FULL STACK ENGINEERING DEFINITION OF READY V1.0
Ticket has an INVEST User Story.
Ticket declares required enabling specifications (if applicable).
Ticket contains all reference assets to perform work (wireframes, diagrams, et al).
Ticket is estimated for complexity.
Ticket is assigned a priority grade.
Ticket is interpreted through a Gherkin lens.
Ticket is commented with an abstract explanation of the solutioned implementation
Ticket has a defined acceptance criteria.
Ticket is certified during Grooming Ceremony as DOR stamped.

USER STORY PRINCIPLES (INVEST)
I - A good user story is INDEPENDENT of all others.
N - A good user story is NEGOTIABLE and is not a contract for a feature.
V - A good user story is VALUABLE representing a vertical slice.
E - A good user story is complexity ESTIMABLE.
S - A good user story is SMALL enough to fit in a single Sprint.
T - A good user story is TESTABLE, even if the test is yet to be written.

HOW TO WRITE ACCEPTANCE CRITERIA (GHERKIN)

cucumber.io/docs/gherkin/reference/

INVEST (mnemonic)

From Wikipedia, the free encyclopedia

The **INVEST mnemonic** for Agile software development projects was created by Bill Wake[1] as a reminder of the characteristics of a good quality Product Backlog Item (commonly written in user story format, but not required to be) or **PBI** for short. Such PBIs may be used in a Scrum backlog, Kanban board or XP project.

Letter	Meaning	Description
I	Independent	The PBI should be self-contained.
N	Negotiable	PBIs are not explicit contracts and should leave space for discussion.
V	Valuable	A PBI must deliver value to the stakeholders.
E	Estimable	You must always be able to estimate the size of a PBI.
S	Small	PBIs should not be so big as to become impossible to plan/task/prioritize within a level of accuracy.
T	Testable	The PBI or its related description must provide the necessary information to make test development possible.

Independent [edit]

One of the characteristics of Agile Methodologies such as Scrum, Kanban or XP is the ability to move PBIs around, taking into account their relative priority - for example - without much effort. If you find PBIs that are tightly dependent, a good idea might be to combine them into a single PBI.

Negotiable [edit]

The only thing that is fixed and set in stone in an agile project is an iteration backlog (and, even then, this "can be clarified and re-negotiated... as more is learned."[2]). While the PBI lies in the product backlog, it can be rewritten or even discarded, depending on business, market, technical or any other type of requirement by team members.

Valuable [edit]

The focus here is to bring actual project-related value to the stakeholder. Coming up with technical PBIs that are really fun to code or design but bring no value to the stakeholders violates one of the Agile Principles, which is to continuously deliver valuable software.[3]

Estimable [edit]

If a PBI size cannot be estimated, it will never be planned or tasked; thus, it will never become part of an iteration. So, there's actually no point in keeping this kind of PBI in the Product Backlog at all. Most of the time, estimation cannot be executed due to the lack of supporting information either in the story description itself or directly from the Product Owner. (Language note- "Estimable" as 'the capability to be estimated' is an American English definition. The British English definition 'of high esteem' may confuse some readers. Some versions of the model use the reference "Estimate-able" which also is not a defined dictionary entry.)

Small [edit]

Try to keep your PBI sizes to typically a few person-days and at most a few person-weeks (a good rule of thumb is that any single Product Backlog Item does not take more than 50% of an iteration; for example a single item won't take more than 5 days for a 2-week / 10 day sprint). Anything beyond that range should be considered too large to be estimated with a good level of certainty - these large PBIs may be called "Epics", where an Epic will take more than one iteration to deliver and, necessarily, will need to be broken down into smaller PBIs that can fit comfortably within Iterations. There's no problem in starting with epic PBIs, as long as they are broken down when the time to place them in an iteration backlog comes closer. This implements Lean software development's Just In Time analysis concept.

Testable [edit]

A PBI should only be considered DONE, among other things, if it was tested successfully. If one cannot test a PBI due to lack of information or access (see "Estimable" above), the PBI should not be considered a good candidate to be part of an iteration Backlog. This is especially true for teams employing TDD - Test Driven Development.