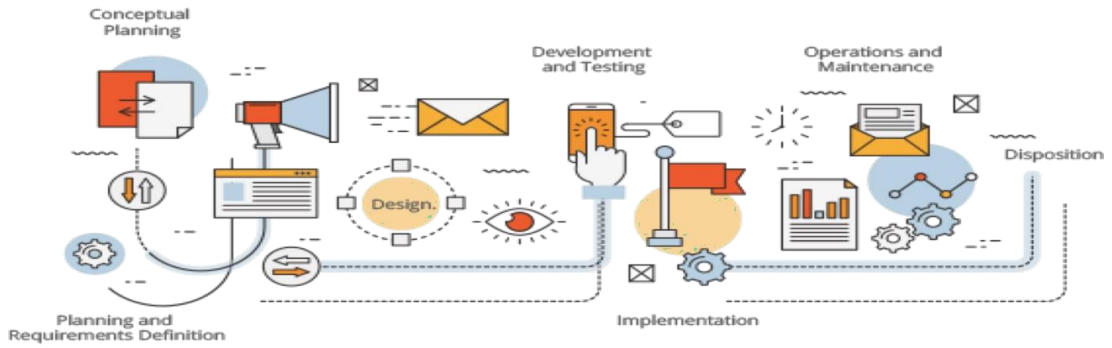
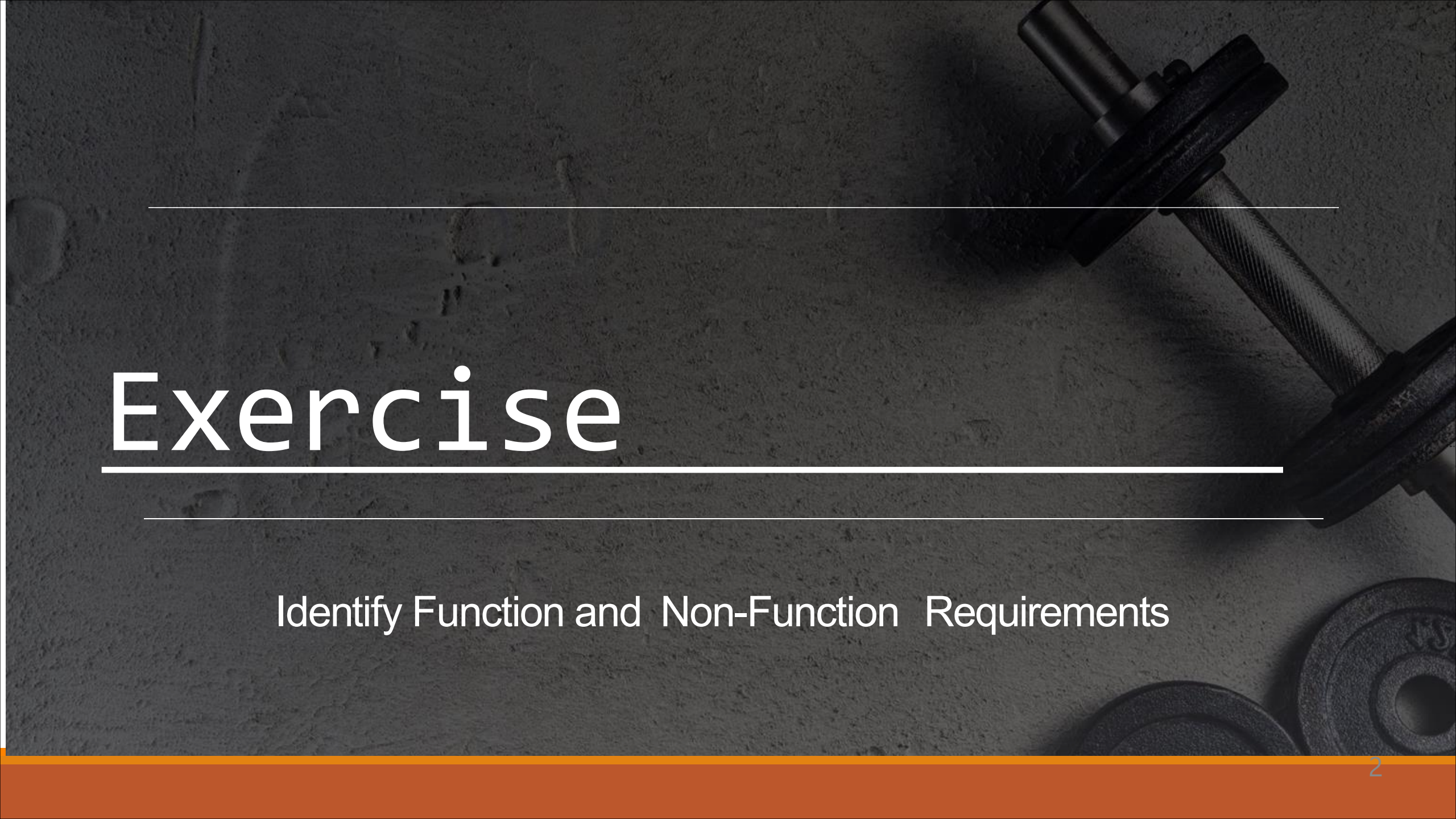
Software Engineering

Requirement Engineering





Exercise

Identify Function and Non-Function Requirements

WHAT ARE FUNCTIONAL REQUIREMENTS?

- "*Any Requirement Which Specifies* ***What The System Should Do.***"
- In other words, they are specific functionalities or actions that the system must perform, for example: "Send email when a new customer signs up" or "Open a new account".

WHAT ARE NON-FUNCTIONAL REQUIREMENTS?

- "*Any Requirement That Specifies **How** The System Performs A Certain Function.*"
- Non-functional requirements do not have an impact on the functionality of the system but they do impact on how it will perform. In short, non-functional requirements are all about system usability.

Identify Function and Non-Function Requirements

- **The software automatically validates customers against the ABC Contact Management System**

  q FR

- **Employees never allowed to update their salary information. Such attempt should be reported to the security administrator.**

  q FR

- **Users must change the initially assigned login password immediately after the first successful login. Moreover, the initial should never be reused.**

  q FR

- **The software system should be integrated with banking API**

  q FR

- **Every unsuccessful attempt by a user to access an item of data shall be recorded on an audit trail.**

  q NFR

- **The software should be portable. So moving from one OS to other OS does not create any problem.**

  q NFR

- **The software system should pass Section 508 accessibility requirement.**

  q NFR

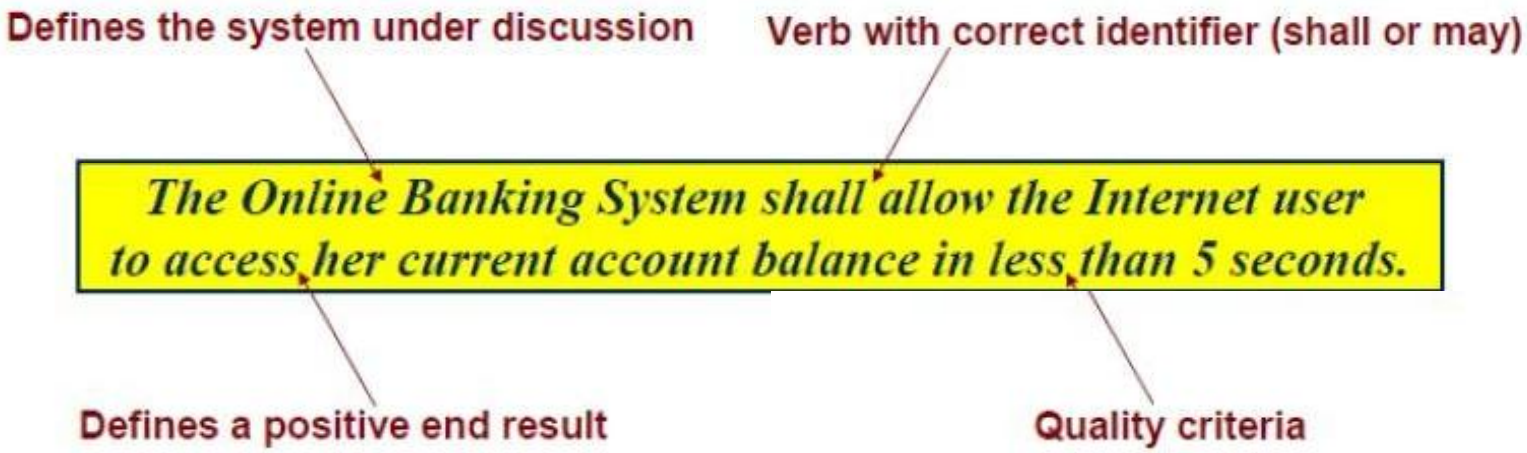
- **The background color for all windows in the application will be blue and have a hexadecimal RGB color value of 0x0000FF.**
  - NFR
- **A website should be capable enough to handle 20 million users with affecting its performance**
  - NFR
- **The Sales system should allow users to record customers sales**
  - FR

Parameters	Functional Requirement	Non-Functional Requirement
What it is	Verb	Attributes
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	Focus on user requirement	Concentrates on the user's expectation.
Documentation	Describe what the product does	Describes how the product works
Type of Testing	Functional Testing like System, Integration, End to End, API testing, etc.	Non-Functional Testing like Performance, Stress, Usability, Security testing, etc.
Test Execution	Test Execution is done before non-functional testing.	After the functional testing
Product Info	Product Features	Product Properties

How to write GOOD REQUIREMENTS

Constraints of a Good requirement

- Unambiguous
- Testable (verifiable)
- Clear (concise, terse, simple, precise)
- Correct
- Understandable
- Feasible (realistic, possible)
- Independent
- Atomic
- Necessary
- Implementation free (abstract)



- Identifies the system under discussion and a desired end result that is wanted within a specified time that is measurable

- The challenge is to seek out the system under discussion, end result, and success measure in every requirement

Writing a good requirement

Unambiguous

- There should be only one way to interpret the requirement. Sometimes ambiguity is introduced by undefined acronyms:
- ***REQ1: The system shall be implemented using ASP.***
- Does ASP mean Active Server Pages or Application Service Provider? To fix this, we can mention a full name and provide an acronym in parentheses:
- ***REQ1: The system shall be implemented using Active Server Pages (ASP).***
- Here's another example:
- ***REQ1: The system shall not accept passwords longer than 15 characters.***
- It is not clear what the system is supposed to do:
- **The system shall not let the user enter more than 15 characters.**
- **The system shall truncate the entered string to 15 characters.**
- **The system shall display an error message if the user enters more than 15 characters.**

Testable (Verifiable)

- *The system shall resist concurrent usage by many users.*
- What number should be considered “many”—10, 100, 1,000?
- Some words can make a requirement untestable
- Some adjectives: robust, safe, accurate, effective, efficient, expandable, flexible, maintainable, reliable, user-friendly, adequate
- Some adverbs and adverbial phrases: quickly, safely, in a timely manner
- Nonspecific words or acronyms: etc., and/or, TBD

Clear (Concise, Terse, Simple, Precise)

- Requirements should not contain unnecessary redundancy or information. They should be stated clearly and simply:

- *REQ1Sometimes the user will enter Airport Code, which the system will understand, but sometimes the closest city may replace it, so the user does not need to know what the airport code is, and it will still be understood by the system.*

- This sentence may be replaced by a simpler one:

- ***REQ1The system shall identify the airport based on either an Airport Code or a City Name.***

Correct

- If a requirement contains facts, these facts should be true:
- ***REQ1Car rental prices shall show all applicable taxes (including 6% state tax).***
- The tax depends on the state, so the provided 6% figure is incorrect.

Understandable

- Requirements should be grammatically correct and written in a consistent style. Standard conventions should be used. The word “shall” should be used instead of “will,” “must,” or “may.”

Feasible (Realistic, Possible)

- The requirement should be doable within existing constraints such as time, money, and available resources:
- ***REQ1: The system shall have a natural language interface that will understand commands given in English language.***
- This requirement may be not feasible within a short span of development time.

Independent

- To understand the requirement, there should not be a need to know any other requirement:
- ***REQ1: The list of available flights shall include flight numbers, departure time, and arrival time for every leg of a flight.***
- ***REQ2: It should be sorted by price.***
- The word “It” in the second sentence refers to the previous requirement. However, if the order of the requirements changes, this requirement will not be understandable.

Atomic

- The requirement should contain a single traceable element:

- ***REQ1The system shall provide the opportunity to book the flight, purchase a ticket, reserve a hotel room, reserve a car, and provide information about attractions.***

- This requirement combines five atomic requirements, which makes traceability very difficult. Sentences including the words “and” or “but” should be reviewed to see if they can be broken into atomic requirements.

Necessary

- A requirement is unnecessary if None of the stakeholders needs the requirement. Or Removing the requirement will not affect the system.

- An example of a requirement that is not needed by a stakeholder is a requirement that is added by developers and designers.
- For example, the fact that a developer thinks that users would like a feature that displays a map of the airport and he knows how to implement it is not a valid reason to add this requirement.

- An example of a requirement that can be removed because it does not provide any new information might look like the following:
- ***REQ1:All requirements specified in the Vision document shall be implemented and tested.***

Implementation free (Abstract)

- Requirements should not contain unnecessary design and implementation information:
- ***REQ1: Content information shall be stored in a text file.***
- How the information is stored is transparent to the user and should be the designer's or architect's decision.

Consistent

•There should not be any conflicts between the requirements. Conflicts may be direct or indirect. Direct conflicts occur when, in the same situation, different behavior is expected:

- **REQ1: Dates shall be displayed in the mm/dd/yyyy format.**
- **REQ2: Dates shall be displayed in the dd/mm/yyyy format.**
- **REQ1: For users in the U.S., dates shall be displayed in the mm/dd/yyyy format.**
- **REQ2: For users in France, dates shall be displayed in the dd/mm/yyyy format.**

Nonredundant

•Each requirement should be expressed only once and should not overlap with another requirement:

•***REQ1: A calendar shall be available to help with entering the flight date.***

•***REQ2: The system shall display a pop-up calendar when entering any date.***

•The first requirement (related to only the flight date) is a subset of the second one (related to any date entered by the user).

Complete

- A requirement should be specified for all conditions that can occur:

- **REQ1: A destination country does not need to be displayed for flights within the U.S.**
- **REQ2: For overseas flights, the system shall display a destination country.**

- What about flights to Canada and Mexico? They are neither “within the U.S.” nor “overseas.”
- All applicable requirements should be specified. This is the toughest condition to be checked. There is really no way to be sure that all the requirements are captured and that one week before the production date one of the stakeholders won’t say, “I forgot to mention that I need one more feature in the application.”

Conflicts

- Conflicts between different nonfunctional requirements are common in complex systems, Spacecraft system:
  - Ø To minimize weight, the number of separate chips in the system should be minimized
  - Ø To minimize power consumption, lower power chips should be used
  - Ø However, using low power chips may mean that more chips have to be used. Which is the most critical requirement?