**Agile User Story**

**<Project Name>**

**OMNEX Systems**

**Street Address**

**City, State Zip Code**

**Apr 02, 2018**

**Table of Contents**

[What is an Agile User Story? 3](#_Toc510453535)

[The Characteristics of a User Story 4](#_Toc510453536)

[What to Avoid When Writing User Stories 5](#_Toc510453537)

[Examples of User Stories: Good and Bad 5](#_Toc510453538)

[User Story with Acceptance Criteria and Acceptance Testing 6](#_Toc510453539)

[Sample template to collect requirement 6](#_Toc510453540)

[User Stories vs Use Case 7](#_Toc510453541)

# What is an Agile User Story?

A user story is an agile project management tool used to define product or system functionality and the associated benefit of the functionality. In an Agile environment, projects are commonly comprised of a large number of user stories representing various levels of system/product user. The user story describes the type of system/product user, what functionality they want, and why they want it (or why it’s beneficial). The format of a user story is:

As a <*type of user*>, I want to <*feature/function*> so that <*reason/benefit/goal*>.

In order to <receive benefit> as a <role>, I can <goal/desire>

As <who> <when> <where>, I <can> because <why>

As <persona>, I can <what?> so that <why?>

The purpose of the user story is to encourage collaboration among the project team for each defined function of the system or product being developed. Agile project management places emphasis on collaboration rather than formal documentation. Because of this, each user story created by the team represents an opportunity for the team to collaborate, decide which user stories will be worked on in the next iteration, further define the functionality (during iteration planning), and accomplish the work.

While user stories represent a simplified approach to defining functionality, the challenge for the project team is developing user stories with the appropriate level of detail.

User Story is only meant to describe a feature, but not describe how to implement it, meaning leaving out the technical aspect, it should describe the behavior or flow from user’s perspective.

A user story is basically a use case. What do your users need the software to actually do? A story should be a unit of work that a team commits to in a sprint. Whether or not that unit requires subtasks should be up to the team.

**Story points for user story estimation**

Sizing of the story point of early adaptors of the scrum, as sometimes a story will be small enough if we do too much slicing vertically, other time it get way too bigger, as we keep on stuffing the feature in one single user story.

**Reason of having story point**

The points are a fuzzy measurement of how big or small a story is, and should be estimated by the engineer(s) who are implementing it or someone with superior knowledge about the work. In organization/team there should have a standard scale for story points measure, so you can compare multiple stories and say and have some reference like you are able to say that those seem like a similar amount of work like that user story.

More often used is Fibonacci numbers, which is fairly standard. The points don’t really “mean” anything, though, they don’t equate to the amount of time spent or effort for implementation, its simple way of calculating a relative complexity or measuring to get to point A to point B.

Definition of Done (DoD), Acceptance Criteria or Condition of Satisfactions (CoS)

As you fine-tune your estimation, the team should be able to reliably pick up as many stories as they can handle. If your process is working well, that number will probably slowly increase over time, but it take 5–10 sprints to master this technique.

Done (DoD) for stories, acceptance criteria or condition of satisfaction (CoS ) . This helps set expectations within the team as to when a team should consider something done. It also help to write detail level of test cases other advance of writing CoS, DoD or acceptance criteria is, you force yourself to think like an end user. What comes out of this approach is much different than writing user story as a PO

# The Characteristics of a User Story

All user stories should be developed with the expectation that once completed, the functionality defined in the user story will add value to the final product. If it does not add value to the finished product then it should be avoided.

User Stories should be developed and written collaboratively by the project team. This development must also include input and feedback from the ultimate customer or product owner. Remember that agile methodology emphasizes collaboration over formal documentation. The user stories represent a tool to facilitate team communication and collaboration instead of a formal specification or requirement. Not only does this collaboration encourage all of the team to contribute to the project, but only through the eyes of the customer or product owner do we know for certain what does or does not add value.

Finding the appropriate level of detail for user stories is often a challenge for the team. User stories should be general enough to provide a description of the functionality and the benefit while also allowing for innovation and creativity for developing a solution. They should not be so detailed as to lock the team into only one way of accomplishing the solution.

In the creation of their user stories, many agile practitioners subscribe to the INVEST acronym created by Bill Wake which states that user stories should be:

**I**ndependent – user stories should not be sequential or locked into a specific order. The team should be able to develop the user stories in any sequence.

**N**egotiable – user stories should be flexible and without too much detail. Details will be added later through team collaboration.

**V**aluable – the user stories should add value to the final product.

**E**stimable – the team must be able to use the user stories to estimate/approximate work.

**S**mall – large or vague user stories are difficult to estimate. User stories should be able to be designed, built, and tested within a single iteration.

**T**estable – the user story should be able to be tested with some type of acceptance criteria or other test (even if it has not yet been defined)

Acceptance criteria, while not formally a part of the user story format, is a crucial component of user stories. As user stories evolve toward iteration planning, the team should collaboratively discuss the acceptance criteria for satisfying each user story. Acceptance criteria are used by the team to understand when the intent of the user story has been met. Additionally, acceptance criteria are often used by the product/system testers to develop the acceptance tests for the user story.

Acceptance tests are developed to test the functionality of each user story. While an acceptance criteria ensures that a certain functionality is included in an iteration, an acceptance test ensures that functionality performs as expected.

# What to Avoid When Writing User Stories

User stories provide an effective tool to encourage communication and collaboration among the team. However, there are several common mistakes that teams make when developing their user stories which often reduce their effectiveness. Some of these mistakes include:

Not including the customer/product owner – the customer/product owner is a critical member of the team in an agile environment. They’re the ultimate authority on what does or does not add value to the product.

Too much detail – user stories should be broad enough to allow for flexibility and a collaborative effort to further define the user story during iteration planning when the team agrees to incorporate it into the next iteration. User stories with too much detail may lock the team into only one way to develop a solution which discourages innovation and creativity.

Lack of acceptance criteria – as user stories evolve, the team must discuss and capture acceptance criteria. Without acceptance criteria, the team will have no ability of knowing when they’ve satisfied the user story. Additionally, a lack of acceptance criteria will result in a lack of acceptance testing which hampers the team’s ability to ensure functionality of the product performs as expected.

# Examples of User Stories: Good and Bad

Below are examples of user stories which include too much detail, not enough detail, and an appropriate amount of detail.

**Too Much Detail:**

As an end user, I want the ability to access the human resource database to generate a staffing report with employee names, dates of birth, social security numbers, addresses, and telephone numbers so that I can periodically update the company’s staffing contact list.

Note that the above example goes into listing details about what the report should include. Including these items in the user story may prevent or discourage the team from including other important content in reports that will be built. This level of detail should be defined later during iteration planning.

**Too Broad:**

As an end user, I want access to the human resource database so that I can periodically update the company’s staffing contact list.

Note that the above example mentions nothing about the ability to generate reports from the human resource database. Only access to the database is mentioned. By leaving out the key information about generating a report, the team might overlook this important functionality and only grant access to the database when the iteration is performed.

**Appropriate Level of Detail:**

As an end user, I want to access the human resource database to generate reports so that I can periodically update the company’s staffing contact list.

The above example ensures that when iteration planning begins, the team can plan for both access and report generating while further defining what information the reports will consist of.

# User Story with Acceptance Criteria and Acceptance Testing

Below is an example of a single user story with both the acceptance criteria and acceptance testing information included.

**User Story:**

As an end user, I want to access the human resource database to generate reports so that I can periodically update the company’s staffing contact list.

**Acceptance Criteria:**

* Ability to gain access to human resource database
* Ability to generate a report which includes
  + Employee names
  + Dates of birth
  + Social security numbers
  + Addresses
  + Telephone numbers
* Ability to use report data to update staffing contact list

**Acceptance Testing:**

* Database is accessible internally but not from outside company firewall
* MS Access report can be generated from database based on query inputs
* MS Access report data can be exported to MS Excel spreadsheet
* Staffing contact list updates contain only most recent data from the database

# Sample template to collect requirement

Find below attached template to collect requirements:



* ID – Unique Identification of User stories
* Theme -A collection of stories by category. A basket or bucket of stories. By its nature collection of stories by category. A basket or bucket of stories.
* Epic -An epic is a big story. A requirement that is just too big to deliver in a single sprint. Epics need to be broken into smaller deliverables (stories). This helps them support the agile principles (e.g. delivering working software frequently, early continuous delivery, regular reflection)
* Module – Functionality group
* Acceptance Criteria -Acceptance Criteria are conditions which a software application should satisfy to be accepted by a user or customer. It mentions the defined standards of a software product must meet. These are a set of rules which cover the system behavior and from which we can make acceptance scenarios
* Note / Screen Attachment / Reference – Screens, Mockups, Prototype
* Priority -
* Priority Order – Sequence of Priority

# User Stories vs Use Case

|  |  |  |
| --- | --- | --- |
|  | **User Stories** | **Use Case** |
| **Definition** | A user story is a short description of something that your customer will do when they come to your website or use your application/software, focused on the value or result they get from doing this thing. They are written from the point of view of a person using your website or application, and written in the language that your customers would use | A use case is a description of a set of interactions between a system and and one or more actors (where ‘actor’ can be people, or other systems.  is a Use Case drawn out into a step-by-step procedure, sometimes accompanied by a flowchart |
| **Similarities** | Generally formulated in users' everyday language. They should help the reader understand what the software should accomplish | Written in users' everyday business language, to facilitate stakeholder communications |
| **Differences** | Provide a small-scale and easy-to-use presentation of information, with little detail, thus remaining open to interpretation, through conversations with on-site customers | * Use cases organize requirements to form a narrative of how users relate to and use a system. Hence they focus on user goals and how interacting with a system satisfies the goals. * Use case flows describe sequences of interactions, and may be worded in terms of a formal model. A use case is intended to provide sufficient detail for it to be understood on its own |
| **Template** | As a <*type of user*>, I want to <*feature/function*> so that <*reason/benefit/goal* | * Title: "goal the use case is trying to satisfy" * Main Success Scenario: numbered list of steps   + Step: "a simple statement of the interaction between the actor and a system" * Extensions: separately numbered lists, one per Extension   + Extension: "a condition that results in different interactions from .. the main success scenario". An extension from main step 3 is numbered 3a, etc |
| **Additional** | * Acceptance Criteria * DoD * Acceptance Testing | •use case id  •use case title  •rationale/description/goal  •actor/user  •preconditions (the things that must have already happened in the system)  •standard path or main success scenario (what will usually happen, described as a series of steps)  •alternate paths or extensions (variations on the above/edge cases)  •post conditions (what the system will have done by the end of the steps |

Sponsor Acceptance

Date:

<Project Sponsor>

<Project Sponsor Title>