# Q-1 What is software? What is software engineering?

Software engineering has two parts: software and engineering.

**Software** is a collection of codes, documents, and triggers that does a specific job and fills a specific requirement.

**Engineering** is the development of products using best practices, principles, and methods.

Software engineering is a concept in and of itself, but to better understand it, you need to know what each part of the term means before you can fully understand how they operate together. It can be difficult to understand, even though it does seem straightforward. That is because the pieces are more complicated than many believe - and working with software engineering for an application is difficult and time-consuming.

## Q-2 Explain types of software.

### Application Software

This is the most common type of computer software, and can be defined as end-user programs that help you perform tasks or achieve a desired outcome. The end-user is the person who is actually using a product or program. (They are the one for whom the “end result” is designed.) Some examples of application software include internet browsers, a CRM tool like Hubspot, a photo-editing software like Adobe or Lightroom, or a word processing application like Microsoft Word. Application software is installed on a computer or mobile device based upon a user’s need. Because this is the most common type of software, there are many options available and users can choose the one that best fits their needs, budget, and expectations. (For example, anyone wanting to look on the internet could use Chrome, Safari, or even Firefox.)

### System Software

System software helps the user, the computer or mobile device, and an application all work together seamlessly. This makes system software crucial to running any kind of application software as well as the whole computer system.

Think about when your laptop or phone has an update. This is system software in action: there is a tweak made to the system software that helps your computer or phone continue to work well and keep applications running. Apple’s iOS is an example of system software, as is Microsoft Windows. System software is always running in the background of your device, but it is never something you will use directly. In fact, the only time most people remember it’s there is when it is time for an update.

## Programming Software

While application software is designed for end-users, and system software is designed for computers or mobile devices, programming software is for computer programmers and developers who are writing code. These are programs that are used to write, develop, test, and debug other software programs. It's helpful to think of these programs as a translator of sorts: they take programming languages like Laravel, Python, C++, and more and translate them into something a computer or phone will understand.

## Driver Software

This software is often considered to be a type of system software. Driver software operates and controls devices that are plugged into a computer. These drivers make it possible for devices to perform their necessary functions. A very good (and practical) example of this is your printer. When you are first setting up your printer to work with your computer, you have to install software to connect the two so that they communicate and print anything you need.

# Q-3 What is SDLC? Explain each phase of SDLC.

Software Development Life Cycle is the application of standard business practices to building software applications. It's typically divided into six to eight steps: Planning, Requirements, Design, Build, Document, Test, Deploy, Maintain. Some project managers will combine, split, or omit steps, depending on the project's scope. These are the core components recommended for all software development projects.

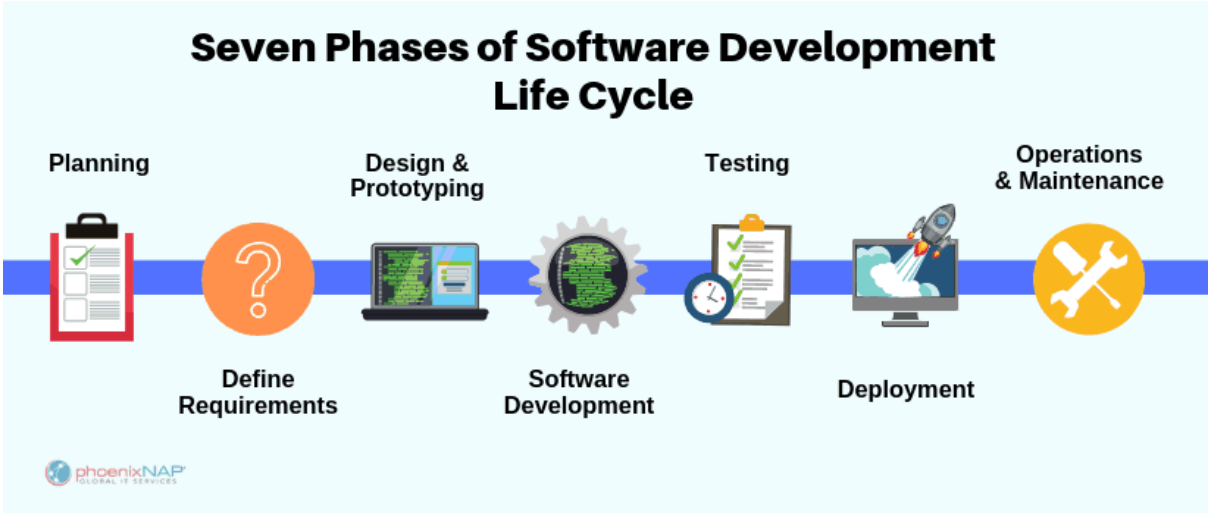
SDLC is a way to measure and improve the development process. It allows a fine-grain analysis of each step of the process. This, in turn, helps companies maximize efficiency at each stage. As computing power increases, it places a higher demand on software and developers. Companies must reduce costs, deliver software faster, and meet or exceed their customers' needs. SDLC helps achieve these goals by identifying inefficiencies and higher costs and fixing them to run smoothly.

## How the Software Development Life Cycle Works

The Software Development Life Cycle simply outlines each task required to put together a software application. This helps to reduce waste and increase the efficiency of the development process. Monitoring also ensures the project stays on track, and continues to be a feasible investment for the company.

Many companies will subdivide these steps into smaller units. Planning might be broken into technology research, marketing research, and a cost-benefit analysis. Other steps can merge with each other. The Testing phase can run concurrently with the Development phase, since developers need to fix errors that occur during testing.

# The Seven Phases of the SDLC




## 1. Planning

In the Planning phase, project leaders evaluate the terms of the project. This includes calculating labor and material costs, creating a timetable with target goals, and creating the project's teams and leadership structure.

Planning can also include feedback from stakeholders. Stakeholders are anyone who stands to benefit from the application. Try to get feedback from potential customers, developers, subject matter experts, and sales reps.

Planning should clearly define the scope and purpose of the application. It plots the course and provisions the team to effectively create the software. It also sets boundaries to help keep the project from expanding or shifting from its original purpose.

## 2. Define Requirements

Defining requirements is considered part of planning to determine what the application is supposed to do and its requirements. For example, a social media application would require the ability to connect with a friend. An inventory program might require a search feature.

Requirements also include defining the resources needed to build the project. For example, a team might develop software to control a custom manufacturing machine. The machine is a requirement in the process.

## 3. Design and Prototyping

The Design phase models the way a software application will work. Some aspects of the design include:

**Architecture –** Specifies programming language, industry practices, overall design, and use of any templates or boilerplate
**User Interface –** Defines the ways customers interact with the software, and how the software responds to input
**Platforms –** Defines the platforms on which the software will run, such as Apple, Android, Windows version, Linux, or even gaming consoles
**Programming –** Not just the programming language, but including methods of solving problems and performing tasks in the application
**Communications –** Defines the methods that the application can communicate with other assets, such as a central server or other instances of the application
**Security –** Defines the measures taken to secure the application, and may include SSL traffic encryption, password protection, and secure storage of user credentials

Prototyping can be a part of the Design phase. A prototype is like one of the early versions of software in the Iterative software development model. It demonstrates a basic idea of how the application looks and works. This "hands-on" design can be shown to stakeholders. Use feedback o improve the application. It's less expensive to change the Prototype phase than to rewrite code to make a change in the Development phase.

## 4. Software Development

This is the actual writing of the program. A small project might be written by a single developer, while a large project might be broken up and worked by several teams. Use an Access Control or Source Code Management application in this phase. These systems help developers track changes to the code. They also help ensure compatibility between different team projects and to make sure target goals are being met.

The coding process includes many other tasks. Many developers need to brush up on skills or work as a team. Finding and fixing errors and glitches is critical. Tasks often hold up the development process, such as waiting for test results or compiling code so an application can run. SDLC can anticipate these delays so that developers can be tasked with other duties.

Software developers appreciate instructions and explanations. Documentation can be a formal process, including wiring a user guide for the application. It can also be informal, like comments in the source code that explain why a developer used a certain procedure. Even companies that strive to create software that's easy and intuitive benefit from the documentation.

Documentation can be a quick guided tour of the application's basic features that display on the first launch. It can be video tutorials for complex tasks. Written documentation like user guides, troubleshooting guides, and FAQ's help users solve problems or technical questions.

## 5. Testing

It’s critical to test an application before making it available to users. Much of the testing can be automated, like security testing. Other testing can only be done in a specific environment – consider creating a simulated production environment for complex deployments. Testing should ensure that each function works correctly. Different parts of the application should also be tested to work seamlessly together—performance test, to reduce any hangs or lags in processing. The testing phase helps reduce the number of bugs and glitches that users encounter. This leads to a higher user satisfaction and a better usage rate.

## 6. Deployment

In the deployment phase, the application is made available to users. Many companies prefer to automate the deployment phase. This can be as simple as a payment portal and download link on the company website. It could also be downloading an application on a smartphone.

Deployment can also be complex. Upgrading a company-wide database to a newly-developed application is one example. Because there are several other systems used by the database, integrating the upgrade can take more time and effort.

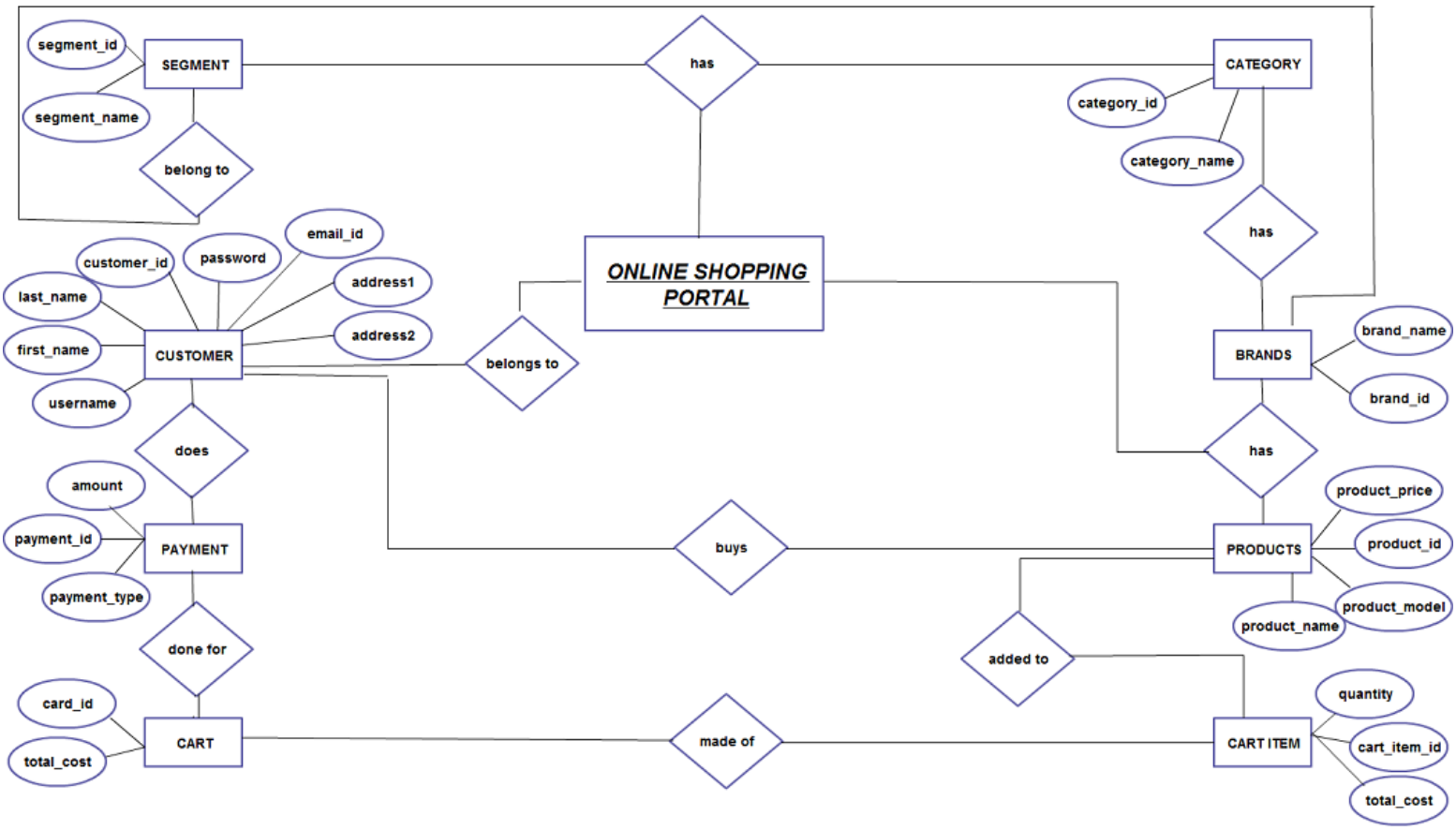
## 7. Operations and Maintenance

At this point, the development cycle is almost finished. The application is done and being used in the field. The Operation and Maintenance phase is still important, though. In this phase, users discover bugs that weren’t found during testing. These errors need to be resolved, which can spawn new development cycles.

In addition to bug fixes, models like Iterative development plan additional features in future releases. For each new release, a new Development Cycle can be launched.

## Q-4 What is DFD? Create a DFD diagram on Flipkart.

**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. Data Flow Diagram can be represented in several ways. The DFD belongs to structured-analysis modeling tools. Data Flow diagrams are very popular because they help us to visualize the major steps and data involved in software-system processes.



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

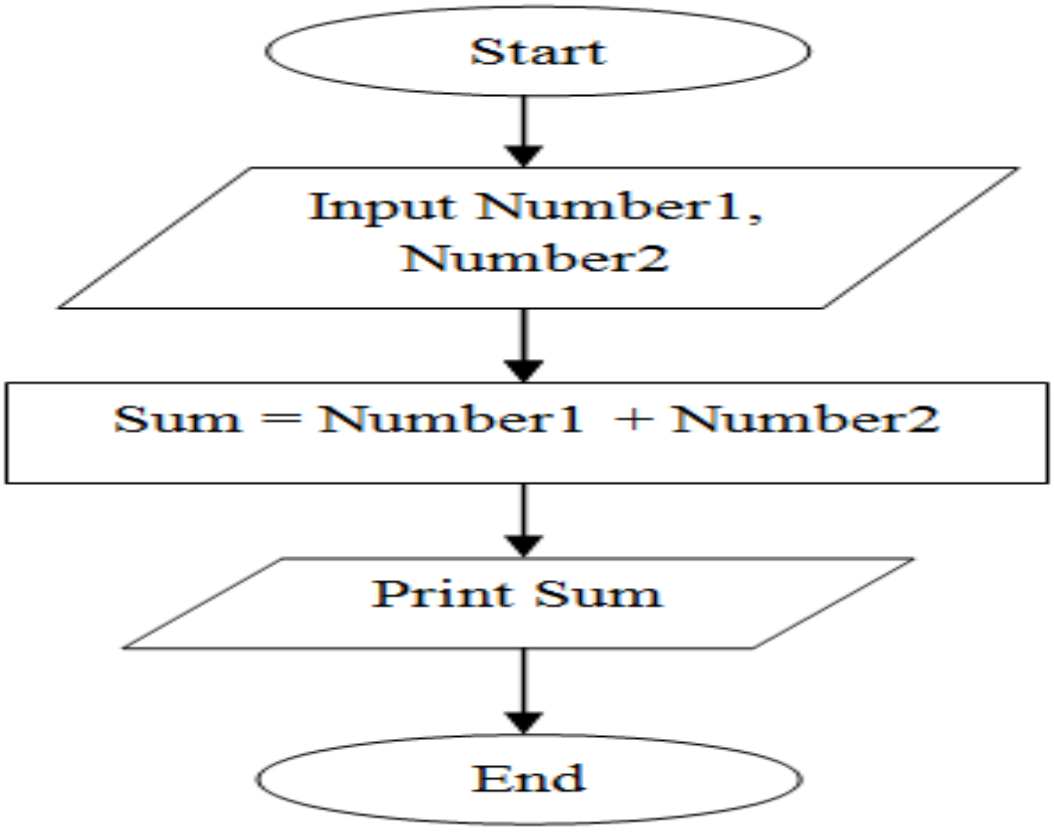
A **flowchart** is a type of diagram that represents a workflow or process. A flowchart can also be defined as a diagrammatic representation of an algorithm, a step-by-step approach to solving a task.

The flowchart shows the steps as boxes of various kinds, and their order by connecting the boxes with arrows. This diagrammatic representation illustrates a solution model to a given problem. Flowcharts are used in analyzing, designing, documenting or managing a process or program in various fields.

Flowcharts are used to design and document simple processes or programs. Like other types of diagrams, they help visualize the process. Two of the many benefits are flaws and bottlenecks may become apparent. Flowcharts typically use the following main symbols:

- A process step, usually called an activity, is denoted as a rectangular box.
- A decision is usually denoted as a diamond.

A flowchart is described as "cross-functional" when the chart is divided into different vertical or horizontal parts, to describe the control of different organizational units. A symbol appearing in a particular part is within the control of that organizational unit. A cross-functional flowchart allows the author to correctly locate the responsibility for performing an action or making a decision, and to show the responsibility of each organizational unit for different parts of a single process.



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

**A use case diagram is used to represent the dynamic behavior of a system. It encapsulates the system's functionality by incorporating use cases, actors, and their relationships. It models the tasks, services, and functions required by a system/subsystem of an application. It depicts the high-level functionality of a system and also tells how the user handles a system.**

## Purpose of Use Case Diagrams

**The main purpose of a use case diagram is to portray the dynamic aspect of a system. It accumulates the system's requirement, which includes both internal as well as external influences. It invokes persons, use cases, and several things that invoke the actors and elements accountable for the implementation of use case diagrams. It represents how an entity from the external environment can interact with a part of the system.**

**Following are the purposes of a use case diagram given below:**

1. **It gathers the system's needs.**
2. **It depicts the external view of the system.**
3. **It recognizes the internal as well as external factors that influence the system.**
4. **It represents the interaction between the actors.**