Example-Based Specification Tests—Guidelines

Consider your tests from the perspective of these guidelines.

√	**Guide the Reader**
	Make sense to four audiences: product owner, tester, programmer, and end-user. ("Would this example make sense in a user manual?")
	Arrange tests to "tell the story" of the system.
	Favor tests that are concise and clear. (Readers tend to gloss over long tests.)
	Put simple cases before complicated ones.
	Make core rules precede user interface rules (or segregate them).
	Include both positive ("happy path") and negative ("sad path") tests.

√	**Tables**
	Use explanatory text between tables.
	Prefer tests in self-contained tables over tests that require multiple tables.
	Make each test *declarative* ("these inputs cause these outputs") rather than *procedural* ("this sequence of steps causes this result"), if possible. (Prefer declarative tests, but you need both.)
	Ensure that no tables are duplicated.
	Describe independent rules in separate tables.
	Keep tables to a manageable size.

√	**Fixtures**
	Make the fixture name meaningful / self-explanatory.
	Make column names meaningful / self-explanatory.
	Use similar column names between different types of fixtures. (E.g., don't use "valid()" in one fixture and "ok()" in another.)

√	**Columns**
	Ensure that all input columns contribute to the output. (Only include non-contributing inputs if you're trying to explicitly show their lack of impact.)
	Include a "comment" or "note" column if needed for clarity.
	Put single concepts in a single column. (E.g., one "date" column, not three columns for "month," "day," and "year.") Think "Whole Value."

√	**Rows**
	Put Boolean inputs in normal order (e.g., F F, F T, T F, T T). For one to three variables, show all combinations. Mark outputs "error" for nonsensical cases.
	Arrange inputs in order of increasing complexity.
	Show typical inputs before showing exceptional inputs.
	Group similar outputs together. (Input order is usually more important.)

√	**Cells**
	Make cell values carry information, where appropriate. ("AllowMixedCASELetters" is more self-explanatory than "AazZ".)
	Do not repeat cell values unnecessarily. (Use different values even when "it doesn't matter.") Don't use values that are "coincidentally" the same as others; use identical values to show things that are tied together.
	Ensure that input & output values check all boundary equivalence classes.