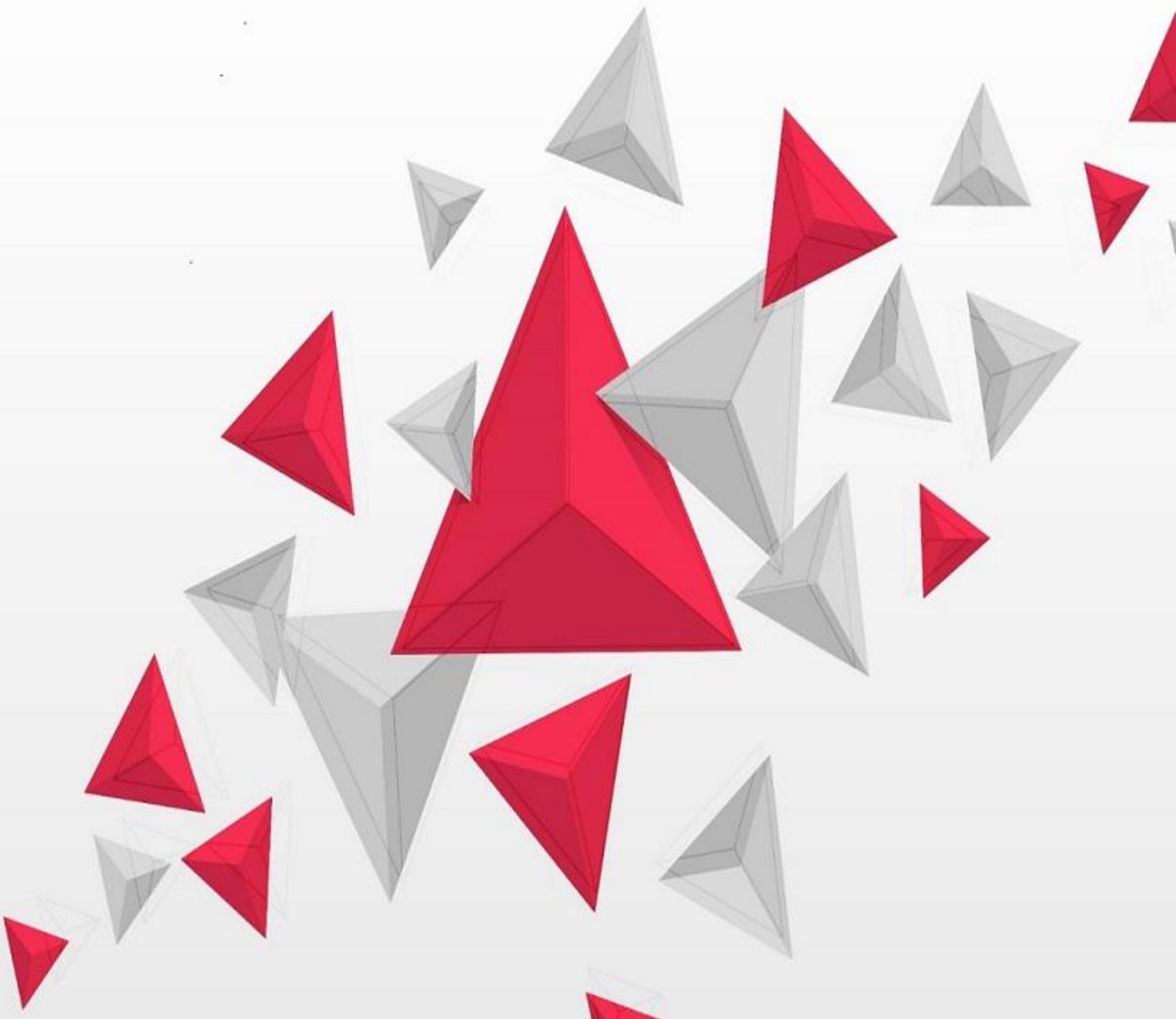


# User Story Guide Book





## USER STORY – INTRODUCTION

Due to ever-changing client requirements and a constantly dynamic economic and business environment, agile software development is now being widely accepted and used as a methodology.

As a quick refresher, **Agile is a set of systems and processes to develop softwares and products**. The most common frameworks for agile are **Scrum, Kanban, and Extreme Programming**.

**User stories** are agile's way of describing the functional requirements. They are a **concise written description (i.e., a 'story') of a piece of functionality that will be valuable to the 'end-user'** – thus, they are aptly named *User Stories*.

The term user story was first heard in 1999 when Kent Beck coined the term in his book on XP (eXtreme Programming). Initially, the stories were written on a card or sticky note and were more of communication starters than requirement documents. With the changing times, the user stories evolved into self-contained documents containing a title, description, specifications, and acceptance criteria.

Going by the definition, a **user story represents a small piece of business value that the agile team can deliver in an iteration**. Iteration is a fixed time duration (2 weeks, 3 weeks) within the project during which the development takes place. A large user story that can not be completed in one iteration is called an '**Epic**', which are further broken down into more manageable user stories.



## ASPECTS OF A USER STORY

There are several aspects to the user stories, and we will now cover all the major ones.

### 1. Short and to the point

User stories are a small yet meaningful division of large functional requirements called 'Epic' in agile terminology. They are precise in the nature of their content and are written to the point without giving many details about the context around the requirement.

Now, there are reasons for it:

- i. Agile stresses more collaboration within the team, and less documentation and user stories resonate with this belief.
- ii. Also, the size of a user story should be small enough to be done in one iteration.

### 2. Easy to estimate

Individual user stories are easy to estimate, aren't very complicated, and provide enough information to calculate the time required to code the functionality described within.

### 3. Easy to change and not rigid

User stories are easy to modify and are not as rigid as their 'use case' counterpart. Considering that agile is all about welcoming changes, it's an essential characteristic of a user story.

### 4. Gives clear and precise acceptance criteria

User stories provide definite and measurable acceptance criteria to the testers so that they can accurately verify the completion of one. This criterion should be agreed upon between the team and the client or product owner\*.

\*Product owner = This term is widely used in agile's 'Scrum' framework and denotes the project's key stakeholder, either a team member or the business's representative. The product owner has a complete vision of what is to be built, prioritizes tasks and features, and guides the team along the way.

### **5. Needs little documentation effort**

User stories don't need extensive brainstorming and documentation like use cases and could be created even by elaborating a sticky note - this helps a lot when you are working in a collaborative environment and have to jot down requirements and make notes on the fly.

### **6. Keeps the technicality out of requirements**

The user stories leave the technical functions or details of 'How' the functionality is to be coded to the technical leads/developers only, and they are never a part of the actual user story. However, if need be, such documents are sometimes added as supplements or appendices to a user story.



## AUDIENCE OF A USER STORY

1. The **Project Manager** will validate whether all the functionalities are listed in their respective user stories and measure their progress and completion in the project burn-down chart
2. **Developers** will use it to understand the requirements and provide estimations against the same
3. The **Business Analysts** will author the user stories, align them with the project scope, get them verified with the product owner, and track them against the user story epic
4. The **Product Owner** will use the user stories as a conversation and discussion base, verify the functionalities being developed, plan the milestones and the future roadmap of the project
5. Any **3<sup>rd</sup> Party Stakeholders** will use it to understand the background and functionality of the features they might be interested in
6. The **Quality Team** will use the user stories to learn about the acceptance criteria, create their test cases and test the acceptance criteria for each of the functionalities
7. The **Audit Team** (whether external or internal) will use the user stories to ensure that all the features have documentation and a signed-off artifact against which they can be tested.



## HOW TO CREATE A USER STORY

So, although the user stories are crisp, writing such tiny information pieces would need a little bit of practice.

But, who is afraid of practice? We will start getting our hands dirty right here!

The golden rule for writing a user story is following the 'statement' method. According to this method, your user story should be in a particular format, i.e. :

*As a <user type here>, I want to <functionality here> so that <benefit achieved>*

So, essentially you are identifying '**Who**' the user is, '**What**' she needs, and '**Why**' she needs it.

But this is a user statement and not a user story, you will say. Yes, correct, but this very statement is the crux, and we obviously will need to add the story around it. So let's do this through an example. Let's consider our existing example of creating the assessment application where we would like to create new users for the application.

**1.** So, our **story statement** now becomes,

*As a 'registered application user', I want to 'have a create user page' so that 'new users could be added to the application'.*

The BA should note that the story statement should be a feature of the software being developed and not a task. Care should be taken that the Who, What, and Why components should be present in the statement.

**2. Description:** We will now go ahead with the Description, where we will elaborate on our statement.

*Once the registered user goes to the Application menu ribbon, he will find a 'Create User' link. Clicking on this link will open a form with the following fields:*

*First Name: Text box accepting alphabets and special characters*

*Last Name: Text box allowing alphabets and special characters*

*Mobile: Text box accepting numerical and special characters*

*Email ID: Text box accepting alphanumeric characters*

*Password: Text box taking alphanumeric characters*

*Confirm Password: Text box accepting alphanumeric characters*

.  
. .  
.

The complete description should be written in a simple, crisp, and easy to understand language without giving much background details and adding any technical jargon, For instance, a 4-page long user story will lose its purpose as it's neither to the point nor easy to estimate.

**3. Acceptance Criteria:** This section will define what the business and product owners require to accept the story and elaborate on the conditions that must be fulfilled if the user story is marked as 'completed'. Example:

- *The logged-in user should be able to enter First Name, Last Name, Mobile Number, Email ID, Password.*
- *The system should validate the text entered in every field as per the field guidelines.*

.  
.

If your user story has more than 5-6 acceptance stories, really analyze the story and see if it makes sense to break it down even further

**4. Attachments:** Any additional notes or references to external documents goes into this section.

For instance, if a mockup or wireframe has been created for the user story, then the same is attached in this section. Mockups for user stories aren't mandatory, but sometimes they are designed for complex requirements to provide more clarity around them.

**5. Assigned to:** In this section, the name of the person, i.e., the developer to whom this requirement is assigned, is written.

**6. Size:** The estimate in terms of efforts required to complete this user story is entered under this section. These efforts could either be in terms of staff hours or

staff days and are usually estimated by the person to whom the user story is assigned.

And, that's it. Short yet effective, isn't it? Since user stories target a specific and immediate need, the business analysts love the user stories.

One last point before we move on to the next section of this lesson - There is no need to have elaborate softwares to create user stories, and mere word or excel based templates should suffice. However, a big team with many user stories under tens of epics might call on specialized software like **Rally**, **Asana**, or **Basecamp** for better management and tracking.





## USER STORY - BEST PRACTICES

Again, I won't leave you with user stories without actionable tips. So, here they go!

### 1. Keep your User Stories short and informal

Remember, don't be too formal or try to stuff too many details in user stories, or you are defeating the purpose of creating one. Focus on what's necessary and leave out non-essential information.

### 2. Write the User Stories collaboratively

Imagine how time-efficient it will be if you write a user story along with the developer working on it while explaining to him the requirements as you write.

### 3. Keep your stories visible and accessible

Stories want to communicate information. Don't hide them on a network drive, but make them visible instead, for instance, by putting all the story statements (belonging to the same epic) on sticky notes when you are brainstorming about it.

### 4. Don't hide that 'User'

Sometimes, we may write a user story like 'As a user, I want to'.

Wait.

What user?

If you don't specify the type of user or customer upfront, your user story starts with ambiguity. Don't commit that mistake.

## **5. User stories and technical details don't go along well**

Neither the business stakeholders nor the end-users know the technicality that goes into developing a user story, and that is why it should be kept separate from the document. Also, a BA should note that the language that goes into writing a user story is simple plain English so that everybody within the team (and even outside) can understand it.

## **6. Lastly, make sure a discussion follows up your user stories on their implementation within the project team.**

Since the user stories are not as detailed as other requirements documents, there is room for ambiguity and confusion. An analyst should strive to put as much information as possible within the user story and then explain any implicit scenarios to the developers working on it.

Applying these tips will help you create not just meaningful user stories but run the project in a true agile fashion!