FUNCTIONAL VS NON-FUNCTIONAL REQUIREMENTS: MAIN DIFFERENCES & EXAMPLES

Link to source: https://theappsolutions.com/blog/development/functional-vs-non-functional-requirements/

Imagine that you want to build a house. It should be two-stored, have a red roof, and several windows. But what about the number of rooms, the color of the walls and the style of the house? Should it be mid-century modern or even Scandinavian? You need to clarify many aspects to make the result meet your requirements.
The same applies to mobile app development. A detailed vision of the project helps business analysts and project managers create better product documentation in the short term. On another hand, if the team need to clarify information during the development stage, development time and costs might increase, as well as the probability that the project may fail.

How can you avoid this?
Simply by defining functional and non-functional requirements for the project. While functional requirements might be quite clear, some of the non-functional requirements are hard to specify.

For instance:
How do we know what acceptable “performance” should look like?
Or
How can we define “maintainability” before any code has been written?

If you want to decrease the project's cost, increase the development of team productivity, and develop a successful and cost-effective project, read on.
Below, we share information on what requirements you need to clarify with a business analyst, functional and nonfunctional requirement examples, and handy tips on how to specify non-functional requirements.

How requirements impact the software development process?

As mentioned, clearly defined requirements are the key to project success. These requirements also help the development team and client to ensure they are working to reach the same goals. Failing to define requirements may cause miscommunication between the team and client, and increase the chances of the project failing.
But wait – there’s more:

- 68% of projects with effective communication, and precise requirements, are likely to deliver project scope and meet quality standards successfully.
- At the same time, 32% of IT projects failed due to sparse estimation during the planning phase and unclear requirements.
- Besides, unclear requirements increase the project timeline and budget up to 60%.
- Also, unclear requirements consume over 41% of the IT development budget for software, staff, and external professional services.

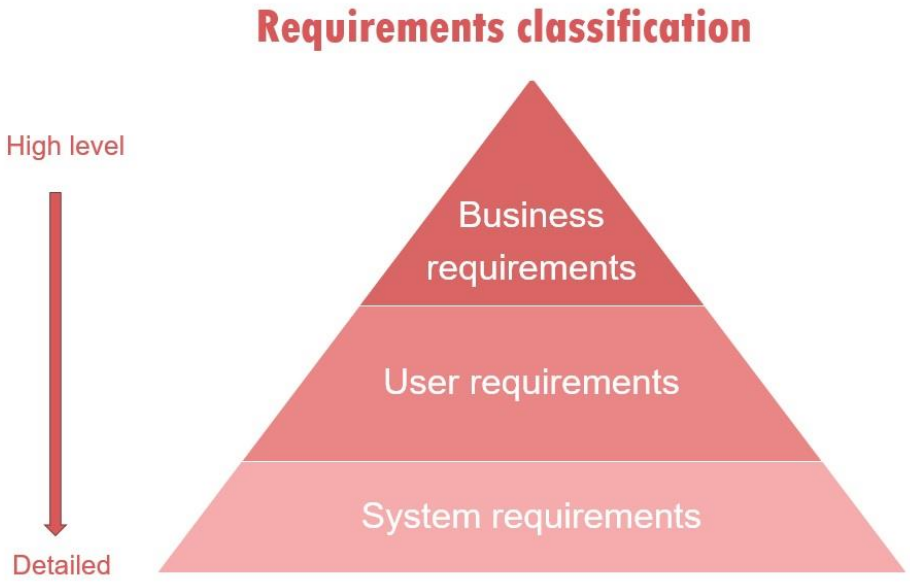
Detailed functional and nonfunctional requirements in software engineering help the team to complete the following tasks:
Define the terms and roles. Requirements help to ensure that the development team and stakeholders are on the same page to avoid misunderstandings in the future.
Reduce communication time. Close cooperation with BA ensures much clearer requirements and less development time. Such an approach reduces the time required for communication during the development stage, as well as the project's cost.

Make the project estimation more precise. Detailed requirements help us to estimate the development time and cost more accurately.
See the possible mistakes beforehand. When the team visualizes the project details during the discovery (inception) phase, they may identify errors in the initial stage of development. Therefore, they save your time and budget.
Create more predictable projects. High-quality requirements and wireframes help to predict the result and develop the project that meets your expectations.
To conclude, detailed requirements help developers and stakeholders to find a common language, save money and time for development, as well as create a project that meets business needs and expectations.
Now we'll move on and learn more about project requirement types.

Classification of requirements

To turn your business idea into a working solution, you need to clarify the following:
Business requirements include high-level statements of goals, objectives, and needs of your project.
Stakeholder requirements help to find what you expect from a particular solution.
Solution requirements describe the product characteristics that will meet your expectations and business needs. They include:

- Functional requirements describe ways a product must behave
- Nonfunctional requirements, also known as quality attributes, describe the general software characteristics



Functional Vs Non-Functional Requirements: The Comparison Table

If you still have a question about the difference between functional and non-functional requirements, check the table below:

Parameters	Functional Requirement	Non-Functional Requirement
Requirement	It is mandatory	It is non-mandatory
Capturing type	It is captured in use case.	It is captured as a quality attribute.
End-result	Product feature	Product properties
Capturing	Easy to capture	Hard to capture
Objective	Helps you verify the functionality of the software.	Helps you to verify the performance of the software.
Area of focus	Focuses on user requirement	Concentrates on the user's expectation and experience.
Documentation	Describe what the product does	Describes how the product works
Product Info	Product Features	Product Properties

Functional requirements

Such requirements describe system behavior under specific conditions and include the product features and functions which web & app developers must add to the solution. Such requirements should be precise both for the development team and stakeholders.
The list of examples of functional requirements includes:

- Business Rules
- Transaction corrections, adjustments, and cancellations
- Administrative functions
- Authentication
- Authorization levels
- Audit Tracking
- External Interfaces
- Certification Requirements
- Reporting Requirements
- Historical Data

If your team uses Agile methodology, they will design most of the requirements in written form. Still, to present some requirements more clearly, the team can visualize them.
The functional requirements may appear in the following forms:
Functional requirements specification document

The documentation includes detailed descriptions of the product's functions and capabilities. These could be a single functional requirements document or other documents, such as user stories and use cases.
As well as the form, the specification document must consist of the following sections:
Purpose. This section includes background, definitions, and system overview.
Overall description. The description document consists of product vision, business rules, and assumptions.
Specific requirements. The requirements might be database requirements, system attributes, and functional requirements.
Here is a project definition example:
"*Admin dashboard - a web portal allowing Admin to view and manage Applicants and Customers, Drivers, vehicles, manage car models, prices and review statistics from both mobile platforms.*"

Use cases
Use cases describe the interaction between the system and external users that leads to achieving particular goals.

Each use case includes three main elements:

- ***Actors*** are the users who will interact with your product.
  Example:
  "*Applicant - a person who wants to use the App and applies for registration.*
  Customer - a person who was approved for registration and can use the App.
  Driver - a person, who was registered in the Driver app and fulfills ride orders for
  Members received from the App."

- ***System functional requirements*** describe the intended behavior of the product.
  Example:
  "*Payment System charges Customer for the ride*."

- ***Goals*** describe all interactions between the users and the system.
  Example:
  "*When the journey is over, Driver marks it has ended in their app*."

User stories
User stories are documented descriptions of software features from the end-user perspective. The document describes scenarios of how the user engages with the solution.
Example:
"*As a Customer, I want to select a car from the carousel so that I can complete the order.* ".
If the development team uses Agile methodology, they will organize user stories in a backlog, an ordered list of product functions.

Functional decomposition
A functional decomposition or work breakdown structure (WBS) illustrates how complex processes and features break into simpler components. By using WBS approach, the team can analyze each part of the project while capturing the full project picture.

Example:

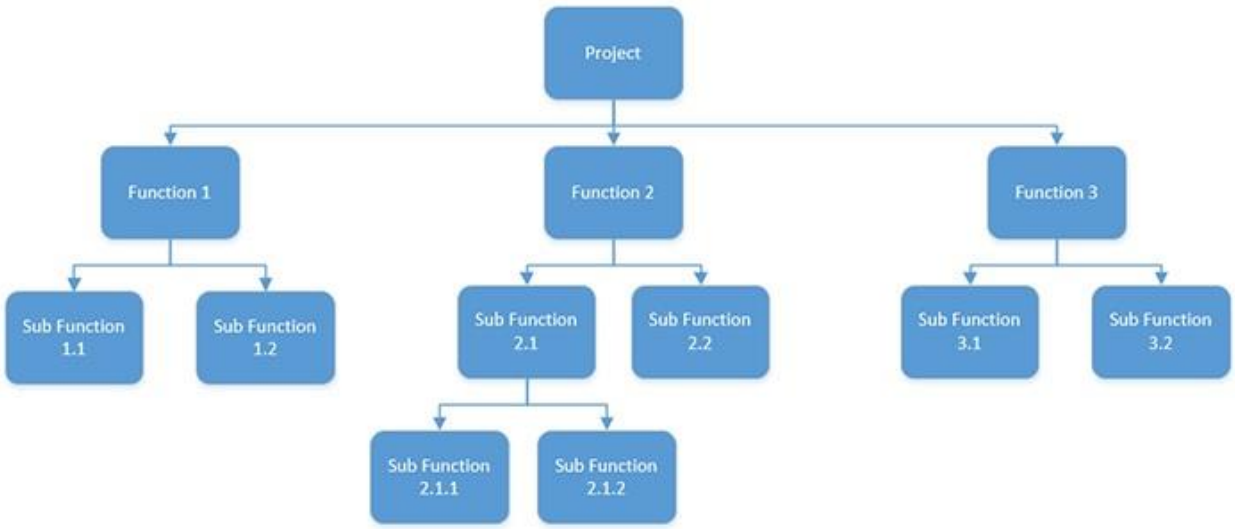


Image source: Batimes

Mobile app prototypes
By using prototypes, stakeholders and teams clarify the project vision and complicated areas of products in development. The development team also uses prototypes to represent how the solution will work and give examples of how users will interact with it.
There are two types of prototypes:

- Throwaway prototypes that can be cheap and fast visual representations of requirements

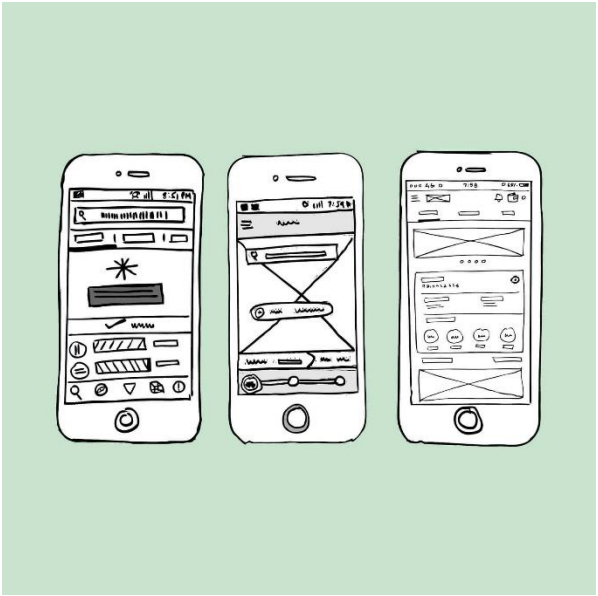


Image source: Justuxdesign

- Evolutionary prototypes, more complex ones, that can later even become the early versions of the product or the MVP.

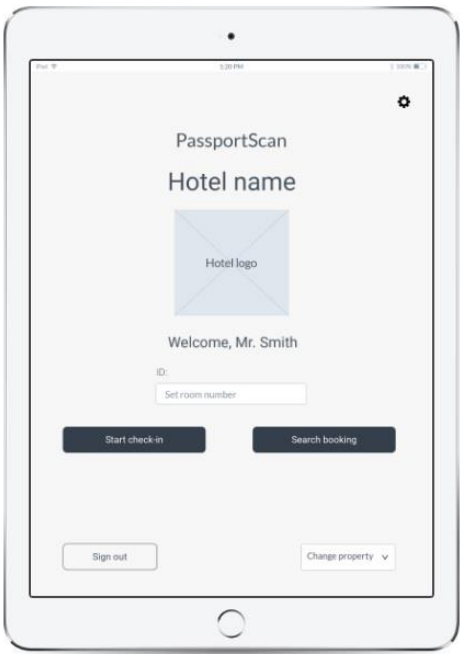


Image source: The App Solutions

Non-functional requirements examples

What is a non-functional requirement?
The definition of non-functional requirements is quality attributes that describe ways your product should behave. The list of basic non-functional requirements includes:

Usability
Usability is the degree of ease with which the user will interact with your products to achieve required goals effectively and efficiently.

Legal or Regulatory Requirements
Legal or regulatory requirements describe product adherence to laws. If your product violates these regulations, it may result in legal punishment, including federal fines.
For example, look at legal requirements for our recent project, a mobile taxi platform:
"*To operate in London, the platform should be licensed by the local transport authority - Transport for London*."

Reliability
Such a metric shows the possibility of your solution to fail. To achieve high reliability, your team should eliminate all bugs that may influence the code safety and issues with system components.

Performance
Performance describes how your solution behaves when users interact with it in various scenarios. Poor performance may lead to negative user experience and jeopardize system safety.
Example:
"*The application shows cars nearby for three seconds.*"

As we said, some non-functional requirements are not so distinct and might be missed by the team and stakeholders due to:
Subjective nature. Different users can view, interpret, and evaluate Nonfunctional characteristics in different ways.

Integrated nature. The goals of one non-functional requirement may conflict with another since they typically have a broad effect on systems.

Don’t know what they [NFRs] are. Unclear terminology, confusing definitions, and the absence of a universally accepted classification scheme make understanding of non-functional requirements a challenge.

Assuming that “Everybody knows.” During the discovery (inception) phase, both the client and the team might forget about some non-functional requirements because some of them are hard to define from the perspective of a business idea. Therefore, they might arise only after the project release.
Still, non-functional requirements mark the difference between a development project’s success and its failure.

How to define non-functional requirements

To determine the majority of non-functional requirements, you should:

- **Use a defined classification** and classify them into three groups: operation, revision, and transition. In this way, the stakeholders and the development team build a consistent language for discussing non-functional needs.
- **With a list of pre-defined elicitation questions**, you may increase the development teams productivity. Besides, you can save time when preparing for elicitation interviews and workshops.
- **Engage with the development team** during the requirements definition to ensure that you are on the same page with the development team.
- **Use 'Invented Wheels'** and reuse the requirements written for other systems, since software systems have a lot in common when comparing nonfunctional requirements.
- **Use automated testing tools** such as Selenium, TestComplete, and Appium. Such tools will help to check your product performance faster and reveal more non-functional requirements.

Final words

Precise functional and non-functional requirements are essential to reduce development costs because, when the requirements are clear, the team can develop the project much faster. The difference between functional and non-functional requirements is as follows:
Functional requirements are easy to define because the business idea drives them. They include all the features of your future project and ways users engage with it.
Experience drives Non-functional requirements. In order, to identify them, you need to analyze the product's performance and make it convenient and useful. Such requirements may appear when the product is being used on a regular basis.
In this way, while functional requirements describe what the system does, non-functional requirements describe how the system works.