What are functional requirements?

Functional requirements are product features that developers must implement to enable the users to achieve their goals. They define the basic system behavior under specific conditions.

Functional requirements should not be confused with other types of requirements in product management:

- **Business requirements** describe the high-level business needs, such as carving a market share, reducing customer churn, or improving the customers' lifetime value.
- **User requirements** cover the different goals your users can achieve using the product and are commonly documented in the form of user stories, use cases, and scenarios.
- **Product requirements** describe how the system needs to operate to meet the business and user requirements. They include **functional requirements** and **non-functional requirements**.

Functional requirements examples

Functional requirements need to be clear, simple, and unambiguous. Here are some examples of well-written functional requirements:

- The system must send a confirmation email whenever an order is placed.
- The system must allow blog visitors to sign up for the newsletter by leaving their email.
- The system must allow users to verify their accounts using their phone number.

But stories can be a useful tool for deriving requirements with the user in mind. For example:

- **User story**: As an existing user, I want to be able to log into my account.
- **Functional requirements**:
    - The system must allow users to log into their account by entering their email and password.
    - The system must allow users to log in with their Google accounts.
    - The system must allow users to reset their password by clicking on "I forgot my password" and receiving a link to their verified email address.

Functional vs. non-functional requirements

When capturing product requirements, it's important to distinguish between functional and non-functional requirements.

To put it simply, functional requirements describe **what the product should do**, while non-functional requirements place constraints on **how the product should do it**. They can be expressed in the following form:

- **Functional requirement**: "The system must do [requirement]."
- **Non-functional requirement**: "The system shall be [requirement]."

Functional requirements – as the name implies – refer to specific product functionality. Defining, measuring, and testing them is usually a straightforward task.

On the other hand, non-functional requirements (also known as "quality requirements" or "quality attributes") are more abstract. They impose constraints on the implementation of the functional requirements in terms of performance, security, reliability, scalability, portability, and so on.

Every functional requirement typically has a set of related non-functional requirements, for example:

- **Functional requirement**: "The system must allow the user to submit feedback through a contact form in the app."
- **Non-functional requirement**: "When the submit button is pressed, the confirmation screen must load within 2 seconds."

What is a user story?

A **user story** is a technique used in Agile software development to capture product requirements from the perspective of a user. Simply put, a user story describes the type of user, what they want, and why. Like a short story a user could tell about something they'll be able to do.

User stories are the smallest unit of work in the Agile framework and act as the building blocks of epics, which in turn add up to form initiatives. In Scrum, user stories are added to the backlog and prioritized during sprints. This ensures that the day-to-day work of the development team contributes to the long-term organizational goals of the company.

- A **user story** focuses on the experience, what the user wants to achieve with the product.

- **Requirements**, on the other hand, focus on functionality – what the product should do.

Requirements are developed at a later stage. During a sprint planning meeting, the team usually decides what stories to tackle. They then move on to discuss the requirements and functionality that each user story requires. Once agreed upon, these requirements are added to the story or incorporated into a separate product requirements document (PRD).

User story template

As a [user], I want to [capability], so that [receive benefit].

- As a new user, I want to sign up using my existing Google account so that I don't have to keep track of another account.
- As a new user, I want to be guided through the platform after registration so that I can discover the core features.
- As a user, I want to receive mobile notifications so that I can get relevant updates whenever I'm on the go.
- As a team manager, I want to see the status of all projects at a glance so that I can easily track everyone's progress.

How to write test cases: A step-by-step guide

Test cases are the blueprints that testers will follow, so they must be clear, thorough, and accurate. Below, we've outlined 10 steps you can take whether you're writing new test cases or revisiting and evaluating existing test cases.

1. Define the area you want to cover from the test scenario.
2. Ensure the test case is easy for testers to understand and execute.
3. Understand and apply relevant test designs.
4. Use a unique test case ID.
5. Use the requirements traceability matrix in testing for visibility.
6. Include a clear description in each test.
7. Add proper preconditions and postconditions.
8. Specify the exact expected result.
9. Utilize suitable testing techniques.
10. Get your test case peer-reviewed before moving forward.

How to write Test Cases (Test Case Example)
Let’s build a test case example based on a specific scenario. Here is a sample case.

- **Test Case ID**: #BST001
- **Test Scenario**: To authenticate a successful user login on Gmail.com
- **Test Steps:**
    - The user navigates to Gmail.com.
    - The user enters a registered email address in the ’email’ field.
    - The user clicks the ‘Next’ button.
    - The user enters the registered password.
    - The user clicks ‘Sign In.’
- **Prerequisites**: A registered Gmail ID with a unique username and password.
- **Browser**: Chrome v 86. Device: Samsung Galaxy Tab S7.
- **Test Data:** Legitimate username and password.
- **Expected/Intended Results:** Once username and password are entered, the web page redirects to the user’s inbox, displaying and highlighting new emails at the top.
- **Actual Results:** As Expected
- **Test Status – Pass/Fail**: Pass

E2E Testing Example

Let's imagine that testers need to confirm that a Gmail account is operational. The following attributes need to be examined:

- To open the Gmail login page, enter the URL into the address box.
- Enter your account using legitimate credentials.
- Access Inbox. Examine your read and unread emails.
- Create a fresh email.
- Respond to and forwarded an email.
- Open the Sent Items folder. Verify your emails there.
- Go to the Spam folder. Verify your emails there.
- To exit Gmail, click the "Logout" button.

Creating End-to-End Test Cases

End to-end testing design framework is comprised of three parts:-

- Build user functions
- Build Conditions
- Build Test Cases

Here is the detailed view:-

Build User Functions

The actions listed below must be carried out as part of the build user functions:

- List the system's features and the ways they are connected.
- List the input, action, and output data for each feature or function.
- Determine the connections between the functions.
- Ascertain whether the function is standalone or reusable.

For instance, if you logged into your bank account and transferred money to a third-party account at another bank.

Build Conditions Based on User Function

The following tasks are considered as part of the build conditions:

- Establishing a set of requirements for every user function specified.
- Sequence, timing and data conditions are examples of conditions.

For example -Checking of more conditions like:

Login Page

- Invalid username and/or password.
- Check your username and password.
- Checking password strength.
- Checking of error messages.

Balance Amount

- Check if the transfer has been completed within 24 hours.
- If you do not have enough money in your account to complete the transfer, the transaction will fail.

Build a Test Scenario

We will construct a test scenario for the specified user function:

In this instance,

- Enter the system.
- Verify your bank balance.
- Transfer your bank balance.

End-to-End Testing	System Testing
End-to-end testing includes testing the behavioral workflow of the software or application.	System testing includes testing the whole software or application as a whole.
The focus is to test user experience and business requirements.	Technical aspects of the software/application are only tested.
Usually performed by testers who are familiar with user experience.	Usually performed by testers that are not involved in the development process.
It includes regression testing, integration testing, and user acceptance testing.	It includes functional testing, performance testing, security testing, and usability testing.
Helps in validating the interfaces of the software or application.	System testing validates the software system's compliance with standards, specifications, and other requirements.
It is usually performed manually.	Performed both manually and automated.