**MODULE: 1**

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

**Software:**

* Software refers to a set of instructions or programs that enable a computer to perform specific tasks. It includes all the various programs, applications, and data that a computer uses to carry out operations. Software can be broadly categorized into two main types:
* **System Software:** This type of software is responsible for managing and controlling computer hardware so that application software can function. Examples include operating systems, device drivers, and utilities.
* **Application Software:** This type of software is designed to perform specific tasks or applications for the user. Examples include word processors, web browsers, games, and business applications

**Software Engineering:**

* Software engineering is a systematic approach to the design, development, testing, and maintenance of software. It involves applying engineering principles to software development, with the goal of creating high-quality, reliable, and maintainable software systems.

1. **Explain types of software**

**System Software:**

* **Operating Systems (OS):** Manage hardware resources and provide services for computer programs. Examples include Windows, macOS, Linux, and Android.
* **Device Drivers:** Facilitate communication between the operating system and hardware devices such as printers, graphics cards, and storage devices.

**Application Software:**

* **Productivity Software:** Helps users perform tasks such as document creation, spreadsheet analysis, and presentation design. Examples include Microsoft Office (Word, Excel, PowerPoint), Google Workspace, and LibreOffice.
* **Web Browsers:** Allow users to access and navigate the internet. Examples include Google Chrome, Mozilla Firefox, and Microsoft Edge.
* **Media Players:** Play audio and video files. Examples include VLC Media Player, Windows Media Player, and iTunes.
* **Graphics Software:** Used for creating and editing images and graphics. Examples include Adobe Photoshop, GIMP, and CorelDRAW.
* **Games:** Provide entertainment through interactive experiences. Examples include Fortnite, Minecraft, and Call of Duty.

**Development Software:**

* **Integrated Development Environments (IDEs):** Assist developers in writing, testing, and debugging code. Examples include Visual Studio, Eclipse, and IntelliJ IDEA.
* **Version Control Systems:** Manage changes to source code over time. Examples include Git, SVN, and Mercurial.

**Utilities:**

* **Antivirus Software:** Protects against malware and viruses. Examples include Norton, McAfee, and Windows Defender.
* **File Management Tools:** Help organize and manipulate files and directories. Examples include File Explorer (Windows), Finder (macOS), and Nautilus (Linux).

**Network Software:**

* **Firewalls:** Monitor and control incoming and outgoing network traffic. Examples include Windows Firewall and iptables.
* **Web Servers:** Serve web pages to users. Examples include Apache, Nginx, and Microsoft IIS.
* **FTP Clients:** Enable file transfer between computers on a network. Examples include FileZilla and WinSCP.

**Embedded Software:**

* **Firmware:** Software embedded in hardware devices. Examples include the firmware in routers, printers, and IoT devices.

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

**SDLC (Software Development Life Cycle):**

* SDLC is a systematic process for planning, creating, testing, deploying, and maintaining information systems. It is a structured approach to software development that aims to produce high-quality software that meets or exceeds customer expectations. The SDLC consists of several phases, each with its own set of activities and deliverables. The typical phases of the SDLC are:

**1. Planning:**

* **Objectives:** Define the project scope, goals, and constraints. Identify risks and create a plan for project management.
* **Activities:** Conduct feasibility studies, define project requirements, estimate resources, and create a project schedule.

**2. Requirements Gathering and Analysis:**

* **Objectives:** Understand and document the needs of end-users and stakeholders.
* **Activities:** Gather, analyze, and document detailed requirements. Create use cases, functional specifications, and system requirements.

**3. Design:**

* **Objectives:** Create a blueprint or design for the software based on the requirements.
* **Activities:** Design the system architecture, user interface, database structure, and overall system functionality. Produce detailed technical specifications.

**4. Implementation (Coding):**

* **Objectives:** Transform the design into actual code.
* **Activities:** Write and test the source code according to the design specifications. This phase involves programming, unit testing, and code reviews.

**5. Testing:**

* **Objectives**: Evaluate the software to identify and fix defects.
* **Activities:** Conduct various testing types, including unit testing, integration testing, system testing, and acceptance testing. Verify that the software meets the specified requirements and is free of critical issues.

**6. Deployment:**

* **Objectives:** Release the software to end-users.
* **Activities:** Plan and execute the deployment process. This may involve data migration, user training, and transitioning the software to its operational environment.

**7. Maintenance:**

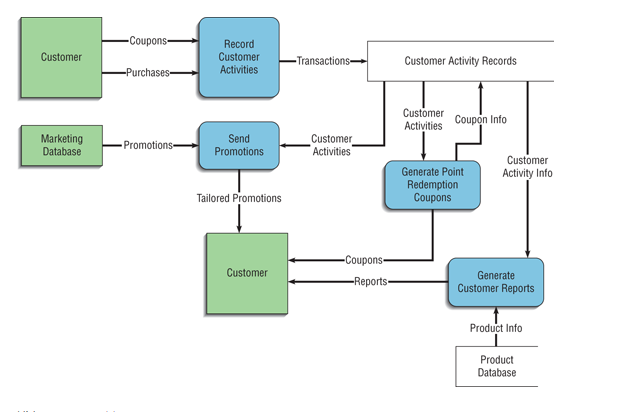
* **Objectives:** Ensure the continued functionality and improvement of the software.
* **Activities:** Address and fix any issues discovered post-deployment. Make updates, enhancements, and modifications based on user feedback and changing requirements.

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

**DFD (Data Flow Diagram):**

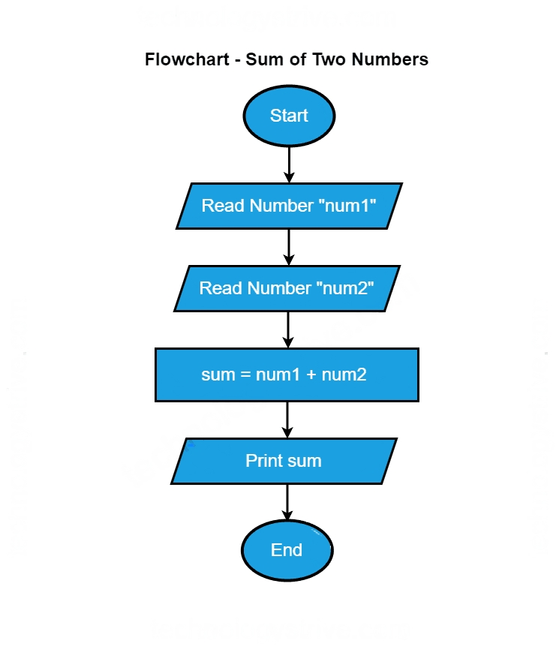
* A Data Flow Diagram (DFD) is a graphical representation of the flow of data within a system. It is a visual tool used to illustrate the movement of data between processes, data stores, and external entities in a system. DFDs are widely used in system analysis and design to model and document how data moves through an information system.

**Key elements in a Data Flow Diagram include**:

* **Processes:** Represent activities or transformations that take place in the system. Processes are typically depicted as circles or ovals.
* **Data Flows:** Represent the movement of data between processes, data stores, and external entities. Data flows are represented by arrows.
* **Data Stores:** Represent repositories of data within the system. These can be databases, file systems, or other data storage mechanisms. Data stores are typically represented as rectangles.
* **External Entities:** Represent sources or destinations of data that interact with the system but are outside of its boundary. External entities are represented as squares or rectangles.
* DFDs come in different levels of detail, allowing analysts to progressively decompose a system into more detailed diagrams. The levels are typically classified as:
* **Context Level DFD:** Provides an overview of the entire system, showing external entities and the major processes.
* **Level 0 DFD:** Represents the main processes and data flows within the system. It provides a high-level view of the system's functionality.
* **Lower-level DFDs:** Decompose the processes of the Level 0 DFD into more detailed sub-processes.
* **DFDs help in:**
* **Understanding System Functionality:** They provide a clear and concise visualization of how data moves through a system.
* **Identifying Data Sources and Destinations:** External entities in DFDs help identify where data originates and where it is consumed.
* **Analyzing System Boundaries:** DFDs define the scope and boundaries of a system by showing what is inside and outside of it.

**5.What is Flow chart? Create a flowchart to make addition of two numbers**

* A flowchart is a graphical representation of a process or system, using different shapes and arrows to depict the steps and the flow of data or control within that process. Flowcharts are commonly used in various fields, including software development, business processes, education, and engineering, to visually represent complex systems and their workflows.
* **Here are some key elements and symbols used in a flowchart:**
* **Start/End Symbol:** Represents the beginning or end of a process. It is usually depicted as an oval or rounded rectangle.
* **Process Symbol:** Represents a specific operation or activity within the process. It is typically depicted as a rectangle.
* **Decision Symbol:** Represents a decision point in the process where the flow can take different paths based on a condition. It is usually depicted as a diamond shape.
* **Input/Output Symbol:** Represents data input or output points in the process. It is typically depicted as a parallelogram.
* **Flow Arrows:** Arrows indicate the flow of control or data between different symbols, showing the sequence in which the steps of the process occur



**6.What is Use case Diagram? Create a use-case on bill payment on paytm.**

* A Use Case Diagram is a type of Unified Modeling Language (UML) diagram that visually represents the interactions between various actors (users or external systems) and a system. Use case diagrams are used in software engineering to capture and define the functional requirements of a system from the perspective of its users. They help to illustrate the different ways that users can interact with a system and the various scenarios or use cases in which the system responds to those interactions.
* **Key components of a Use Case Diagram include:**
* **Actors:** Actors represent the external entities (users, systems, or other entities) that interact with the system. They are typically depicted as stick figures. Actors can be primary actors, who initiate a use case, or secondary actors, who assist in the completion of a use case.
* **Use Cases:** A use case represents a specific functionality or behavior of the system from the perspective of an actor. It describes a sequence of interactions between the actor and the system. Use cases are depicted as ovals.
* **Associations:** Lines or arrows connecting actors and use cases represent associations. An association between an actor and a use case indicates that the actor interacts with that use case.
* **System Boundary:** A boundary box or rectangle encloses the use cases and actors, representing the scope of the system under consideration.

