# MODULE 1 :OVERVIEW OF IT industry(SDLC)

## 1. What is software? What is software engineering?

- Software is a set of programs, Set of Instruction Used to execute specific task or Function.
- Two Type of software: -
    1. Application Software
    2. System Application
    3. Driver Software
    4. Middleware Software
    5. Programming Software

**1) Application Software :** These are programs designed to perform specific tasks for users. They can be further divided into categories like.

**2) System Software**: Software that provides a platform for running application software and manages hardware resources. Examples include device drivers, utility programs, and software development tools.

**3) Driver Software**: A driver in software provides a programming interface to control and manage specific lower-level interfaces that are often linked to a specific type of hardware, or other low-level service. In the case of hardware, the specific subclass of drivers controlling physical or virtual hardware devices are known as device drivers.

**4) Middle ware Software**: Software that acts as an intermediary between different software applications or components. It helps facilitate communication and data exchange. Examples include web servers, application servers, and messaging

middleware.

**5) Programming Software**: Programming software is a software which helps the programmer in developing other software. Compilers, assemblers, debuggers, interpreters etc. are examples of programming software. Integrated development environments (IDEs) are combinations of all these software.

## ***Software engienring***

- Software Engineering is the process of designing, developing, testing, and maintaining software. It is a systematic and disciplined approach to software development that aims to create high-quality, reliable, and maintainable software. Software engineering includes a variety of techniques, tools, and methodologies, including requirements analysis, design, testing, and maintenance.

### Key Principles of Software Engineering

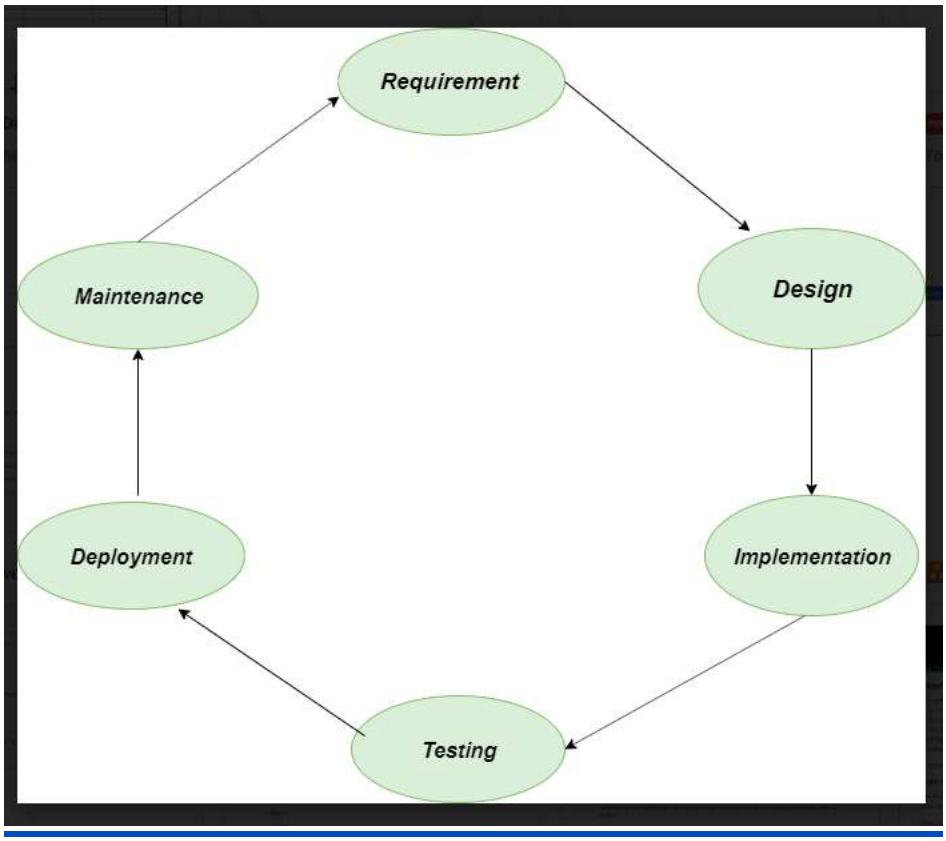
- Modularity: Breaking the software into smaller, reusable components that can be developed and tested independently.
- Abstraction: Hiding the implementation details of a component and exposing only the necessary functionality to other parts of the software.
- Encapsulation: Wrapping up the data and functions of an object into a single unit, and protecting the internal state of an object from external modifications.
- Reusability: Creating components that can be used in multiple projects, which can save time and resources.
- Maintenance: Regularly updating and improving the software to fix bugs, add new features, and address security vulnerabilities.
- Testing: Verifying that the software meets its requirements and is free of bugs.
- Design Patterns: Solving recurring problems in software design by providing templates for solving them.
- Agile methodologies: Using iterative and incremental development processes that focus on customer satisfaction, rapid delivery, and flexibility.
- Continuous Integration & Deployment: Continuously integrating the code changes and

deploying them into the production environment.

# SDLC

- SOFTWARE DEVELOPMENT LIFE CYCLE
- 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.

## Phases of SDLC




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 as Development.

things you need to know about this phase:

This is the longest phase in 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.

## DFD(dataflow diagram)

- DFD is the abbreviation for Data Flow Diagram. The flow of data of a system or a process is represented by DFD. It also gives insight into the inputs and outputs of each entity and the process itself. DFD does not have 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 analyzing existing as well as proposed system.
- It provides an overview of
    - What data is system processes.

- What transformation are performed.
- What data are stored.
- What results are produced , etc.

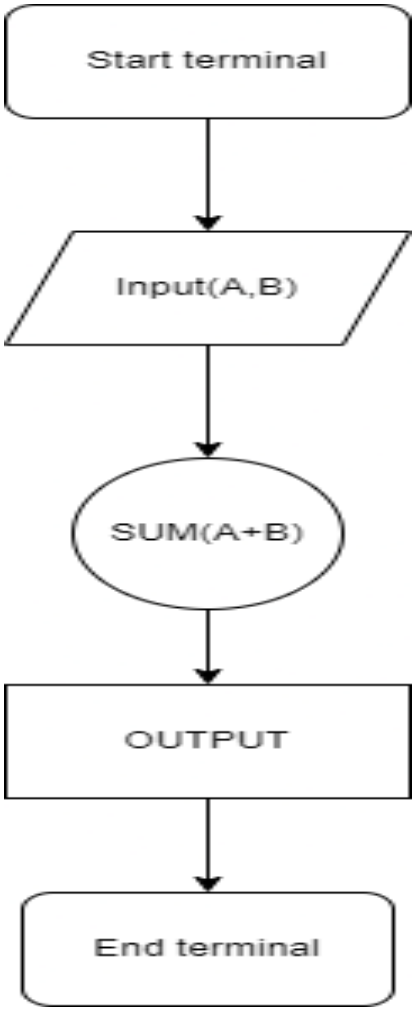
## DFD for Flipcart

https://drive.google.com/drive/folders/1qBQDwurF6mcv94ESxDDmKzbd4ZrzRqoG?usp=drive_link

## FLOWCHART

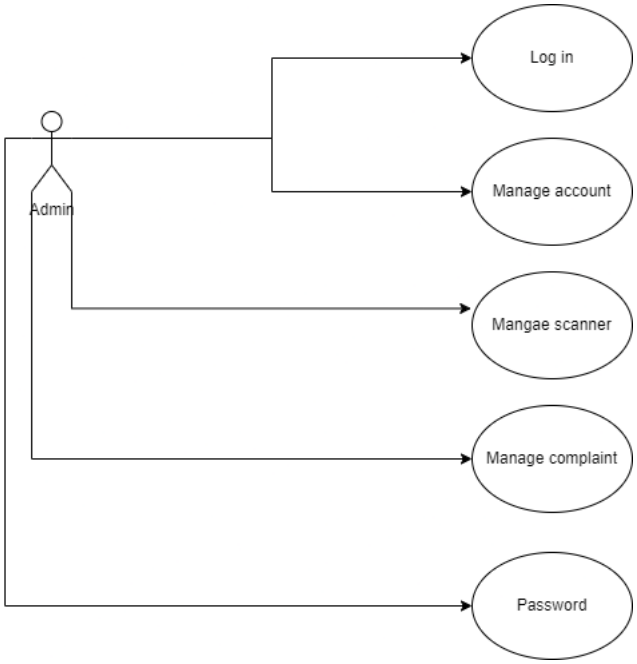
- A flowchart is a type of diagram that represents a workflow or process. A flowchart can also be defined as a diagrammatic representation of an algorithm, a step-by-step approach to solving a task.

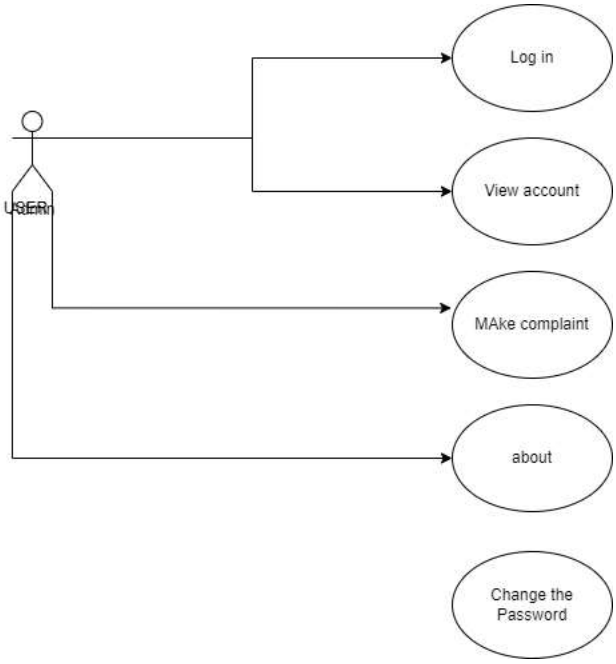
## EXAMPLE of flowchart




## USECASE diagram

A use case diagram is a graphical depiction of a user's possible interactions with a system. A use case diagram shows various use cases and different types of users the system has and will often be accompanied by other types of diagrams as well. The use cases are represented by either circles or ellipses.


Log in
Admin
Manage account
Mangae scanner
Manage complaint
Password


Log in
USER
View account
MAke complaint
about
Change the Password