USER STORIES

WHAT ARE USER STORIES?

It is the smallest unit of work to be done which carries a certain business value with it.

A user story created to capture a product functionality from user's perspective.

WHY CREATE USER STORIES?

- **Stories keep the focus on the user:** 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

1. Once a story has been written, it's time to integrate it into the workflow.
2. During a sprint or iteration planning meeting, the team decides what stories they'll tackle that sprint. Teams discuss the requirements and functionality that each user story requires.
3. Another step in this meeting is to score the stories based on their complexity or time to completion.
4. Teams use t-shirt sizes, the Fibonacci sequence, or planning poker to make proper estimations.

The most commonly used standard format for a User Story creation is stated below:

As a <user role/customer, I want to < goal to be accomplished> so that I can <reason of the goal>

Example:

As a WhatsApp user, I want a camera icon in the chat write box to capture and send pictures so that I can click and share my pictures simultaneously with all my friends.

3 C'S OF USER STORY

- CARD: Cards are where User stories are written. A card forms the basis for a conversation. When a card is placed om the board, the team is committed to discussing that specific user requirement. The cards are usually 3 inches by 5 inches.

- CONVERSATION: Once the user stories are written, they are then opened to have more discussions, conversations around these requirements. Many meetings like backlog grooming, sprint planning, release planning etc. form the stage for these conversations and information exchange on user stories. The focus is to have more of ongoing conversations and less of detailing the requirements upfront.

- CONFIRMATION: The C is to build user stories acceptance criteria.

    - Acceptance Criteria: Acceptance Criteria of any user story are the conditions that should be fulfilled to satisfy the user or product owner before marking the user story complete. Teams have the acceptance criteria to be fulfilled as their definition of done.
      These are the conditions of contentment on the user story card. They can be written on the back side of the card.

The format for acceptance criteria is as follows:

"Given some precondition when I do some action then I expect the result".

Example (w.r.t for the above user story):

- Please consider chatting with a friend and I should be able to capture a picture.
- When I click on a picture, I should be able to add a caption to the image before sending it.
- If there is some problem with starting my phone camera, an error message like "Camera could not be started." etc., should be shown accordingly.

Hence, the User story defines the requirement for any functionality or feature while the Acceptance Criteria defines the "Definition of done" for the user story or the requirement.

WHO CAN CREATE A USER STORY?

1. Developer.

2. Team member.

3. product owner.->ownership will be with him

4. Some times ->scrum master.

Acceptence criteria come up with the team, product owner decide to priority or depriortise. So, acceptence criteria is with product owner.

Case 1:

Before 3 years, I was working on a Mobile Application Project and the product was an application that was designed for the delivery people.

You would have seen a delivery person coming to your place for delivery. They have a mobile phone on which they ask you to give your signature after delivery. This signature reflects on the portal of the courier service providers like DTDC, FedEx etc.

Let's imagine that the mobile app is just launched and their portals are already existing and up.

Problem: For a Sprint your Product owner has a user story for this mobile app that "As a Portal Admin, I should be able to view the signature taken by the delivery person at the time of delivery". Here the portal (web app) has been changed and updated accordingly to reflect the signature.

As a QA you have to verify if the signature captured in the mobile app is reflecting as expected in the portal.

If you look at this user story, it looks simple, but there is a hidden requirement here that "For historical deliveries, there was no signature reflection functionality, so what should happen if the portal guys view historical deliveries?" Should historical data be wiped out? Should we allow crashes or errors for such data?

Of course not at all, this should be handled graciously.

Solution: When the respective DB tables are updated to add a new column for the Signature location, the old data should have a NULL or 0 value which should be checked and a message stating "No signature exists" should be shown.

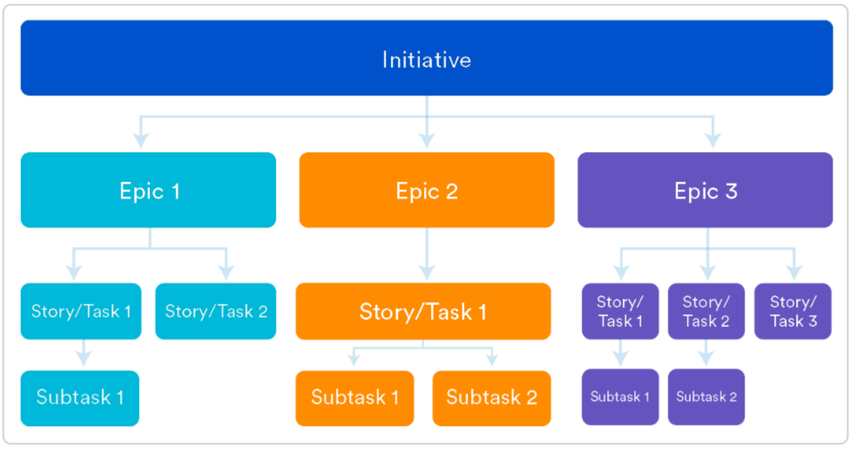
This can be called a miss by the Product Owner or Business Analyst, but this has to be done. Implementing one feature successfully but breaking something along with it is not desirable by the customers. This needs to be done along with the same user story and in the same sprint.

WHAT IS AN AGILE EPIC?

An epic is a large body of work that can be broken down into several smaller stories, or sometimes called “Issues” in Jira. Epics often encompass multiple teams, on multiple projects, and can even be tracked on multiple boards. Epics are almost always delivered over a set of sprints.

EPICS, AND INITIATIVES?

- 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.



Example:

As a restaurant customer, I want to reserve a table online, so I can ensure I have a place to eat at my preferred time.

Acceptance criteria: The system should show available times for reservations. After reservation, the customer should receive a confirmation.

Four main types of user stories:

- Simple: These are individual or self-contained user stories that focus on a particular user or type of product.

- Epic: Groups of related user stories come together to form epics. They may involve multiple users working together or independently or multiple needs for a single type of user to achieve some goal or benefit.

- Thematic: These are major investments and strategies that group epics together. Thematic user stories highlight how a company will achieve wider goals.

- Scaled Agile Framework (SAFe): These user stories add extra details such as a benefit hypothesis, cost of delay, or nonfunctional requirements.

INVEST

INVEST is the acronym you can use to know if you are on track to writing good user stories.

1. **Independent**

   Every agile user story must be independent and should stand on its own. It helps in prioritizing, also called user story priority.

   Also, you can move them to the product backlog if the agile user story is independent.

2. **Negotiable**

   When implementing the agile user story format, there's a collaboration between teams and customers.

   This is where the negotiation comes in. You will have to decide on what to implement and what to skip.

3. **Valuable**

   When writing user stories, remember that they must be valuable to the customers.

4. **Estimable**

   You must be able to estimate an agile user story.

   If a user story is not estimable, then the scope of that story is not well understood. When estimating agile user stories, you can easily distinguish between high and low-effort stories.

5. **Small**

   When writing user stories, know that it must take less effort (days or a few weeks) to implement it.

6. **Testable**

   One of the best indicators of writing better user stories is testability. When a customer cannot tell if you have implemented their story, the entire team's effort may be in vain. To avoid this, write the acceptance criteria before implementing the agile user story.

USER STORIES EXAMPLES

User Story_1: As a registered user, I want to be able to reset my password in case I forget it, so that I can regain access to my account.

Acceptance Criteria:

- The "Forgot Password" option should be available on the login page.
- Upon clicking "Forgot Password," the user should be prompted to enter their email address.
- After entering the email address, the user should receive a password reset link via email within 5 minutes.
- Clicking on the password reset link should redirect the user to a page where they can create a new password.
- After successfully resetting the password, the user should be able to log in with the new password.

User Story_2: As a customer, I want to be able to filter products by category on the e-commerce website, so that I can easily find items I'm interested in.

Acceptance Criteria:

- The website should have a visible filter option labelled "Categories" on the product listing page.
- Clicking on the "Categories" filter should display a dropdown menu listing available product categories.
- Selecting a category from the dropdown should update the product listing to show only items belonging to that category.
- The selected category should be highlighted or indicated clearly to the user.
- The filter should be responsive and work seamlessly on both desktop and mobile devices.

User Story_3: As an admin user, I want to be able to add new tasks to the project management tool, so that I can keep track of upcoming work.

Acceptance Criteria:

- There should be an "Add Task" button visible on the dashboard for admin users.
- Clicking on the "Add Task" button should open a modal or form where the admin can enter task details such as title, description, priority, and due date.
- All fields marked as required should be validated before allowing submission.
- After adding a new task, it should appear immediately in the task list on the dashboard.
- The admin should be able to edit or delete a task after it has been added.

AGILE WORKFLOWS

An agile workflow is a series of stages agile teams use to develop an application, from ideation to completion.

Begin with simplicity, avoiding over-engineering.

Recommended basic workflow states for software teams:

- TO DO
- IN PROGRESS
- CODE REVIEW
- DONE

Additional Workflow States:

- Some teams add states like AWAITING QA and READY TO MERGE for finer tracking.

Autonomous Teams and Workflow:

- Agile teams evolve to handle diverse tasks autonomously.
- Developer involvement across the workflow stages is a hallmark of agility.

Adaptable Workflows:

- Healthy workflows adapt to team needs.
- Address pain points in retrospectives.
- Importance of flexible workflow configuration in issue trackers.

Transition to Agile Process Flow:

- Create statuses for each work type.
- Lean set of statuses reflecting different phases of work.
- Consider metrics for reporting and organization-wide transparency.

Optimizing Workflow:

- Implement Work In Progress (WIP) limits to manage flow.
- Monitor throughput and identify bottlenecks.

- Continuous optimization around WIP limits improves efficiency.

Scaling Agile Workflow:

- Challenges in maintaining consistency across multiple agile teams.
- Benefits of shared workflows for collaboration and transition of work.
- Adaptation and collaboration lead to improved workflows.

Agile Workflow with Jira Software:

- Ability to share workflows while customizing visualization on agile boards.
- Flexibility in representation without compromising shared workflow assets.

Continuous Improvement:

- Agile process development should be iterative.
- Regular retrospectives and adaptation based on team dynamics and culture changes.

Agile Retrospectives Overview:

A reflection opportunity for agile teams after completing a time-boxed work period (typically a sprint).

Aims to identify successes, challenges, and areas for improvement.

Importance of Agile Retrospectives:

- Aligns with Agile Manifesto principles, promoting adaptation and continuous improvement.
- Ensures issues are addressed promptly, fostering team accountability and efficiency.

Discussion Points:

- Covers both technical and team-related aspects.
- Emphasizes actionable solutions for improvement.

Agile Retrospective Techniques:

Various methods to refresh retrospectives and maintain momentum:

- 4 L's
- Dot voting
- Lean coffee
- One word
- Past two months map
- Sailboat
- Start, Stop, Continue
- Question cards

Best Practices:

- Set duration and frequency expectations.
- Come prepared with a structured agenda.
- Rotate facilitator roles for diverse perspectives.
- Foster collaborative and honest discussions.
- Ensure shared responsibility and accountability.
- Conclude with actionable items for the next sprint.
- Document meeting notes for accessibility and accountability.

Agile Retrospective Agenda:

- Set the stage
- Gather input
- Brainstorm ideas
- Pick solutions
- Close

1. **Product Backlog:** It's a prioritized to-do list for the project, managed by the Product Owner. It contains all the tasks, features, enhancements, and fixes that need to be done on the product. Items are added, removed, or adjusted based on feedback and changing priorities. It guides the team on what to work on next and serves as a roadmap for the product's development.

2. **Sprint Backlog:** A list of tasks and user stories identified by the development team to be completed during a specific sprint. It is derived from the product backlog and contains the items that the team commits to deliver during the sprint.

3. **Sprint:** A time-boxed period, usually 2-4 weeks long, during which a set of features, enhancements, or fixes are developed and delivered. It is a fundamental unit of development in Scrum.

4. **Product Owner:** A key role in Scrum responsible for representing the interests of stakeholders, maintaining the product backlog, and ensuring that the team delivers value to the business.

5. **Scrum Master:** Another key role in Scrum responsible for facilitating the Scrum process, removing impediments, and helping the team work effectively. They also ensure that the team adheres to Scrum principles and practices.

6. **Increment:** A visible, usable, and potentially releasable version of the product produced during a sprint. It is the sum of all the completed product backlog items at the end of a sprint.

7. **Daily Scrum:** Also known as the daily standup, it is a short, time-boxed meeting (usually 15 minutes) held every day during a sprint where team members discuss their progress, plans for the day, and any obstacles they are facing.

8. **Sprint Retrospective:** A meeting held at the end of each sprint where the team reflects on their performance, discusses what went well, what could be improved, and identifies actions to enhance their process for the next sprint. It's an opportunity for continuous improvement.