Software Engineering Assignment

MODULE: 1 (SDLC)

Q1) What is software? What is software engineering?

Ans: Software is a collection of instructions or programs that tell a computer how to perform specific task. It acts as the interface between the user and the computer hardware.

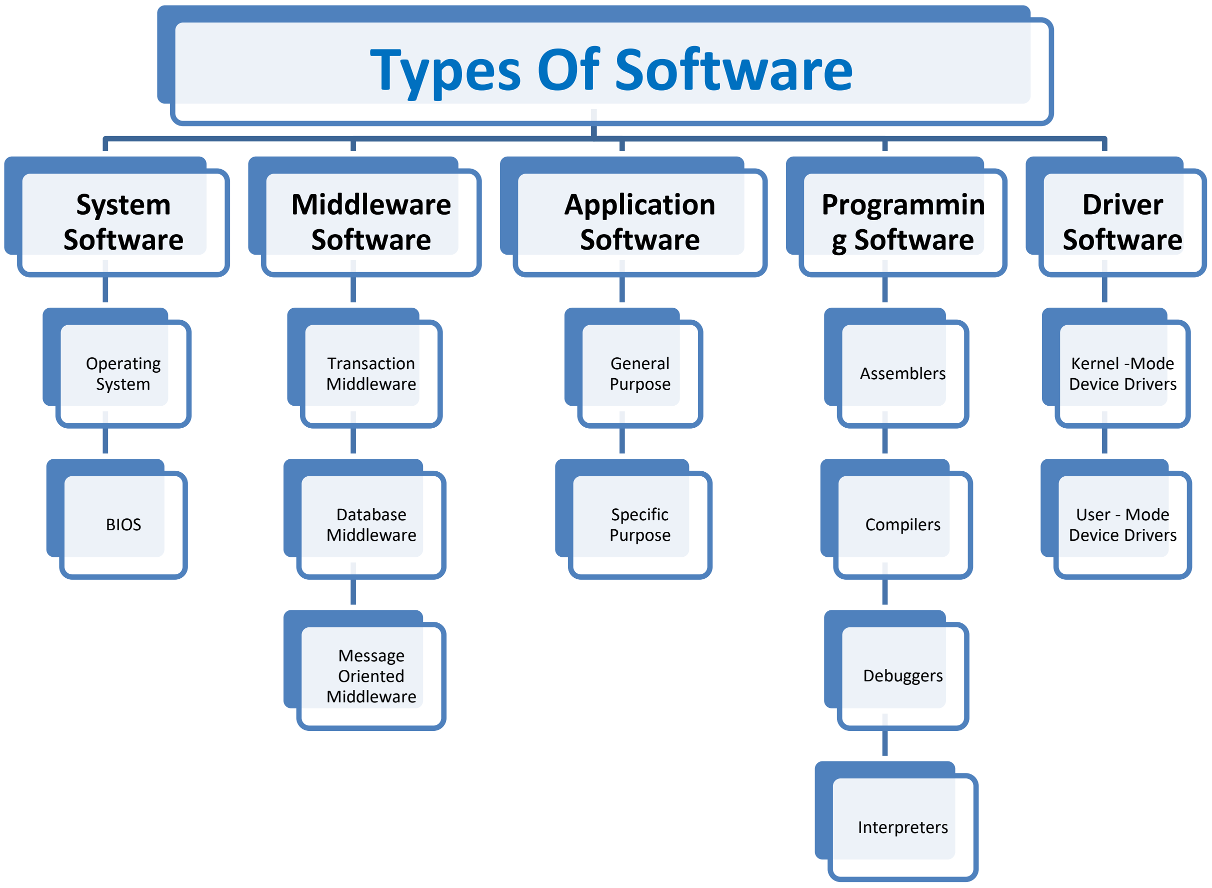
- Software engineering is the systematic application of engineering principles to the development, operation, and maintenance of software. It involves a range of activities, including:
  - Requirements Gathering: Understanding what users need from the software.
  - System Design: Planning the architecture and components of the software.
  - Implementation: Writing the code and building the software.
  - Testing: Ensuring the software functions as intended and is free of defects.
  - Deployment: Releasing the software for use.
  - Maintenance: Updating and improving the software over time.

Q2) Explain types of software.

Ans: There are many type of software categories among them only two are explained below:

- System Software: This type of software manages hardware components and provides a platform for running application software. Examples include operating systems (like Windows, macOS, Linux) and device drivers.
- Application Software: Designed to help users perform specific tasks, application software includes a wide range of programs. Examples are word processors (like Microsoft Word), spreadsheets (like Excel), and web browsers (like Chrome).

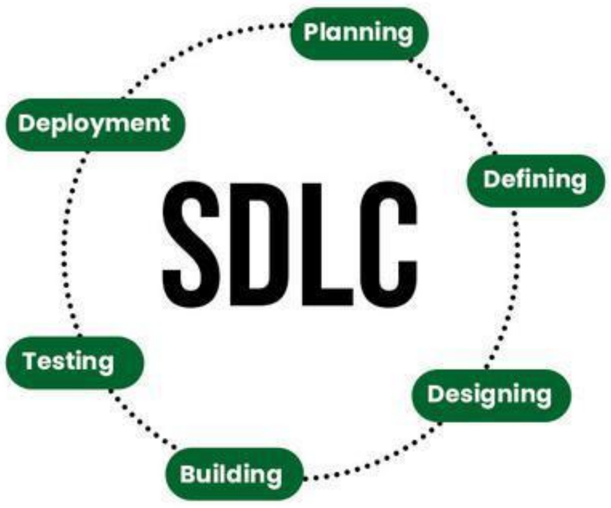
- And other software Categories are given in the Below chart :



Q3) What is SDLC? Explain each phase of SDLC.

Ans: Software development life cycle (SDLC) is a structured process that is used to design, develop, and test good-quality software. SDLC, or software development life cycle, is a methodology that defines the entire procedure of software development step-by-step.

➢ The Software Development Life Cycle (SDLC) is a structured process that outlines the stages involved in software development. Here are the typical phases:



Planning:

- **Objective**: Define the project's scope, goals, and feasibility.
- **Activities**: Conduct feasibility studies, gather requirements, and outline project timelines and resources.

Requirements Analysis:

- **Objective**: Gather and document the needs and requirements of the stakeholders.
- **Activities**: Engage with users and stakeholders, create detailed specifications, and prioritize requirements.

Design:

- **Objective**: Create the architecture of the software.
- **Activities**: Develop high-level and detailed designs, including user interfaces, system interfaces, and database structures.

Implementation (or Coding):

- **Objective**: Convert design specifications into actual code.
- **Activities**: Developers write code, conduct unit tests, and integrate components.

Testing:

- **Objective**: Identify and fix defects before the software is deployed.
- **Activities**: Conduct various tests (unit, integration, system, acceptance) to ensure the software meets requirements and is free of bugs.

Deployment:

- **Objective**: Release the software to users.
- **Activities**: Deploy the software to the production environment, provide training, and support users during the transition.

Maintenance:

- **Objective**: Ensure the software remains functional and relevant.
- **Activities**: Perform updates, bug fixes, and enhancements based on user feedback and changing requirements.

Q4) What is DFD? Create a DFD diagram on Flipkart.

Ans: A Data Flow Diagram (DFD) is a visual representation of the flow of data within a system. It illustrates how data moves from one process to another, showing the relationships between various components. DFDs are useful for understanding how data is processed and can help in system analysis and design.

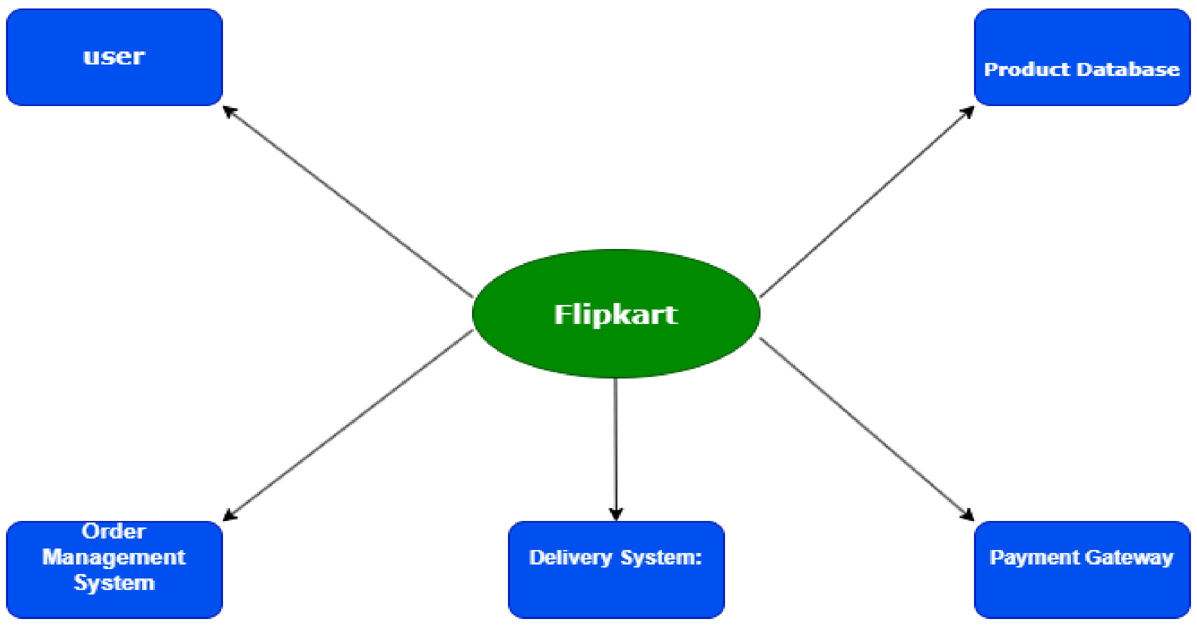
Key Components of DFDs:

1. **Processes**:
    - Represented by circles or ovals.
    - Show transformations or operations that occur on the data.
2. **Data Stores**:
    - Represented by open-ended rectangles.
    - Indicate where data is stored within the system (e.g., databases, files).
3. **Data Flows**:
    - Shown as arrows.
    - Indicate the movement of data between processes, data stores, and external entities.
4. **External Entities**:
    - Represented by squares or rectangles.
    - Show sources or destinations of data outside the system (e.g., users, external systems).

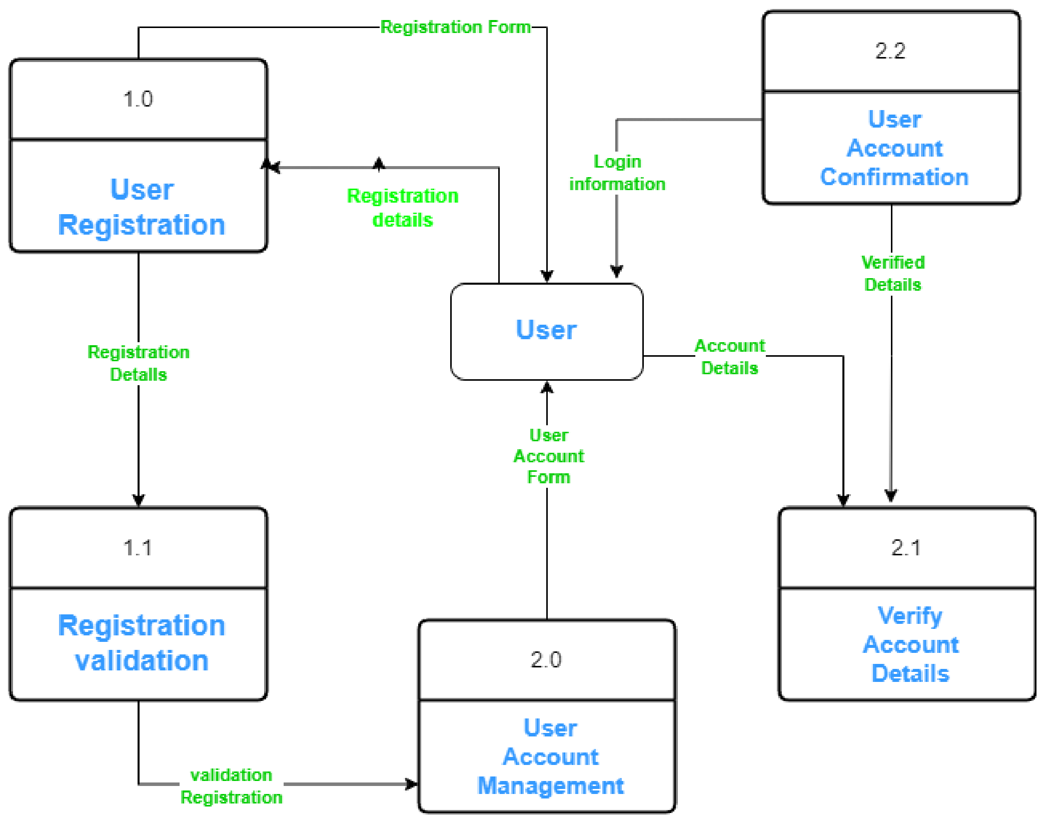
Types of DFDs:

- Context DFD (Level 0 DFD): Provides a high-level overview of the system, showing how it interacts with external entities.
- Level 1 DFD: Breaks down the system into major processes while still maintaining a high-level view.
- Level 2 DFD: Further decomposes processes into sub-processes, providing more detail about data flow and interactions.

➢ Here's a very basic and simplified example of a DFD diagram for Flipkart:



Level 0 DFD [Data Flow Diagram] of Flipkart



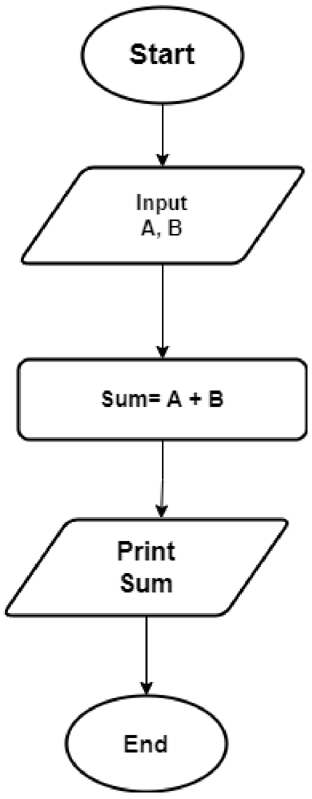
Flipkart Login DFD [Data Flow Diagram]

Q5) What is Flow chart? Create a flowchart to make addition of two numbers.

Ans: A flowchart is a diagrammatic representation used to illustrate the step-by-step sequence of operations, tasks, or decisions involved in a process or system. It uses various symbols, such as arrows, rectangles, diamonds, and ovals, to show the flow of control or data from one step to another.

- This straightforward flowchart shows how to add two numbers :

Print Sum of two Number:



- In this flowchart:

- "Start" is represented by an oval or rounded rectangle.
- "Input" A and B is represented by a parallelogram.
- "Sum" A and B is represented by a rectangle.
- "print sum" is represented by a parallelogram.
- "End" is represented by an oval or rounded rectangle.

Arrows connecting each step show how the steps flow into one another.

Q6) What is Use case Diagram? Create a use-case on bill payment on paytm

Ans: A Use Case Diagram is a type of behavioral diagram used in software and systems engineering to visualize the functional requirements of a system. It captures the interactions between users (actors) and the system, focusing on what the system should do without going into implementation details.

Key Components of a Use Case Diagram:

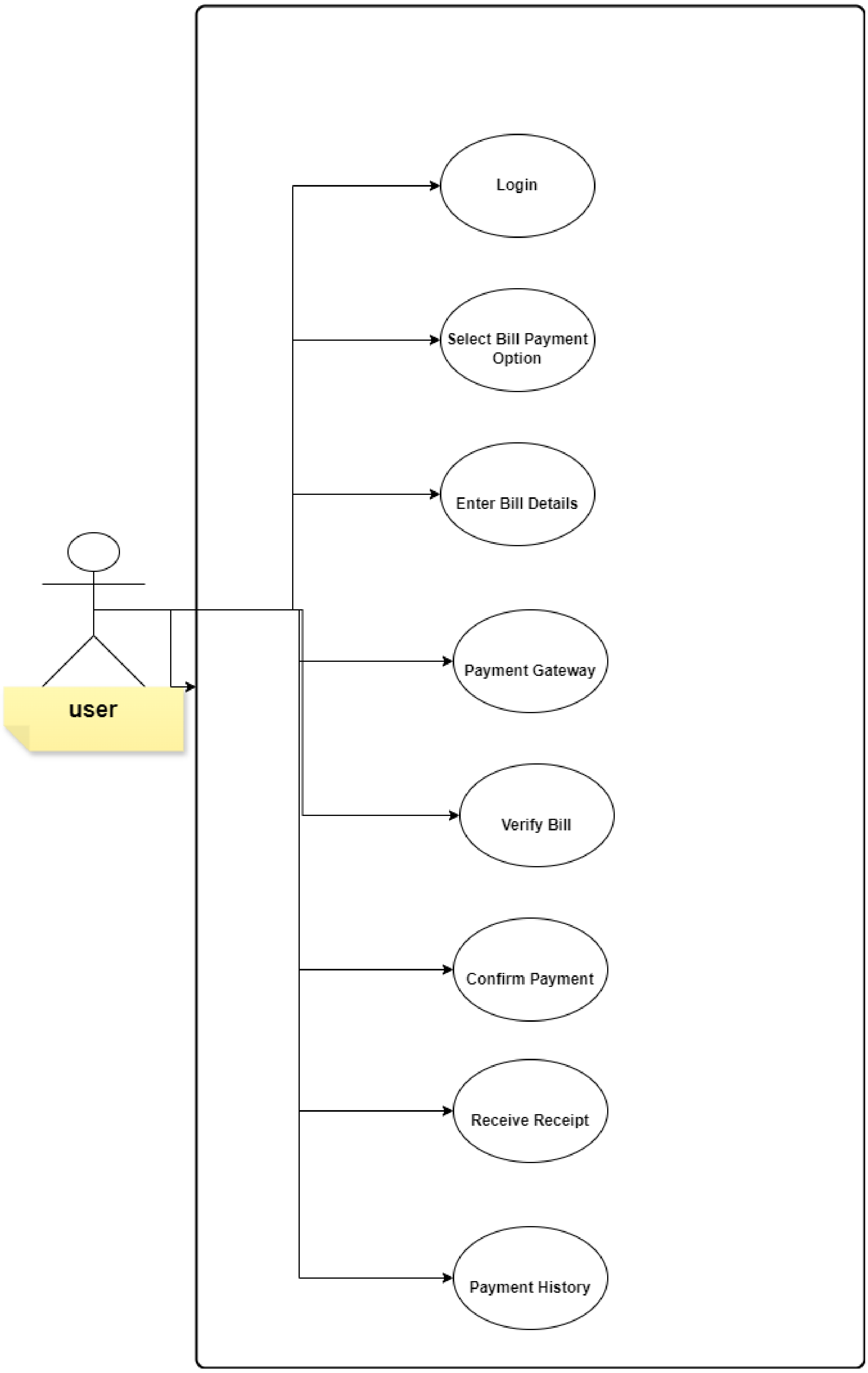
1. Actors:
   - Represent external entities (users, other systems, or hardware) that interact with the system.
   - Actors can be human users or other systems.
   - Example: A customer, an admin, or a payment gateway.

2. Use Cases:
    - Represent specific functions or actions that the system performs to achieve certain goals for the actors.
    - Use cases are typically described in simple verb-noun format.
    - Example: Place Order, Login, Process Payment.
3. System Boundary:
    - This is a rectangle that encapsulates all the use cases and shows the scope of the system being designed. Actors are placed outside the boundary.
4. Relationships:
    - Association: Connects actors to use cases, showing the interaction between them.
    - Include: A use case that always happens as part of another use case (common functionality).
    - Extend: A use case that happens optionally, under certain conditions.
    - Generalization: Indicates inheritance between actors or use cases (e.g., one actor inherits the roles of another).

Flow of Use Case:

1. Login/Sign Up: User logs in to their Paytm account or signs up if they don't have an account.
2. Select Bill Payment: The user selects the "Bill Payment" option from the menu.
3. Enter Bill Details: The user inputs the required bill details like bill number, account number, or biller.
4. Verify Bill: Paytm sends the information to the biller and retrieves the amount due.
5. Make Payment: The user selects a preferred payment method (e.g., UPI, Credit Card) and confirms the payment.
6. Confirm Payment: The payment gateway processes the transaction, and the system updates the user on the success or failure of the payment.
7. Receive Receipt: After successful payment, the user receives a digital receipt, and the biller is updated.
8. Payment History: The user can view previously paid bills in the system.

User Case Diagram


user
Login
Select Bill Payment Option
Enter Bill Details
Payment Gateway
Verify Bill
Confirm Payment
Receive Receipt
Payment History