Types of Non-Functional Requirements Examples

Hamid Akhtar ·
Published in Coinmonks
10 min read · Jun 10, 2023

You’re on a mission to create the perfect house. Having rooms, doors, and windows is essential — the functional requirements. But let’s take it up a notch. What if this house could be more than just functional? What if it could be extraordinary?

That’s where non-functional requirements come into play. They are the secret sauce that elevates a house from ordinary to exceptional.

Non-functional requirements are all about the details that make a difference — the way it looks, the level of safety it provides, and the overall experience it offers. They ensure that your house not only functions but it shines with style, impresses with its performance, and brings joy to those who inhabit it. By incorporating non-functional requirements, you’re not just building a house but creating a masterpiece product that will stand the test of time and leave everyone in awe.

What are Non-Functional Requirements?

Non-functional requirements are specifications that define how well software should perform. They cover areas such as speed, security, reliability, and usability. Unlike functional requirements that focus on what software must do, non-functional requirements determine how effectively it should do it. These requirements ensure that the software meets quality standards and user expectations.

Factors like measurement scenarios, workload, architectural constraints, and scalability expectations must be considered when documenting non-functional requirements. They are crucial for creating software that performs well and provides a positive user experience.

Also Read: What is Non-Functional Testing?

Examples of Non-Functional Requirements

Non-functional requirements (NFRs) define how a system works and its limitations. NFR requirements make the product affordable, user-friendly, and accessible..

Here are types of non functional requirements:

- **Performance**: Fast-loading video games without lag.
- **Reliability:** Car starts every time you turn the key.
- **Usability:** Phones with a simple and easy-to-use interface.
- **Security**: Banking websites with strong protection and security for personal information.
- **Scalability:** Websites handling a large number of visitors without crashing.
- **Compatibility:** Printers working with different types of computers.

Learn More: Cross Browser Testing with Non-Functional Tests

Functional and Non-Functional Requirements: Comparison

Here is an example that will help you easily understand the concept of functional and non-functional.

- Imagine you want to buy a new smartphone. Functional requirements will tell you what features the smartphone should have, like a high-resolution camera, a fast processor, and a large storage capacity.
- Non-functional requirements will describe the general qualities of the smartphone, such as its battery life, durability, and user-friendliness. These requirements ensure that the smartphone performs its functions well and meets your expectations in terms of overall performance and user experience.

Also Read: Functional and Non-Functional Testing Checklist

Types of Non-Functional Requirements (with Examples)

1. **Performance and Scalability:** This refers to the speed at which the system can complete tasks and its ability to handle increased demand. For instance, an e-commerce website should be capable of accommodating many visitors during peak shopping seasons.
2. **Portability and compatibility:** This pertains to how well the system functions with other systems and its ease of being moved to different environments. For example, a mobile app

should be compatible with various devices and operating systems.

3. **Reliability, maintainability, and availability:** This encompasses the system's ability to work without errors, and maintain and its availability for use. For instance, a hospital's electronic medical records system should be reliable, easy to maintain, and accessible 24/7.
4. **Security** relates to the system's protection against unauthorized access. For example, an online banking system should have robust security measures to safeguard customers' personal and financial information.
5. **Localization**: refers to the system's adaptability to different languages and cultures. For instance, a social media platform should be capable of displaying content in multiple languages and allowing users to customize their profiles based on cultural preferences.
6. **Usability:** This denotes the ease of use of the system. For example, a video conferencing app should have a simple, intuitive interface for easy navigation.
7. **Manageability**: This relates to how easy it is to manage and maintain the system. For example, a content management system should provide a user-friendly interface that enables administrators to easily add, edit, and delete content.
8. **Data integrity**: This pertains to the accuracy and consistency of the system's data. For example, a customer relationship management system should ensure customer data remains accurate and up-to-date.
9. **Capacity** refers to the system's ability to handle data or traffic. For instance, a cloud storage service should be capable of managing large amounts of data from multiple users.
10. **Availability:** This denotes how often the system is accessible. For example, an online shopping website should be available 24/7 to allow customers to shop anytime.
11. **Utility:** This relates to the system is usefulness to its users. For example, a weather app should provide users with accurate and timely weather information.
12. **Interoperability** refers to how well the system can work with other systems. For instance, a healthcare information exchange system should be capable of exchanging patient data with other healthcare systems.
13. **Environmental**: This pertains to the system's ability to operate in different environmental conditions. For example, a GPS navigation system should be able to function in various weather conditions and temperatures.

Remember, non-functional requirements are crucial because they ensure the system functions as intended and meets user expectations, just like functional requirements.

What is Non-Functional Requirements Gathering?

Non-functional requirements gathering is about understanding the qualities you want your system or software to have rather than just what it does. It includes things like reliability, performance, usability, and security. You can gather these requirements by talking to people, conducting workshops, and using surveys. Once you have all the information, you can design, develop, and evaluate your system. It's important to ensure your system meets your expectations and performs well in real-world conditions.

How NFRs help development teams and project managers?

- **Vision and Focus**: NFRs provide a clear vision of the qualities your system needs to possess. They guide your efforts, ensuring you stay on track and deliver exceptional results.
- **Architectural Brilliance:** As technical experts, you have the power to make crucial design decisions. NFRs act as your playbook, helping you choose the right technologies, frameworks, and infrastructure to create scalable and maintainable systems.
- **Performance Powerhouse**: Just like athletes aiming for peak performance, NFRs drive you to optimize system speed, responsiveness, and efficiency. This means fine-tuning your code, optimizing database queries, and delivering a top-notch user experience.
- **Risk Management**: NFRs help you identify and mitigate potential risks. By incorporating security, reliability, and availability requirements, you build robust defenses, protect your systems, and keep the game strong.
- **Prioritization Prowess:** In the game of development, there are always trade-offs. NFRs empower you to prioritize effectively, balancing functional and non-functional requirements within project constraints to score big wins.
- **Seamless Team Communication**: NFRs are your secret language, allowing you to communicate effectively with your teammates and stakeholders. Discuss system behavior, performance expectations, and usability, ensuring everyone is on the same page.
- **Validation**: NFRs are your referee, enabling you to evaluate and validate your system's compliance with desired qualities. You ensure a victorious outcome by conducting thorough testing, analysis, and usability assessments.

Follow-Up Read: Roles and Responsibilities of a Test Manager

How to include NFRs while planning? (10 Techniques)

These 10 techniques ensure that the system not only works but also meets the quality expectations of its users.

1. Conduct interviews

- You talk to stakeholders like users and experts to understand their expectations regarding non-functional aspects of the system.
- It's like having a conversation to gather valuable insights.

2. Workshops

- It's like a group discussion where stakeholders come together to share their ideas and opinions about non-functional requirements.
- It encourages collaboration and helps in brainstorming.
- You get input from multiple people instead of just one person's perspective.

3. Surveys and questionnaires

- You create forms or questionnaires and distribute them to stakeholders. Collecting responses from many people gives you a broader understanding of their preferences and opinions on non-functional requirements.
- It's a more structured way of gathering information.

4. Document analysis

- You review existing documents like project specifications or industry standards to find any non-functional requirements already documented or implied.
- It helps you understand what has already been considered.
- You're looking for any relevant information that's already available.

5. Prototyping

- You create a simplified version or a mock-up of the system to give stakeholders a hands-on experience.

- By getting their feedback, you can identify non-functional usability, performance, and aesthetics requirements.
- You're showing them how the system might work. That sounds like a great way to get their input.

6. Observation

- Instead of just talking to people, you directly observe them using similar systems or working in their actual environment.
- This helps you understand their needs and challenges and identify non-functional workflow requirements and usability requirements.
- You can see firsthand how they interact with the system by observing them.

7. Benchmarking

- This helps you set performance goals or compliance requirements for non-functional aspects based on what is already considered good or acceptable.
- You're looking at what others have done well and using that as a reference for your own system.

8. Expert judgment

- Seek advice from domain experts who have experience dealing with non-functional requirements.
- Their expertise helps identify important requirements and potential issues that might be overlooked.
- You're tapping into their knowledge to ensure you're on the right track. That's valuable input.

9. Risk analysis

- By analyzing potential risks related to the system, you can identify non-functional requirements necessary for mitigating those risks.
- For example, if there's a risk of the system crashing, you would need requirements related to reliability and performance.
- It's crucial to consider and address the potential risks through appropriate requirements.

10. Analyzing use cases

- Use cases describe how users interact with the system to achieve specific goals.
- By analyzing these use cases, you can identify non-functional requirements crucial for the successful execution of those use cases, like performance or scalability.
- You're looking at specific scenarios to understand what the system needs to do well. That's a practical approach.

Advantages of Non-Functional Requirement

Non-functional requirements are really important for making a system work well. Imagine you're a company that develops a self-driving car. You want to ensure that your car performs flawlessly and meets high standards.

- First, you set performance requirements to ensure the car responds quickly and accurately to different situations. You specify how fast the car should react to obstacles and how smoothly it should navigate turns.
- Next, you focus on reliability requirements to ensure the car operates without failures. You implement metrics to measure the car's ability to run for long periods without encountering any issues, minimizing the risk of accidents or malfunctions.
- Security requirements are also crucial. You want to protect the car's control systems from hackers and unauthorized access. You implement encryption protocols, authentication mechanisms, and rigorous access controls to safeguard the car's operation and prevent potential threats.
- Additionally, the car needs to meet scalability requirements. As you anticipate a growing user base, you design the car to handle increased demands, such as accommodating more passengers, optimizing energy consumption, and adapting to varying traffic conditions.
- Usability is another focus. You aim to create an intuitive user interface with clear displays and easy-to-use controls, enabling passengers to interact with the car effortlessly and enhancing their overall experience.
- Lastly, the car must comply with industry regulations and standards, such as traffic and safety regulations. You ensure that the car's non-functional requirements align with these legal requirements, making the car roadworthy and legally compliant.

Disadvantages of Non-Functional Requirement

Non-functional requirements have their challenges.

- First, they can be subjective and a bit tricky to define precisely. This might make things more complicated when trying to implement them.
- Secondly, conflicts can arise between different non-functional requirements or even with functional requirements. It can be a juggling act to balance them all.
- Thirdly, measuring and prioritizing non-functional requirements can be a bit challenging, as they are often less tangible than functional requirements.

But, despite these challenges, non-functional requirements are crucial for a system's performance, security, and user-friendliness. They play a vital role in building reliable and high-quality systems.

Best Practices for Documenting Non-Functional Requirements

When documenting non-functional requirements, there are a few key things to keep in mind.

- **First, use clear and specific language to describe each requirement.** For example, imagine you're designing a website like Amazon. You would document non-functional requirements that state how the website should be easy to navigate, load quickly, and have a smooth checkout process. These requirements are specific and leave no room for confusion.
- **Next, it's helpful to categorize the requirements into different groups**. Take Tesla as an example. They categorize their non-functional requirements based on performance and scalability. This means they specify how their electric vehicles should have fast acceleration, long battery life, and be able to handle increased demand as more people adopt electric cars.
- **Another important aspect is to have measurable criteria**. For instance, let's say you're working on a vacation rental platform like Airbnb. Your non-functional requirements include criteria such as the website should load within three seconds and the booking process should be completed in four steps. These measurable criteria allow you to assess if the requirements have been met during testing objectively.
- **It's also crucial to involve stakeholders in the process**. Let's go back to Amazon as an example. They actively seek feedback from customers to understand their needs and preferences. This input helps shape their non-functional requirements related to usability and user experience. Involving stakeholders ensures that all relevant perspectives are considered, resulting in a better end product.
- **Lastly, maintaining proper version control and traceability is essential.** Think of a company like Microsoft. They have a systematic approach to managing non-functional requirements

by keeping track of changes and updates. This ensures everyone is on the same page and helps maintain consistency throughout the project.

By following these practices, companies can effectively document non-functional requirements, resulting in products and services that meet user expectations, perform well, and provide a positive experience.

Closing Notes

In summary, non-functional requirements (NFRs) are essential considerations in various industries. NFRs ensure that businesses provide top-notch quality and meet user expectations. So, remember the importance of NFRs in different industries and their role in delivering exceptional customer experiences.

Go beyond basic performance testing and embrace real-world scenarios with BrowserStack. Understand the diverse traffic landscape and effortlessly test your application on the device-browser-OS combinations your target audience will likely use.

Source: https://www.browserstack.com/guide/non-functional-requirements-examples

https://substack.com/@hamidakhtar?utm_source=profile-page

Nfr Software Development Software Engineering

Follow

Written by Hamid Akhtar

44 Followers · Writer for Coinmonks

Let us talk!!! hmdlabee@gmail.com Telegram: @binarycole

More from Hamid Akhtar and Coinmonks

Hamid Akhtar in Coinmonks

iOS Development on Windows: A Complete Guide to Xcode for Windows

What is Xcode?

9 min read · Oct 28, 2023

46

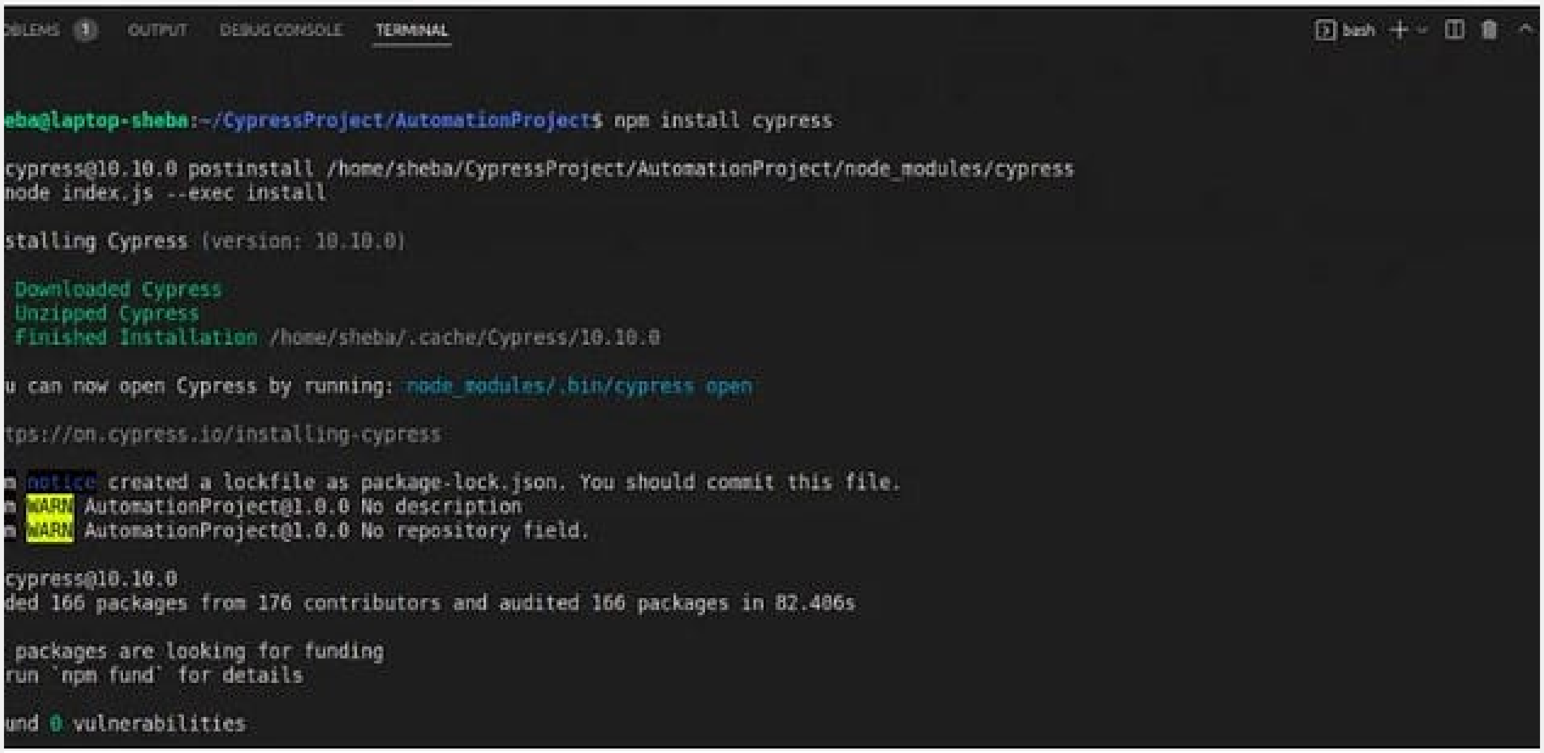
Crypto Big Stories in Coinmonks

These Crypto Gems Will Print the Next Set of Millionaires in This Bull Market

My top 5 hidden crypto gems to make you a millionaire. But you need to invest smartly to avoid losing money.

· 7 min read · Feb 19, 2024

395 5



```
eba@laptop-sheba:~/CypressProject/AutomationProject$ npm install cypress

cypress@10.10.0 postinstall /home/sheba/CypressProject/AutomationProject/node_modules/cypress
node index.js --exec install

stalling Cypress (version: 10.10.0)

 Downloaded Cypress
 Unzipped Cypress
 Finished Installation /home/sheba/.cache/Cypress/10.10.0

u can now open Cypress by running: node_modules/.bin/cypress open

tps://on.cypress.io/installing-cypress

m notice created a lockfile as package-lock.json. You should commit this file.
m WARN AutomationProject@1.0.0 No description
m WARN AutomationProject@1.0.0 No repository field.

cypress@10.10.0
ded 166 packages from 176 contributors and audited 166 packages in 82.406s

 packages are looking for funding
run `npm fund` for details

und 0 vulnerabilities
```

Hamid Akhtar in Coinmonks

Conditional Testing inCypress: Tutorial

When you perform one action or a different one, you are using conditional testing, a basic programming pattern. It is also referred to as...

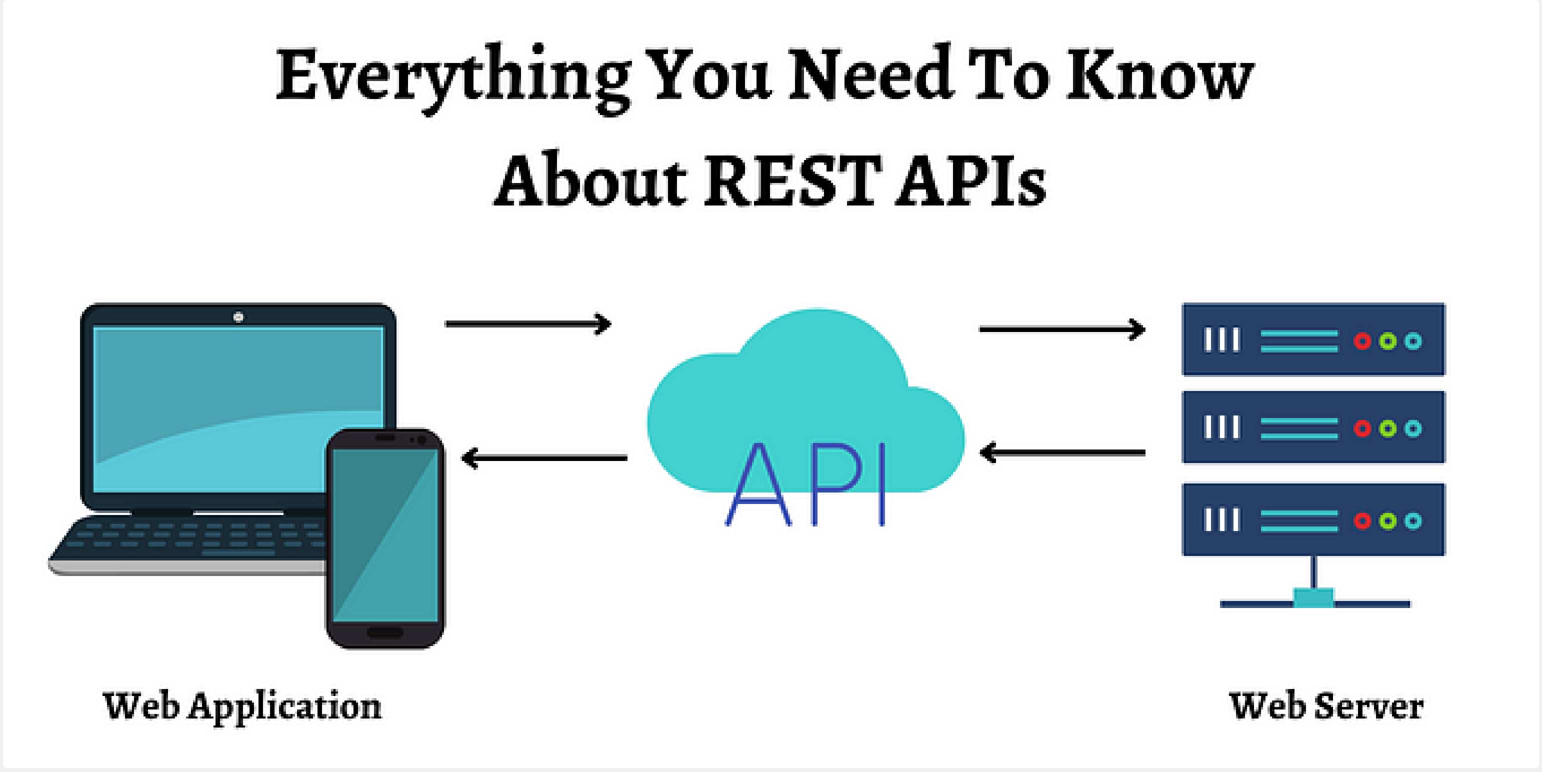
10 min read · Mar 2, 2023

12

See all from Hamid Akhtar

See all from Coinmonks

Recommended from Medium



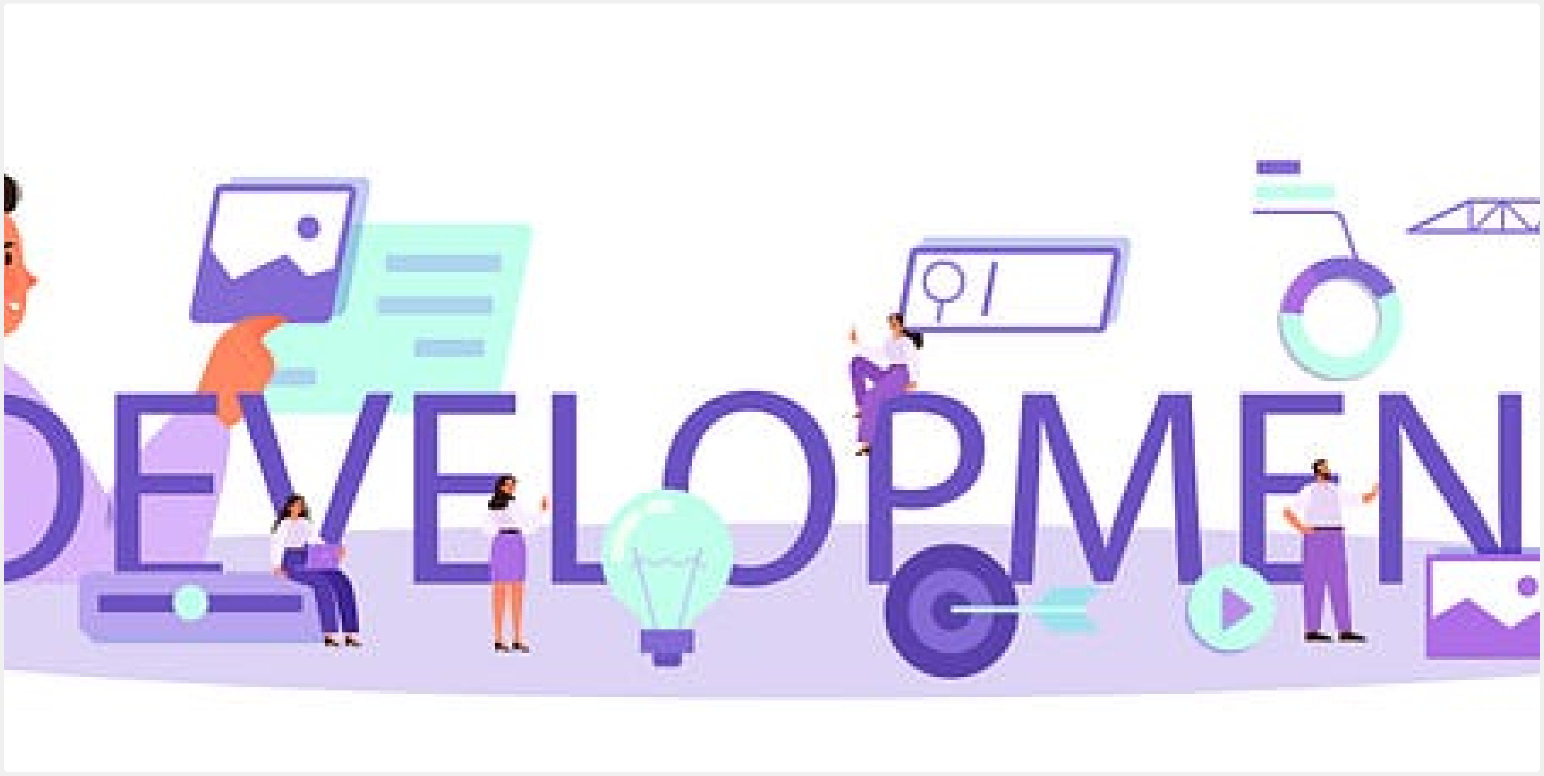
Amira Rahma

Everything You Need To Know About REST APIs

In the digital age, where applications and services seamlessly connect and interact, REST APIs have emerged as the unsung heroes behind the...

14 min read · Sep 24, 2023

22



System Design By CHK

System Design — Architecting Enterprise-Level Microservices

In the rapidly evolving landscape of software development, enterprises are constantly striving to build robust, scalable, and efficient...

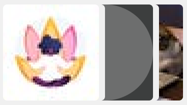
5 min read · Mar 10, 2024

75

Lists

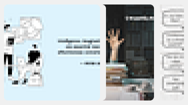
General Coding Knowledge

20 stories · 1030 saves



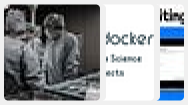
Stories to Help You Grow as a Software Developer

19 stories · 913 saves



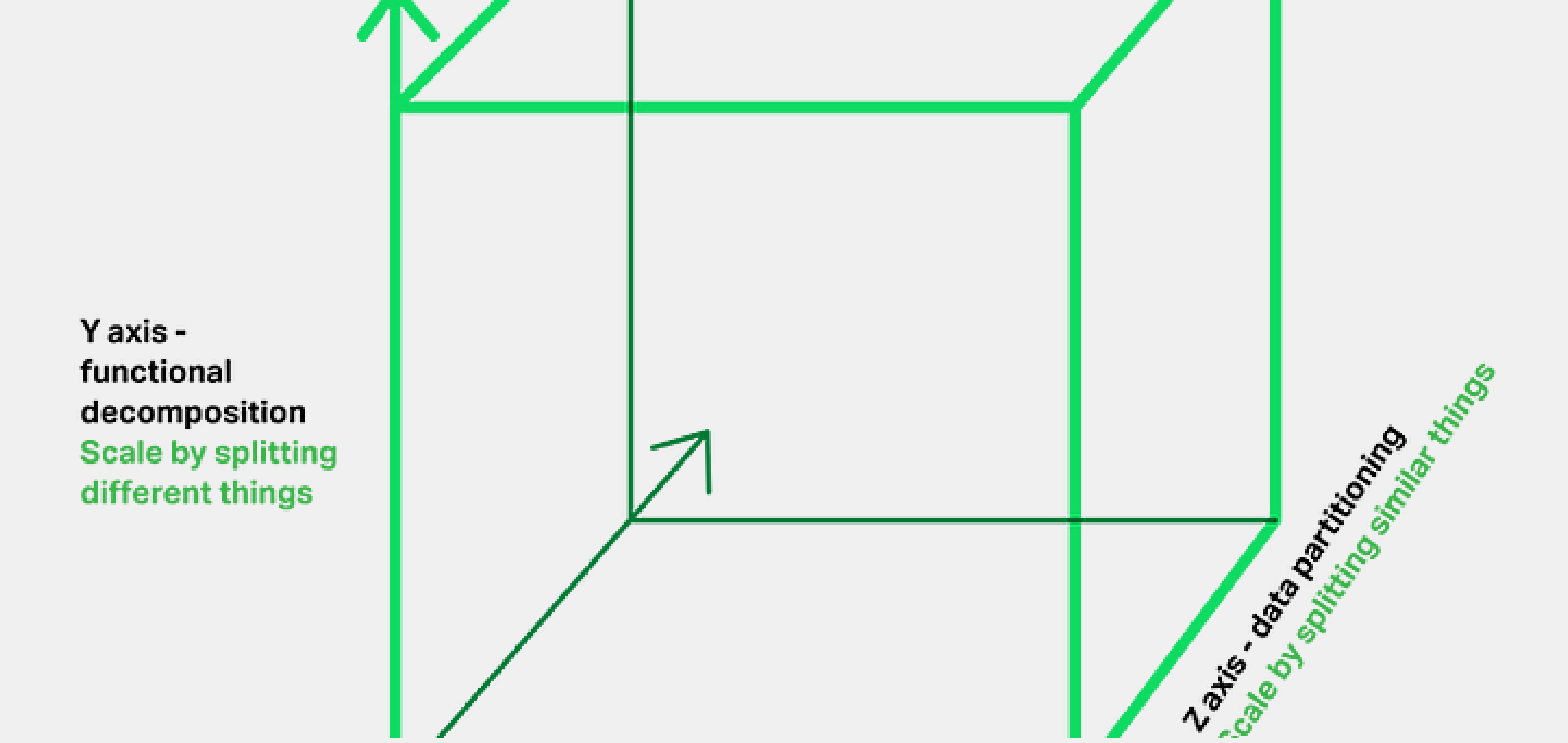
Leadership

48 stories · 269 saves



Coding & Development

11 stories · 513 saves


Y axis - functional decomposition
Scale by splitting different things
Z axis - data partitioning
Scale by splitting similar things

Avicsebooks

Achieving Scalability with Scale Cube

What is a Scale Cube

6 min read · Feb 10, 2024

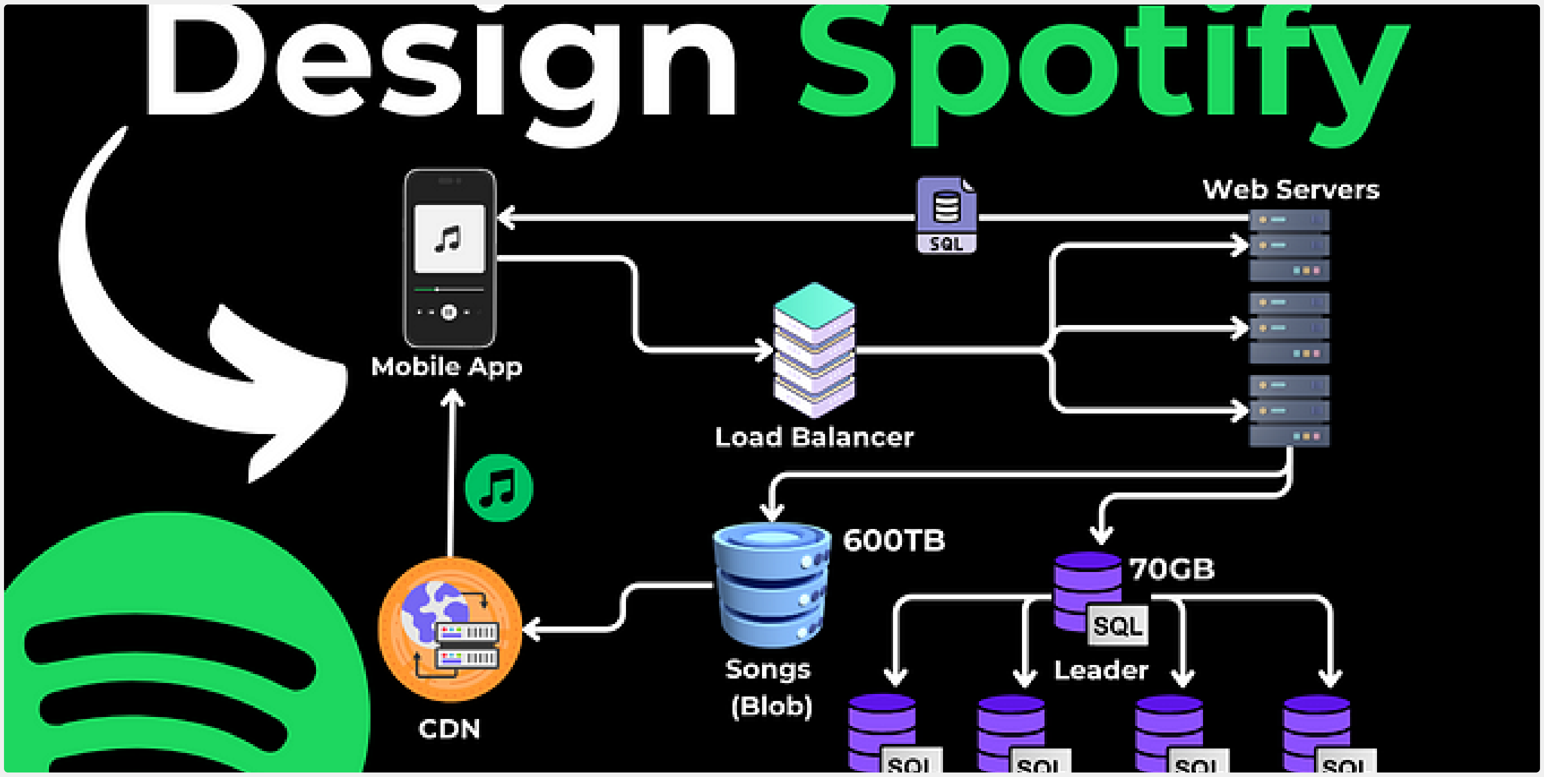
Grow Solutions

Functional and Non-Functional Requirements: The Ultimate Checklist with Examples

Introduction

18 min read · Nov 17, 2023

59



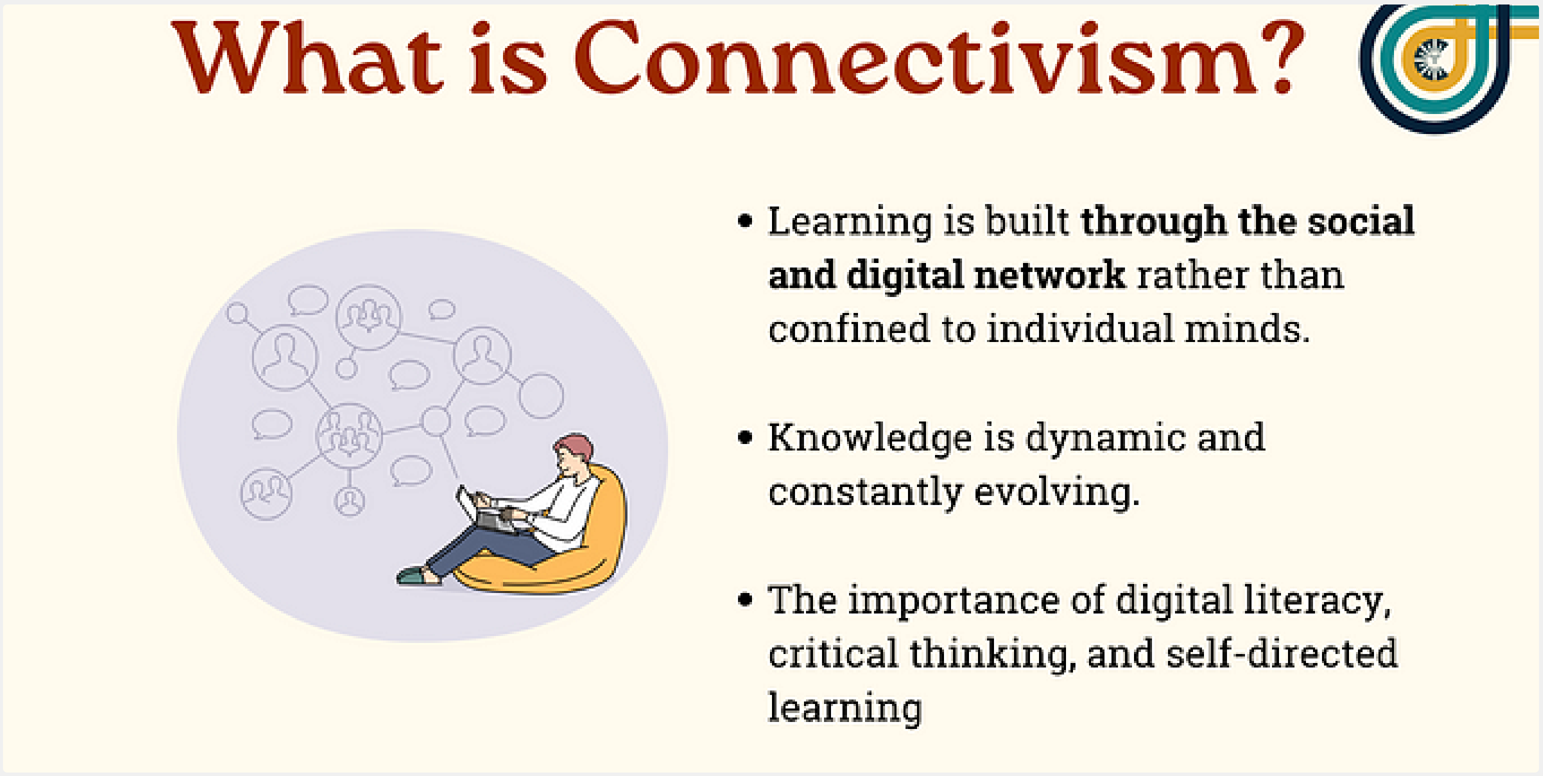
Hayk Simonyan in Level Up Coding

System Design Interview Question: Design Spotify

High-level overview of a System Design Interview Question - Design Spotify.

6 min read · Feb 21, 2024

3.8K 29



What is Connectivism?

- Learning is built **through the social and digital network** rather than confined to individual minds.
- Knowledge is dynamic and constantly evolving.
- The importance of digital literacy, critical thinking, and self-directed learning

YingYu Chen

#5 Learning Design: Connectivism (and my inpirational PLN)

Do you use social networks to learn and connect with people? Yes. That’s connectivism.

3 min read · Feb 24, 2024

50

See more recommendations