User Story

- A user story is an informal, **general explanation of a software feature** written from the **perspective of the end user** (i.e., what they want from the software or system and why).
- Its purpose is to articulate how a software feature will provide value to the customer.
- These stories use mostly **non-technical language** to **provide context for the development team.**
- After reading a user story, **the team knows why they are building**, **what they're building, and what value it creates**.
- Overall, **it's an end goal**, not a feature, expressed from the software user's perspective.

Why create user stories?

- For development teams new to agile, user stories sometimes seem like an added step. a collection of stories keeps the **team focused on solving problems for real users.**
- User stories make the project more manageable and motivate the team by **dividing complex tasks into small parts**.
- With the end goal defined, the team can **work together** to decide how best to **serve the user and meet that goal.**
- Stories encourage the team to **think critically and creatively** about how to best solve for an end goal.

User Story Format

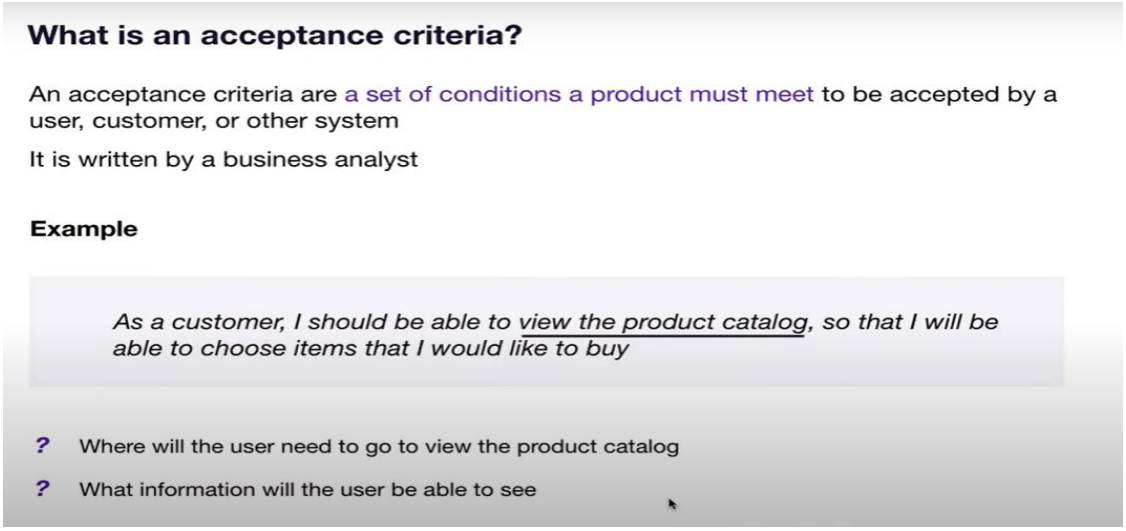
As a <**role**> , I want <**function**> so that <**benefit**>.

- It answers to the questions **who, what & why**.
- A good user story will follow the criteria termed as **INVEST**.

- **I: Independent** - User stories should not be interdependent. You must ensure that any changes to a User Story do not affect another. This is to avoid increasing the work burden.

- **N: Negotiable-** Stories should be discussable. As the team gets more familiar with the story context, additional ideas may emerge, the wording of the story may change, and details may be added.
- **V: Valuable** - While creating user stories, you must make it understandable and clearly state the value of the product to the user.
- **E**: **Estimable** - The development of the goals highlighted by the user story should be measurable. This will allow your team to determine their priorities as well as their working schedule.
- **S**:**Small** - User stories need to be short.
- **T**: **Testable -** Finally, the user story needs to have an achievable goal that can be tested to see if it delivers on user expectations.

Acceptance Criteria



User Story Example

User Story for creating a new account

User Story - As a **new Amazon app user**, I want to **create a new Amazon account** , so that I can **easily buy the product**.

Acceptance Criteria

Scenario: Create new Amazon account.

Given: The user can open the app/website.

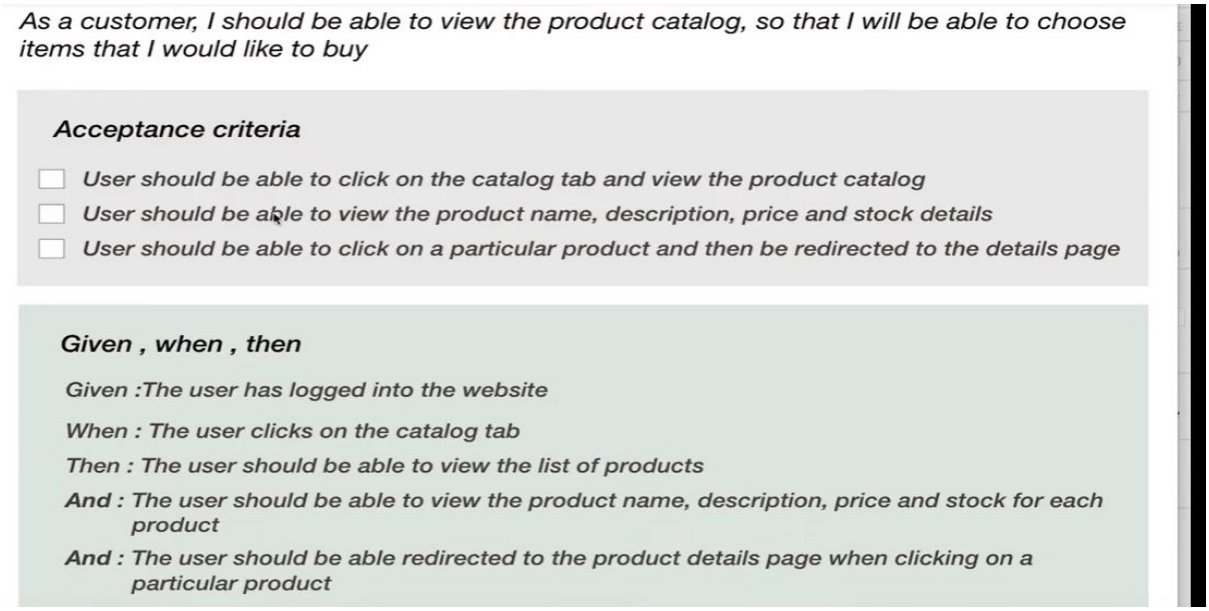
When: The user clicks signup button.

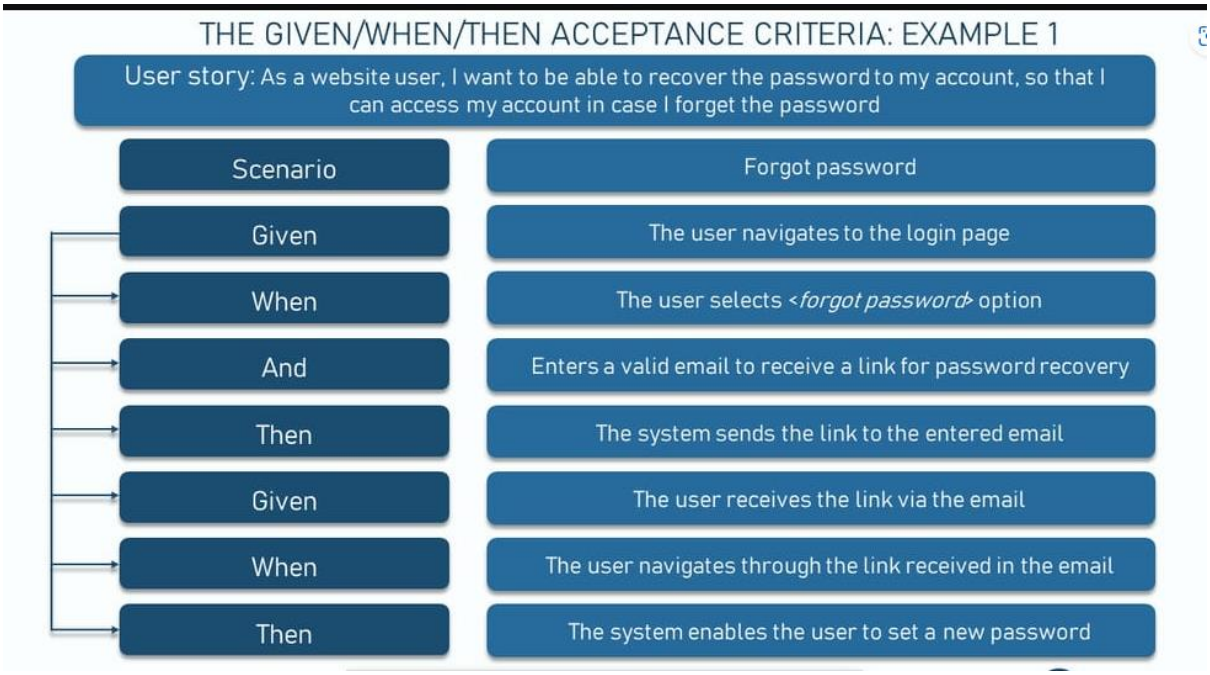
And: The user enters the personal details in the promoted form.

Given: The user receives an OTP via email or SMS for verification.

When: The user enters OTP manually.

Then: The user can start using the app for buying the desired product.

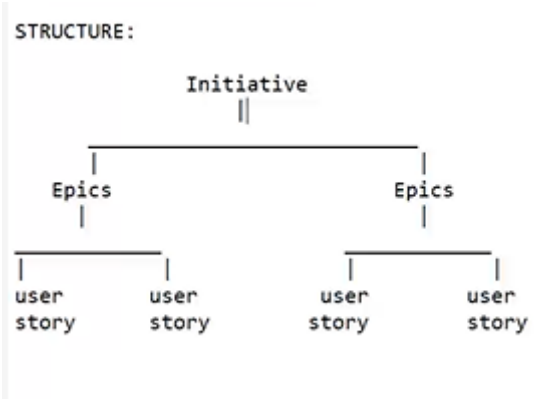




Intiative:are collections of epic that drive toward a common goal. EX:Scool management system [title of the project]

Epics:are large bodies of work that can be broken down into several smaller task. EX: loginpg,attendance,fees[modules]

user story:are short requirements or requests written from the prespective of end user. EX:breif details about the modules.



3 C's of User Story

1. **Card:** This refers to the representation of the user story. The card typically contains the title of the user story, a short description of the user's requirement or need.
2. **Conversation:** This refers to the discussion that takes place around the user story. The conversation typically involves clarifying questions and discussions around the acceptance criteria, and may involve input from developers, testers, and other stakeholders.
3. **Confirmation:** This refers to the acceptance criteria that must be met in order for the user story to be considered complete. The confirmation criteria are typically defined during the conversation phase and may include things like specific user actions, expected outcomes, or performance criteria. By defining the acceptance criteria up front, teams can ensure that everyone is on the same page and that the user story is well-defined.

User story for Amazon app

1.Registration

User Story 1#
As an Amazon app user, I can register the application by entering my email, password, and name.

Acceptance criteria

Scenario: Create new Amazon account.

Given: The user can open the application.

When: The user clicks signup button.

And: The user enters the name, email, and password in the promoted form.

Then: The system created account for the user.

User Story 2 #
As an Amazon app user, I will receive the OTP for verification via email or SMS once I have registered the application.

Acceptance criteria

Scenario: OTP verification after creating account.

Given: The user receives an OTP via email or SMS for verification.

When: The user enters OTP manually.

Then: The user can start using the app for buying the desired product.

2.Login

User Story 1#

As an Amazon app user, I can login the application by entering my name and password.

Acceptance criteria

Scenario: Login my account.

Given: The user can open the application

When: The user clicks login button.

And: The user enters the name and password in the promoted form.

Then: The user logged into application.

User Story 2#

As an Amazon app user, I can reset my password if I have forgotten the password.

Acceptance criteria

Scenario: Forgot password.

Given: The user navigates to the login page.

When: The user clicks <forgot password> option.

And: The user enters a valid email to receive a link for password recovery.

Then: The system sends the link to the entered email.

Given: The user receives the link via the email.

When: The user navigates through the link received in the email.

Then: The system enables the user to set a new password.

3.Search product

User Story 1#

As an Amazon app user, I can search the product by category.

Acceptance criteria

Scenario: Search product.

Given: The user navigates to the home page.

When: The user searches the product by category.

Then: The system generates the sorted results based on category.

4.Filter product

User Story 1#

As an Amazon app user, I can filter results by price.

Acceptance criteria

Scenario: Filter product.

Given: The user can find the filter option in the sorted result page.

When: The user clicks the filter option.

And: The user filter the product by price.

Then: The system generates the filtered results.

5.View Product list

User Story 1#

As an Amazon app user, I can view the product list so that I can purchase.

Acceptance criteria

Scenario: View product list

Given: The user can see a thumbnail image for each product.

When: The user clicks thumbnail to view the product details.

And: The user can click add to cart option.

Then: The system adds the product to the cart from detailed page.

6.Review Cart

User Story 1#

As an Amazon app user, I can review my cart, so that I can make adjustments prior to checkout.

Acceptance criteria

Scenario: Review cart

Given: The user navigates to the cart page.

When: The user can adjust the quantity of the item .

And: The user can remove the cart item.

Then: The user can view the final cost and tax before shipping.

7.Checkout product

User story 1#

As an Amazon app user, I can checkout the product from cart page.

Acceptance Criteria

Scenario: Checkout product

Given: The user can trigger the checkout from cart page.

When: The user can enter the shipping address.

And: The user can choose payment options.

Then: The user can verify payment via the payment processor.

Given: The system can show the total cost including tax and shipping.

When: The user can place the order.

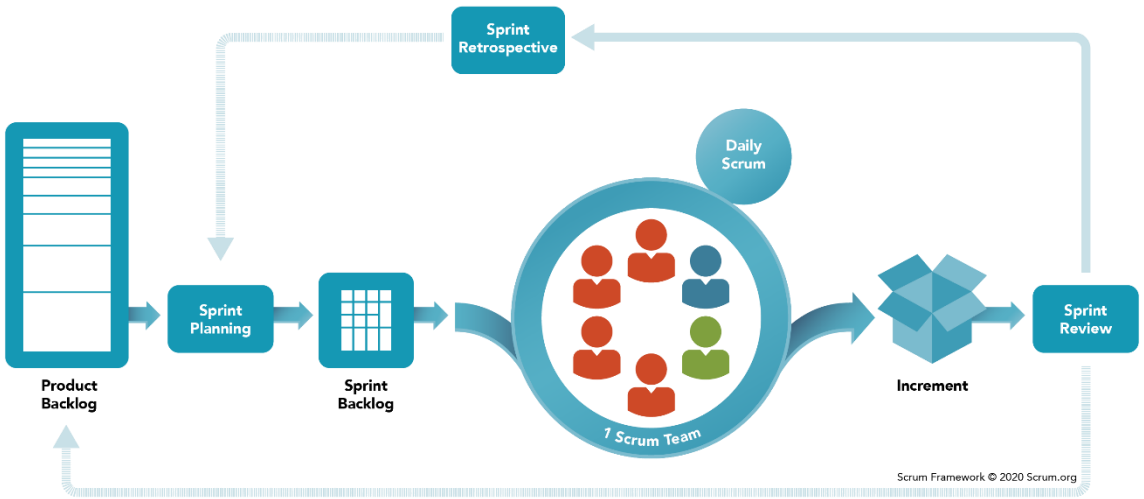
Then: The system confirms the order via email or SMS.

Scrum

- **Scrum** is the type of Agile framework. It is a framework for managing and organizing work , especially in software development.
- It emphasizes teamwork, collaboration, and iterative progress.

Silent features of Scrum are:

- Scrum is light-weighted framework.
- Scrum emphasizes self-organization.
- Scrum is simple to understand.
- Scrum framework help the team to work together.



Roles – Scrum defines three main roles

1.Product Owner

- They are focused on understanding business, customer, and market requirements, then prioritizing the work to be done.

- Effective product owners:
    - Build and manage the product backlog.
    - Closely partner with the business and the team to ensure everyone understands the work items in the product backlog.
    - Give the team clear guidance on which features to deliver next.

2.Scrum Master - The scrum master is the master of scrum framework, who **ensures the scrum framework is followed**. Scrum has a clearly defined set of roles and rituals that should be followed and the scrum master works with each member of the scrum team **to guide and coach the team through the scrum framework**.

3.Development Team- On a scrum team, a developer is anyone on the team that is delivering work, including designing, coding, and testing the product.

Artifacts

Scrum artifacts help manage the work:

1.Product Backlog

A prioritized list of tasks or user stories , maintained by the product owner, that represents all the work needed to be done on the project. The most important items are shown at the top of the product backlog so the team knows what to deliver first.

2.Sprint Backlog

The set of product backlog items selected for the sprint by the developers (team members), plus a plan for delivering the increment and realizing the sprint goal.

3.Increment

A sum of usable sprint backlog items completed by the developers in the sprint that meets the definition of done.

Scrum Events

Sprint-Work is broken down into small, manageable chunk called “sprint”. A Sprint is a time box of one month or less. Each sprint should

bring the product closer to the product goal. A new Sprint starts immediately after the completion of the previous Sprint.

Sprint Planning-The entire scrum team establishes the sprint goal, what can be done, and how the chosen work will be completed. Planning should be timeboxed to a maximum of 8 hours for a month-long sprint, with a shorter timebox for shorter sprints.

Daily Scrum- The developers (team members delivering the work) inspect the progress toward the sprint goal and adapt the sprint backlog as necessary, adjusting the upcoming planned work. A daily scrum should be timeboxed to 15 minutes each day.

Sprint Review- The scrum team inspects the sprint outcome, If the product still has some non-achievable features, it will be checked in this stage and then passed to the Sprint Retrospective stage.

Sprint Retrospective - In this stage quality or status of the product is checked. The team identifies improvements to make the next sprint more effective.

Test Plan

In software development, a test plan defines your testing team's test strategy, goals, and scope, which ultimately work together to ensure that all your software components are testes sufficiently before a release.

Efficient test plan:

1. Define the release scope.
2. Schedule timelines.
3. Define test objectives.
4. Determine test deliverables.
5. Design the test strategy.
6. Plan test environment and test data.

How to create a test plan

1. Define the release scope

Before any test activity occurs, it's important to define the scope of testing for your release. This means defining the features or functions that need to be included in the release, considering any constraints and dependencies that can affect the release, and determining what type of release it is.

Examples of questions to ask when defining the release scope include:

- Are there new features being released in this version?
- What are the risk areas?
- Are there any particularly sticky areas where you've seen regressions in the past?
- What type of release is it? Is this a maintenance release that includes bug fixes? Is this a minor feature release? Is this a major feature release?
- What does being "done" actually look like for your team?

For example, what information would you require if your organization has just launched a new e-commerce site and wants to test it before it launches?

Whether it's talking with the developers to understand the scope of the project, or working with a product manager to walkthrough new functionalities and user flow, defining the scope ensures that accurate information is being shared and that there is a common understanding of the product's goals, expectations, and features.

2. Schedule timelines

Specify release deadlines to help you decide your testing time and routine. Here are some pointers for determining timelines:

- Consult your project manager to understand the current release timeline.

- Look at past release times and schedules.
- Consider the timeframes for development: Your development team might have a set schedule for finishing development work. Make sure you comprehend that timeframe so you can adjust the testing schedule.
- Add some extra wiggle room: It's common to encounter unexpected delays. Including extra time for unforeseen events can help you stick to your plan.
- Review and update the schedule frequently to ensure the test timetable is attainable.

3. Define test objectives

A test objective is a reason or purpose for designing and executing a test. These objectives ultimately help guide and define the scope of testing activities.

Examples of general test objectives include:

- Identifying and reporting defects
- Testing new features
- A certain level of test coverage

Examples of objectives for specific types of testing include:

- **Functional testing objectives:** Ensure the software works as it should. Examples of goals for this objective include: Validating user workflows, data processing, and verifying input/output parameters.
- **Performance testing objectives:** Ensure the software is efficient and can handle various loads. Examples of goals for this objective include: Verifying software reaction time, throughput, and scalability.
- **Security testing objectives:** Uncover program security flaws. Examples of goals for this objective include: Verifying authentication and authorization features and identifying potential threats.

- **Usability testing objectives:** Concentrate on ease of use and user experience. Examples of goals for this objective include: Validating software accessibility, verifying user flow, and identifying user-related issues.

Measure testing with the right metrics

Metrics assess the overall quality of a release, the progress of your testing, and the effectiveness of your testing (for a particular test cycle or the entirety of your testing).

They provide visibility into your testing process and overall product quality, ultimately helping your team decide if your release is ready to ship. Here are some metric formulas you might consider:

Defect Density

- Defect Density = Defect count/size of the release (lines of code)

Example: If your software has 150 defects and 15,000 lines of code, its defect density is 0.01 defects per line of code.

Test Coverage

- Test Coverage = (Total number of requirements mapped to test cases / Total number of requirements) x 100.

Defect Detection Efficiency (DDE)

- DDE = The percentage of defects detected during a phase / Total number of defects

Time to Market

- TTM = The time it takes for your company to go from idea to product launch