Are Your User Stories Well-written?

In Agile software development, user stories are minimal units of work. They describe a feature in non-technical language, as if from the end-user perspective. User stories typically follow a simple template:

As a <type of user>, I want <some goal> so that <some reason>.

Following the so-called INVEST principle, a good user story should be:

- “I” ndependent (of all others)
- “N”egotiable (not a specific contract for features)
- “V”aluable (or vertical)
- “E”stimable (to a good approximation)
- “S”mall (so as to fit within an iteration)
- “T”estable (in principle, even if there isn’t a test for it yet)

Use the following checklists to find out if your user stories are well-written:

☐ Are standard document templates used for capturing requirements?

☐ Are primary & secondary source/requester of requirements referred?

☐ Are the actors consistent with the general set of actors used?

☐ Does the So That clearly state the benefit of the user story?

☐ Do they follow INVEST principles?

☐ For any story that likely requires a screen, is there a mockup?

☐ Can you identify the epic that the stories relate to?

☐ Can you identify the priorities of the stories within an epic?

☐ Are user access rights considered?
Are Your Acceptance Criteria Well-formulated?

In Agile software development, “acceptance criteria,” sometimes also referred to as “the definition of done,” describes a set of predefined requirements a user story should meet, or scope of work the engineering team should complete for the user story to be considered finished.

Use the following checklists to find out if acceptance criteria for your user stories are well-formulated:

- Is it a requirement (what) and not a solution (how)?
- Is it a requirement and not a statement or a goal?
- Does the requirement not use wording that is open-ended (is not open to misinterpretation)?
- Is the requirement simple and clear to understand?
- Does the requirement refrain from making assumptions about the reader’s knowledge of external systems or business processes? If no, are these assumptions clearly stated elsewhere?
- Does the requirement refrain from using non-specific abbreviations and words?
- Does the requirement contain enough information within itself to be of value without a dependency on another requirement at the same level?
- Is the requirement written in an active voice and present time?
- Does the requirement contain/specify quantified data to be used as input for test scenarios?
- If it is a low level requirement, can it be traced to a high level requirement it was spawned from?
- Are negative scenarios considered in the acceptance criteria?
- Is there a worked example that you understand?
- Is it clear how to demo the story?