An Introduction to EARS
(the Easy Approach to Requirements Syntax)

The Easy Approach to Requirements Syntax (EARS) is an approach that provides structure to requirements written in text. It makes use of a small number of keywords and a syntax (structure). Requirements are expressed in a consistent format that is easy for audiences to understand. Unlike more extensive standards, EARS requires very little training and no specialized tools.

EARS Structure

The generic structure of an EARS requirement is:

While <optional precondition(s)>, when <optional trigger>, the <system name> shall <system response(s)>

Each requirement may have:

- Zero to many preconditions
- Zero or one trigger
- One system name
- Zero to many system responses

Summary of EARS Patterns

EARS makes use of a small number of patterns for constructing requirements. The clauses of a requirement written in EARS always appear in the same order.

Pattern Name	Pattern
Ubiquitous	The <system name> shall <system response>
Event-driven	When <trigger>, the <system name> shall <system response>
Unwanted behavior	If <trigger>, then the <system name> shall <system response>
State-driven	While <precondition(s)>, the system name> shall <system response>
Optional feature	Where <feature is included> the <system name> shall <system response>
Complex	Combine the above patterns to specify more complex requirements

EARS Pattern Definitions and Examples

Ubiquitous requirements

Ubiquitous requirements are requirements that are always active. They do not have a precondition or trigger. They are not performed in response to an event nor are they performed in response to a defined system state.

Example: The Farm Fresh Mobile App shall present all information in English

Event-driven requirements

Event-driven requirements are initiated when and only when a triggering condition occurs or is detected.

Example: When an item is added to the shopping cart, the Farm Fresh Mobile App shall decrement the quantity available for that item

Unwanted behavior

These requirements handle unwanted behavior including error conditions, failures, faults, disturbances, and other undesired events.

Example: If a customer under the age of 18 attempts to purchase an age-restricted product, then the Farm Fresh Mobile App shall not place that item in the shopping cart.

State-driven requirements

State-driven requirements are triggered while the system is in a specific state.

Example: While awaiting a response from the credit card processing vendor, the Farm Fresh Mobile App shall prevent the user from performing any other functions.

Optional features

Optional features are only invoked in systems that include a particular optional feature.

Example: Where location services is available and enabled, the Farm Fresh Mobile App shall use location services to establish the current location of the customer.

Complex requirements

Some requirements involve multiple triggers, states, and/or optional features. A combination of the keywords from the EARS rule patterns can be used to express these requirements.

Example: When the customer provides a credit card to pay for an order, if the credit card payment vendor rejects the card, then the Farm Fresh Mobile App shall request an alternate form of payment.