Software Testing - User Stories

User story is one of the primary development artifact for the XP project teams. A user story contains just enough information so that development team can reasonably give estimate about completing, tester can discuss how it will validated and customer can see its value.

One of the common question that we hear most of the time is, how user stories are different from use cases. User stories are much simpler than use cases. User stories are very easy to create, discuss and develop. They also do not contain any technical details.

Typically good user stories are defined in the following format

As a I would like to do so that

While discussing User Stories, you should make sure that they are not very big. A big user story leaves the chances of ambiguity and absence of clarity in the mind of development team. Ideally, you should be able to break user stories so that it can be finished by two person in a week's time.

If you are working as a tester in a agile project and see any story taking longer to finish, consider this as an opportunity to ask question and see if this can be broken by functionality or platform or in any other way.

Once a story is broken to the last level, this story is assigned some points based on how long it will take a pair of programmer to finish it. These user stories are than broken down in tasks and assigned to appropriate developers. Typically in the start of project, you create a deck of user stories and for each iteration or sprint you choose stories based on their priority and time it would take to complete them.

User stories are often written on the paper medium like index cards or post-its. This also ensures that there is no unnecessary details on the user stories. Once development team decides on the stories that will be developed in a sprint, it is further divided into tasks and resources are allocated accordingly.

Most of the time, these user stories are defined only in acceptance term. For example,

As a user I should be able to register with the site so that I can receive newsletter is a good example of user story. As a tester, it should be your job to find out negative scenarios and assumptions in this story. For example, what information user should provide to register? Are there any distinction between optional and required information? Should we validate email address of user before registering?.. and many more.

Remember, one of the main reason for using user stories is to have an invitation to further conversation. According to Ron Jeffries, every user story in XP have three components - Cards (Physical Medium) , Conversation (Discussion Surrounding them) and Confirmation (tests that verify them). One of the widely used acronym for good user stories is called INVEST. Getting right user stories is essential for the success of any agile project and hence this acronym INVEST is used to justify the investment that you make in creating user stories.

According to this acronym,

- **I** stands for independent
- **N** for negotiable
- **V** for valuable
- **E** for Estimable
- **S** for small and
- **T** for testable

As a tester in agile projects, it is very important for you to make sure that every story you work on conforms to the characteristics defined by these terms. There are good reasons for doing this.

Stories should be independent so that it is easier to plan them. Consider the situation where in you need to decide stories to be included in the Sprint and most of the stories are dependent on each other.

Stories should be negotiable, user stories are not contract with the customer they are a platform to clear understanding between everyone involved. Any one should be able to challenge its intention as well as implementation.

User stories should be valuable to some one. You will have situation where in your stories will not bring direct value to your customer, but it should bring value to some one, it could be developer or tester. For example As a tester I should be able to access application below UI layer so that Automation is not dependent on the UI. As oppose to We should have functionality of accessing application with out UI.

Stories should be estimable, remember we are not saying accurate estimate, but it should be reasonable estimate. Since stories are negotiable, these estimates can change but initially it should be possible to estimate reasonable on how much time it would take to complete a story.

Most of the characteristics defined above make sure that user stories are clear and small. This is very important, most of time big stories are also vague. It is also possible to keep team moving with smaller stories as it gives a sense of accomplishing or finishing something. Typically, you should try to complete at least 2 stories per sprint (two programmers and one tester for two week).

Since user stories are requirements which drives the development you should make sure that these stories are testable. You can find this by looking for some characteristics like ambiguity, clarity etc which you might have used in verifying requirement in traditional development model.

Remember, user stories are the crux of agile software development methodology and as a tester you should make sure that your team INVEST in user stories.