1) What is software? What is software engineering?

Software:

Software refers to a collection of instructions or programs that tell a computer how to perform specific tasks. It includes everything from operating systems and applications to games and utilities. Essentially, software comprises code written in programming languages that is executed on hardware (like computers or mobile devices) to produce the desired functionality.

Software Engineering:

Software engineering is the systematic approach to the design, development, maintenance, and evolution of software. It involves applying engineering principles and practices to software development, aiming to ensure the reliability, efficiency, and maintainability of software systems. Key aspects of software engineering include requirements gathering, design, coding, testing, deployment, and maintenance, all while adhering to quality standards and project management principles.

2) Explain types of software.

1) System Software:

- **Operating Systems (OS):** Examples include Windows, macOS, Linux, Android, iOS. They manage computer hardware and provide a platform for running applications.

- **Device Drivers:** Software that enables communication between hardware devices and the operating system.

- **Utilities:** Tools that perform specific tasks like disk cleanup, antivirus scanning, file management, etc.

2) Application Software:

- ➔ **Productivity Software:** Tools used for creating documents, presentations, spreadsheets, etc. (e.g., Microsoft Office, Google Workspace).

- ➔ **Database Software:** Manages data and allows users to store, organize, and retrieve information (e.g., Oracle, MySQL, Microsoft SQL Server).

- ➔ **Graphics Software:** Used for creating, editing, and manipulating visual content (e.g., Adobe Photoshop, Illustrator).

- ➔ **Media Software:** Includes media players, editors, and management tools (e.g., VLC Media Player, Adobe Premiere).

- ➔ **Communication Software:** Enables communication between users or devices (e.g., email clients, instant messaging apps).

- ➔ **Version Control Systems:** Manage changes to source code over time (e.g., Git, SVN).

3) Development Software:

- **Programming Languages:** Tools and environments used to write, edit, and compile code (e.g., Python, Java, Visual Studio).

- **Integrated Development Environments (IDEs):** Software suites that provide comprehensive tools for software development (e.g., IntelliJ IDEA, Visual Studio Code).

- **Version Control Systems:** Manage changes to source code over time (e.g., Git, SVN).

4) Embedded Software:

- Software embedded into hardware devices to control their functionalities (e.g., firmware in printers, routers, IoT devices).

5) Enterprise Software:

- Software used by organizations to manage their operations (e.g., ERP systems, CRM systems, HR management software).

6) Web-based Software:

- Applications accessed via a web browser over the internet (e.g., web applications, online banking systems).

7) Open Source Software:

- Software whose source code is freely available for anyone to use, modify, and distribute (e.g., Linux operating system, Apache web server).

8) Closed Source Software (Proprietary Software):

- Software whose source code is not publicly available, and usually requires a license or purchase to use (e.g., Microsoft Office, Adobe Photoshop).

3) What is SDLC ? explain each phase of SDLC.

SDLC stands for Software Development Life Cycle. It is a structured process that outlines the steps or phases involved in developing software. Each phase in the SDLC aims to ensure the quality and reliability of the software product being developed, while also managing costs and time effectively. The typical phases of the SDLC are as follows:

1. Requirements gathering and analysis: This phase involves gathering information about the software requirements from stakeholders, such as customers, end-users, and business analysts.

2. 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.

3. Implementation or coding: The design is then implemented in code, usually in several iterations, and this phase is also called Development.

things you need to know about this phase:

- This is the longest phase in the SDLC model.
- This phase consists of Front end + Middleware + Back-end.
- **In front-end:** Development of coding is done even SEO settings are done.
- **In Middleware:** They connect both the front end and back end.
- **In the back-end:** A database is created.

4. Testing: The software is thoroughly tested to ensure that it meets the requirements and works correctly.

5. Deployment: After successful testing, The software is deployed to a production environment and made available to end-users.

6. Maintenance: This phase includes ongoing support, bug fixes, and updates to the software.

4) What is DFD? Create a DFD diagram on flipkart.

DFD stands for Data Flow Diagram. It is a graphical representation of the flow of data through a system or process. DFDs are commonly used in software engineering and business analysis to visualize and describe the flow of data within a system, regardless of its physical or software components.

Components of a DFD:

1. **Processes:** Represented by circles or ovals, processes in a DFD denote activities or transformations that are performed on the data. Each process typically takes inputs, processes them, and produces outputs.
2. **Data Flows:** Represented by arrows, data flows in a DFD show the movement of data between processes, data stores, and external entities. They indicate the path that data takes throughout the system.
3. **Data Stores:** Represented by rectangles, data stores in a DFD represent where data is held within the system. They can represent databases, files, or any other storage medium where data is stored for later use.
4. **External Entities:** Represented by squares, external entities in a DFD represent sources or destinations of data outside the system being analyzed. These can be users, other systems, or external data sources.

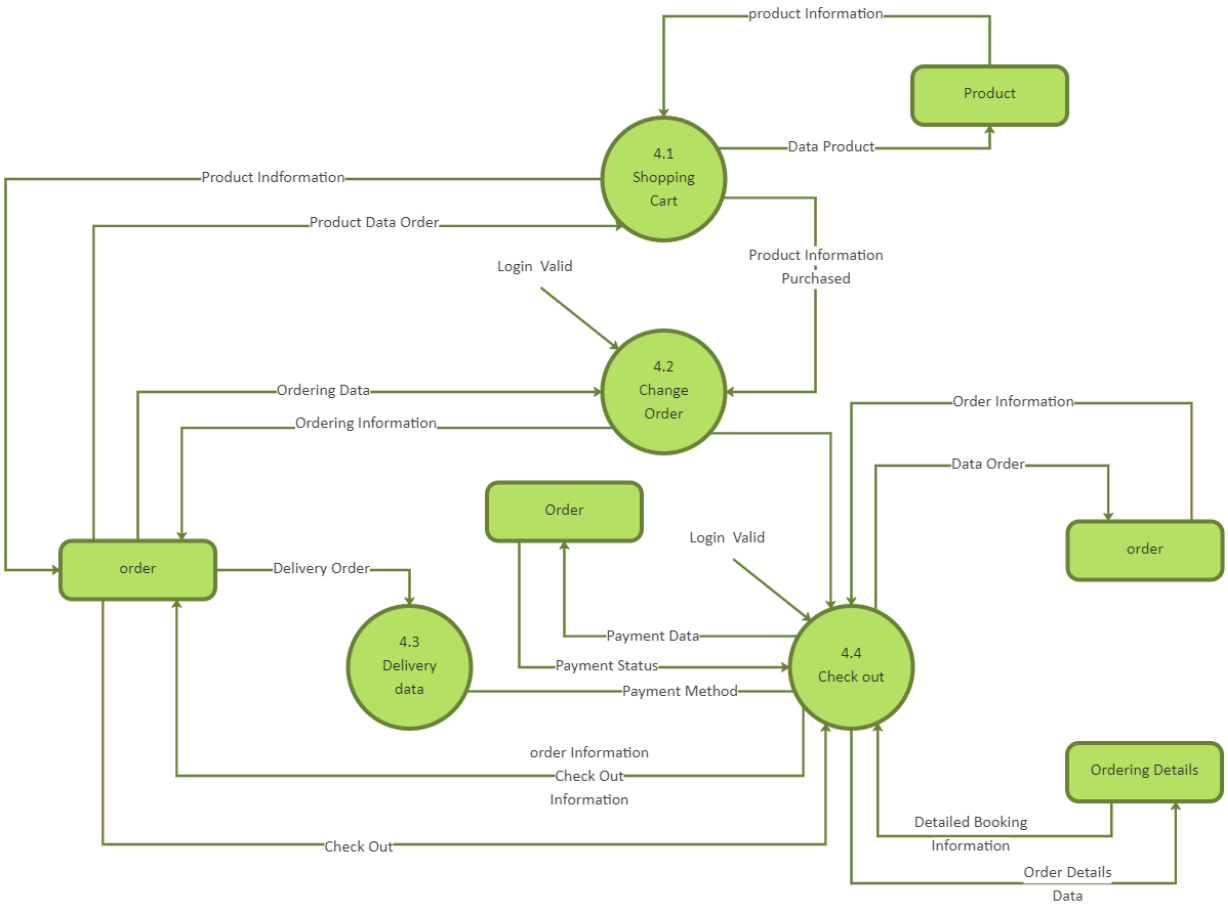
Types of DFDs:

- **Context Diagram (Level 0 DFD):** Provides an overview of the entire system, showing the interactions between the system and external entities. It only shows one process representing the entire system.
- **Level 1 DFD:** Decomposes the context diagram into more detailed processes, data flows, data stores, and external entities. It provides a more detailed view of the system's functionalities.
- **Level 2 DFD (and higher levels):** Further decomposes processes from Level 1 into more detailed subprocesses, showing more intricate details of how data flows through the system.

Benefits of DFDs:

- **Clarity and Understanding:** DFDs provide a clear and visual representation of how data moves through a system, making it easier for stakeholders to understand the system's workings.
- **Analysis and Design:** They help in analyzing the system requirements and designing new systems or improving existing ones by identifying inefficiencies or areas for improvement.
- **Communication:** DFDs serve as a communication tool between stakeholders, helping to ensure that everyone has a shared understanding of the system and its data flows.

DFD diagram on flipkart



5) What is a flow chart ? Create a flowchart to make addition of 2 numbers.

A flowchart is a visual representation of a process or algorithm, typically using symbols connected with arrows to show the sequence of steps. It is a graphical tool that helps in understanding the flow of a complex system or process, breaking it down into simpler components and illustrating their relationships and dependencies.

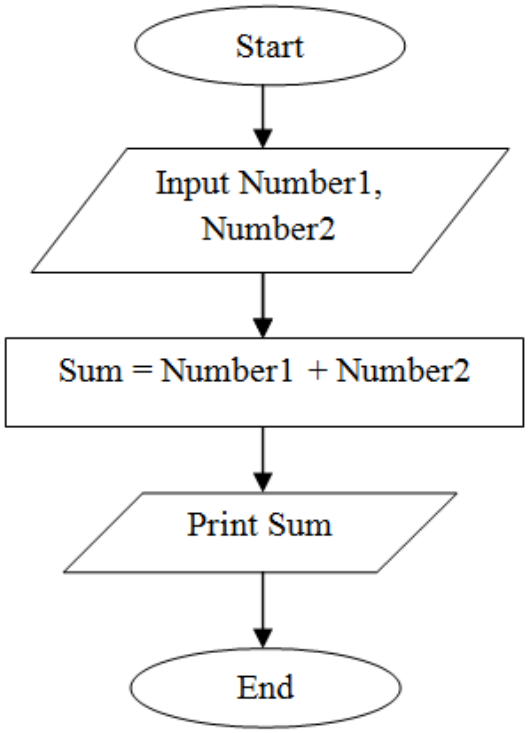
Components of a Flowchart:

1. **Terminal (Start/End):**
    - Denotes the beginning and end of a process. Usually represented by an oval shape with the word "Start" or "End" inside.
2. **Process:**
    - Represents a specific action or operation within the process. Typically depicted as a rectangle with a concise description of the action.
3. **Decision (Conditional):**
    - Represents a point in the process where a decision must be made. Usually shown as a diamond shape with a question inside, and arrows branching out to different paths based on the decision.
4. **Input/Output:**
    - Represents where data enters or exits the process. Shown as a parallelogram shape, often labeled with the input/output data or action.
5. **Flow Arrows:**
    - Arrows connecting the symbols indicate the direction of flow, showing the sequence of actions or decisions in the process.

Uses of Flowcharts:

- **Process Documentation:** Flowcharts are used to document and visualize business processes, workflows, and procedures. They help in understanding how tasks are performed and identifying potential improvements.
- **Algorithm Design:** In software development and computer science, flowcharts are used to design and describe algorithms. They provide a step-by-step representation of logical sequences and decision points in code.
- **Problem Solving:** Flowcharts are used in problem-solving scenarios to analyze and map out possible solutions and decision-making processes.
- **Training and Education:** Flowcharts are used as educational tools to teach complex processes or concepts in a structured and visual manner.

Flowchart to make addition of 2



6) What is a use-case diagram ? Create a use-case diagram on bill payment on paytm.

A use-case diagram is a type of behavioral diagram in Unified Modeling Language (UML) that depicts the interactions between users (actors) and a system, showcasing the various ways users interact with the system to achieve specific goals or tasks. Use-case diagrams are widely used in software engineering to capture and communicate the functional requirements of a system from a user's perspective.

Components of a Use-Case Diagram:

1. **Actors:**
    - Represented by stick figures, actors in a use-case diagram are external entities (users or other systems) that interact with the system being modeled. Each actor typically corresponds to a role that interacts with the system to achieve specific goals.
2. **Use Cases:**
    - Represented by ovals, use cases describe specific functionalities or tasks that the system provides to its users. Each use case represents a discrete unit of work or a specific user goal that the system can accomplish.
3. **Relationships:**
    - **Association:** A solid line connecting an actor to a use case indicates that the actor interacts with the system to perform that use case.
    - **Include Relationship:** A dashed line with an arrowhead from one use case to another indicates that one use case includes the functionality of another use case. It signifies that the included use case is always part of the base use case.

    - **Extend Relationship:** A dashed line with an arrowhead from one use case to another indicates that one use case can extend another use case under certain conditions. It signifies optional or exceptional behavior.

Benefits of Use-Case Diagrams:

- **Requirements Elicitation:** Use-case diagrams help in capturing functional requirements from a user's perspective, ensuring that all user goals and tasks are identified.
- **Visualization:** They provide a clear visual representation of system functionality and user interactions, aiding in communication between stakeholders, designers, and developers.
- **System Design:** Use-case diagrams serve as a foundation for designing system architecture and defining interfaces and interactions between subsystems.
- **Scope Definition:** They help in defining the scope and boundaries of the system, focusing on what the system should do from a user's point of view.

Use-case diagram on bill payment on paytm