**Software documentation**

[**https://en.wikipedia.org/wiki/Software\_documentation**](https://en.wikipedia.org/wiki/Software_documentation)

**Software documentation** is written text or illustration that accompanies computer software. It either explains how it operates or how to use it, and may mean different things to people in different roles.

Documentation is an important part of software engineering. Types of documentation include:

1. Requirements – Statements that identify attributes, capabilities, characteristics, or qualities of a system. This is the foundation for what will be or has been implemented.
2. Architecture/Design – Overview of software. Includes relations to an environment and construction principles to be used in design of software components.

Architecture documentation (also known as software architecture description) is a special breed of design document. In a way, architecture documents are third derivative from the code (design document being second derivative, and code documents being first). Very little in the architecture documents is specific to the code itself. These documents do not describe how to program a particular routine, or even why that particular routine exists in the form that it does, but instead merely lays out the general requirements that would motivate the existence of such a routine. A good architecture document is short on details but thick on explanation. It may suggest approaches for lower level design, but leave the actual exploration trade studies to other documents.

Another breed of design docs is the comparison document, or trade study. This would often take the form of a *whitepaper*. It focuses on one specific aspect of the system and suggests alternate approaches. It could be at the user interface, code, design, or even architectural level. It will outline what the situation is, describe one or more alternatives, and enumerate the pros and cons of each. A good trade study document is heavy on research, expresses its idea clearly (without relying heavily on obtuse jargon to dazzle the reader), and most importantly is impartial. It should honestly and clearly explain the costs of whatever solution it offers as best. The objective of a trade study is to devise the best solution, rather than to push a particular point of view. It is perfectly acceptable to state no conclusion, or to conclude that none of the alternatives are sufficiently better than the baseline to warrant a change. It should be approached as a scientific endeavor, not as a marketing technique.

A very important part of the design document in enterprise software development is the Database Design Document (DDD). It contains Conceptual, Logical, and Physical Design Elements. The DDD includes the formal information that the people who interact with the database need. The purpose of preparing it is to create a common source to be used by all players within the scene. The potential users are:

* Database designer
* Database developer
* Database administrator
* Application designer
* Application developer

When talking about Relational Database Systems, the document should include following parts:

* Entity - Relationship Schema ([enhanced](https://en.wikipedia.org/wiki/Enhanced_Entity-Relationship_Model) or not), including following information and their clear definitions:
  + Entity Sets and their attributes
  + Relationships and their attributes
  + Candidate keys for each entity set
  + Attribute and Tuple based constraints
* Relational Schema, including following information:
  + Tables, Attributes, and their properties
  + Views
  + Constraints such as primary keys, foreign keys,
  + Cardinality of referential constraints
  + Cascading Policy for referential constraints
  + Primary keys

It is very important to include all information that is to be used by all actors in the scene. It is also very important to update the documents as any change occurs in the database as well.

1. Technical – Documentation of code, algorithms, interfaces, and APIs.

It is important for the code documents associated with the source code (which may include README files and API documentation) to be thorough, but not so verbose that it becomes overly time-consuming or difficult to maintain them. Various how-to and overview documentation guides are commonly found specific to the software application or software product being documented by API writers. This documentation may be used by developers, testers, and also the end-users using the software application. Today, a lot of high-end applications in the field of power, energy, transportation, networks, aerospace, safety, security, industry automation and a variety of other domains are seen. Technical documentation has become important within such organizations as the basic and advanced level of information may change over a period of time with architecture changes. Code documents are often organized into a *reference guide* style, allowing a programmer to quickly look up an arbitrary function or class.

1. End user – Manuals for the end-user, system administrators and support staff.

Typically, the user documentation describes each feature of the program, and assists the user in realizing these features. A good user document can also go so far as to provide thorough troubleshooting assistance. It is very important for user documents to not be confusing, and for them to be up to date. User documents need not be organized in any particular way, but it is very important for them to have a thorough index. Consistency and simplicity are also very valuable. User documentation is considered to constitute a contract specifying what the software will do. API Writers are very well accomplished towards writing good user documents as they would be well aware of the software architecture and programming techniques used. See also [technical writing](https://en.wikipedia.org/wiki/Technical_writing).

User documentation can be produced in a variety of online and print formats. However, there are three broad ways in which user documentation can be organized.

1. **Tutorial:** A tutorial approach is considered the most useful for a new user, in which they are guided through each step of accomplishing particular tasks
2. **Thematic:** A thematic approach, where chapters or sections concentrate on one particular area of interest, is of more general use to an intermediate user. Some authors prefer to convey their ideas through a knowledge based article to facilitating the user needs. This approach is usually practiced by a dynamic industry, such as Information technology, where the user population is largely correlated with the troubleshooting demands
3. **List or Reference:** The final type of organizing principle is one in which commands or tasks are simply listed alphabetically or logically grouped, often via cross-referenced indexes. This latter approach is of greater use to advanced users who know exactly what sort of information they are looking for.

A common complaint among users regarding software documentation is that only one of these three approaches was taken to the near-exclusion of the other two. It is common to limit provided software documentation for personal computers to online help that give only reference information on commands or menu items. The job of tutoring new users or helping more experienced users get the most out of a program is left to private publishers, who are often given significant assistance by the software developer.

1. Marketing – How to market the product and analysis of the market demand.

For many applications it is necessary to have some promotional materials to encourage casual observers to spend more time learning about the product. This form of documentation has three purposes:-

1. To excite the potential user about the product and instill in them a desire for becoming more involved with it.
2. To inform them about what exactly the product does, so that their expectations are in line with what they will be receiving.
3. To explain the position of this product with respect to other alternatives.