Writing Good requirements techniques

Introduction to Writing Good Requirements Techniques

Writing clear and testable requirements ensures that the system can be properly validated

1. Why Good Requirements Matter:

- Prevents misunderstandings and ensures all stakeholders are on the same page.
- Reduces project risk by ensuring that developers, testers, and designers know exactly what is expected.
- Makes requirements testable, traceable, and verifiable.

2. Common Pitfalls in Requirements Writing:

- **Ambiguity**: Vague terms that can be interpreted in multiple ways (e.g., “fast,” “user-friendly”).
- **Incomplete requirements**: Missing critical information (e.g., conditions under which a function should occur).
- **Non-testable requirements**: Requirements that cannot be verified through tests or measurable criteria.

3. Techniques for Writing Good Requirements:

- Use clear and concise language to ensure understanding.
- Write testable requirements: Every requirement must be verifiable through inspection or tests.
- Ensure traceability: Requirements should be linked to higher-level business needs or stakeholder requirements.

Bad Example	Good Example
1. The system should be fast.	1. The system shall respond to user inputs within 2 seconds.
2. The UI must be user-friendly.	2. The UI shall allow users to navigate between screens with fewer than 3 clicks.
3. The app should work offline.	3. The app shall retain core functionality (viewing documents) when offline.
4. The system should be scalable.	4. The system shall handle up to 10,000 concurrent users without performance degradation.
5. The data must be secure.	5. All user data shall be encrypted using 256-bit AES encryption.
6. The system should handle large files.	6. The system shall process files up to 2GB without performance degradation.
7. The system should support different formats.	7. The system shall support uploading and processing files in .pdf, .docx, and .txt formats.
8. The system should allow customization.	8. The system shall allow users to customize their dashboard layout with widgets.
9. The software should be easy to maintain.	9. The software shall include detailed inline documentation for developers.
10. The website should load quickly.	10. The website shall load within 3 seconds on a 5Mbps internet connection.

Requirement Traceability

Stakeholder Requirements to System Requirements

User stories are mapped to system requirements, ensuring traceability from stakeholder needs to system implementation

1. User Story Format:

- "As a [user], I want [feature], so that [benefit]."
- **Captures user needs:** Focuses on the user's perspective and goals.
- **Easy to understand:** Written in plain language for clarity and simplicity.
- **Focus on benefits:** Helps identify the value provided to the user.

2. System Requirement Format:

- Defines how the system shall implement the user need in a technical manner.
- **Technical language:** Specifies what the system must technically do to fulfill the user story.
- **Measurable outcomes:** Ensures the system's behavior is testable and verifiable.
- **System constraints:** Captures any conditions or performance limitations

3. Ensuring Traceability:

- **Traceability Matrix:** Link user stories to system requirements for traceability.
- **Requirement tracking tools:** Use tools like Jira or Confluence to track requirements in real-time.
- **End-to-end visibility:** Ensures stakeholder needs are mapped and implemented in system design.

User Story (Stakeholder Requirement)	System Requirement (Technical Language)
User Story 1: "As a customer, I want to filter products by price so that I can find the best deals."	**System Requirement 1**: "The system shall provide a filter function that allows users to sort products by price range (min to max) in real-time."
User Story 2: "As an admin, I want to create and manage user accounts so that I can control access to the system."	**System Requirement 2**: "The system shall provide an interface for admin users to create, edit, and delete user accounts with role-based access control."
User Story 3: "As a user, I want to receive a confirmation email after placing an order so that I know it was successful."	**System Requirement 3**: "The system shall automatically generate and send an order confirmation email to the user's registered email address within 30 seconds of order completion."
User Story 4: "As a shopper, I want to save items to a wishlist so that I can review them later."	**System Requirement 4**: "The system shall allow users to add products to a wishlist, stored in the database, accessible from their account on future logins."
User Story 5: "As a user, I want to reset my password if I forget it so that I can regain access to my account."	**System Requirement 5**: "The system shall provide a password recovery option, sending a one-time link to reset the user's password, valid for 30 minutes."

Functional and Non-Functional System Requirements

Functional requirements focus on the system's behavior, while non-functional requirements define its properties

1. Functional System Requirements:

- **Reflect system behavior:** Specify what the system should do in response to inputs or events.
- **User-focused:** Relate to specific functionalities required by stakeholders or users.
- **Testable and measurable:** Can be verified through actions, tests, or simulations.
- **Example:** "The system shall allow users to log in using their email and password."

2. Non-Functional System Requirements:

- **Describe system properties:** Specify how the system is in terms of performance, security, and usability.
- **System quality attributes:** Focus on aspects like reliability, performance, and security that impact how the system operates.
- **Typically not user-facing:** Concerned with system attributes that affect its background operations or performance.
- **Example:** "The system shall store passwords securely using 256-bit encryption."

3. Ensuring Completeness:

- **Functional:** Captures required system actions and interactions based on user needs.
- **Non-functional:** Defines the system's operational properties
- **Complement each other:** Functional and non-functional requirements work together to ensure the system behaves as expected and operates efficiently.

Functional Requirement	Non-Functional Requirement
FR1: "The system shall allow users to upload images to their profile."	**NFR1**: "The system shall store images with a maximum file size of 5MB."
FR2: "The system shall generate a report summarizing user activity."	**NFR2**: "The system shall have an availability of 99.9% uptime."
FR3: "The system shall allow users to reset their passwords."	**NFR3**: "All passwords shall be stored securely using 256-bit encryption."
FR4: "The system shall enable users to search for products by category."	**NFR4**: "The system shall support up to 10,000 concurrent users without degradation in performance."
FR5: "The system shall provide users with the ability to download invoices in PDF format."	**NFR5**: "The system shall ensure that all downloadable invoices are generated within 5 seconds."
FR6: "The system shall allow administrators to assign roles to users."	**NFR6**: "The system shall ensure role assignments are fully logged and auditable for 5 years."

Mustergültige Anforderungen - die SOPHIST Templates für Requirements

MASTER Requirements Template Overview

Exemplary requirements - the SOPHIST templates for requirements

MASTER templates provide a structured approach to writing system requirements, ensuring consistency and traceability

1. Purpose of MASTER Templates:

- **Standardize writing:** Ensures uniformity in structuring requirements, making them easy to understand and implement.
- **Improve traceability:** Facilitates the tracing of requirements from high-level stakeholder needs down to specific system actions or properties.
- **Enhance clarity and testability:** Each template ensures that requirements are written in a clear, testable format.

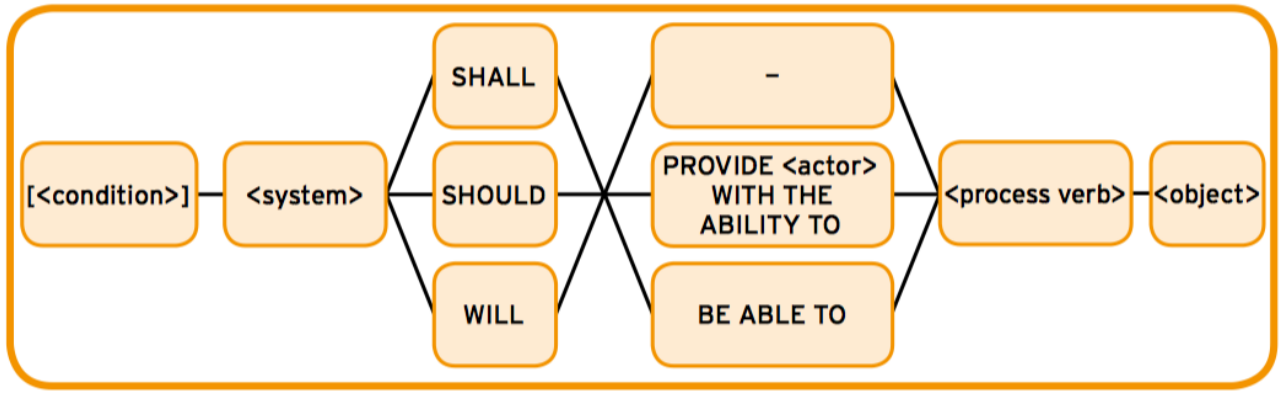
2. Types of MASTER Templates:

- **Functional MASTER:** Defines what the system must do under specific conditions.
- **Property MASTER:** Specifies the properties or attributes the system must have.
- **Process MASTER:** Describes how certain processes or activities must be carried out within the system.

3. Usage of MASTER Templates:

- **Consistency across teams:** Ensures that different teams working on the same system maintain a consistent structure in how requirements are written.
- **Scalability:** MASTER templates can be applied to both small-scale and large-scale systems, ensuring scalability.
- **Testability and traceability:** By using these templates, every requirement can be tested and traced back to its source, improving system validation.

Functional Requirements MASTER Template

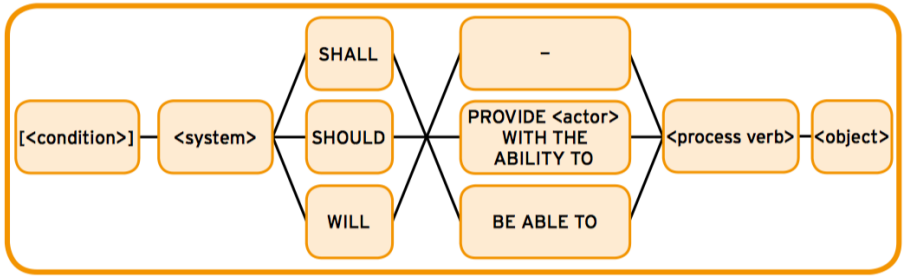


For example: *The document editor shall provide the user with the ability to create new documents.*

- **Condition:** Shall be specified with the ConditionMASTER, LogicMASTER, EventMASTER or TimeMASTER templates.
- **System:** Represents the system or component that should provide the functionality.
- **Liability:** Modal verbs are used to express liability:
  - Shall: Statement is legal binding and mandatory.
  - Should: Statement is desired, but not mandatory.
  - Will: Statement is recommended, but not mandatory.
- **Activity type:** The different types of system activities are the following [RdS14a]:
  - Independent system action: (-) The system performs the process by itself.
  - User interaction: (PROVIDE <actor> WITH THE ABILITY TO) The system provides the user with some process functionality.
  - Interface requirement: (BE ABLE TO) The system performs the process dependent on a third factor.

Process verb: Represents the process: procedures (functionality) and actions to be provided by the system

MASTER Template - Functional Requirements Example



<condition>, <system>, SHALL/SHOULD/WILL, <actor>, <process verb>, and **<object>**

<When a user enters valid login credentials>, the **<authentication system> SHALL <authenticate>** the **<user>** and grant access to their profile.

<When a user clicks "Add to Cart">, the **<e-commerce platform> SHALL <add>** the **<selected product>** to the **<shopping cart>**.

<If the temperature exceeds 75°C>, the **<server cooling system> SHALL <activate>** the **<cooling mechanism>** to reduce the temperatur

<When a user submits the contact form>, the **<website backend> SHALL <send>** the **<form data>** to the **<support team>** for follow-up.

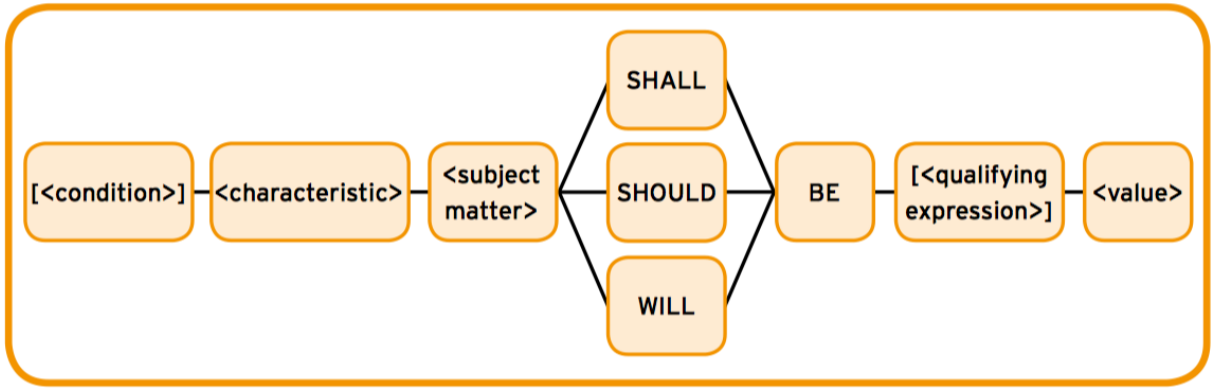
<If an invalid credit card is provided>, the **<payment gateway> SHALL <display>** an **<error message>** to the **<user>**.

<Upon exceeding the cloud storage limit>, the **<cloud storage system> SHALL <notify>** the **<user>** and **<restrict>** further **<uploads>**.

<When the system detects 15 minutes of user inactivity>, the **<banking application> SHALL <log out>** the **<user>** for security purposes.

<Upon receiving a maintenance request>, the **<facility management system> SHALL <assign>** the **<request>** to the **<appropriate technician>**.

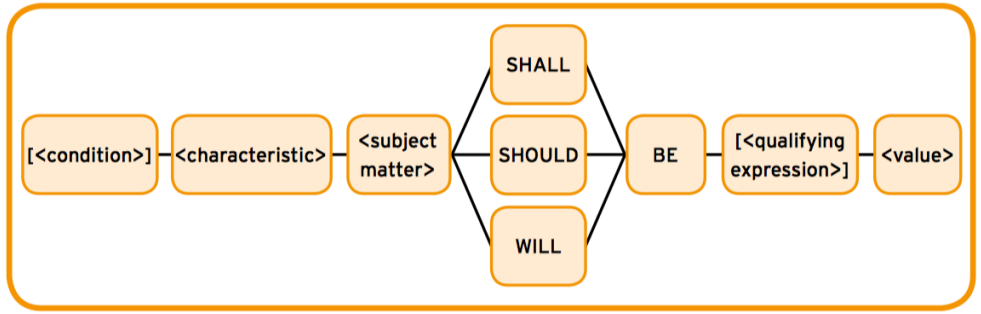
Property Requirements MASTER Template



For example: *The design of the website should be responsive.*

- **Characteristic**: Defines the property of the subject matter (e.g., speed, size).
  - This is the key attribute of the system that needs to be defined.
- **Subject Matter**: Represents the system, sub-system, or component that has the property.
  - Identifies the part of the system to which the characteristic applies.
- **Liability**: Uses modal verbs (SHALL, SHOULD, WILL) to express the level of obligation.
  - Shall: Legal binding and mandatory.
  - Should: Desired but not mandatory.
  - Will: Recommended but not binding.
- **Qualifying Expression:** Specifies the range or limits of the property.
  - Describes constraints like "greater than," "less than," etc.
- **Value**: The specific value associated with the characteristic.
  - This is connected through the verb "to BE.

MASTER Template - Property Requirements Example



<condition>, <characteristic>, <subject matter>, SHALL/SHOULD/WILL, <qualifying expression>, and <value>

<When operating at full load>, the **<processor> SHALL BE <clock speed> <at least 3.2 GHz>**.

<In temperatures above 35°C>, the **<battery> SHALL BE <operating capacity> <at least 80%>** of its normal capacity.

<During regular operation>, the **<display> SHALL BE <brightness level> <adjustable from 100 nits to 500 nits>**.

<Under heavy usage>, the **<system memory> SHOULD BE <response time> <less than 10 milliseconds>**.

<In power-saving mode>, the **<screen> WILL BE <refresh rate> <limited to 30Hz>** to conserve energy.

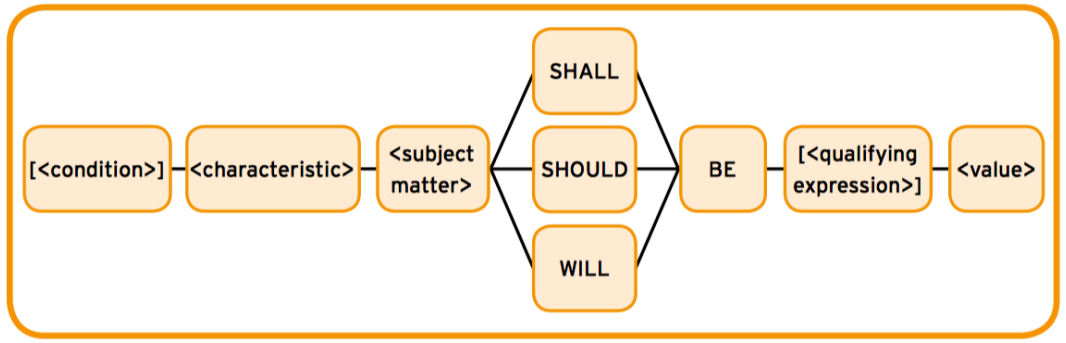
<When connected to a high-speed network>, the **<network adapter> SHALL BE <data transfer rate> <at least 1 Gbps>**.

<At startup>, the **<operating system> SHALL BE <boot time> <no more than 20 seconds>**.

<Under normal usage conditions>, the **<hard drive> SHALL BE <noise level> <less than 30 dB>**.

<During playback of 4K video content>, the **<graphics card> SHALL BE <frame rate> <at least 60 frames per second>**.

Environmental Requirements MASTER Template

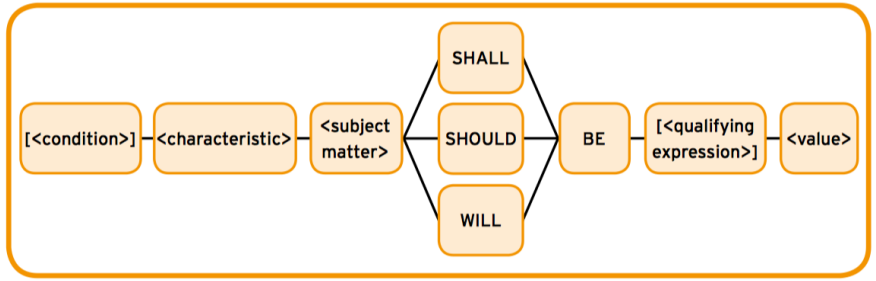


The fixed values are designed in a way that make the requirement belongs to the system and not the environment

For example: *The charger of the device shall be designed in a way the system can be operated in a range 100-240V/50-60Hz.*

- **Component of Subject Matter:** Specifies which component of the system the environment requirement applies to.
  - Identifies which part of the system needs to operate in a specific environment.
- **Liability:** Expressed through modal verbs (SHALL, SHOULD, WILL) to define the level of obligation.
  - **Shall**: Binding and mandatory.
  - **Should**: Desired but not mandatory.
  - **Will**: Recommended but not binding.
- **Designed in a Way**: Indicates how the component should be designed to meet environmental conditions.
  - Ensures that the design accounts for the environment in which the system operates.
- **Condition**: Describes the environmental condition under which the component operates.
  - E.g., temperature, humidity, or power range.
- **Characteristic**: Defines the characteristic the system must meet under these conditions.
  - Describes system properties that must be maintained in the environment.
- **Qualifying Expression and Value**: Specifies measurable expressions (e.g., voltage range, operating temperature).
  - Quantifies the environmental requirement.

MASTER Template - Property Requirements Example



<condition>, <characteristic>, <subject matter>, SHALL/SHOULD/WILL, BE <qualifying expression>, <value>

<When the system is exposed to temperatures below 0°C>, the **<heating unit> SHALL <be designed in a way>** that the system **<can be operated>** at **<optimal efficiency>**.

<In the event of power failure>, the **<backup generator> SHALL <be designed in a way>** to **<operate>** the system **<for at least 4 hours>**.

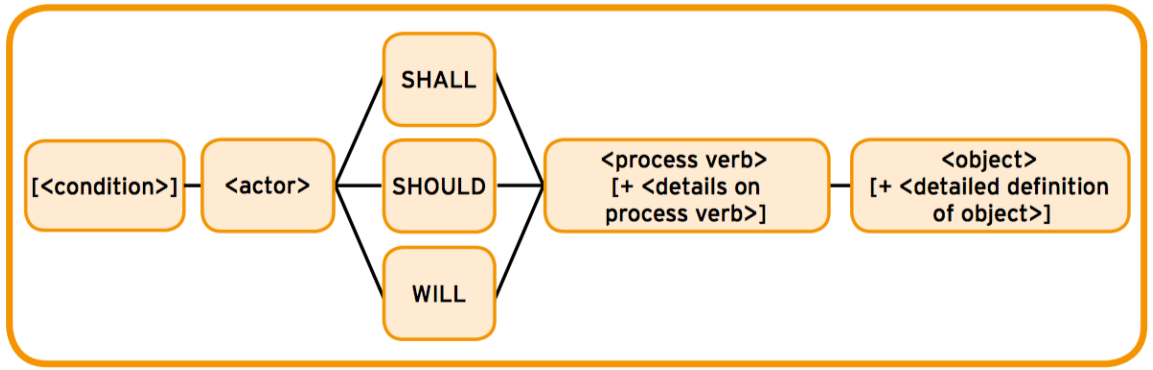
<When exposed to direct sunlight>, the **<solar panel system> SHALL <be designed in a way>** that it **<can be operated>** in temperatures of **<up to 50°C>**.

<If the system is exposed to dust and particles>, the **<air filtration system> SHALL <be designed in a way>** that it **<can be operated>** without performance degradation.

<When operating at high altitudes (above 3000 meters)>, the **<pressure-sensitive components> SHALL <be designed in a way>** that they **<can be operated>** without pressure-related failures.

<In environments with high electromagnetic interference (EMI)>, the **<communication system> SHALL <be designed in a way>** that it **<can be operated>** with minimal signal disruption.

Process Requirements MASTER Template

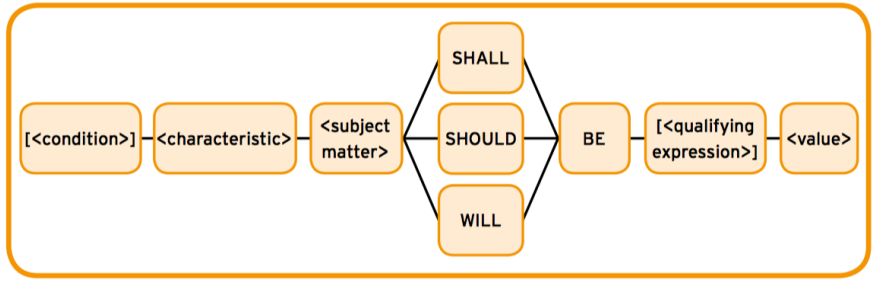


Process requirements are related to activities or legal-contractual requirements, as well as non-functional requirements.

For example: *The software developers should work according to the Personal Software Process (PSP).*

- **Condition**: Describes when the process requirement is relevant.
  - Specifies the trigger for the process (e.g., during development).
- **Actor**: Represents the actor responsible for performing the process.
  - Unlike other templates, the focus is on who performs the process, not the system.
- **Liability**: Uses modal verbs (SHALL, SHOULD, WILL) to indicate the level of obligation.
  - **Shall**: Binding and mandatory.
  - **Should**: Desired but not mandatory.
  - **Will**: Recommended but not binding.
- **Process Verb**: Describes the action or process the actor is responsible for performing.
  - This may include additional details about the process.
- **Object**: Specifies the target or result of the process.
  - Provides a detailed definition of the object or result expected from the process.

MASTER Template - Process Requirements Example



<condition>, <characteristic>, <subject matter>, SHALL/SHOULD/WILL, BE <qualifying expression>, <value>

<When a customer places an order>, the **<warehouse operator> SHALL <pick>** the **<items from stock>** and **<prepare them for shipping>**.

<If a system error is detected>, the **<monitoring service> SHALL <log>** the **<error>** and **<notify the system administrator>**.

<If the software update is approved>, the **<IT administrator> WILL <deploy>** the **<update>** to all **<company devices>**.

<When a customer calls support>, the **<call agent> SHALL <log>** the **<call details>** and **<create a support ticket>**.

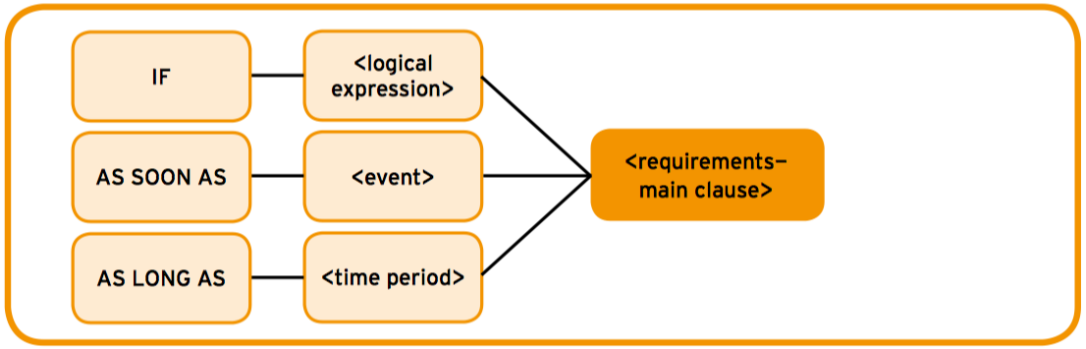
<When a purchase order is created>, the **<purchasing manager> SHALL <review>** the **<order>** and **<approve or request changes>**.

<If the contract is signed>, the **<legal department> WILL <file>** the **<contract>** and **<update the company's legal records>**.

<When a customer request is received>, the **<customer service representative> SHOULD <respond>** to the **<request>** within **<24 hours>**.

<When the monthly payroll period ends>, the **<payroll specialist> SHALL <calculate>** the **<salaries>** and **<process employee payments>**.

Conditions MASTER Template

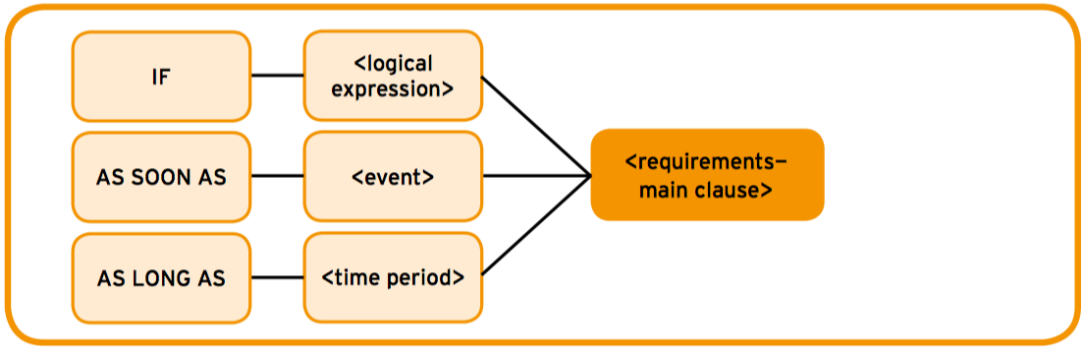


Process requirements are related to activities or legal-contractual requirements, as well as non-functional requirements.

For example: *If the server could not found what was requested.*

- Condition Types:
  - **IF**: Represents logical expressions that trigger a requirement based on a logical condition.
  - **AS SOON AS**: Triggers the requirement immediately when a specific event occurs.
  - **AS LONG AS**: Specifies that the requirement applies for the duration of a given time period or event.
- **Logical Expressions:** Defines a condition based on a logical comparison or evaluation.
  - Example: If the user provides invalid input, the system must display an error message.
- **Event-based Conditions:** Triggered by an event such as an action by the user or an external system event.
  - Example: As soon as the user clicks “Submit”, the system shall process the form.
- **Time-based Conditions:** Apply while the system is in a specific state or during a defined time period.
  - Example: As long as the user is logged in, the system must display real-time notifications.

MASTER Template – Condition Examples



<condition>, <logical expression/event/time period>, <system>, SHALL/SHOULD/WILL, <main clause>

<IF the user enters an invalid password>, the **<system> SHALL <display an error message>**.

<AS SOON AS the temperature sensor detects a rise above 30°C>, the **<cooling system> SHALL <activate to maintain safe operating conditions>**.

<AS LONG AS the user remains inactive for 15 minutes>, the **<system> SHALL <prompt the user to resume or log out>**.

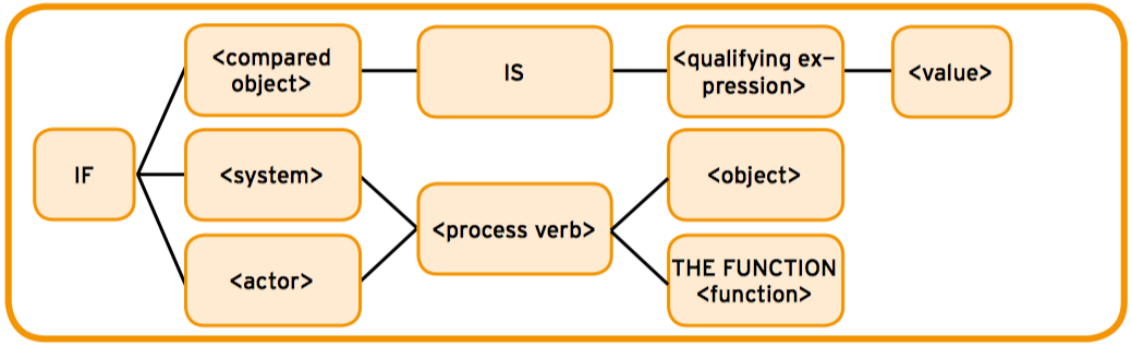
<IF the system detects a network connection failure>, the **<system> SHALL <switch to offline mode>**.

<AS SOON AS the battery charge falls below 20%>, the **<device> SHALL <reduce performance to conserve energy>**.

<IF the user cancels the operation>, the **<system> SHALL <revert all changes made during the session>**.

<AS SOON AS the motion sensor detects movement>, the **<security system> SHALL <trigger the alarm and notify the security team>**.

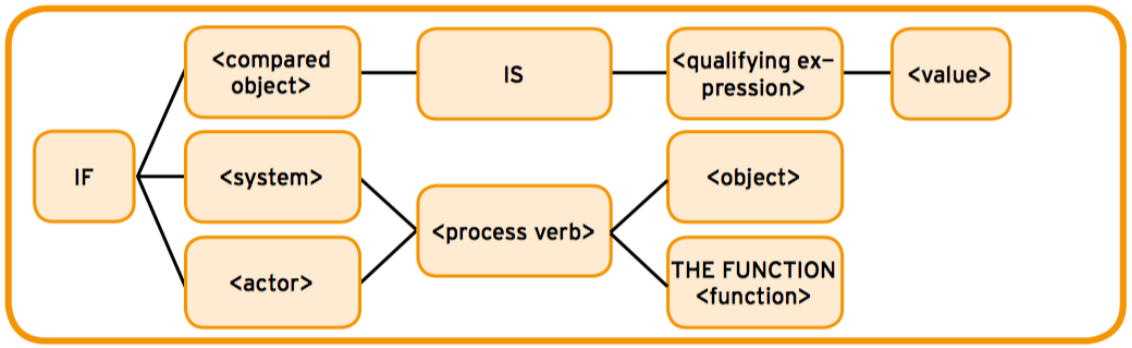
Logical Conditions MASTER Template



.The logical statement is made through a compared object, an actor or a system

- **If Condition**: Specifies the condition or trigger for the logical statement.
  - E.g., "If the system reaches a certain temperature."
- **Compared Object/System/Actor**: The subject of the logical comparison.
  - Specifies whether the object, system, or actor is involved in the logical condition.
- **Process Verb:** Describes the system or actor's action or state.
  - Defines the functionality or state that the logical condition affects.
- **Qualifying Expression**: Defines the range or expression for comparison (e.g., equal to, greater than).
  - Specifies how the logical condition is evaluated.
- **Value**: The measurable value for the comparison.
  - E.g., "The temperature must be greater than 75°C."

MASTER Template – Logical Conditions Examples



<compared object>, <system>, IS, <qualifying expression>, <value>, <process verb>, <object>, and **<function>**

<IF the temperature sensor> IS <greater than 50°C>, the **<cooling system> SHALL <activate>** and **<reduce the temperature>**.

<IF the battery> IS <less than 10%>, the **<system> SHALL <dim the display>** to **<preserve battery life>**.

<IF the internet connection> IS <unavailable>, the **<system> SHALL <switch to offline mode>** to **<allow limited functionality>**.

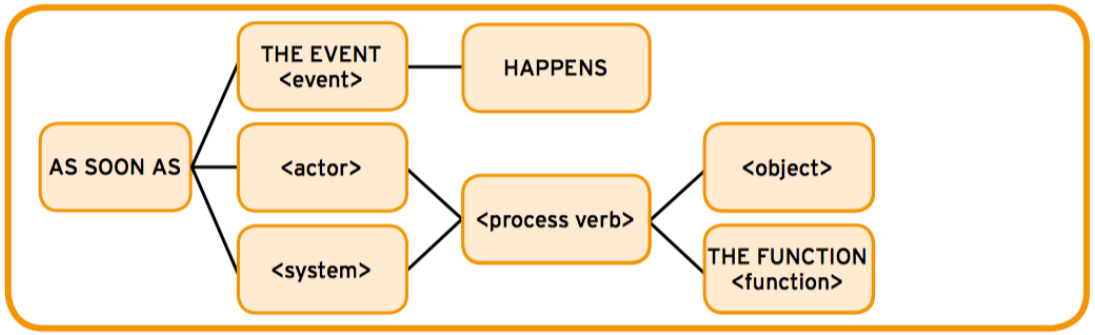
<IF the user> IS <an administrator>, the **<system> SHALL <grant access>** to **<configuration settings>**.

<IF the file size> IS <greater than 100 MB>, the **<system> SHALL <compress the file>** before **<uploading>**.

<IF the network> IS <experiencing latency>, the **<system> SHALL <reduce video quality>** to **<maintain a stable connection>**.

<IF the storage capacity> IS <below 5%>, the **<system> SHALL <notify the user>** to **<free up space>**

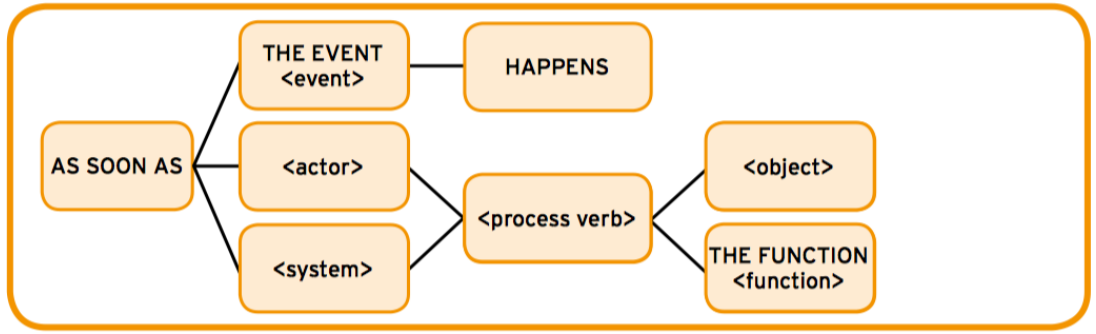
Event-Driven Requirements MASTER Template



The term event summarizes the possible events that may affect the system

- **Event**: Describes the triggering event that initiates the requirement.
  - E.g., "As soon as a user logs in."
- **Actor/System**: Defines who or what is involved when the event happens.
  - Specifies the actor or system component that responds to the event.
- **Happens**: Marks the point at which the event occurs.
  - Indicates the moment when the event is recognized by the system.
- **Process Verb**: Describes the action or function that takes place once the event occurs.
  - E.g., "Log the user activity."
- Object: Specifies the target or result of the process.
  - Describes the object or function affected by the event.

MASTER Template – Event-Driven Requirements Examples



<AS SOON AS the event happens>, <actor/system>, <process verb>, <object>, and **<function>**:

<AS SOON AS the user presses the power button>, the **<system> SHALL <start>** the **<boot process>** and **<load the operating system>**.

<AS SOON AS the temperature exceeds 75°C>, the **<cooling system> SHALL <activate>** and **<reduce the temperature>** to prevent overheating.

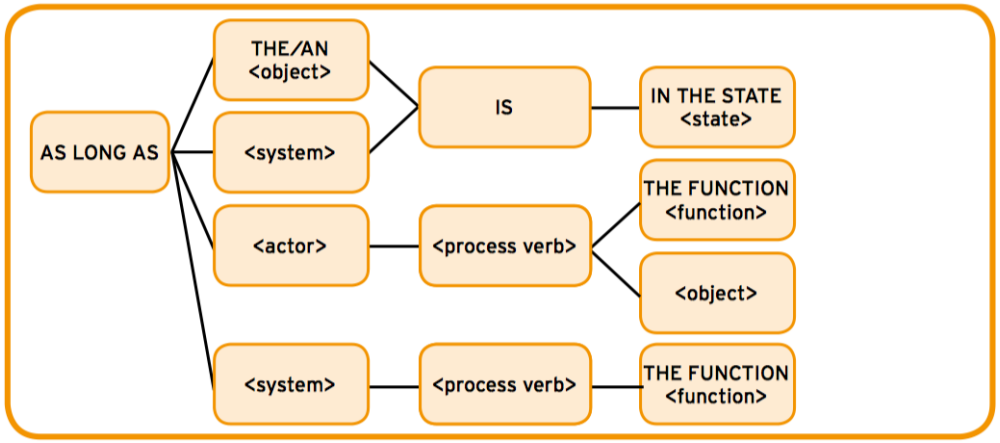
<AS SOON AS the user submits the form>, the **<backend system> SHALL <validate>** the **<input data>** and **<store it in the database>**.

<AS SOON AS the motion sensor detects movement>, the **<security system> SHALL <trigger>** the **<alarm>** and **<notify the security team>**.

<AS SOON AS the door is unlocked>, the **<access control system> SHALL <log>** the **<entry event>** and **<activate the door mechanism>**.

<AS SOON AS the battery level drops below 10%>, the **<system> SHALL <dim the display>** and **<limit background processes>** to conserve battery.

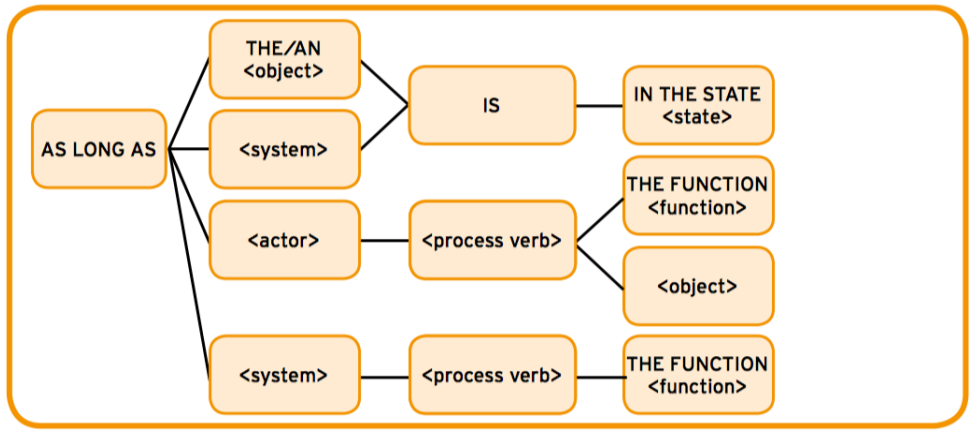
Time-Based Requirements MASTER Template



Is used to specify a certain period of time when a system or object may have temporary behaviors. Both, conditions and requirements, end at the same time

- **As Long As:** Specifies the duration or condition under which the requirement applies.
  - E.g., "As long as the system is in standby mode."
- **Actor/System:** Defines who or what is involved during the specified time period.
  - Identifies the actor or system affected by the time constraint.
- **Is in State:** Describes the system or actor's state during the time period.
  - Specifies the condition that must be met (e.g., in a specific mode or state).
- **Process Verb:** Describes the action that occurs during the specified time.
  - E.g., "Monitor system performance."
- **Object/Function:** Specifies the object or function affected by the process during the specified time period.
  - Describes the target or result of the time-bound action.

MASTER Template – Time-Based Requirements Examples



<AS LONG AS the object/system/actor is in a specific state>, <system>, <process verb>, <function>

<AS LONG AS the user is logged in>, the **<system> SHALL <display>** real-time **<notifications>**.

<AS LONG AS the temperature sensor is in the "overheat" state>, the **<cooling system> SHALL <operate>** to **<reduce the temperature>**.

<AS LONG AS the battery level is below 20%>, the **<system> SHALL <limit background processes>** to **<conserve battery life>**.

<AS LONG AS the device is connected to Wi-Fi>, the **<system> SHALL <sync>** all **<data>** to the cloud.

<AS LONG AS the vehicle is moving>, the **<infotainment system> SHALL <restrict access>** to certain **<features>** for safety reasons.

<AS LONG AS the file transfer is in progress>, the **<system> SHALL <display>** the **<transfer status>** to the user.

<AS LONG AS the door lock is in the "unlocked" state>, the **<security system> SHALL <track>** all **<entry events>**.

Easy Approach to Requirements Syntax

EARS Requirements Template Overview

EARS methodology, helping writers select the appropriate template for each situation

1. Why Use EARS?: ✍️

- Simplifies the process of writing requirements by providing predefined templates for different types of system behaviors.
- Reduces ambiguity and ensures that requirements are easily understandable by all stakeholders.
- Writing requirements using structured syntax ensures consistency across the project.

2. Types of EARS Requirements: 📝

- **Generic Requirements:** Cover basic, unconditional system behaviors.
- **Ubiquitous Requirements:** Apply under all conditions, without the need for triggers.
- **Event-Driven Requirements:** Triggered when a specific event occurs.
    - Example: "WHEN the door is opened, the system shall activate the alarm."

3. Benefits of EARS: 🎯

- Ensures that all requirements are written in a clear, testable, and verifiable format.
- Helps avoid common issues such as vague or incomplete requirements.
- A well-structured requirement is easier to validate through testing.

Easy Approach to Requirements Syntax

EARS Generic Requirement

Generic requirement syntax in the EARS system, with preconditions and triggers that lead to a system response

- **Precondition**: Defines necessary conditions to invoke the requirement.
  - Specifies what must be true before the system response is activated

  .
- **Trigger**: Describes the event that initiates the requirement.
  - Specifies what causes the system to act.

- **System Response**: Represents the system behavior that occurs once the precondition and trigger are met.
  - Defines how the system reacts to the trigger event.

Generic Requirement Syntax:

<optional preconditions> <optional trigger> the <system name> shall <system response>

EARS - Generic Requirements Examples

<optional preconditions> <optional trigger> the <system name> shall <system response>:

<If the user is logged in>, the **<system> shall <display personalized content>**.

<If the temperature exceeds 40°C>, the **<cooling system> shall <activate>**.

<If the power button is pressed>, the **<system> shall <shut down>** safely.

<If the file is not found>, the **<system> shall <return an error message>**.

<If the user requests a password reset>, the **<system> shall <send an email with reset instructions>**.

<When the user clicks "Submit">, the **<system> shall <validate the form data>**.

<If no network connection is available>, the **<system> shall <switch to offline mode>**.

<If the battery level drops below 20%>, the **<device> shall <enter power-saving mode>**.

<If the sensor detects motion>, the **<alarm system> shall <trigger the security alarm>**.

<When the vehicle is in reverse>, the **<backup camera> shall <activate>**.

Easy Approach to Requirements Syntax

EARS Ubiquitous Requirement

Ubiquitous requirement in the EARS system, defining a fundamental property of the system without conditions or trigger

- **No Preconditions or Triggers:** Ubiquitous requirements define fundamental properties of the system.
  These requirements always apply and are not dependent on external events or conditions.
- **Format**: Simple structure <system name> shall <system response>.
  The requirement is always true for the system, without any specific conditions.
- **Example**:
  "The software shall be written in Java."

Ubiquitous Requirement Syntax:

<system name> shall <system response>

EARS - Ubiquitous Requirements Examples

<system name> shall <system response>

The **<software application> shall <be written in Java>**.

The **<security system> shall <encrypt all data>** stored on the server.

The **<user interface> shall <provide a login screen>** when the application starts.

The **<website> shall <comply with accessibility standards>**.

The **<operating system> shall <allow users to manage files>** using a file explorer.

The **<device> shall <connect to Wi-Fi>** if available.

The **<software> shall <log all user activities>** for auditing purposes.

The **<server> shall <perform daily backups>** of all critical data.

The **<system> shall <provide multi-language support>** for users.

The **<application> shall <store user preferences>** locally on the device.

Easy Approach to Requirements Syntax

EARS Event-Driven Requirement

For Event-driven requirements in the EARS system responds to specific events

- **Triggered by Events:** Event-driven requirements are activated only when a specific event or trigger occurs.
  This type of requirement relies on the occurrence of an event to initiate a system response.
- **Format**: Uses the WHEN keyword to define the trigger.
  Structure: WHEN <optional preconditions> <optional trigger> the <system name> shall <system response>.
- **Example**:
  "When a DVD is inserted into the DVD player, the OS shall spin up the optical drive."

Event-Driven Requirement Syntax:

WHEN <optional preconditions> <optional trigger> the <system name> shall <system response>

EARS - Event-driven Requirements Examples

WHEN <trigger> the <system name> shall <system response>

WHEN a **user presses the power button**, the **device shall shut down** safely.

WHEN the **door sensor** detects that the door is opened, the **security system shall trigger the alarm**.

WHEN the **vehicle** is placed in reverse gear, the **backup camera shall activate**.

WHEN the **battery level** falls below 20%, the **device shall enter power-saving mode**.

WHEN the **user clicks the "Submit" button**, the **web application shall validate the form data**.

WHEN the **network connection** is lost, the **system shall switch to offline mode**.

WHEN the **user logs out**, the **system shall clear all session data**.

WHEN a **motion sensor** detects movement, the **security camera shall start recording**.

WHEN a **USB device** is connected, the **operating system shall mount the device** and allow file access.

Easy Approach to Requirements Syntax

EARS Unwanted Behaviors

Unwanted behavior requirement in the EARS system, capturing how the system responds to failures or disturbances

- **Capturing Failures**: Unwanted behaviours, such as failures, disturbances, or deviations, need specific responses.
  - These requirements focus on system reactions to abnormal conditions.
- **Format**: Uses the IF/THEN keywords to define conditions and the system's response.
  - **Structure**: IF <optional preconditions> <optional trigger>, THEN the <system name> shall <system response>.
- **Example**:
  - "IF a system error occurs, THEN the system shall log the error and notify the user.

Unwanted Behaviors Requirement Syntax:

IF <optional preconditions> <optional trigger>, THEN the <system name> shall <system response>.

EARS - Unwanted Behaviors Requirements Examples

IF <optional preconditions> <optional trigger>, THEN
the <system name> shall <system response>.

IF the **user has attempted to log in three times with incorrect credentials**, **THEN** the **system shall lock the account** and **notify the security team**.

IF the **server temperature exceeds 85°C**, **THEN** the **system shall shut down** to prevent damage.

IF the **user's session times out**, **THEN** the **system shall log the user out** and **redirect them to the login page**.

IF the **network connection is lost during data transmission**, **THEN** the **system shall pause the transmission** and **attempt to reconnect**.

IF the **disk space falls below 10%**, **THEN** the **system shall notify the administrator** and **limit further data storage**.

IF the **vehicle detects a loss of traction**, **THEN** the **system shall activate the anti-lock braking system (ABS)**.

IF the **user cancels a transaction before completion**, **THEN** the **system shall revert all changes** and **display a cancellation confirmation**.

IF the **power supply drops below 90%**, **THEN** the **system shall switch to backup power** and **log the event**.

IF the **door is left open for more than 5 minutes**, **THEN** the **security system shall send a notification** and **activate a warning light**.

Easy Approach to Requirements Syntax

EARS Optional Requirements

Optional requirement in the EARS system, defining how the system responds when a certain feature is included

- **Triggered by Features**: Optional requirements are only valid if a certain feature is present.
  - These requirements depend on specific system configurations or capabilities.
- **Format**: Uses the WHERE keyword to define the condition based on a feature.
  - **Structure**: WHERE <feature is included> the <system name> shall <system response>.
- **Example**:
  - "Where an HDMI port is present, the software shall allow the user to select HD content for viewing."

Optional Requirement Syntax:

WHERE <feature is included> the <system name> shall <system response>

EARS – Optional Requirements Examples

WHERE <feature is included> the <system name> shall <system response>

WHERE a **GPS module is included**, the **system shall provide real-time location tracking**.

WHERE a **backup battery is present**, the **system shall switch to battery power** during a power outage.

WHERE a **high-resolution display is available**, the **system shall allow 4K video playback**.

WHERE a **fingerprint scanner is integrated**, the **system shall use biometric authentication** for login.

WHERE a **Wi-Fi connection is detected**, the **system shall automatically sync data to the cloud**.

WHERE an **HDMI port is available**, the **system shall support external monitor connections**.

WHERE the **vehicle is equipped with sensors**, the **system shall provide lane departure warnings**.

WHERE a **dedicated graphics card is installed**, the **system shall enable advanced rendering features**.

WHERE the **device supports Bluetooth**, the **system shall allow wireless connection to peripherals**.

WHERE a **security camera is available**, the **system shall offer real-time video streaming**.

Easy Approach to Requirements Syntax

EARS Complex Requirements

Optional requirement in the EARS system, defining how the system responds when a certain feature is included

- **Combining Conditions**: Complex requirements are created by combining multiple conditions.
  - These requirements are more intricate and may involve multiple events, states, or actions.
- **Keywords**: The combination of WHEN, IF, THEN, WHILE, and WHERE allows for detailed conditional requirements.
  - Use these keywords to define the sequence and interaction of multiple conditions.
- **Example**:
  - "When the landing gear button is depressed once, if the software detects that the landing gear does not lock into position, then the software shall sound an alarm."

EARS – Optional Requirements Examples

multiple conditions using keywords like **IF**, **WHEN**, **WHILE**, **THEN**, and **WHERE**:

WHEN the **user presses the start button**, **IF** the **battery level is above 20%**, **THEN** the **system shall initiate boot-up**.

IF the **temperature exceeds 50°C**, **AND** the **fan is not running**, **THEN** the **cooling system shall activate**.

WHEN the **sensor detects motion**, **AND** the **light level is below 30%**, **THEN** the **system shall turn on the lights**.

IF the **user logs in with admin credentials**, **AND** the **two-factor authentication is enabled**, **THEN** the **system shall send a verification code**.

WHILE the **user is uploading files**, **IF** the **network connection is lost**, **THEN** the **system shall pause the upload** and **retry** when the connection is restored.

WHEN the **vehicle is moving**, **IF** the **driver opens a door**, **THEN** the **system shall trigger an alert** and **lock the door**.

IF the **user is inactive for 10 minutes**, **AND** the **session is not in a critical operation**, **THEN** the **system shall log the user out automatically**.

WHEN the **temperature is below freezing**, **AND** the **engine is running**, **THEN** the **system shall activate the engine heater**.

IF the **fire alarm is triggered**, **AND** the **emergency exits are blocked**, **THEN** the **system shall alert emergency responders** and **unlock alternative exits**.

Summary and Q&A

Writing requirements with templates contribute to project success by providing clear, testable, and traceable requirements

1. MASTER Templates

- Structured for system-level requirements.
- Ensures traceability and consistency.
- Testable and measurable for validation.

2. EARS Templates

- Simplifies writing for different system behaviors.
- Reduces ambiguity with predefined structures.
- Focuses on clear and concise requirements.

3. Writing Good Requirements

- Use clear and unambiguous language.
- Ensure every requirement is testable.
- Maintain traceability to goals and objectives.