

# Agile

- [Definition of Done](#)
- [Stand up](#)
- [Agile Trainings](#)
- [User Story Template](#)
- [Agile Discussion Jan 25 2021](#)
- [ADO Work Item Standardization](#)

# Definition of Done

We consider a User Story or Defect to be complete when it has met the following criteria:

- Meets the Acceptance Criteria for the story
- Engineer has done their own testing and verified that everything is working as expected
- Smoke tests (ensure that it builds and runs) for Utopia project and any others as needed
- Tested by the team tester as applicable
- Pull-requested into a feature
- As part of the pull-request, the branch is reviewed by at least two other Engineers
- All tasks on the story are marked as completed

# Stand up

3 Standup Questions (per story)

- How far along are you?
- Will you complete in time?
- Who can help ? (How can we adjust the collaboration)

# Agile Trainings

[Agile-GettingStarted.pptx](#)

Agile Training to go over the basics of Agility, covering:

- Agile Manifesto Values and Principles
- User Story Writing
- Requirements Hierarchy: Epics, Features, User Stories
- Estimation techniques
- High-level overview of Scrum and Kanban

[ProductOwnerTraining.pptx](#)

Product Owner Training

[ScrumMasterTraining.pptx](#)

Scrum Master Training

# User Story Template

## User Story

Write user story text description.

### Note

User stories should be written according to the following pattern:

As a , I want  so that .

Important: a user story should consider needs of particular type (most often role) of user.

User stories should follow INVEST principles:

- Independent. Reduced dependencies = easier to plan;
- Negotiable. Details added via collaboration;
- Valuable. Provides value to the customer;
- Estimable. Too big or too vague = not estimable;
- Small. Can be done in less than a week by the team;
- Testable. Good acceptance criteria.

Note: large (those which won't fit into a single sprint) user stories, also called epics, should be eventually split into smaller user stories (if it's possible at the moment).

Each user story will go through the following lifecycle before implementation begins:

1. Writing a user story (anyone is allowed to write a user story, most often it'll be Product Owner);
2. Discussion (all teams should take part in the final discussion):
  1. [Optional] Breaking an epic into smaller user stories;
  2. Splitting into sub-tasks (with tech specifics);
  3. Verification of acceptance criteria completeness;
  4. Update user story contents if needed;
3. Mark as ready. Goes to the implementation pipeline when it's time.

## Activity

Specify the related activity.

[Optional] provide a short description of the activity if necessary.

### Note

Activities (also known as themes) are common groups (usually features) that combine user stories together. As an example, "Account" activity may contain user stories describing registration, password recovery, profile updates, etc. Granularity of activities should be chosen in a way that doesn't lead to activities with too many user stories (in such cases these activities should be split into smaller ones).

When combined with user stories, activities present a good visualization of the backlog, called user story map.

## Type

Epic or User story. Epics should only be used as a temporary stub for user story(ies). See definition of an epic above.

## Status

Status of user story readiness in terms of the lifecycle. A user story should only be marked as ready when all sections were filled in and reviewed by team members.

An example:

- In icebox (new);
- In backlog (discussion in progress);
- Ready to go (discussed and verified);
- In progress, in testing, etc.

## Components

Provide a list of affected components (e.g. back-end, front-end, etc.).

## Sketches/wireframes/mockups

All graphical data goes here (with some captions if necessary).

An example:

Sketch 1  
Image not found or type unknown

## Workflow

Describe the whole workflow from user's perspective, step by step, with all necessary details.

All stakeholders should be able to understand contents of this section, so no sophisticated technical details should be placed here.

An example:

1. User opens web app link.
2. User enters credentials.
3. User logs in and sees the dashboard.

## Acceptance Criteria

Describe details that are required to validate the software once it's implemented: UI elements behavior, validation logic, messages of all types (validation errors, hints, etc.).

An example:

- Security
  - No other web pages except Sign In should be available for non-authenticated users.
  - When a non-authenticated user opens a link leading to one of internal pages and then logs in, the app should redirect the user from Sign In to the page corresponding to the original link
- UI elements
  - "Login" input field
    - Required field
    - Maximum 255 characters

## Tech Notes

Describe all low level technical details here, like API specification, error handling conventions, timeout values, etc. May also contain some diagrams, (pseudo) code snippets, etc.

## Links

Provide links to related epics/user stories, mockups, API specification, other high-level design documents, etc.

# Agile Discussion Jan 25 2021

## Named Roles:

Splitting of PO and Scrum Master

- Product Owner (PO) - similar to Product Manager, prioritize sprint work based off of the product manager. Remove blockers along with the scrum master and work with scrum master/PO from other teams
- Scrum Master - manages the logistics/ceremonies and coaching the team
- Quinn and Royce become the POs - need to find who will be SM on each team (Bryan and Abe)
- Meet with POs to discuss organization of backlog, aggregation of team boards, reorganize around features

Do a full split of the teams, not a large kanban of items.

## Ceremonies in Order:

- Will be covered in training, SM will create and organize

## Product Council:

- Discuss processes, backlog, prioritization
- Separate from current Stakeholder meeting

## Release Cycle:

Faster releases - Goal of constant releases during the day

## Action Items:

- Check with Abe and Bryan to become SMs - Royce & Quinn
- Training on SM/PO roles (Agile training - 3hr WED, SM/PO - 1.5-2hrs each FRI and then following WED 8:30am MST) - Bryce



- Product Council meeting - setup weekly meeting (Eventually include all process stakeholders: Marc, Brian, Jesse, Eric, Emily)

# ADO Work Item Standardization

## User Stories

USER STORY 76232\*

76232 [Feature/Project/Criteria] Title should be specific

No one selected

1 Comment

Add Tag

State

New

Area

Utopia\Development\Automages

Reason

New

Iteration

Utopia

Description

The Why

Add context for the why. What problem does it address. Can use user stories when helpful to add clarity, but not required.  
This is not the place to tell a developer the "how" of the story. If the dev team has determined a specific approach that should be described in the technical details.

Technical Details

The How

- Gotchas
- Where to look for specific information
- Details that will help on the technical side of things.
- Any configuration setup needed
- Specific error messages
- Different projects involved or impacted

Acceptance Criteria

This is the **work that needs to be done** to accomplish the why

- Use cases (including fail scenarios).
- definition of done (look at measure of success (qualitative or quantitative)),

QA

Specifics for QA testing

Indicate if there's a DB change

## Bugs

The primary difference between a bug and a user story is that the bug should include the Repro Steps

BUG 76345\*

76345 [Feature/Project/Criteria] Title should be specific

No one selected

0 Comments

Add Tag

StateNew

AreaUtopia\Development\Automages

ReasonNew

IterationUtopia

Description

The Why

For bugs this should include **Repro Steps**  
Add context for the why. What problem does it address. Can use user stories when helpful to add clarity, but not required.  
This is not the place to tell a developer the "how" of the story. If the dev team has determined a specific approach that should be described in the technical details.

Technical Details

The How

- Gotchas
- Where to look for specific information
- Details that will help on the technical side of things.
- Any configuration setup needed
- Specific error messages
- Different projects involved or impacted

Acceptance Criteria

This is the **work that needs to be done** to accomplish the why

- Use cases (including fail scenarios).
- definition of done (look at measure of success (qualitative or quantitative))

QA

Specifics for QA testing

Indicate if there's a DB change