**Assignment**

**MODULE: 1 SE – Overview of IT Industry**

1. **What is software? What is software engineering?**

* **Software:** Software is a set of instructions, data or programs used to operate computers and execute specific tasks. It is the opposite of hardware, which describes the physical aspects of a computer. Software is a generic term used to refer to applications, scripts and programs that run on a device.

Software is typically categorized into three main types:

1. **System Software**: This includes the operating system and all utility programs that manage computer resources at a low level. e.g., Windows, macOS, Linux
2. **Application Software**: These are programs designed to perform specific tasks for users. Examples include word processors (e.g., Microsoft Word), web browsers (e.g., Google Chrome), graphic design tools (e.g., Adobe Photoshop), and business software (e.g., accounting programs)
3. **Development Software**: This category encompasses the tools used by software developers to create, debug, maintain, or otherwise support software. It includes programming languages (e.g., Python, Java), compilers, interpreters, integrated development environments (IDEs), and version control systems.

* **Software engineering**: software engineering is the discipline of designing, creating, and maintaining software systems using engineering principles and techniques.

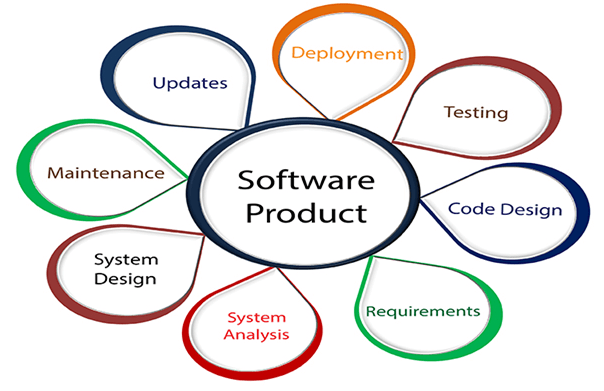
It involves various stages such as requirements analysis, design, coding, testing, deployment, and maintenance to ensure that software products are reliable, scalable, and efficient.

Software engineers apply engineering principles and knowledge of programming languages to build software solutions for end users.

The term software engineering is the product of two words software & engineering.

The software is a collection of integrated programs. Software subsists of carefully-organized instructions and code written by developers on any of various particular computer languages. Computer programs and related documentation such as requirements, design models and user manuals.

Engineering is the application of scientific and practical knowledge to invent, design, build, maintain, and improve frameworks, processes, etc.



Software Engineering is required for:

* To manage large software
* For more stability
* Cost Management
* To manage the dynamic nature of software
* For better quality Management

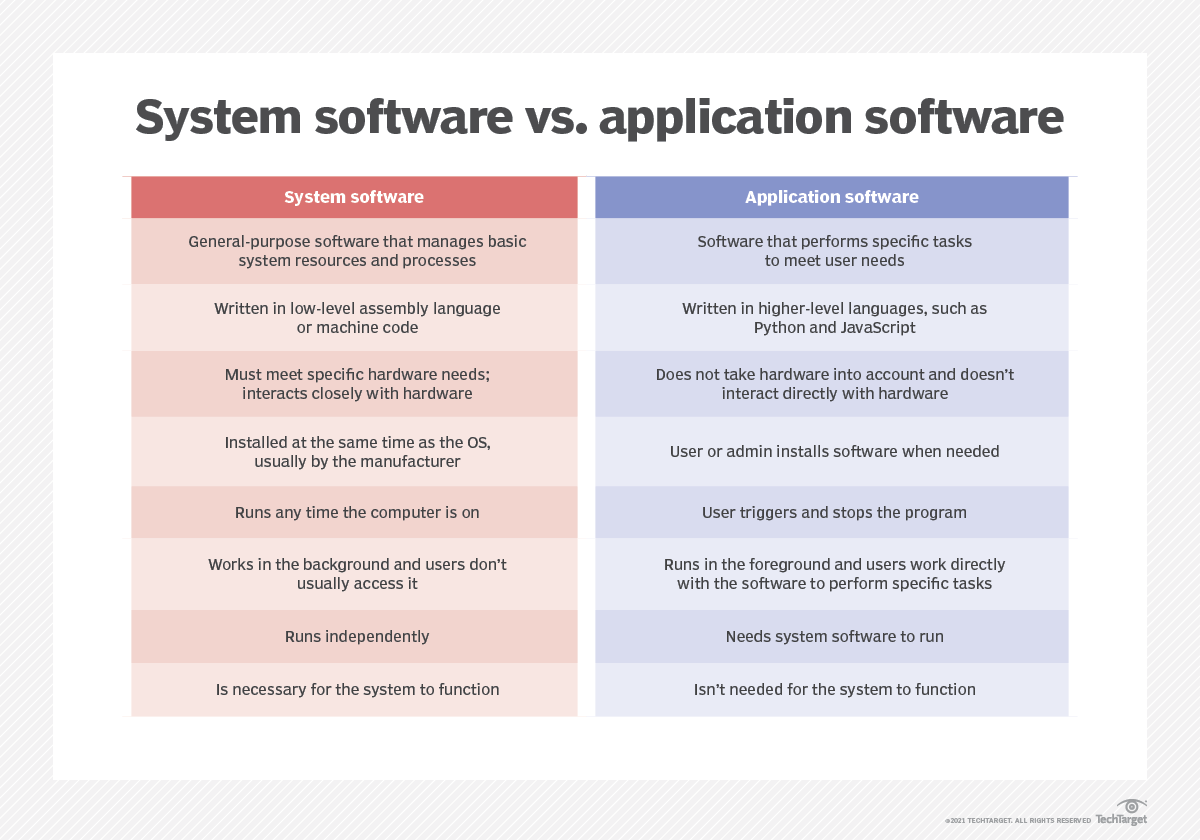
Characters of good software engineering:

* Exposure to systematic methods, i.e., familiarity with software engineering principles.
* Good technical knowledge of the project range (Domain knowledge).
* Good programming abilities
* Good communication skills. These skills comprise of oral, written, and interpersonal skills.
* High motivation
* Sound knowledge of fundamentals of computer science.
* Intelligence.
* Ability to work in a team
* Discipline, etc.

1. **Explain types of software.**

* Software is a set of instructions, data or programs used to operate computers and execute specific tasks. It is the opposite of hardware, which describes the physical aspects of a computer. Software is a generic term used to refer to applications, scripts and programs that run on a device. It can be thought of as the variable part of a computer, while hardware is the invariable part.

The two main categories of software are application software and system software. An application is software that fulfils a specific need or performs tasks. System software is designed to run a computer's hardware and provides a platform for applications to run on top of.



Among the various categories of software, the most common types include the following:

1. **Application software:** The most common type of software, application software is a computer software package that performs a specific function for a user, or in some cases, for another application. An application can be self-contained, or it can be a group of programs that run the application for the user. Examples of modern applications include office suites, graphics software, databases and database management programs, web browsers, word processors, software development tools, image editors and communication platforms.
2. **System software:** These software programs are designed to run a computer's application programs and hardware. System software coordinates the activities and functions of the hardware and software. In addition, it controls the operations of the computer hardware and provides an environment or platform for all the other types of software to work in. The OS is the best example of system software; it manages all the other computer programs. Other examples of system software include the firmware, computer language translators and system utilities.
3. **Driver software:** Also known as device drivers, this software is often considered a type of system software. Device drivers control the devices and peripherals connected to a computer, enabling them to perform their specific tasks. Every device that is connected to a computer needs at least one device driver to function. Examples include software that comes with any nonstandard hardware, including special game controllers, as well as the software that enables standard hardware, such as USB storage devices, keyboards, headphones and printers.
4. **Middleware**: The term middleware describes software that mediates between application and system software or between two different kinds of application software. For example, middleware enables Microsoft Windows to talk to Excel and Word. It is also used to send a remote work request from an application in a computer that has one kind of OS, to an application in a computer with a different OS. It also enables newer applications to work with legacy ones.
5. **Programming software:** Computer programmers use programming software to write code. Programming software and programming tools enable developers to develop, write, test and debug other software programs. Examples of programming software include assemblers, compilers, debuggers and interpreters.
6. **Embedded Software:** Designed to operate hardware and perform specific control functions within larger systems. Embedded Systems Software Found in various devices like automotive control systems, medical devices, and consumer electronics (e.g., software in a microwave oven or a car's navigation system).
7. **Entertainment Software:** Provides entertainment and recreational activities.

Video Games: Software designed for interactive play (e.g., Fortnite, Minecraft, The Legend of Zelda). Media Players: Applications used to play audio and video content (e.g., iTunes, Spotify, Windows Media Player).

1. **Educational Software:** Facilitates learning and education. E-Learning Platforms Provide online courses and training programs (e.g., Khan Academy, Coursera). Educational Games: Designed to teach specific skills or knowledge through gameplay (e.g., Duolingo for language learning, educational software for kids like ABCmouse).
2. **Business Software:** Supports business activities and improves productivity. Customer Relationship Management (CRM): Manages interactions with customers and clients (e.g., Salesforce, HubSpot). Enterprise Resource Planning (ERP): Integrates core business processes (e.g., SAP, Oracle ERP).

Accounting Software: Manages financial transactions and accounting records (e.g., QuickBooks, Xero).

1. **What is SDLC? Explain each phase of SDLC.**

* The Software Development Life Cycle (SDLC) is a structured process that outlines the stages involved in developing software from initial concept to final deployment and maintenance. It provides a systematic approach to planning, creating, testing, and deploying software, ensuring high-quality products that meet or exceed user expectations. The SDLC typically involves several phases, each with specific tasks and deliverables.

The software development lifecycle (SDLC) is the cost-effective and time-efficient process that development teams use to design and build high-quality software. The goal of SDLC is to minimize project risks through forward planning so that software meets customer expectations during production and beyond. This methodology outlines a series of steps that divide the software development process into tasks you can assign, complete, and measure.

**Benefits of SDLC:**

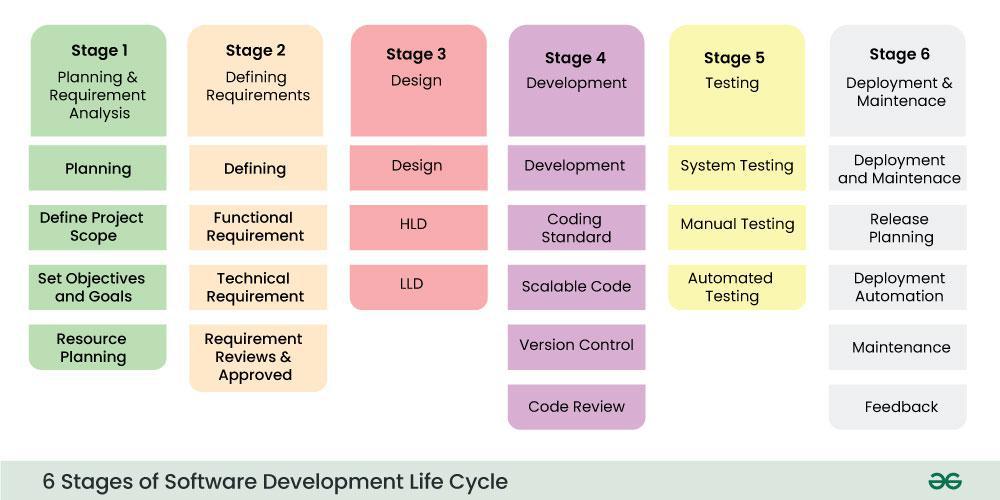
* Increased visibility of the development process for all stakeholders involved
* Efficient estimation, planning, and scheduling
* Improved risk management and cost estimation
* Systematic software delivery and better customer satisfaction

**Importance of SDLC:**

The SDLC is crucial for several reasons:

* **Structure:** Provides a clear, structured approach to software development, ensuring all necessary steps are followed.
* **Quality:** Helps in delivering high-quality software that meets user requirements and expectations.
* **Cost and Time Management:** Facilitates efficient resource allocation, time management, and cost control.
* **Risk Management:** Identifies and mitigates risks early in the development process.
* **Documentation:** Ensures thorough documentation at each phase, aiding in future maintenance and upgrades.

**6 phases of SDLC:**



1. **Planning:**

* Objective: Establish the project’s scope, feasibility, resources, and schedule.
* Activities:
* Conduct feasibility studies (technical, economic, operational).
* Define project goals and objectives.
* Identify resources, budget, and timeline.
* Develop a project plan and risk management strategy.

1. **Requirements Analysis:**

* Objective: Gather and document the functional and non-functional requirements.
* Activities:
* Interview stakeholders and users.
* Collect and analyze user requirements.
* Create requirement specifications.
* Validate and obtain approval of requirements.

1. **Design:**

* Objective: Outline the software architecture and detailed design.
* Activities:
* Develop high-level design (architecture diagrams).
* Create detailed design (data models, interface designs).
* Select technology stack.
* Document the design specifications.

1. **Implementation (Coding):**

* Objective: Translate design documents into executable software.
* Activities:
* Set up development environment.
* Write and compile code.
* Perform unit testing.
* Document code and develop user manuals.

1. **Testing:**

* Objective: Ensure the software meets the required standards and functions correctly.
* Activities:
* Develop and execute test plans.
* Perform various tests (unit, integration, system, acceptance).
* Identify and fix defects.
* Conduct performance and security testing.

1. **Deployment:**

* Objective: Install the software in the production environment and make it available to users.
* Activities:
* Prepare the deployment environment.
* Deploy the software.
* Conduct post-deployment testing.
* Provide user training and support.

1. **Maintenance:**

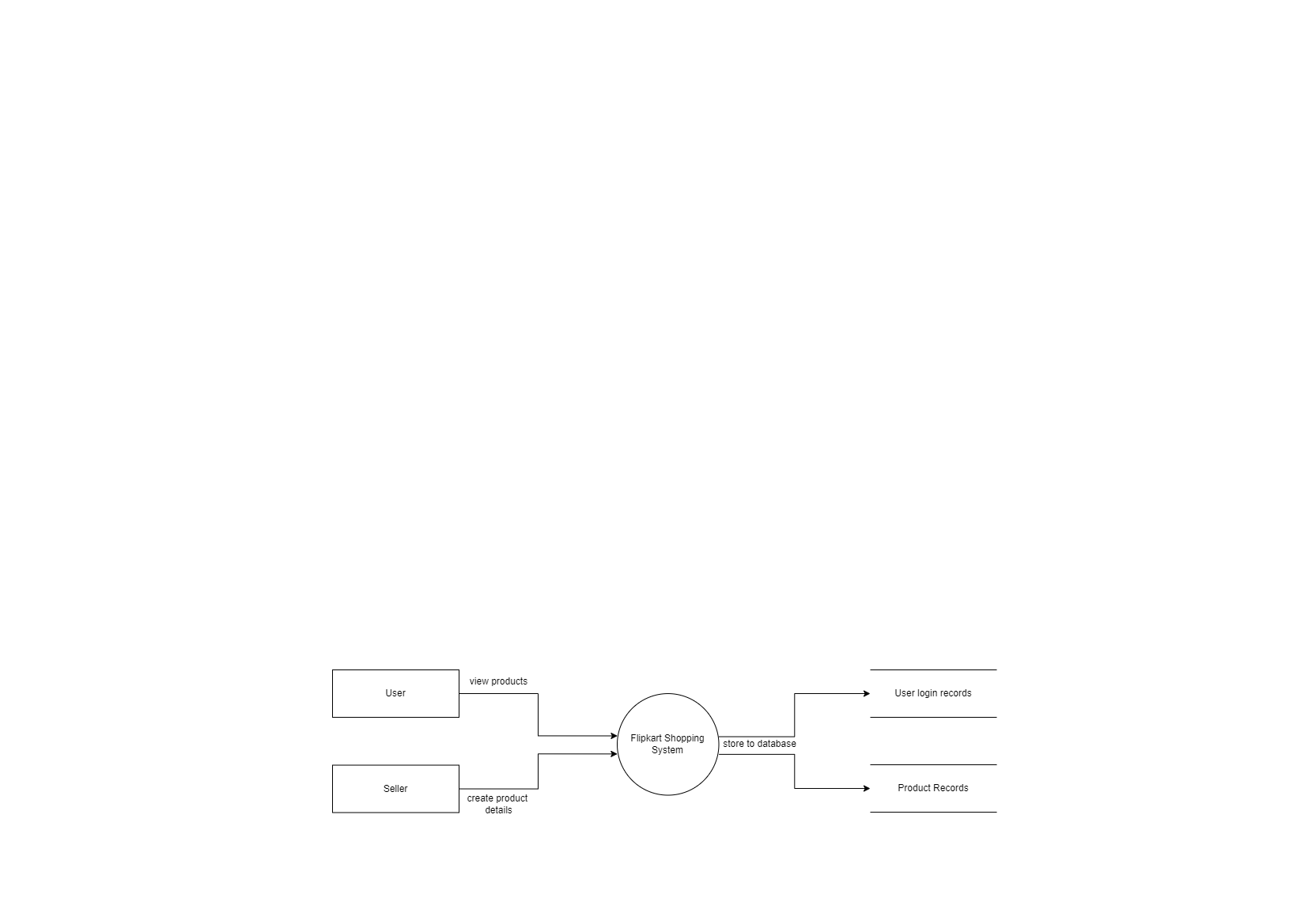
* Objective: Provide ongoing support, fix bugs, and implement improvements.
* Activities:
* Monitor the software for issues.
* Perform regular updates and patches.
* Enhance functionality as needed.
* Provide technical support to users.

1. **What is DFD? Create a DFD diagram on Flipkart.**

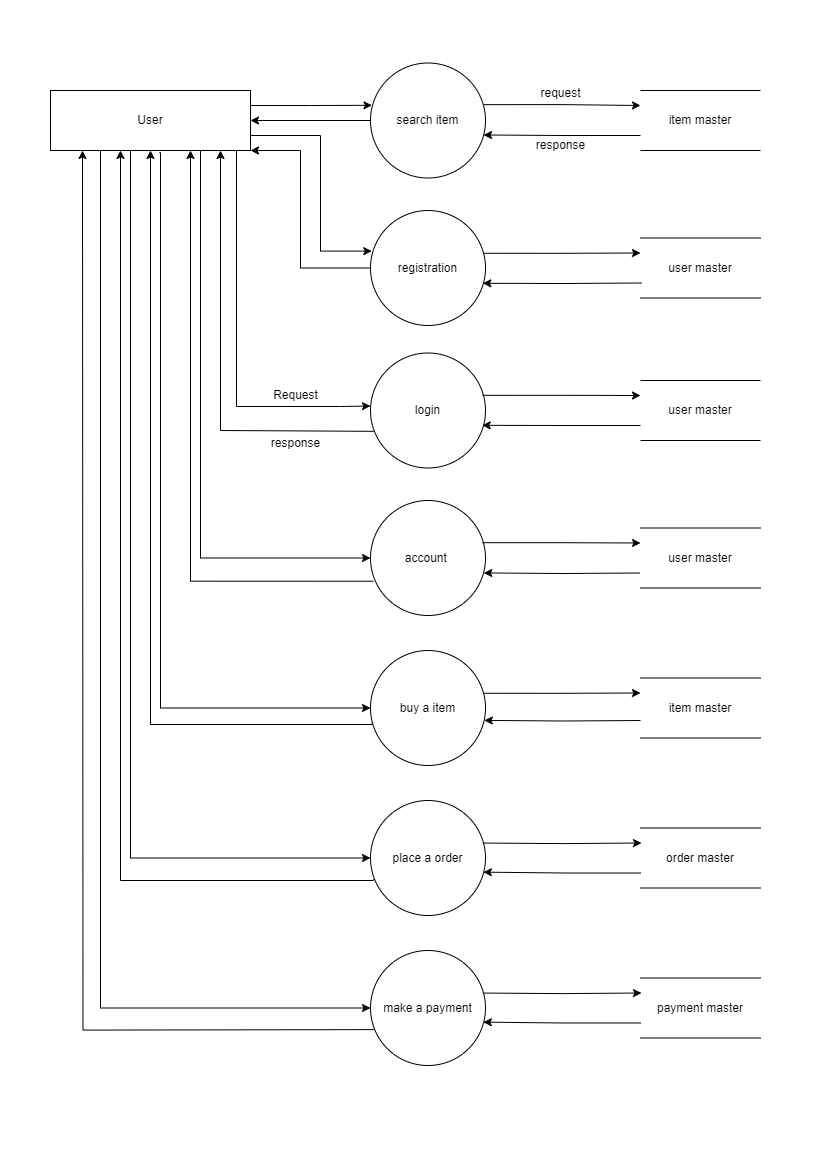
* DFD is the abbreviation for Data Flow Diagram. The flow of data in a system or process is represented by a Data Flow Diagram (DFD). It also gives insight into the inputs and outputs of each entity and the process itself. Data Flow Diagram (DFD) does not have a control flow and no loops or decision rules are present. Specific operations, depending on the type of data, can be explained by a flowchart. It is a graphical tool, useful for communicating with users, managers and other personnel. it is useful for analysing existing as well as proposed systems.

It provides an overview of,

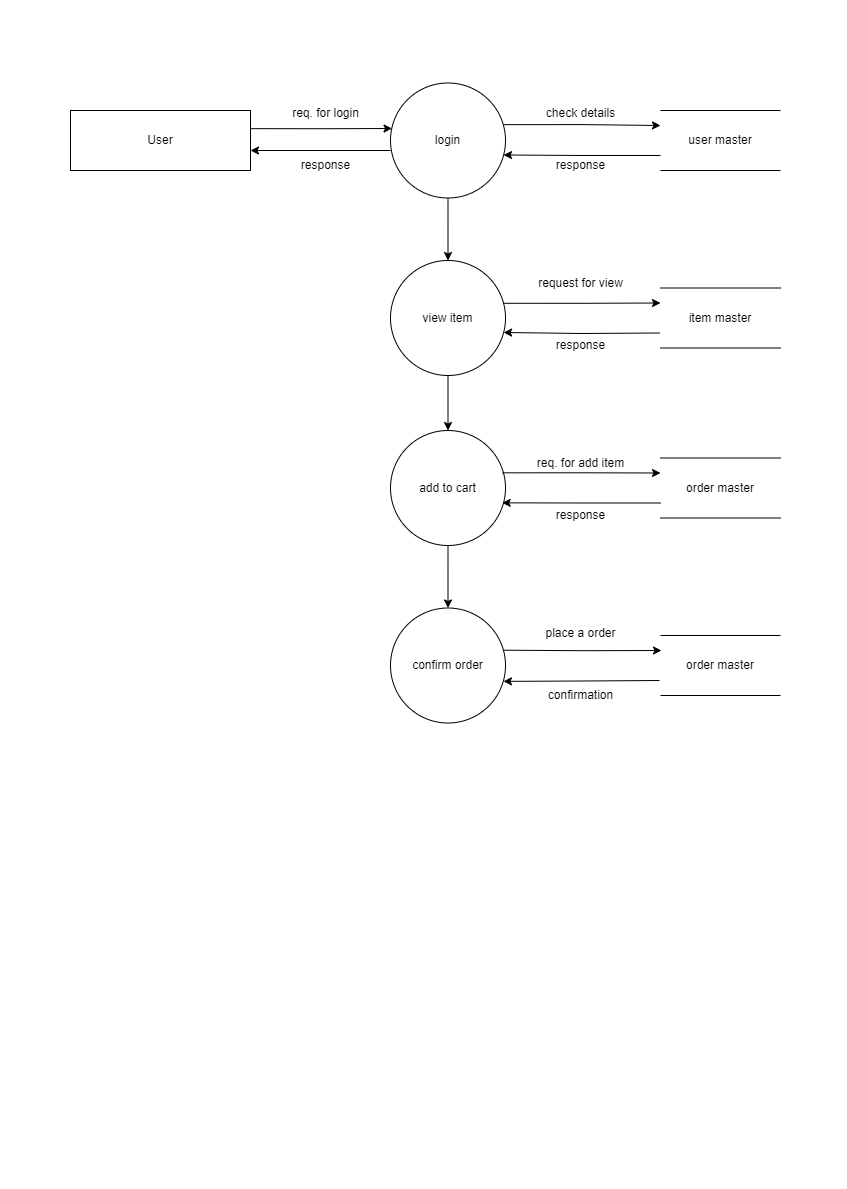
* What data is system processes.
* What transformation are performed.
* What data are stored.
* What results are produced, etc.
* Context level DFD – 0 level:



* 1st level DFD:



* 2nd level DFD (buy a item module):



1. **What is Flow Chart? Create a flowchart to make addition of two numbers.**

* A flowchart is a graphical representation of a process or a system that uses various symbols to denote different steps, actions, or decisions. It helps in visualizing the flow of operations and understanding the sequence of activities in a clear and structured manner. Flowcharts are commonly used in planning, documenting, and analysing processes in various fields such as computer programming, business management, and engineering.

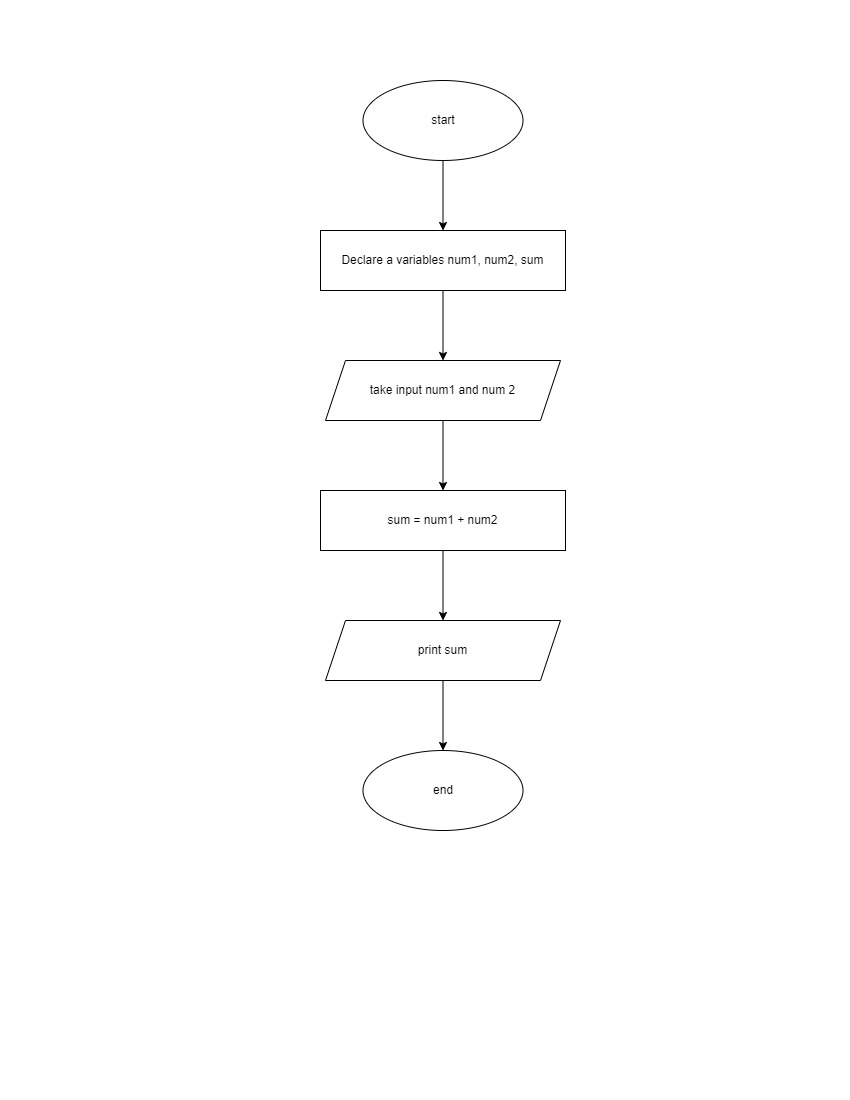
Each step in the process is represented by a different symbol, and arrows are used to show the flow from one step to the next. Flowcharts are commonly used to design and document complex processes, making them easier to understand and analyze.

**Key Elements of a Flowchart:**

1. **Symbols**: Different shapes represent different types of actions or steps:
   * **Oval**: Represents the start and end points of the process.
   * **Rectangle**: Indicates a process or action step, such as a calculation or an operation.
   * **Diamond**: Represents a decision point that can lead to different branches in the flow.
   * **Parallelogram**: Used for input and output operations.
   * **Arrows**: Indicate the direction of the process flow from one step to the next.
2. **Flow Lines**: Arrows that connect the symbols and show the direction of the process flow.

**Benefits of Using Flowcharts:**

* **Clarity**: Visualizing the steps makes complex processes easier to understand.
* **Communication**: Provides a clear way to communicate how a process works.
* **Problem-Solving**: Helps in identifying bottlenecks and inefficiencies in a process.
* **Documentation**: Acts as a reference for process documentation and standardization.
* Flow chart for adding two numbers:



**Explanation:**

1. **Start**: The beginning of the process.
2. **Declaration:** process of declaring variables.
3. **Input Num1**: The user is prompted to input the first number.
4. **Input Num2**: The user is prompted to input the second number.
5. **Sum = num1 + num2**: The process step where the two numbers are added together.
6. **Output Sum**: The result of the addition is displayed.
7. **End**: The end of the process.
8. **What is Use Case Diagram? Create a use-case on bill payment on paytm.**

* A use case diagram is a type of behavioural diagram defined by the Unified Modeling Language (UML) that represents the interactions between users (actors) and a system to achieve a goal. It visually depicts the functional requirements of a system, showing how various actors interact with different use cases (functions or services) that the system provides.

**Key Components of a Use Case Diagram:**

* **Actors**: Entities that interact with the system, which can be users, other systems, or devices. Actors are usually represented by stick figures.
* **Use Cases**: Functions or services that the system provides to the actors, represented by ovals. Each use case should deliver a specific goal or task for the actor.
* **System Boundary**: A rectangle that defines the scope of the system, enclosing all the use cases.
* **Relationships**: Lines or arrows that connect actors to use cases, indicating interactions. Common types of relationships include:
* **Association**: A line connecting an actor to a use case, showing that the actor uses the functionality of the use case.
* **Include**: A dashed arrow from one use case to another, indicating that the behavior of the second use case is included in the first.
* **Extend**: A dashed arrow from one use case to another, showing that the second use case extends the behavior of the first, usually under certain conditions.
* **Generalization**: An arrow pointing from a specialized actor or use case to a more general actor or use case, indicating inheritance.

**Benefits of Use Case Diagrams:**

* **Clarity**: Provides a clear and simple visual representation of system functionality and interactions.
* **Communication**: Enhances communication between stakeholders, developers, and users by clearly illustrating requirements.
* **Organization**: Helps in organizing system requirements and identifying user interactions.
* **Scope Definition**: Defines the boundaries of the system and helps in understanding what is inside and outside the scope of the system.
* **Use-case on bill payment on paytm:**

**Actors:**

* **User:** The individual using the Paytm app to pay a utility bill.
* **Paytm System:** The backend system of Paytm facilitating the transaction.
* **Utility Service Provider:** The entity providing the utility service (e.g., electricity company, gas bill, credit card bill).

**Use Cases:**

* **Login:** The user logs into their Paytm account.
* **Select Utility Bill Payment:** The user navigates to the utility bill payment section.
* **Enter Utility Details:** The user inputs necessary details like customer account number and bill amount.
* **Verify Utility Details:** Paytm system verifies the entered details with the utility service provider.
* **Choose Payment Method:** The user selects a payment method (e.g., Paytm wallet, credit card, bank account).
* **Make Payment:** The user confirms and makes the payment.
* **Receive Confirmation:** The user receives a confirmation of the payment from Paytm.
* **Notify Utility Provider:** Paytm notifies the utility service provider about the payment.
* **Use Case Diagram:**

