Story and Task/Bug Templates

Story

Naming Conventions

The name (title) of each story/task/bug should be in **imperative or declarative** style.

- Imperative Style - Sentence that gives an order or command

```
Example: Implement export PDF reports for all jobs
```

- Declarative Style - Makes a statement, provides a fact, offers an explanation or conveys information

```
Example: Download file attachment does not work as expected
```

```
Example: Worksheet file upload should not create new entry
```

The name of the story should state the following criteria:

- What will be implemented.
- What will be integrated.
- What will be changed/removed/deleted/added.

The name of the story should not include:

- Description in the name (title) of the story.
- Steps to reproduce.

Business Requirements

The business requirements must be an input to the Sprint Refinement and Sprint Planning ceremonies. The business requirements can be supplemented or amended during these ceremonies. If the story does not state clearly the business requirements described in the sections below, the story should not be accepted as a ready work.

- Objective (Context) - what is the current functionality and what is expected against the current functionality (if exists)?
- Goal of the story (who are the actors) in form of: As a [persona], I want to [action] so that [benefit], (customer perspective). For example, searching for a game could include the following options:
  - Free text search - As a parent, I want to search for a specific keyword so that I can quickly navigate the game.
  - Browse by category: age group - As a parent, I want to find an age-appropriate game that my kids will easily pick up.

Task/Bug

Task/Issues: represent regular business tasks. In Jira, standard issues are where daily work is discussed and carried out by team members. For software teams, standard issues (like sub-task, bugs or stories) estimate and track the effort required to build an interaction or other end goal for your product. The task is a technical, detailed explanation for a specific part of a software feature written from the perspective of the software developer. This definition gives a better explanation of why have a well-defined task is important.

Why & How	Description

Is there a best practice for defining a regular task?	The description of the regular task should follow this structure:
	• **Current Implementation**
	  ◦ When no previous implementation exists: No Implementation.
	  ◦ Ex. Client's Job is not displayed in email templates.
	• **New Requirement**
	  ◦ Which classes are affected by the changes?
	  ◦ Which methods are affected by the changes?
	  ◦ Which table/columns are affected by the changes?
	  ◦ Is there a complete date fetching?
	  ▪ If yes, propose query agreed on Task Breakdown.
	  ◦ Describe the proposed solution for each of the above points agreed on Task Breakdown.
	  ◦ Is there dependency to other tasks? - Relation to other tasks in case of dependency with other tasks should be noted. Task dependency means an order of completing tasks that matters.

Is there a best practice for defining a bug?	The description of a Bug should be composed of:
	• **Context**.
	• **Steps to reproduce.**
	  ◦ Environment.
	  ◦ Test user.
	  ◦ Feature (flow).
	  ◦ Steps to reproduce.
	• **Expected behavior -** includes UI/UX, API and Database
	• **Actual behavior -** includes UI/UX, API and Database.
	• **Prerequisites**
	• **Screenshots or screen recordings in case of a complex bug**

When do defining tasks take place?	On task breakdown Scrum ceremonies.

        - Browse by category: type of education - As a parent, I want to find a game that will help my child improve their knowledge and skills in a specific area.
- Scope
- Prerequisites, assumptions and options
- Acceptance criteria or Acceptance Tests

Technical Specification

The technical specifications must be an outcome of the Sprint Refinement and Sprint Planning ceremonies and must be well defined input to the Task Breakdown ceremony.

Providing an answer to each of the following questions, will guide you on how to complete the technical specification.

- Is there a need of UI/UX design?
- Which components are affected?
    - UI/UX
        - What components will be used?
        - Do we have that component?
            - Ex. we need a dropdown with multi selection that can preselect items.
        - Do we have it in a shared library?
        - What component should we use?
        - Where is that component located?
    - How the current flow will be affected by the new changes?
        - Are there any dependencies for that screen?
        - Are there any shared libraries?
        - Are there any related tasks/stories?
    - Database
        - Do we need to add/remove column or tables?
            - Yes: check if other teams need to be involved or informed about any data changes.
        - Is there a database migration?
        - Does the data change somehow affects the next deployment?
        - Migration actions?
        - Fallback plan if migration failed?
        - How the migration will be tested?
    - Backend
        - Does the new change introduce a new end-point?
            - Yes: prepare a proper documentation
        - Which Services/Modules will be affected by the change?
        - Is there an existing feature/flows affected by the change?
            - Yes: specify in which files the changes will be implemented.
        - Does the change involve refactoring of existing codebase?
        - Is there a dependency to third party libraries?
        - Does the new requirement involve communication with other APIs?
        - How the change will be tested during development phase?
        - Where the new changes will be applied in the code?
        - Are there any Team Dependencies?
        - Is there a feature that is in progress in the same application/screen/DB…?
        - Are there any related tasks/stories?

Why defining tasks in a detailed way during task breakdown?	During the task breakdown ceremony, each team member should contribute with suggestions and ideas about the solutions of each task. Each solution should be well-considered. The candidate solution should be documented in the task on a way to define how and exactly where the changes should be applied. The tasks have to be people agnostic, meaning each tasks should be well-defined as each team member can be a proper candidate to work on the task.
Why well-documenting tasks matters?	In a multi-product team organization, observing the tasks between the different products is the only way for syncing everyone about the technical changes. On each task, a specific team member will be able to see what is the current implementation, what are the requirements against the current implementation, and exactly where the changes should be applied. This approach doesn't help only for keeping the people up to date for the ongoing work but also helps finding out what activities and changes have been taken for a specific feature in the history during the planning of upcoming features.
What if there is a dependency between two tasks?	More info in parent section.