Requirements Rules

This document outlines rules for writing high-quality requirements. Each rule is associated with a specific characteristic of a good requirement.

Index

C1: Necessary
RULE 20
RULE 30

C2:: Appropriate
RULE 2
RULE 3
RULE 31

C4: Complete
RULE 1
RULE 2
RULE 6
RULE 7
RULE 9
RULE 11
RULE 23
RULE 24
RULE 25
RULE 27
RULE 33
RULE 34
RULE 35
RULE 39
RULE 40
RULE 41

C5: Singular
RULE 9
RULE 18
RULE 19
RULE 20
RULE 21
RULE 22
RULE 23
RULE 39

C6:: Feasible
RULE 26
RULE 33

C7: Verifiable
RULE 1
RULE 2
RULE 3
RULE 4
RULE 5
RULE 6
RULE 7
RULE 8
RULE 9
RULE 10
RULE 11
RULE 12
RULE 13
RULE 15

RULE 16
RULE 17
RULE 18
RULE 24
RULE 26
RULE 27
RULE 28
RULE 31
RULE 32
RULE 33
RULE 34
RULE 35

C8 (Correct) Rules:
RULE 1
RULE 6
RULE 11
RULE 12
RULE 14
RULE 16
RULE 26
RULE 27
RULE 32
RULE 33
RULE 36

Characteristic: C1 - Necessary

Characteristic_Name: Necessary
Description: Requirements should define essential capabilities, functions, or constraints. Avoid including unnecessary information.

Rule: R20 - Purpose Phrases

Rule_ID: R20

Rule_Name: Purpose Phrases

Rule_Text: Eliminate any phrases that describe the "purpose of," "intent of," or "reason for" the requirement. Focus solely on the required behavior or characteristic.

Rule_Detail: Including purpose phrases can add unnecessary words and shift the focus away from the core requirement. The rationale behind a requirement should be captured in supporting documentation or attributes, not in the requirement statement itself.

Rule_Example: Unacceptable ("The Record_Subsystem shall display the Name of each Line_Item so that the Operator can confirm that it is the correct Item.")
Improved ("The Record_Subsystem shall display the Name of each Line_Item per <Display Standard XYZ>.")
Rationale (The text "so that the Operator can confirm that it is the correct Item" should be included in the rationale attribute.)

Rule: R30 - Unique Expression

Rule_ID: R30

Rule_Name: Unique Expression

Rule_Text: Ensure that each requirement is stated once and only once within the

documentation. Avoid redundancy.

Rule_Detail: Stating a requirement multiple times can lead to inconsistencies if one instance is updated and others are not. It also makes requirements management more complex.

Rule_Example: (This rule is more about document organization, so a before/after example isn't as applicable. Imagine a scenario where the same performance requirement is listed in both a "Functional Requirements" section and a "Performance Requirements" section. R30 dictates it should only be in one.)

Characteristic: C2 - Appropriate

Characteristic_Name: Appropriate

Description: Requirements should be suitable for the system level and avoid specifying design or implementation details.

Rule: R2 - Active Voice

Rule_ID: R2

Rule_Name: Active Voice

Rule_Text: Write requirements in the active voice. The entity responsible for fulfilling the requirement must be the subject of the sentence (e.g., "The system shall...", not "The action shall be done by the system...").

Rule_Detail: Active voice makes it clear who or what is responsible for carrying out the action, reducing ambiguity.

Rule_Example: Unacceptable ("During Login, the Identity of the Customer shall be confirmed.")
Improved ("The Accounting_System shall confirm the Customer_Identity.")

Rule: R3 - Clear Subject-Verb Agreement

Rule_ID: R3

Rule_Name: Clear Subject-Verb Agreement

Rule_Text: Verify that the subject and verb in each requirement statement agree grammatically (e.g., "The system shall perform...", not "The system must performing...")

Rule_Detail: Correct grammar is essential for clarity. Subject-verb agreement ensures the sentence is grammatically sound and avoids misinterpretation.

Rule_Example: Acceptable ("The <business> shall ..."; "The <business unit> shall ..."; "The <SOI> shall ..."; "The <subsystem> shall ..."; and "The <system element> shall ...")
Unacceptable (“The User shall …”)
Improved (“The <SOI> shall …”)

Rule: R31 - Solution-Free Language

Rule_ID: R31

Rule_Name: Solution-Free Language

Rule_Text: Phrase requirements to define what needs to be achieved, not how to achieve it. Avoid specifying design or implementation details (e.g., "The system shall respond in ≤ 30 seconds," not "The system shall use Algorithm X to respond...").

Rule_Detail: This rule promotes flexibility in design and avoids over-constraining the solution. Requirements should specify what the system must do, and not how it should do it.

Rule_Example: A requirement should state "The system shall provide a secure login" rather than "The system shall use RSA encryption for login".

Characteristic: C4 - Complete

Characteristic_Name: Complete

Description: Requirements should be fully stated and include all necessary information.

Rule: R1 - Structured Statements

Rule_ID: R1

Rule_Name: Structured Statements

Rule_Text: Formulate requirements as complete sentences adhering to a defined structure: "<subject (entity)> shall <verb (action)> <object> <response>". Use "shall" to denote mandatory actions.

Rule_Detail: A consistent structure makes requirements easier to understand and less prone to missing elements. The basic structure helps ensure all necessary parts of a requirement are present.

Rule_Example:

Rule: R2 - Active Voice

Rule_ID: R2

Rule_Name: Active Voice

Rule_Text: Use active voice, clearly identifying the responsible entity as the subject.

Rule_Detail: Active voice makes it clear who or what is responsible for carrying out the action, reducing ambiguity.

Rule_Example: Unacceptable ("During Login, the Identity of the Customer shall be confirmed.")
Improved ("The Accounting_System shall confirm the Customer_Identity.")

Rule: R6 - Common Units of Measure

Rule_ID: R6

Rule_Name: Common Units of Measure

Rule_Text: Express all quantities with SI/metric units.

Rule_Detail: Consistent units of measure are crucial for avoiding errors and misinterpretations, especially in engineering and scientific contexts.

Rule_Example: Unacceptable ("With power applied, The Circuit_Board shall <...> a temperature of less than 30 degrees.")
Improved ("With power applied, The Circuit_Board shall <...> a temperature of less than 30 degrees Celsius.")

Rule: R7 - Avoid Vague Terms

Rule_ID: R7

Rule_Name: Avoid Vague Terms

Rule_Text: Remove subjective or imprecise language. Replace terms like "some," "many," "approximately," "adequate," etc., with quantifiable measures.

Rule_Detail: Vague terms lead to ambiguity and make requirements difficult to verify. Requirements should be stated in a way that leaves no room for interpretation.

Rule_Example: The GPS shall display the User_Location in accordance with <Display Standard XYZ>.

Rule: R9 - Avoid Open-Ended Clauses

Rule_ID: R9

Rule_Name: Avoid Open-Ended Clauses

Rule_Text: Do not use phrases that imply an undefined continuation, such as "including but not limited to," "etc.," or "and so on."

Rule_Detail: Open-ended clauses make it unclear what is and isn't included in the requirement, leading to potential disputes and verification problems.

Rule_Example: Unacceptable ("The system shall process data, including user input, sensor readings, etc.")
Improved(
"The system shall process user input."
"The system shall process sensor readings.")

Rule: R11 - Separate Clauses

Rule_ID: R11

Rule_Name: Separate Clauses

Rule_Text: Write one requirement per sentence. Do not combine multiple requirements into a single statement.

Rule_Detail: Combining multiple requirements makes them harder to trace, allocate, and verify. Each requirement should be distinct and stand on its own.

Rule_Example: Instead of "The system shall log in the user and display the dashboard," write two separate requirements.

Rule: R23 - Supporting Diagrams/Models

Rule_ID: R23

Rule_Name: Supporting Diagrams/Models

Rule_Text: Note if the requirement is better expressed or clarified by a diagram, model, or Interface Control Document (ICD), though you are not generating these.

Rule_Detail: While the requirement itself is text-based, this rule acknowledges that visual aids can be essential for understanding complex relationships or behaviors.

Rule_Example: "An example is a requirement on voltage at turn on which involves a magnitude, rise time, overshoot, and dampening time along with tolerances." (A diagram would be very helpful here)

Rule: R24 - Pronouns

Rule_ID: R24

Rule_Name: Pronouns

Rule_Text: Avoid using pronouns like "it," "they," "this," or "that." Always use the specific entity or object.

Rule_Detail: Pronouns can be ambiguous, especially in complex documents. Using the specific name avoids any confusion about what is being referred to.

Rule_Example: Instead of "It shall display the data," write "The System shall display the data."

Rule: R25 - Headings

Rule_ID: R25

Rule_Name: Headings

Rule_Text: Ensure the requirement is understandable without relying on any section headings.

Rule_Detail: Requirements should be self-contained. If a requirement relies on a heading for context, it is not complete.

Rule_Example: A requirement in a "Security" section should still explicitly state any security-related conditions, rather than assuming the reader will infer them from the heading.

Rule: R27 - Explicit Conditions

Rule_ID: R27

Rule_Name: Explicit Conditions

Rule_Text: Clearly define all triggering conditions and events. Avoid ambiguity in describing when the requirement applies.

Rule_Detail: Many requirements are conditional (they only apply under certain

circumstances). These conditions must be clearly stated to avoid uncertainty.

Rule_Example: "If a condition that applies to a need or requirement is not stated explicitly within the need statement or requirement statement, the need or requirement statement is not Complete (C4), Verifiable/Validatable (C7), nor Correct (C8) unless the condition clause is included."

Rule: R33 - Range of Values

Rule_ID: R33

Rule_Name: Range of Values

Rule_Text: Include acceptable tolerances or ranges for all quantitative requirements.

Rule_Detail: In the real world, values are rarely exact. Specifying a range or tolerance makes the requirement more realistic and verifiable.

Rule_Example: "The system shall measure the temperature with an accuracy of ±0.5 degrees Celsius."

Rule: R34 - Measurable Performance

Rule_ID: R34

Rule_Name: Measurable Performance

Rule_Text: Define requirements in terms of quantifiable metrics.

Rule_Detail: This is closely related to avoiding vague terms. Requirements should define how much or how fast, not just that something should happen.

Rule_Example: Unacceptable (The <SOI> shall be online.)
Improved (The <SOI> shall have an Availability of greater than xx% over a period of greater than yyy hours.)

Rule: R35 - Temporal Dependencies

Rule_ID: R35

Rule_Name: Temporal Dependencies

Rule_Text: Specify timing and sequence constraints clearly (e.g., "within 10 seconds," "before event X," "after step Y").

Rule_Detail: Many systems operate in time-critical environments. Temporal dependencies ensure that timing and sequencing are correctly defined.

Rule_Example: "The system shall display the alert within 2 seconds of receiving the sensor data."

Rule: R39 - Style Guide

Rule_ID: R39

Rule_Name: Style Guide

Rule_Text: Adhere to a consistent style guide (if provided; otherwise, apply generally accepted English grammar and technical writing conventions).

Rule_Detail: A style guide ensures uniformity in formatting, terminology, and overall presentation, making the document easier to read and understand. This reduces the chance of misinterpretations due to inconsistent formatting.

Rule_Example: A style guide might specify (
How to format headings and subheadings.
Whether to use the Oxford comma.
How to format tables and figures.
Preferred terminology for certain concepts.)

Rule: R40 - Decimal Format

Rule_ID: R40

Rule_Name: Decimal Format

Rule_Text: Follow a consistent decimal format (if specified).

Rule_Detail: Consistency in decimal formatting is crucial, especially in technical documents with many numerical values. This avoids confusion and potential errors in interpreting the numbers.

Rule_Example: Some regions use a comma as the decimal separator (e.g., 1,234.56), while others use a period (e.g., 1.234,56). A style guide should specify which to use.
Consistent use of leading zeros (e.g., 0.5 instead of .5)

Rule: R41 - Related Needs

Rule_ID: R41

Rule_Name: Related Needs

Rule_Text: (If applicable) Note the relationship of the requirement to its originating need.

Rule_Detail: Tracing requirements back to their originating needs provides context and justification for the requirement. This traceability is essential for requirements management and change control.

Rule_Example: Include a field or attribute in the requirement that references the specific need that the requirement fulfills.

Characteristic: C5 - Singular

Characteristic_Name: Singular

Description: Requirements should express a single thought or assertion. Avoid combining multiple requirements into one.

Rule: R9 - Open-ended Clauses

Rule_ID: R9

Rule_Name: Open-ended Clauses

Rule_Text: Avoid open-ended, non-specific clauses such as “including but not limited to”, “etc.” and “and so on”. Open-ended clauses imply there is more required without stating exactly what.

Rule_Detail: Open-ended clauses make it unclear what is and isn't included in the requirement, leading to potential disputes and verification problems.

Rule_Example: Unacceptable ("The system shall process data, including user input, sensor readings, etc.")
Improved(
"The system shall process user input."
"The system shall process sensor readings.")

Rule: R18 - Single Thought Sentence

Rule_ID: R18

Rule_Name: Single Thought Sentence

Rule_Text: Express only one requirement or assertion per sentence. (e.g., "The system shall do X" is acceptable. "The system shall do X and Y" is not). This makes the requirement easier to understand and verify.

Rule_Detail: A sentence with multiple "shall" statements implies multiple requirements. Keeping it to one makes it singular and easier to manage.

Rule_Example: Unacceptable ("The system shall log the user in and display the main menu.")
Improved ("The system shall log the user in.")
("The system shall display the main menu after successful login.")

Rule: R19 - Combinators

Rule_ID: R19

Rule_Name: Combinators

Rule_Text: Avoid using conjunctions (and, or) to combine multiple requirements. (e.g., "The system shall do A and the system shall do B" should be two separate requirements). This ensures each requirement is distinct.

Rule_Detail: Similar to R18, conjunctions combine actions, making it unclear if they are truly separate requirements.

Rule_Example: Unacceptable ("The system shall store data and generate reports.")
Improved ("The system shall store data.")
("The system shall generate reports.")

Rule: R20 - Purpose Phrases

Rule_ID: R20

Rule_Name: Purpose Phrases

Rule_Text: Avoid phrases that state the purpose of the need statement or requirement statement. Such phrases include “for the purpose of’, “in order to”, “so that”, “as a means to”, “intended to”, “used to”, and “to”.

Rule_Detail: As previously mentioned, these phrases add extra information that doesn't define the core requirement itself, violating the "singular" focus.

Rule_Example: Unacceptable ("The system shall display the alert so that the operator is notified.")
Improved ("The system shall display the alert.")

Rule: R21 - Parentheses

Rule_ID: R21

Rule_Name: Parentheses

Rule_Text: Use parentheses carefully to avoid introducing ambiguity or multiple interpretations. (e.g., "The system shall (if condition A) do X" is acceptable if clear. "The system shall do X (Y or Z)" could be ambiguous). This ensures clarity in conditional statements.

Rule_Detail: Parentheses can create branching logic within a requirement, making it less singular if not used carefully.

Rule_Example: Acceptable - "The system shall (if the user is an administrator) display the settings menu."
Ambiguous - "The system shall display the data (in a table or a chart)." (Is it one requirement with two display options, or two separate display requirements?)

Rule: R22 - Enumeration

Rule_ID: R22

Rule_Name: Enumeration

Rule_Text: If listing items, ensure it doesn't create multiple implicit requirements. (e.g., "The system shall support A, B, and C" is acceptable if the support is the same for all. If the support differs, separate requirements are needed). This avoids hidden requirements.

Rule_Detail: Enumeration is acceptable if the action applies uniformly to all items. If there are variations, it's not truly singular.

Rule_Example: Acceptable - "The system shall support login via username, email, and phone number." (Same "support" for all).
Unacceptable - "The system shall process orders, invoices, and reports." (Processing might be different for each).

Rule: R23 - Supporting Diagram, Model, or ICD

Rule_ID: R23

Rule_Name: Supporting Diagram, Model, or ICD

Rule_Text: Note the need for these. (e.g., if describing a complex interface, state "See ICD-123"). This acknowledges when visual aids are necessary for clarity.

Rule_Detail: While the requirement itself is singular, this rule recognizes that complex requirements may need external clarification, which doesn't violate singularity.

Rule_Example: "The system shall communicate with the external device (see Interface Control Document XYZ)."

Rule: R39 - Style Guide

Rule_ID: R39

Rule_Name: Style Guide

Rule_Text: Adhere to a consistent style guide. (e.g., formatting, terminology). This ensures uniformity and readability.

Rule_Detail: As previously mentioned, a style guide contributes to clarity and consistency, which indirectly supports the "singular" characteristic by reducing the chance of misinterpretation.

Characteristic: C6 - Feasible

Characteristic_Name: Feasible
Description: Requirements should be achievable within the constraints of the project.

Rule: R26 - Avoid Absolutes

Rule_ID: R26

Rule_Name: Avoid Absolutes

Rule_Text: Do not use terms that imply impossibility or unachievability, such as "all," "always," "every," or "never." (e.g., "The system shall always respond" is often unrealistic. "The system shall respond within 5 seconds" is more feasible). This ensures requirements are realistic.

Rule_Detail: Absolute terms often set unrealistic expectations that cannot be met within project constraints (time, budget, technology).

Rule_Example: Unfeasible - "The system shall always be available 24/7 with zero downtime."
Feasible - "The system shall have an availability of 99.9%."

Rule: R33 - Range of Values

Rule_ID: R33

Rule_Name: Range of Values

Rule_Text: Include tolerances to reflect real-world limitations. (e.g., manufacturing tolerances, environmental variations). This acknowledges that perfect values are rarely achievable.

Rule_Detail: Tolerances acknowledge that systems operate within a range of acceptable values, making requirements more realistic and feasible to achieve and verify.

Rule_Example: Unfeasible - ("The component shall have a weight of exactly 10 grams.")
Feasible - ("The component shall have a weight of 10 ± 0.5 grams.")

Characteristic: C7 - Verifiable

Characteristic_Name: Verifiable
Description: Requirements should be stated in a way that can be objectively verified.

Rule: R1 - Structured Statements

Rule_ID: R1

Rule_Name: Structured Statements

Rule_Text: Use the specified sentence structure: "<subject (entity)> shall <verb (action)> <object> <response>". This provides a consistent framework for verification.

Rule_Detail: A structured statement makes it clear what needs to be verified. Each element (subject, verb, object, response) provides a specific point of verification.

Rule_Example: "The system shall display the user's name within 2 seconds of login."
Subject - The system
Verb - shall display
Object - the user's name
Response - within 2 seconds of login
Each of these can be verified.

Rule: R2 - Active Voice

Rule_ID: R2

Rule_Name: Active Voice

Rule_Text: Use active voice. (e.g., "The system shall transmit data" is verifiable. "Data shall be transmitted" is less clear on who transmits). This clarifies who performs the action, aiding in verification.

Rule_Detail: Active voice clearly identifies the actor, making it easier to determine who is responsible for the action and how to verify it.

Rule_Example: Verifiable - ("The server shall send the email notification." (We verify the server))
Less Verifiable - ("The email notification shall be sent." (Who sends it?))

Rule: R3 - Appropriate Subject-Verb

Rule_ID: R3

Rule_Name: Appropriate Subject-Verb

Rule_Text: Ensure the subject (system, user, etc.) is logically capable of performing the action. (e.g., "The user shall enter the password" is appropriate. "The system shall enter the password" is not). This ensures logical consistency.

Rule_Detail: The subject must be able to perform the verb. This ensures that the requirement makes sense and can be verified.

Rule_Example: Appropriate - "The operator shall initiate the shutdown sequence."

(Operators can do this)
Inappropriate - "The sensor shall display the data." (Sensors measure, they don't display)

Rule: R4 - Defined Terms

Rule_ID: R4

Rule_Name: Defined Terms

Rule_Text: Highlight any technical terms that require definition in a glossary. (e.g., if using "widget," ensure "widget" is defined). This avoids ambiguity.

Rule_Detail: If terms are not defined, verification becomes subjective. Definitions provide a clear basis for verifying compliance.

Rule_Example: If the requirement uses "response time," the glossary must define exactly what "response time" means (e.g., from user input to screen display).

Rule: R5 - Definite Articles

Rule_ID: R5

Rule_Name: Definite Articles

Rule_Text: Use "the" instead of "a" or "an" for specific entities. (e.g., "The system shall display the error message" is specific. "The system shall display an error message" is less precise). This improves clarity. [cite: 43, 44, 45]

Rule_Detail: "The" implies a specific instance, making verification clearer. "A" or "an" can be vague.

Rule_Example: Specific - "The system shall display the low battery warning." (A particular warning)
Vague - "The system shall display an error message." (Any error message?)

Rule: R6 - Common Units of Measure

Rule_ID: R6

Rule_Name: Common Units of Measure

Rule_Text: Use SI/metric units. (e.g., "meters," "seconds," "Jos"). This allows for objective measurement. [cite: 45, 46]

Rule_Detail: Standard units are essential for objective verification. Non-standard units can lead to confusion and errors.

Rule_Example: "The system shall operate at a frequency of 50 Hz" (Hertz is a standard unit).

Rule: R7 - Vague Terms

Rule_ID: R7

Rule_Name: Vague Terms

Rule_Text: Avoid vague terms. (e.g., replace "user-friendly" with specific usability criteria). This ensures testability. [cite: 46, 47]

Rule_Detail: Vague terms are subjective and hard to verify. Replace them with measurable criteria.

Rule_Example: Vague - "The system shall be easy to use."
Verifiable - "The system shall allow a user to complete a transaction in under 3 minutes."

Rule: R8 - Escape Clauses

Rule_ID: R8

Rule_Name: Escape Clauses

Rule_Text: Eliminate phrases that weaken the enforceability of the requirement. (e.g., remove "if possible," "as appropriate"). This makes the requirement mandatory. [cite: 47, 48]

Rule_Detail: These clauses provide loopholes, making the requirement optional and difficult to verify.

Rule_Example: Weak - "The system shall display the alert if possible."
Strong - "The system shall display the alert."

Rule: R9 - Open-Ended Clauses

Rule_ID: R9

Rule_Name: Open-Ended Clauses

Rule_Text: Avoid open-ended clauses. (e.g., remove "etc.," "and so on"). This ensures the requirement is fully defined. [cite: 48, 49]

Rule_Detail: As previously discussed, these clauses make it unclear what needs to be verified.

Rule_Example: Incomplete - "The system shall process data, including temperature, pressure, etc."
Complete - "The system shall process temperature and pressure data."

Rule: R10 - Superfluous Infinitives

Rule_ID: R10

Rule_Name: Superfluous Infinitives

Rule_Text: Remove unnecessary uses of "to be able to," "to be capable of," etc. (e.g., "The system shall process data" is better than "The system shall be able to process data"). This makes the requirement concise. [cite: 49, 50]

Rule_Detail: These phrases add unnecessary words without adding meaning. Conciseness improves clarity.

Rule_Example: Wordy - "The system shall be capable of storing 1TB of data."
Concise - "The system shall store 1TB of data."

Rule: R11 - Separate Clauses

Rule_ID: R11

Rule_Name: Separate Clauses

Rule_Text: Use one clause per requirement. (e.g., separate "The system shall do X and Y" into two requirements). This allows for independent verification.

Rule_Detail: If a requirement has multiple parts, it's harder to verify each part individually. Separation allows for focused verification.

Rule_Example: Combined - "The system shall log in the user and display the profile."
Separate -
"The system shall log in the user."
"The system shall display the user's profile after successful login."

Rule: R12 - Correct Grammar

Rule_ID: R12

Rule_Name: Correct Grammar

Rule_Text: Ensure grammatical accuracy. This avoids misinterpretation.

Rule_Detail: Grammatical errors can change the meaning of a requirement, making it difficult to verify the intended behavior.

Rule_Example: Subject-verb agreement, correct tense, etc.

Rule: R13 - Correct Spelling

Rule_ID: R13

Rule_Name: Correct Spelling

Rule_Text: Ensure correct spelling. This avoids errors and improves clarity.

Rule_Detail: Misspellings can lead to confusion and make the document unprofessional.

Rule: R15 - Logical Expressions

Rule_ID: R15

Rule_Name: Logical Expressions

Rule_Text: Use quantifiable logic. (e.g., use Boolean operators, mathematical inequalities). This allows for precise testing.

Rule_Detail: Logic expressions (AND, OR, NOT, >, <, =) provide precise conditions for verification.

Rule_Example: "The system shall activate the alarm if (temperature > 100°C) AND (pressure > 50 psi)."

Rule: R16 - Use of “Not”

Rule_ID: R16

Rule_Name: Use of “Not”

Rule_Text: Minimize the use of negative statements. Phrase requirements

positively. (e.g., "The system shall display an error" is better than "The system shall not fail to display an error"). This improves clarity and verifiability.

Rule_Detail: Positive statements are generally easier to verify than negative ones. It's easier to prove something does happen than that it doesn't happen.

Rule_Example: Negative - ("The system shall not allow invalid input.")
Positive - ("The system shall validate input and display an error message for invalid input.")

Rule: R17 - Use of Oblique Symbol

Rule_ID: R17

Rule_Name: Use of Oblique Symbol

Rule_Text: Avoid using "/" to mean "and/or." (e.g., write "The system shall do A and B" or "The system shall do A or B" instead of "The system shall do A/B"). This avoids ambiguity.

Rule_Detail: "/" is ambiguous. "And" and "or" have precise logical meanings.

Rule_Example: Ambiguous - "The system shall support user authentication via password/biometrics."
Clear - "The system shall support user authentication via password or biometrics." (Either is acceptable)
"The system shall support user authentication via password and biometrics." (Both are required)

Rule: R18 - Single Thought Sentence

Rule_ID: R18

Rule_Name: Single Thought Sentence

Rule_Text: Use single-thought sentences. (one requirement per sentence). This allows for focused verification.

Rule_Detail: A sentence with multiple "shall" statements implies multiple requirements. Keeping it to one makes it singular and easier to manage.

Rule_Example: Unacceptable ("The system shall log the user in and display the main menu.")
Improved ("The system shall log the user in.")
("The system shall display the main menu after successful login.")

Rule: R24 - Pronouns

Rule_ID: R24

Rule_Name: Pronouns

Rule_Text: Avoid pronouns. (use specific nouns instead of "it," "they," etc.). This removes ambiguity.

Rule_Detail: Pronouns can be ambiguous, especially in complex documents. Using the specific name avoids any confusion about what is being referred to.

Rule_Example: Instead of "It shall display the data," write "The System shall

display the data."

Rule: R26 - Absolutes

Rule_ID: R26

Rule_Name: Absolutes

Rule_Text: Avoid absolutes. (use "each" instead of "all," "every," "always," "never"). This makes the requirement more realistic.

Rule_Detail: Absolute terms often set unrealistic expectations that cannot be met within project constraints (time, budget, technology).

Rule_Example: Unfeasible - "The system shall always be available 24/7 with zero downtime."
Feasible - "The system shall have an availability of 99.9%."

Rule: R27 - Explicit Conditions

Rule_ID: R27

Rule_Name: Explicit Conditions

Rule_Text: Define conditions explicitly. (e.g., "When the user clicks button X," not "Under certain circumstances"). This makes the requirement testable.

Rule_Detail: Many requirements are conditional (they only apply under certain circumstances). These conditions must be clearly stated to avoid uncertainty.

Rule_Example: "If a condition that applies to a need or requirement is not stated explicitly within the need statement or requirement statement, the need or requirement statement is not Complete (C4), Verifiable/Validatable (C7), nor Correct (C8) unless the condition clause is included."

Rule: R28 - Multiple Conditions

Rule_ID: R28

Rule_Name: Multiple Conditions

Rule_Text: Break down requirements with complex conditions. (e.g., separate "If A and B or C, then do X" into simpler requirements). This improves clarity and verifiability.

Rule_Detail: Complex logic is hard to test. Breaking it down simplifies verification.

Rule_Example: Complex - "If (user is admin AND time is after 5 PM) OR (system is in maintenance mode), then display the warning."
Simpler -
"If the user is an administrator and the time is after 5 PM, then the system shall display the warning."
"If the system is in maintenance mode, then the system shall display the warning."

Rule: R31 - Solution Free

Rule_ID: R31

Rule_Name: Solution Free

Rule_Text: Keep requirements solution-free. (describe what, not how). This allows for design flexibility.

Rule_Detail: If a requirement specifies a solution, you can only verify that specific solution, not other ways of meeting the need.

Rule_Example: Solution-specific - "The system shall use RSA encryption to secure communication."
Solution-free - "The system shall secure communication using industry-standard encryption."

Rule: R32 - Universal Qualification

Rule_ID: R32

Rule_Name: Universal Qualification

Rule_Text: Use "each" instead of "all," "any," or "both" when appropriate. This ensures the requirement applies to individual instances.

Rule_Detail: "Each" clarifies that every item must meet the requirement, not just the set as a whole.

Rule_Example: Ambiguous - "All messages shall be encrypted." (Does this mean the entire set of messages, or every single message?)
Clear - "Each message shall be encrypted."

Rule: R33 - Range of Values

Rule_ID: R33

Rule_Name: Range of Values

Rule_Text: Include ranges and tolerances. (e.g., "5V ± 0.1V"). This reflects real-world variability.

Rule_Detail: Real-world measurements are never exact. Ranges and tolerances make verification realistic.

Rule_Example: "The temperature shall be between 20°C and 25°C."

Rule: R34 - Measurable Performance

Rule_ID: R34

Rule_Name: Measurable Performance

Rule_Text: Use quantifiable metrics. (e.g., "100 transactions per second"). This allows for objective measurement.

Rule_Detail: Quantifiable metrics provide specific values to verify.

Rule_Example: Non-quantifiable - "The system shall be fast."
Quantifiable - "The system shall process a request in under 1 second."

Rule: R35 - Temporal Dependencies

Rule_ID: R35

Rule_Name: Temporal Dependencies

Rule_Text: Define time dependencies. (e.g., "within 2 seconds," "after event Y"). This clarifies timing requirements.

Rule_Detail: Timing is crucial for many systems. These dependencies specify when actions should occur.

Rule_Example: "The system shall display the error message within 5 seconds of the invalid input."

Characteristic: C8 - Correct

Characteristic_Name: Correct
Description: Requirements should be accurate, unambiguous, and free from errors.

Rule: R1 - Structured Statements

Rule_ID: R1

Rule_Name: Structured Statements

Rule_Text: Use the correct sentence structure: "<subject (entity)> shall <verb (action)> <object> <response>". This ensures a standard format.

Rule_Detail: A structured statement makes it clear what needs to be verified. Each element (subject, verb, object, response) provides a specific point of verification.

Rule_Example: "The system shall display the user's name within 2 seconds of login."
Subject - The system
Verb - shall display
Object - the user's name
Response - within 2 seconds of login
Each of these can be verified.

Rule: R6 - Units of Measure

Rule_ID: R6

Rule_Name: Units of Measure

Rule_Text: Explicitly state units when stating quantities. (e.g., "10 meters," "5 seconds," not just "10" or "5"). This avoids ambiguity.

Rule_Detail: Consistent units of measure are crucial for avoiding errors and misinterpretations, especially in engineering and scientific contexts.

Rule_Example: Unacceptable ("With power applied, The Circuit_Board shall <...> a temperature of less than 30 degrees.")
Improved ("With power applied, The Circuit_Board shall <...> a temperature of less than 30 degrees Celsius.")

Rule: R11 - Separate Clauses

Rule_ID: R11

Rule_Name: Separate Clauses

Rule_Text: Separate combined conditions. (e.g., "If A, then X. If B, then Y" instead of "If A and B, then X and Y"). This allows for independent verification.

Rule_Detail: If a requirement has multiple parts, it's harder to verify each part individually. Separation allows for focused verification.

Rule_Example: Combined - "The system shall log in the user and display the profile."
Separate -
"The system shall log in the user."
"The system shall display the user's profile after successful login."

Rule: R12 - Correct Grammar

Rule_ID: R12

Rule_Name: Correct Grammar

Rule_Text: Use correct grammar. This ensures the requirement is understandable.

Rule: R14 - Correct Punctuation

Rule_ID: R14

Rule_Name: Correct Punctuation

Rule_Text: Use correct punctuation. This avoids misinterpretation.

Rule: R16 - Avoid "Not"

Rule_ID: R16

Rule_Name: Avoid "Not"

Rule_Text: Minimize negative statements. (e.g., phrase as "The system shall do X" instead of "The system shall not fail to do X"). This improves clarity.

Rule_Detail: Positive statements are generally easier to verify than negative ones. It's easier to prove something does happen than that it doesn't happen.

Rule_Example: Negative - ("The system shall not allow invalid input.")
Positive - ("The system shall validate input and display an error message for invalid input.")

Rule: R26 - Avoid Absolutes

Rule_ID: R26

Rule_Name: Avoid Absolutes

Rule_Text: Avoid absolutes. (e.g., use "each" or "every instance" instead of "always"). This makes the requirement realistic.

Rule_Detail: Absolute terms often set unrealistic expectations that cannot be met within project constraints (time, budget, technology).

Rule_Example: Unfeasible - "The system shall always be available 24/7 with zero downtime."

Feasible - "The system shall have an availability of 99.9%."

Rule: R27 - Explicit Conditions

Rule_ID: R27

Rule_Name: Explicit Conditions

Rule_Text: Be explicit about conditions. (e.g., "When the user does A," not "Under certain conditions"). This makes the requirement testable.

Rule_Detail: Many requirements are conditional (they only apply under certain circumstances). These conditions must be clearly stated to avoid uncertainty.

Rule_Example: "If a condition that applies to a need or requirement is not stated explicitly within the need statement or requirement statement, the need or requirement statement is not Complete (C4), Verifiable/Validatable (C7), nor Correct (C8) unless the condition clause is included."

Rule: R32 - Universal Qualification

Rule_ID: R32

Rule_Name: Universal Qualification

Rule_Text: Avoid jargon or terms that require specific domain knowledge without definition. This ensures the requirement is understandable to all stakeholders.

Rule_Detail: "Each" clarifies that every item must meet the requirement, not just the set as a whole.

Rule_Example: Ambiguous - "All messages shall be encrypted." (Does this mean the entire set of messages, or every single message?)
Clear - "Each message shall be encrypted."

Rule: R33 - Range of Values

Rule_ID: R33

Rule_Name: Range of Values

Rule_Text: Add tolerances to numerical values. (e.g., "10 ± 0.5"). This reflects real-world variability.

Rule_Detail: Real-world measurements are never exact. Ranges and tolerances make verification realistic.

Rule_Example: "The temperature shall be between 20°C and 25°C."

Rule: R36 - Consistent Terms/Units

Rule_ID: R36

Rule_Name: Consistent Terms/Units

Rule_Text: Use consistent terminology and units throughout the document. (e.g., always use "transmit" or "send," not a mix of both). This avoids confusion.

Rule_Detail: Inconsistent use of terms or units introduces ambiguity and increases the chance of errors in interpreting and implementing the

requirements.

Rule_Example: E.g (Always use "meters" and "seconds," not "m" and "sec" inconsistently.)

