User Stories:

What is a user story:

* Simply told, a user story is a brief, simple and precise description of a specific feature written by a developer from customer point of view.
* As the user story is written from the customer or user’s viewpoint, it is written in such an informal and straightforward way that the user does not have any difficulty understanding it.
* The user story is articulated so explicitly that even a non- specialist can understand it.
* Writing user stories is a good way of encapsulating the user requirements with a focus on their goals.

3 C’s User:

* Cards
* Conversation
* Confirmation

Cards:

* The cards type is not essential, what matters is what is written on it. Writing the user stories on cards on cards helps keep them short but enables all concerned to know what the story is about.
* Format used for writing the user stories on the cards is: As a (User) ,I want to (accomplish this task),so that I can(achieve this objective or goal).
* Once the product owner finalizes the user story for a particular Sprint(time-box), they hand the card to the developer.

Conversation:

* It depicts the customer requirements, but these requirements should be discussed further and if needed, refined and given to the developers.
* When the team holds a conversation on the user story, it changes into a deliverable.
* Conversation also helps bring clarity about the user story to the team members as sometimes it may be hard to interpret.
* Team should thoroughly discuss the story, from understanding the customers requirements to what needs to be delivered.
* Conversation can be of any type like verbal, through emails, etc… The purpose is to exchange views and ideas.
* Conversation do not end once the team starts working on the user story.
* As they progress in work, team members would learn more things, and exchange of views and ideas would continue.

Confirmation:

* The Confirmation is that Acceptance Criteria that should be fulfilled and tested to ensure that the user requirements have been met and delivered correctly.
* The acceptance criteria provide the basis for acceptance tests that would determine the acceptability of a product.
* The product owner should test the functionality and confirm that it meets the customer requirements described in the user story.

What are the 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.
* Stories fit neatly into agile frameworks like [scrum](https://www.atlassian.com/agile/scrum) and [kanban](https://www.atlassian.com/agile/kanban).
* 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:

* **Stories keep the focus on the user -** 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 those goals.
* **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.

User templet and examples:

* As a [user], I [want to], [so that].
* Initiative

Epic

Story

Examples of user stories:

* Example:

Create a task

as a Student,

I want to add a new task to my list,

so that I can keep update track my tasks.

Acceptance criteria:

* An acceptance criteria are a set of conditions a product must meet to be accepted by user, customer, or other user. Written by a business analyst.

A checklist for writing acceptance criteria:

1.check to everyone involved

2.can be tested or verified

3.Either pass or fail

4.Focus on the outcomes, not how the outcome is achieved

5.As specific as possible.

Common mistakes to avoid when writing acceptance criteria:

1.Ignoring the user perspective

2.Telling the developers “how” to do the work

3.Writing acceptance criteria too early

4.Waiting until development is underway in the sprint to write the acceptance criteria

5.The criteria are broad

6.The criteria are vague

7.There are a cumbersome number of criteria (a large quantity may indicate you need to break up the work into smaller parts).

Example Of Acceptance Criteria:

1.Clicking on the "Add Task" button should open a form with fields for entering task details.

2.The task form should include fields for entering a task title, description, due date, and priority.

3.I should be able to set a due date for the task. If no due date is specified, it should default to "No due date."

4.Clicking the Create Task button should close the form and add the new task to the task list.

5.Upon successful creation, the new task should immediately appear in the task list, reflecting the entered details.

6.Completed tasks should be distinguishable from active tasks, using visual cues or a separate section.

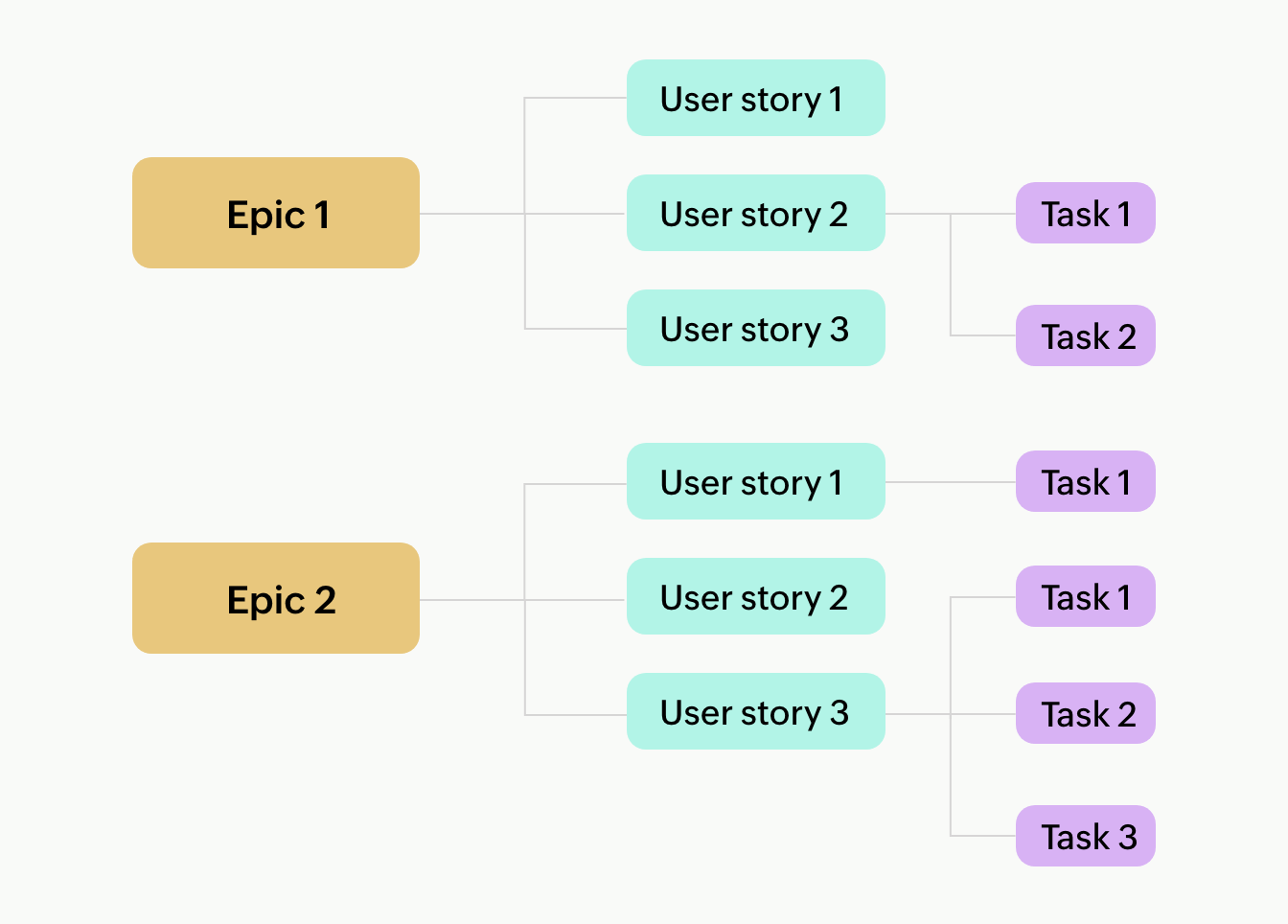
Invest:

* I- Independent (of all others) : Dependencies cause delays. You don’t get user-ready working software until both or all dependent stories are complete.
* N- Negotiable (not a specific contract for features) : Treating the story as an evolving conversation between product owner and development team builds a shared understanding and harnesses everyone’s expertise: the product owner knows the benefits you’ll bring the users and the development team knows the best way to do this.
* V- Valuable (or [**vertical**](http://guide.agilealliance.org/guide/incremental.html)) : Your goal is to deliver valuable working software.
* E- Estimable (to a good approximation) : T**he story gives the development team enough detail to estimate the size of the story.**
* S- Small (so as to fit within an iteration) : **The story is the smallest piece of work that will deliver useful software.**
* T- Testable (in principle, even if there isn’t a test for it yet) **: The story is clear enough that you can assess if the story is done.**

Epic:

Each epic story is listed vertically with the columns begin for the user.

* User Registration: As a new user, I want to register for an account by providing my email and creating a password so that I can access the application.
* Email Verification: As a registered user, I want to receive a verification email with a unique link to confirm my email address and activate my account.
* Login: As a registered user, I want to log in using my email and password to access the features of the application.
* Forgot Password: As a user, I want to request a password reset link via email in case I forget my password.
* Role-Based Access: As an administrator, I want to assign different roles (e.g., admin, regular user) to control access levels and permissions within the application.
* Two-Factor Authentication: As a security-conscious user, I want the option to enable two-factor authentication for an extra layer of account security.



Product Backlog :

* A product backlog is a comprehensive list of all the tasks, features, and improvements that need to be completed to deliver a product.
* Think of it as a to-do list for your product development process, but it’s not just any to-do list.
* The product backlog is owned by the product owner.
* The product owner adds, changes, and prioritizes items in the backlog as the project’s needs evolve.
* It helps the team understand what needs to be done next and allows them to focus on the most valuable tasks. The product owner uses the backlog to guide the team’s work, ensuring that they are always working on the most important features.
* Product backlog is different from a user story.
* Product backlog is made up of may diff items, which can include user stories, but also bugs, tasks, and other work items.
* If we consider the product backlog as a book, then user stories are like individual chapters. Each one is important and contributes to the overall narrative, but they are part of a larger whole.

Product Owner:

* A product owner is responsible for ensuring the success of a project in Scrum.
* A Scrum framework is an [Agile methodology](https://www.simplilearn.com/benefits-of-agile-methodology-article) that facilitates communication and self-organization within a team.
* The Product Owner is the primary point of contact on behalf of the customer to identify the[product requirements](https://www.simplilearn.com/what-is-requirement-analysis-article) for the development team.
* The Product Owner must communicate with the development team to explain the product features to be implemented.
* The role of the Product Owner is critical for companies that are keen to move to an agile-based product development methodology.
* The role of the Product Owner is to maximize the value addition of the products that are developed by the [agile scrum team.](https://www.simplilearn.com/everything-you-need-to-know-about-becoming-agile-scrum-master-article)
* The Product Owner has to collaborate and work closely with various stakeholders such as customers, business leaders, development teams, [project managers](https://www.simplilearn.com/what-qualifications-do-you-need-to-be-a-project-manager-article), and other [stakeholders.](https://www.simplilearn.com/stakeholders-impact-on-the-projects-article)