Summary: *A user story is an informal, general explanation of a software feature written from the perspective of the end user. Its purpose is to articulate how a software feature will provide value to the customer.*

It's tempting to think that user stories are, simply put, software system requirements. But they're not.

A key component of agile software development is putting people first, and a user story puts end users at the center of the conversation. These stories use non-technical language to provide context for the development team and their efforts. After reading a user story, the team knows why they are building, what they're building, and what value it creates.

User stories are one of the core components of an agile program. They help provide a user-focused framework for daily work — which drives collaboration, creativity, and a better product overall.

What are agile user stories?

A user story is the smallest unit of work in an agile framework. It’s an end goal, not a feature, expressed from the software user’s perspective.

A user story is an informal, general explanation of a software feature written from the perspective of the end user or customer.

The purpose of a user story is to articulate how a piece of work will deliver a particular value back to the customer. Note that "customers" don't have to be external end users in the traditional sense, they can also be internal customers or colleagues within your organization who depend on your team.

User stories are a few sentences in simple language that outline the desired outcome. They don't go into detail. Requirements are added later, once agreed upon by the team.

Stories fit neatly into agile frameworks like scrum and kanban. In scrum, user stories are added to sprints and “burned down” over the duration of the sprint. Kanban teams pull user stories into their backlog and run them through their workflow. It’s this work on user stories that help scrum teams get better at estimation and sprint planning, leading to more accurate forecasting and greater agility. Thanks to stories, kanban teams learn how to manage work-in-progress (WIP) and can further refine their workflows.

User stories are also the building blocks of larger agile frameworks like epics and initiatives. Epics are large work items broken down into a set of stories, and multiple epics comprise an initiative. These larger structures ensure that the day-to-day work of the development team (on stores) contributes to the organizational goals built into epics and initiatives.

Why create user stories?

For development teams new to agile, user stories sometimes seem like an added step. Why not just break the big project (the epic) into a series of steps and get on with it? But stories give the team important context and associate tasks with the value those tasks bring.

User stories serve a number of key benefits:

- **Stories keep the focus on the user.** A to-do list keeps the team focused on tasks that need to be checked off, but a collection of stories keeps the team focused on solving problems for real users.

- **Stories enable collaboration.** With the end goal defined, the team can work together to decide how best to serve the user and meet that goal.

- **Stories drive creative solutions.** Stories encourage the team to think critically and creatively about how to best solve for an end goal.

- **Stories create momentum.** With each passing story, the development team enjoys a small challenge and a small win, driving momentum.

Working with user stories

Once a story has been written, it's time to integrate it into your workflow. Generally a story is written by the product owner, product manager, or program manager and submitted for review.

During a sprint or iteration planning meeting, the team decides what stories they'll tackle that sprint. Teams now discuss the requirements and functionality that each user story requires. This is an opportunity to get technical and creative in the team's implementation of the story. Once agreed upon, these requirements are added to the story.

Another common step in this meeting is to score the stories based on their complexity or time to completion. Teams use t-shirt sizes, the Fibonacci sequence, or planning poker to make proper estimations. A story should be sized to complete in one sprint, so as the team specs each story, they make sure to break up stories that will go over that completion horizon.

How to write user stories

Consider the following when writing user stories:

- **Definition of "done"** — The story is generally "done" when the user can complete the outlined task, but make sure to define what that is.

- **Outline subtasks or tasks** — Decide which specific steps need to be completed and who is responsible for each of them.

- **User personas** — For whom? If there are multiple end users, consider making multiple stories.

- **Ordered Steps** — Write a story for each step in a larger process.

- **Listen to feedback** — Talk to your users and capture the problem or need in their words. No need to guess at stories when you can source them from your customers.

- **Time** — Time is a touchy subject. Many development teams avoid discussions of time altogether, relying instead on their estimation frameworks. Since stories should be completable in one sprint, stories that might take weeks or months to complete should be broken up into smaller stories or should be considered their own epic.

Once the user stories are clearly defined, make sure they are visible for the entire team.

User story template and examples

User stories are often expressed in a simple sentence, structured as follows:

“As a [persona], I [want to], [so that].”

Breaking this down:

- "As a [persona]": Who are we building this for? We’re not just after a job title, we’re after the persona of the person. Max. Our team should have a shared understanding of who Max is. We’ve hopefully interviewed plenty of Max’s. We understand how that person works, how they think and what they feel. We have empathy for Max.
- “Wants to”: Here we’re describing their intent — not the features they use. What is it they’re actually trying to achieve? This statement should be implementation free — if you’re describing any part of the UI and not what the user goal is you're missing the point.
- “So that”: how does their immediate desire to do something this fit into their bigger picture? What’s the overall benefit they’re trying to achieve? What is the big problem that needs solving?

For example, user stories might look like:

- As Max, I want to invite my friends, so we can enjoy this service together.
- As Sascha, I want to organize my work, so I can feel more in control.
- As a manager, I want to be able to understand my colleagues progress, so I can better report our sucess and failures.

This structure is not required, but it is helpful for defining done. When that persona can capture their desired value, then the story is complete. We encourage teams to define their own structure, and then to stick to it.

Getting started with agile user stories

User stories describe the why and the what behind the day-to-day work of development team members, often expressed as *persona* + *need* + *purpose*. Understanding their role as the source of truth for what your team is delivering, but also why, is key to a smooth process.

Start by evaluating the next, or most pressing, large project (e.g. an epic). Break it down into smaller user stories, and work with the development team for refinement. Once your stories are out in the wild where the whole team can see them, you’re ready to get to work.

User stories are a key component of agile software development. They are short, simple descriptions of a feature or functionality from the perspective of a user. User stories are used to capture requirements in an agile project and help the development team understand the needs and expectations of the users.

Here are some key characteristics of user stories:

1. User-centric: User stories focus on the needs of the user and what they want to achieve with the software.
2. Simple: User stories are short and simple descriptions of a feature or functionality.
3. Independent: User stories can stand on their own and do not rely on other user stories.
4. Negotiable: User stories are open to discussion and can be refined and modified based on feedback from stakeholders.
5. Valuable: User stories provide value to the user and the business.
6. Estimable: User stories can be estimated in terms of time and effort required for implementation.
7. Testable: User stories can be tested to ensure they meet the needs of the user.
8. Prioritized: User stories are prioritized based on their importance to the user and the business goals.
9. Iterative: User stories are developed iteratively, allowing for feedback and changes throughout the development process.
10. Consistent: User stories follow a consistent format, making them easy to understand and work with.
11. Contextual: User stories are written in a way that provides context to the development team, helping them understand the user’s needs and goals.
12. Acceptance criteria: User stories have clear and specific acceptance criteria that define when the story is considered “done” and ready for release.
13. Role-based: User stories are written from the perspective of a specific user role, helping to ensure that the development team is building features that are relevant and useful to that user.
14. Traceable: User stories are tracked and linked to specific features and functionality in the software, making it easy to trace back to the original user need.

In agile software development, user stories are typically written on index cards or in a digital format, and are used to drive the development process. The development team uses user stories to plan and prioritize work, estimate the effort required for implementation, and track progress towards completing the user stories.

By using user stories in agile software development, teams can ensure that they are building software that meets the needs of the users and delivers value to the business.

In Agile software development and product management User Story refers to a short, informal, and simple description of software features that are required by the end-users in the software system. Its main purpose is to provide software features that will add value to the customer requirements. User stories are considered an important tool in Incremental software development. Mainly a user story defines the type of user, their need, and why they need that. So in simple, a user story is a simple description of requirements that needs to be implemented in the software system.

Pattern of User Story:

User stories are completely from the end-user perspective which follows the Role-Feature-Benefit pattern.

As a [type of user], I want [an action], so that [some reason]

For example :
As the project manager of a construction team, I want our team-messaging app to include file sharing and information update so that my team can collaborate and communicate with each other in real-time as a result the construction project development and completion will be fast.

Writing User Stories :
User stories are from a user perspective. So when user stories are written, users are given more importance during the process. Some points outlined which are taken into consideration during writing user stories like

1. Requirements
2. Tasks and their subtasks
3. Actual user
4. Importance to user words/feedback
5. Breaking user stories for larger requirements

With this also some other principles which are given importance during creating user stories are discussed below.

INVEST Principle of User story :
A good user story should be based on INVEST principle which expresses the quality of the user story because in base a good software product is completely dependent upon a good user story. In 2003 INVEST checklist was introduced by Bill Wake in an article.

1. **Independent –**
   Not dependent on other.
2. **Negotiable –**
   Includes the important avoid contract.

3. **Valuable –**
   Provide value to customer.
4. **Estimable –**
   It should be estimated.
5. **Small –**
   It should be simple and small not complex.
6. **Testable –**
   It should be evaluated by pre-written acceptance criteria.

3 C's in User Stories :

1. **Card –**
   Write stories on cards, prioritize, estimate and schedule it accordingly.
2. **Conversation –**
   Conduct conversations, Specify the requirements and bring clarity.
3. **Confirmation –**
   Meet the acceptance criteria of the software.

Working with User Stories :
Below points represents working with user stories

1. Once the user story is fully written then it undergoes review and verification.
2. In project workflow meetings it is reviewed and verified then added to the actual workflow.
3. Actual requirements and functionality are decided based on the stories.
4. User stories are scored based on their complexity and the team starts work based on user stories.

Importance of creating User stories :

1. Stories clear idea about requirements
2. Makes it easy to understand the features
3. Delivers higher customer satisfaction
4. Fasten development process
5. Creates an effective work environment
6. Enables collaboration between teams
7. Delivery of valuable software

Advantages of using User Stories in Agile Software Development:

1. User stories help to keep the focus on the user’s needs and expectations, which leads to better customer satisfaction.
2. User stories are easy to understand and can be created quickly, which speeds up the requirements gathering process.
3. User stories are flexible and can be refined and modified easily as requirements change.
4. User stories are independent, which makes it easier to prioritize and plan the work.
5. User stories are small and manageable, which makes it easier to estimate effort and track progress.
6. User stories promote collaboration between stakeholders, which leads to better communication and understanding.
7. User stories help to reduce scope creep and feature bloat, as they focus on the essential needs of the user.
8. User stories encourage a customer-focused mindset, as they keep the team focused on delivering value to the user.
9. User stories facilitate a user-centered design approach, as they involve the user in the development process and allow for user feedback.
10. User stories promote transparency, as they make it clear what the team is working on and why.
11. User stories enable incremental delivery, as they allow the team to deliver small, usable pieces of functionality to the user quickly.
12. User stories facilitate testing, as they provide clear acceptance criteria that define when a story is considered “done” and ready for testing.
13. User stories encourage creativity and innovation, as they provide a framework for the team to explore different solutions to meet the user’s needs.

Disadvantages of using User Stories in Agile Software Development:

1. User stories may not provide enough detail or clarity to fully capture the requirements.
2. User stories may not be sufficient for complex or large-scale projects.
3. User stories may be subjective and influenced by the biases of the stakeholders.
4. User stories may not capture all of the requirements, which can lead to gaps in the software.
5. User stories may not be suitable for projects with a high degree of technical complexity.
6. User stories may not capture non-functional requirements, such as performance, scalability, or security, which are critical to the success of the software.

7. User stories may not be appropriate for projects with a high degree of interdependence between features or components, as they are designed to be independent.
8. User stories may not be suitable for teams that lack experience with Agile methodologies or have difficulty working collaboratively.
9. User stories may not provide enough detail for remote or distributed teams, as they rely heavily on face-to-face communication and collaboration.
10. User stories may not capture the full context or user journey, which can lead to a fragmented understanding of the user's needs and goals.
11. User stories may not be suitable for regulatory or compliance-driven projects, as they may not provide enough documentation or traceability for audits or reviews.
12. User stories may not be appropriate for projects with a high degree of uncertainty or risk, as they require a clear understanding of the user's needs and goals.

Overall, the advantages of using User Stories in Agile Software Development outweigh the disadvantages, as they provide a user-centric approach to requirements gathering, which leads to better customer satisfaction and a more effective development process. However, it is important to use User Stories in conjunction with other techniques such as modeling and prototyping to ensure that all requirements are captured and addressed in the software.