SE OVERVIEW OF IT INDUSTRY

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

**Ans:-**

**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.**

**Software Engineering :-**

**It is a branch of engineering that deals with the development of software products. It operates within a set of principles, best practices, and methods that have been carefully honed throughout the years, changing as software and technology change. Software engineering leads to a product that is reliable, efficient, and effective at what it does. While software engineering can lead to products that do not do this, the product will almost always go back into the production stage.**

1. **Explain types of software?**

**Ans:-**

* **Application software :-**

**The most common type of software, application software is a computer software package that performs a specific function for a user, or in some cases, for another application. An application can be self-contained, or it can be a group of programs that run the application for the user. Examples of**[**modern applications**](https://www.techtarget.com/searchcio/feature/The-rise-of-modern-applications-Why-you-need-them)**include office suites, graphics software, databases and database management programs, web browsers, word processors, software development tools, image editors and communication platforms.**

* **System software :-**

**These software programs are designed to run a computer's application programs and hardware. System software coordinates the activities and functions of the hardware and software. In addition, it controls the operations of the computer hardware and provides an environment or platform for all the other types of software to work in. The OS is the best example of system software; it manages all the other computer programs. Other examples of system software include the** [**firmware**](https://www.techtarget.com/whatis/definition/firmware), **computer language translators and system** [**utilities**](https://www.techtarget.com/whatis/definition/utility).

* **Driver software :-**

**Also known as device drivers, this software is often considered a type of system software. Device drivers control the devices and peripherals connected to a computer, enabling them to perform their specific tasks. Every device that is connected to a computer needs at least one** **device driver to function. Examples include software that comes with any nonstandard hardware, including special game controllers, as well as the software that enables standard hardware, such as USB storage devices, keyboards, headphones and printers.**

* **Middleware :-**

**The term *middleware* describes software that mediates between application and system software or between two different kinds of application software. For example, middleware enables Microsoft Windows to talk to Excel and Word. It is also used to send a remote work request from an application in a computer that has one kind of OS, to an application in a computer with a different OS. It also enables newer applications to work with legacy ones.**

* **Programming software :-**

**Computer programmers use programming software to write code. Programming software and programming tools enable developers to develop, write, test and**[**debug**](https://www.techtarget.com/searchsoftwarequality/definition/debugging)**other software programs. Examples of programming software include assemblers, compilers, debuggers and interpreters.**

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

**Ans :-**

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

**An SDLC (software development life cycle) is a big-picture breakdown of all the steps involved in software creation (planning, coding, testing, deploying, etc.). Companies define custom SDLCs to create a predictable ,** [**iterative framework**](https://phoenixnap.com/glossary/iterative-development)**that guides the team through all major stages of development.**



**Fig. SDLC life cycle**

* **The character and the exact number of phases in an SDLC vary between businesses and projects. The most common models are variations on the following steps:**

1. **Planning & Requirement analysis.**
2. **Product design.**
3. **Implementation.**
4. **Testing & Integration/Quality Assurance.**
5. **Deployment.**
6. **Post-production Maintenance.**

### 1. Planning and Requirement Gathering :-

**The client defines a problem that needs to be solved, which further forms the basis for finalizing the requirements. We gather all the information from the client and users (if required) to create requirement documents so that the development team understands what they are building. Then follows planning, which includes distributing work across teams, setting milestones, creating timelines for delivery, getting cost estimates, conducting a risk analysis, and devising a plan to mitigate the risks. Creating a**[**proof of concept**](https://www.netsolutions.com/insights/proof-of-concept-poc/)**is also part of this stage, where the technical feasibility is checked and validated before proceeding with the development work.**

### 

### 2. Design :-

**The design phase of the software development life cycle focuses on**[**creating software architecture**](https://www.netsolutions.com/insights/why-software-architecture-matters-to-build-scalable-solutions/)**, prototypes, and**[**user experience design**](https://www.netsolutions.com/experience-design)**. Here’s an overview of what these activities entail:**

**Software Architecture: Refers to creating an organized flow of elements in a software code for quality assurance, clarity, and accessibility. You can refer to software architecture as the blueprint for the development team.**

**Prototype:** **The UI/UX team builds a**[**prototype version**](https://www.netsolutions.com/insights/poc-vs-mvp-vs-prototype/)**of the software to validate its look and flow of design elements. It lets the development team and the stakeholders visualize the overall look and feel of the software.**

### 3. Develop :-

**In this SDLC phase, coders work on bringing the concept into reality. The developers create KLOCs (thousands of lines of code) using programming languages they feel comfortable with. The development team aims to achieve developer velocity while ensuring quality delivery. The development team might release the software in one go, as with waterfall development, or can choose to deliver software in fragments (segmented into features, i.e., Agile development). Once the code is ready, the development team shares it with the testing team for review.**

### 

### 4. Testing and Quality Assurance :-

**This application development life cycle phase focuses on testing the written code for bugs and other inconsistencies. The testing and**[**quality assurance team**](https://www.netsolutions.com/quality-assurance-testing)**works to test and report the bugs to the development team. The testing team can rely on manual or automated testing (per their expertise and defined process). We can say that the testers and the development team work together to ensure foolproof software delivery.**

### 5. Deployment :-

**The entire software or a part of it goes into the production environment phase after developing, testing, fixing, retesting, and validating. If you follow the Agile SDLC process, deployment could be the launching of MVP and other features. However, in the case of Waterfall, deployment refers to launching a fully-fledged product in the market. If the end-users experience any issue with the software, it moves back to the software development team for reconsideration and fixing.**

### 6. Maintenance :-

**New requirements will likely drop with new technologies and changing user requirements. The software development team needs to iterate through the entire software development life cycle to accommodate these requirements to work on the recent change. Thus, maintenance implies that software requires updates from time to time, which you must frequently do to uphold the software’s value proposition. The common types of maintenance include:**

* **Corrective Maintenance :  
  Removing the existing bugs from the software to improve its performance. These errors come to light, generally through user reviews.**
* **Perfective Maintenance :  
  New requirements keep evolving from time to time. And, to stay relevant and valuable, the software needs to be updated to accommodate these changes. That is the idea behind perfective maintenance.**

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

**Ans:-** **Data flow diagram (DFD) is a diagram being used frequently in software design. It visually represents the flow of data throughout processes in a given system. DFD shows the kind of information that will be input to and output from processes as well as where the data will be stored.**