What Are Software Development Requirements?

Let's start with the basics: what do we mean by product requirements in web development and why are they essential? When we're working on creating a website or any other software product, **requirements refer to everything the system needs to function properly,** as well as any potential issues we might encounter along the way.

To ensure that the requirements are thorough, we need to address a few key questions:

- **What are the client's business goals?** Understanding the client's business goals is crucial as it determines the purpose behind the product creation. These goals can range from expanding audience reach, entering new markets, or validating an idea. The level of complexity required for the product centers on these objectives. Sometimes, a minimally viable product with basic features serves, while other times, a comprehensive, multi-functional product adapted to various languages is necessary.
- **Who are the people we're targeting with this product, and how will they use it?** Understanding the target audience's demographics, activity field, and tech awareness helps determine the product's complexity. For example, when developing a fitness app aimed at health-conscious people aged 25-40, we can provide features such as workout tracking and nutritional recommendations. Knowing how people will use it - whether on the go or during focused workouts - influences interface design and performance optimization.
- **What will the end product look like?** This question aims to define the visual design and functionality of the product being developed. For example, if we're talking about a fitness app, the final product might have a sleek interface with sections for workout tracking, goal setting, and nutritional advice. The idea is to create an easy-to-use tool that empowers individuals to improve their fitness by providing them with personalized workout plans, progress tracking, and nutritional insights.
- **What technology will we use to make it happen?** This can mean making choices about programming languages, frameworks, and database systems to ensure that the application's functionality and performance meet user needs and industry standards.
- **What problems might arise, and how can we avoid or solve them?** It refers to the fact that the team needs to anticipate and plan for potential problems that may arise with the product, such as technical failures, compatibility issues, maintenance issues due to excessive user base growth, user dissatisfaction, or something else. We need to come up with a strategy to prevent or solve these problems, such as thorough testing, intuitive design, and responsive customer.

Functional requirements

Functional requirements are like a checklist for a system as they specify what the system must do. Let’s take a fitness app as an example. In this case, functional requirements would outline tasks such as:

- Creating user profiles
- Logging workouts
- Tracking progress
- Displaying workout details
- Sending notifications about upcoming workouts

If these requirements aren't met, your software won't function properly. Imagine you purchased an online course, but after you made the payment, it didn't show up in your email inbox right away, or maybe it didn't show up at all. This is because one of the functional requirements was for the course to be emailed to you after payment, but it didn't happen the way it was supposed to support.

Non-functional requirements

These aren't just about the features themselves; they say how those features should work and be built. Let's look at the fitness app as an example. Non-functional requirements ensure the app loads quickly, has an easy-to-use interface, works well on different devices, and can handle many users without crashing. For instance, the app should load within 3 seconds on average and no single action should take longer than 1 second.

These requirements are all about:

- How easy it is to use
- How well it runs
- How reliable it is

Simply put, non-functional requirements determine exactly how the system should operate: its speed, what's going on behind the scenes, and so on. For instance, how swiftly you should receive a training course after payment, or how fast you can access your personal account and its features.

Functional vs. Non-Functional Requirements

When we compare these two, here's what we see:

Functional requirements are essential; the system can’t run unless these are met.

Non-functional requirements, on the other hand, aren't mandatory; **the system can still work even if these aren't fully met.**

Now, let's explore the other distinctions below.

Functional Requirements:

- What the system does
- Outline system capabilities
- Can be directly tested
- Can be easily defined
- Deals with features
- System-oriented
- Cover user registration, data processing
- User requirements are prioritized.

Non-Functional Requirements:

- How the system does it
- Focus on performance, security, etc.
- Not directly testable
- Hard to be defined
- Deals with attributes
- User-oriented
- Cover response time, security
- User expectations and needs are prioritized.

Examples of Functional vs Non-functional Requirements:

Now, let's look at some real-world examples to understand how these software requirements work in practice. Take Airbnb, for example.

Let's break down the functional requirements for Airbnb:

- **Effortless sign-up and log-in:** users must be able to easily create accounts using email or social media credentials.
- **Simplified search:** the platform should provide a straightforward interface for users to find apartments, with filters for location, dates, price range, property type, and amenities.

- **Comprehensive listings:** listings should include high-quality photos, detailed descriptions, availability calendars, cancellation policies, and accurate pricing.
- **Direct communication:** Airbnb's messaging feature should enable direct communication between guests and hosts.
- **Secure booking:** the payment system must ensure secure transactions, supporting multiple payment methods while protecting users' financial information with encryption and fraud prevention measures.
- **Share experiences:** guests should be able to leave detailed reviews and ratings for properties and hosts.

Let's outline the non-functional requirements for Airbnb:

- **Fast and reliable performance:** Airbnb's platform needs to respond quickly, ensuring smooth browsing, searching, and booking experiences, especially during busy times.
- **Data security:** Airbnb must prioritize safeguarding users' personal and financial data by adhering to industry.
- **User-friendly design:** the Airbnb app should be easy to navigate, with a clear layout and responsive design.
- **Device compatibility:** Airbnb should make sure that users can access the platform from any device.
- **Compliance and trust:** Airbnb needs to implement rigorous verification procedures for hosts and listings, ensuring compliance with local regulations.

Functional requirements:

- Users must be able to schedule appointments with healthcare providers using the system.
- The system must update inventory levels in real-time as items are bought and sold.
- Users should be able to easily transfer funds between their accounts.
- The system needs to provide personalized suggestions based on the user's browsing and purchase history.

Non-functional requirements:

- Web pages should take no longer than 3 seconds to load.
- The system must comply with GDPR to safeguard user data.
- The streaming service should handle up to 1 million concurrent users during peak times.
- The mobile app must be easy to use and maintain an average rating of at least 4 stars.

Functional and non-functional requirements are essential for project success. Starting with them helps you know where you're headed and what's needed. On the flip side, if you don't document

requirements well, you risk failure. And not just for this project - your team's reputation is on the line too.

CASE STUDY

Imagine you're a student working on a software development project for a local hospital aiming to improve its management, patient care, and data security. The hospital wants to implement a Hospital Management System (HMS) that will streamline administrative tasks, enhance patient interaction, and ensure compliance with healthcare regulations.

Your task is to carefully read the hospital's requirements and identify which are functional and which are non-functional. Here is the detailed description of the hospital's requirements:

1. Patient Management:
    - The system should allow doctors and nurses to create, view, and update patient records. Each record must include details like patient ID, name, age, gender, address, contact information, emergency contact, medical history, current medications, ongoing treatments, and test results.
    - Patients should be able to view their own records and update certain information, such as contact details and emergency contact, through a secure patient portal.
2. Appointment Scheduling:
    - Patients should have the ability to book, modify, or cancel appointments online based on doctors' availability, with options to choose the preferred time and date.
    - The system should send automated appointment reminders via email and SMS to reduce no-show rates.
    - Hospital staff should be able to manage appointment slots, reschedule, or cancel appointments if needed.
3. Billing and Payment Processing:
    - The HMS should generate detailed invoices for services, treatments, and medications for each patient.
    - Patients should be able to view, download, and pay their bills online through secure payment options.
4. Data Security and Privacy:
    - The system must comply with healthcare data regulations, ensuring that all patient data is encrypted both in storage and in transit.
    - Secure login with multi-factor authentication should be implemented for all hospital staff and patients to protect sensitive data from unauthorized access.

- Only authorized medical staff should have access to patient medical records, while billing staff should only have access to billing information.

5. Medical Reporting and Analytics:
    - The system should automatically generate daily reports for hospital administrators, detailing patient admissions, discharges, transfers, and current occupancy.
    - The HMS should also generate monthly analytics on commonly treated conditions, average length of stay, and patient demographics to support hospital decision-making.
6. User Interface and Accessibility:
    - The system should feature an intuitive interface that requires minimal training for hospital staff to perform routine tasks, such as checking in patients, scheduling appointments, and processing billing.
    - The interface should be accessible on both desktop and mobile devices, allowing staff to access it from various devices within the hospital.
7. System Availability and Performance:
    - The system should be available 24/7 to ensure uninterrupted access for both staff and patients, with a maximum downtime of 0.1% annually.
    - The HMS should respond to user requests, such as retrieving patient records or scheduling an appointment, within 1 second to avoid delays in hospital operations.
8. Integration with Existing Systems:
    - The HMS should integrate with the hospital's existing lab and pharmacy systems.
    - Lab results should automatically be updated in the patient’s medical records as soon as they’re available, allowing doctors to view test outcomes in real time.
    - The pharmacy system should also be linked to track prescriptions and availability of medications for each patient.