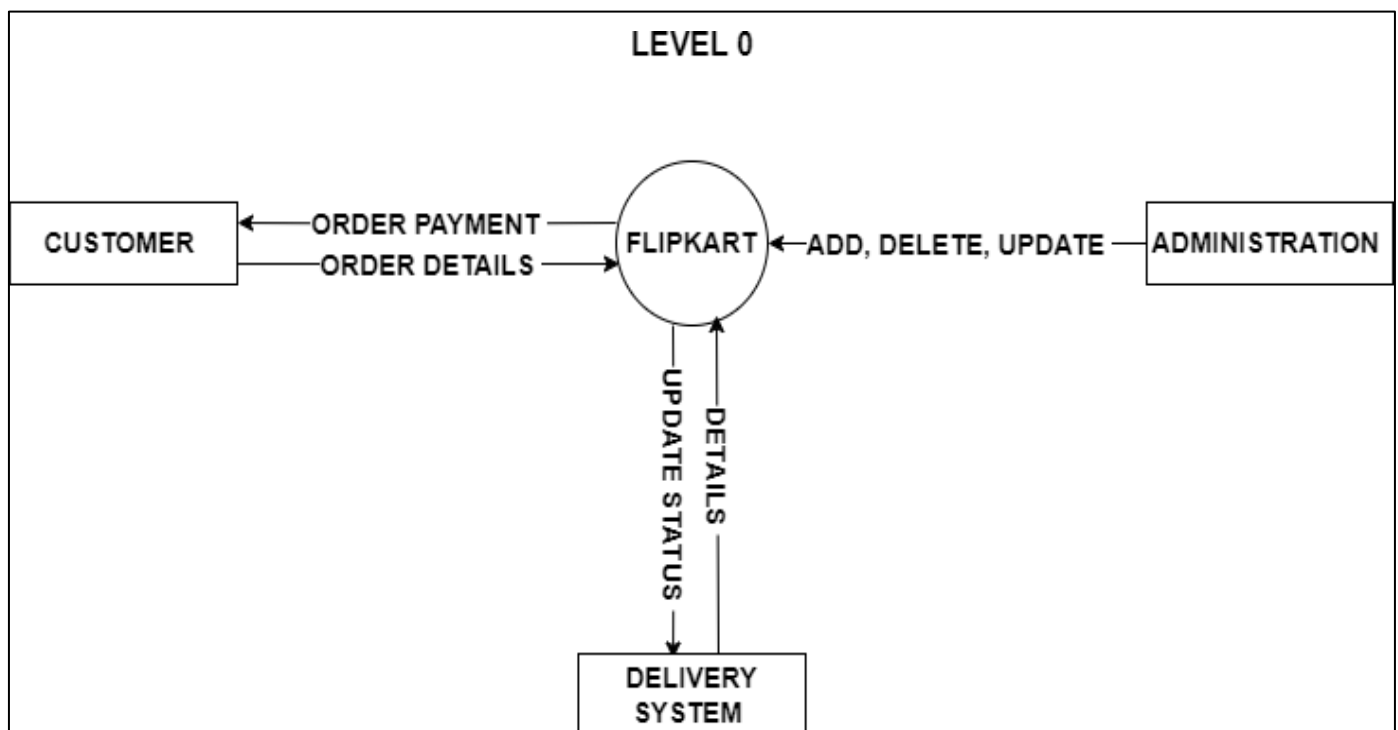
- A **Data Flow Diagram (DFD)** is a graphical representation of the flow of data through a system, illustrating how inputs are transformed into outputs. It shows the interaction between processes, data stores, and external entities in a system.

- **DFD Symbols:-**

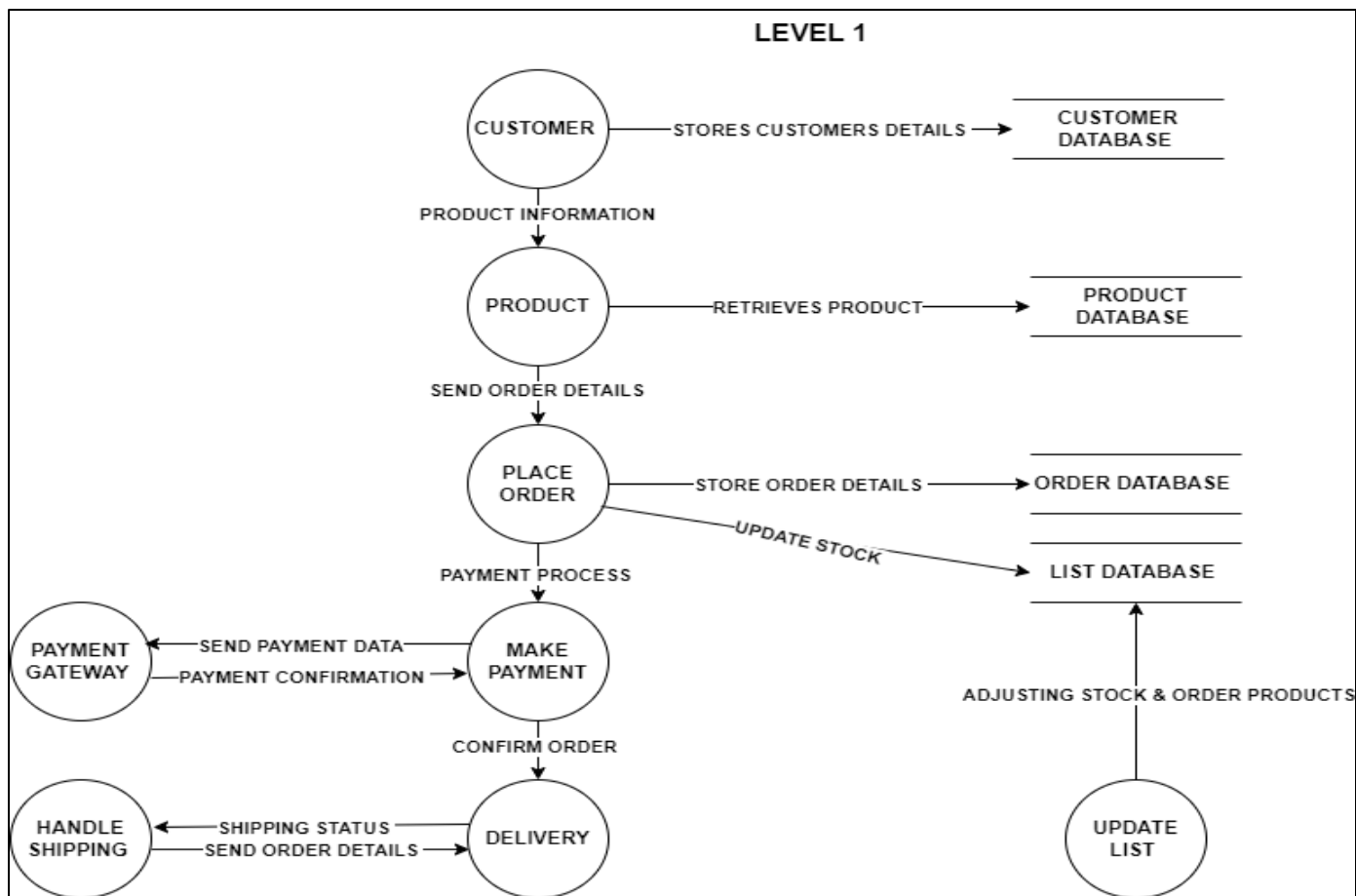


- **DFD diagram on Flipkart:-**

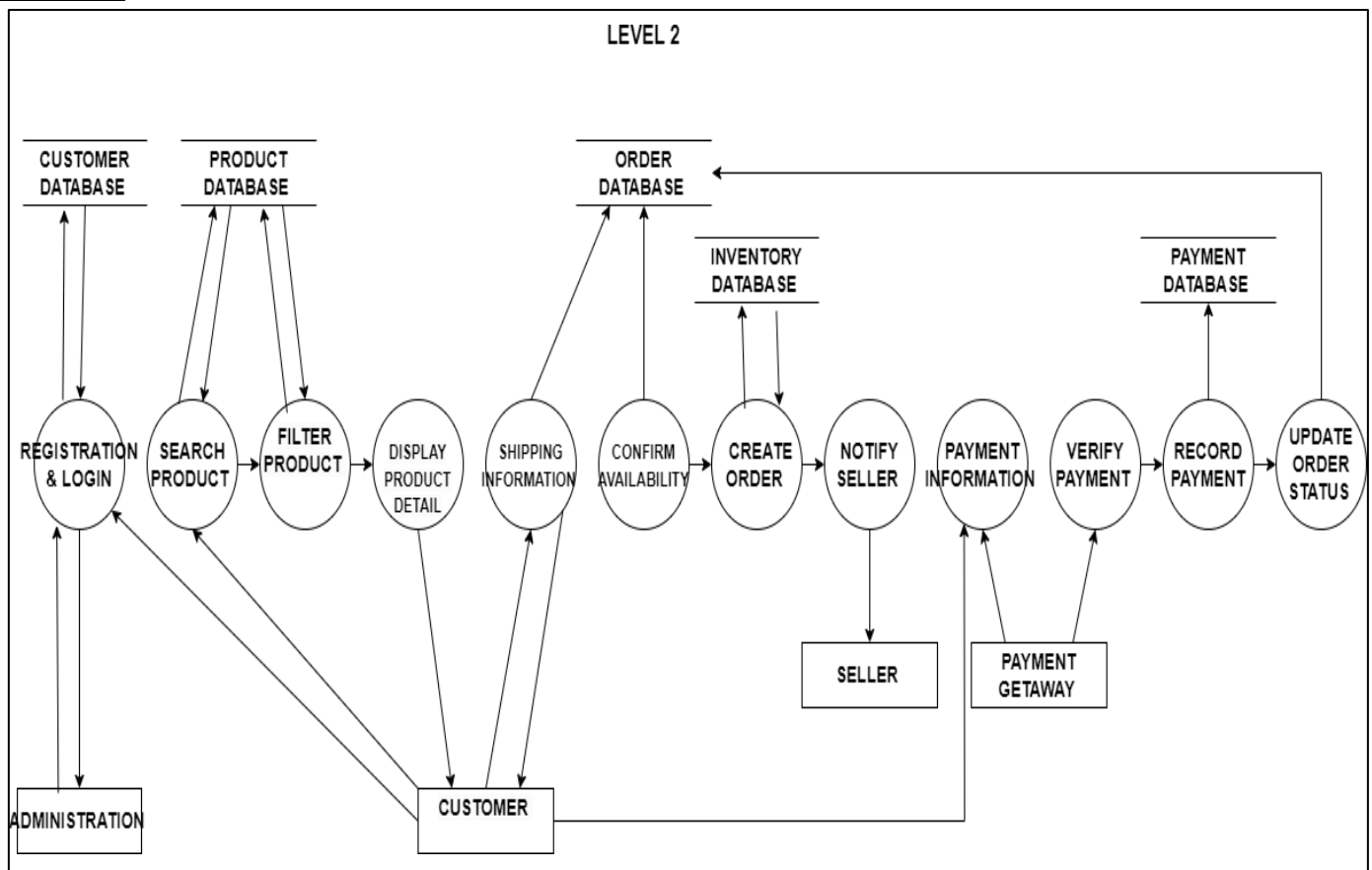
~ LEVEL 0:-



~ LEVEL 1:-








~ LEVEL 2:-



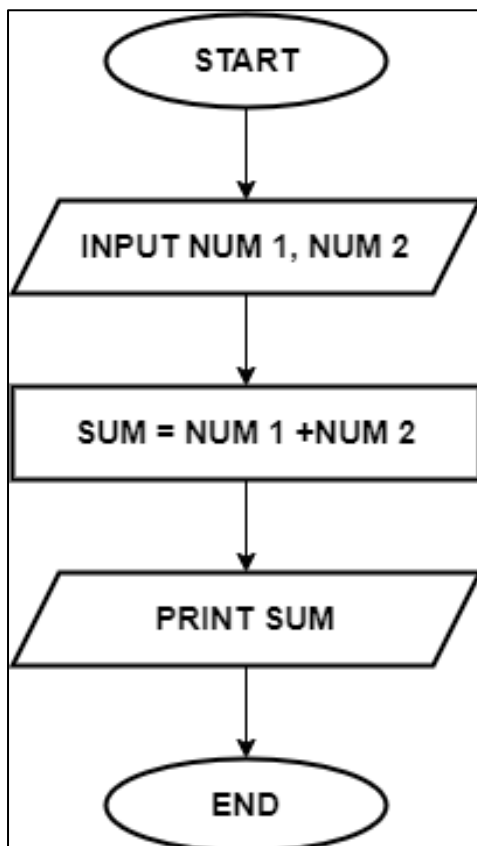
### 37.What is Flow chart? Create a flowchart to make addition of two numbers.

- A **flowchart** is a type of diagram that represents a workflow or process. A flowchart can also be defined as a diagrammatic representation of an algorithm, a step-by-step approach to solving a task.

- **Flowchart Symbols:-**

| Symbol  | Name         |
|---|--------------|
|  | Start/end    |
|  | Arrows       |
|  | Input/Output |
|  | Process      |
|  | Decision     |

- **Flowchart to make addition of two numbers**



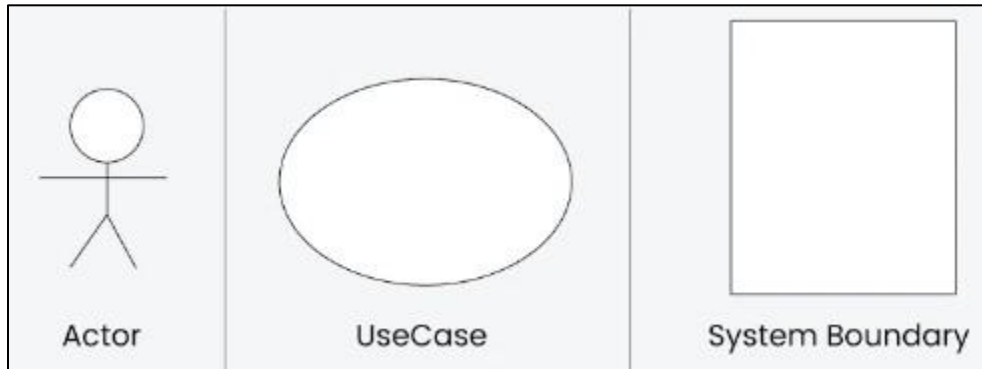
### **38.Flow Chart:- How do flowcharts help in programming and system design?**

- Flowcharts are a useful tool in programming and system design because they help visualize the logic and flow of a program:
- ~ Planning:- Flowcharts can help programmers plan the logic behind new programs.
- ~ Understanding:- Flowcharts can help programmers understand and work with less familiar code structures by mapping existing programs.
- ~ Communication:- Flowcharts can help programmers communicate their ideas to other team members, supervisors, and external stakeholders.
- ~ Debugging:- Flowcharts can help programmers identify potential problem areas and debug code that is not working.
- ~ Efficiency:- Flowcharts can help programmers develop efficient coding by clearly showing where data is going to end up.
- ~ Organization:- Flowcharts can help organize big-picture thinking and provide a guide when it comes time to code.
- ~ Collaboration:- Flowcharts can help large teams better understand a project process and any necessary data flows.
- ~ Blueprinting:- Flowcharts can be used for multiple projects, saving time and effort in the future.
- ~ Data management:- Flowcharts can help analyze a process to ensure no functions, inputs, or outputs get overlooked.

### 39.What is Use case Diagram? Create a use-case on bill payment on pay tm.

- A **Use Case Diagram** is a type of Unified Modeling Language (UML) diagram that represents the interaction between users or external systems and a system under consideration to accomplish specific goals. It provides a high-level view of the system's functionality by illustrating the various ways users can interact with it.

- **Use Case Diagram Symbols:-**



- **Use-case on bill payment on paytm:-**

