Azure DevOps Workshop
By Crystal Tenn
1.24.19

IIBA® Tampa Bay Chapter

BA Reference Guide

What is a Work Item?

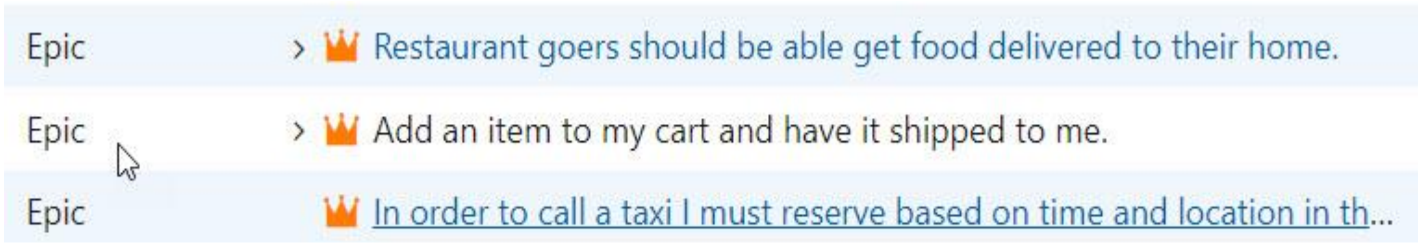
A work item in Azure DevOps is a generic term to encompass Epics, Features, User Stories, Tasks, Bugs, and Test Cases. It generally has a form with many fields, can be assigned to people, track history, tag, prioritize, and variety of other options depending on the type of work item.

What is each Work Item in more detail?

Epics

Epics are large bodies of work that cannot be delivered in a single iteration and must be broken down into several smaller tasks (they span multiple sprints). There is no standard form to represent epics. Some teams use the familiar user story formats (As A, I want, So That or In Order To, As A, I want) while other teams represent the epics with a short phrase.

Epic Examples:

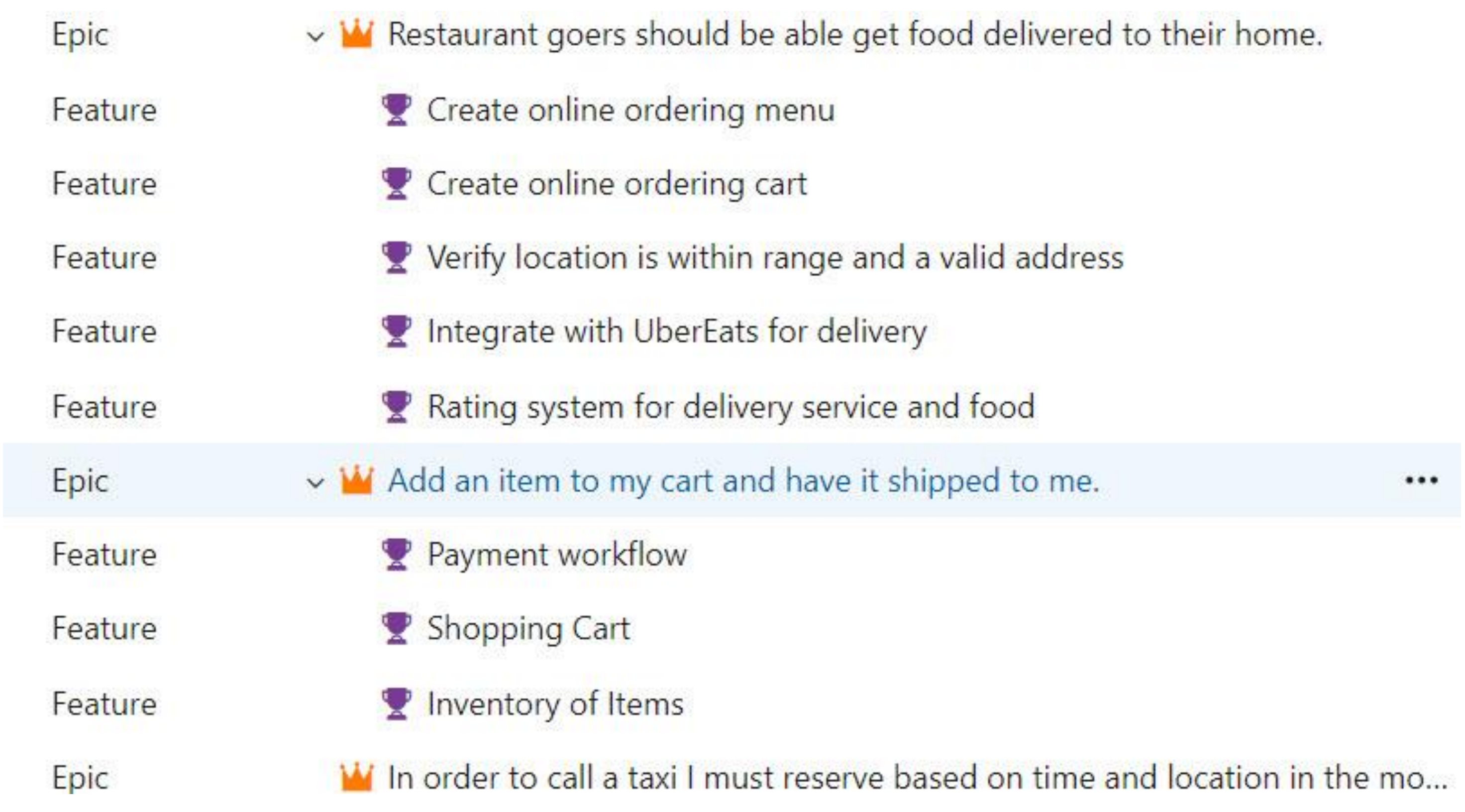


Epic	Restaurant goers should be able get food delivered to their home.
Epic	Add an item to my cart and have it shipped to me.
Epic	In order to call a taxi I must reserve based on time and location in th...

Features

Features are children of Epics. A feature is a distinguishing characteristic or capability of a software application or library (e.g., performance, portability, or functionality).

Feature Examples:



Epic	Restaurant goers should be able get food delivered to their home.
Feature	Create online ordering menu
Feature	Create online ordering cart
Feature	Verify location is within range and a valid address
Feature	Integrate with UberEats for delivery
Feature	Rating system for delivery service and food
Epic	Add an item to my cart and have it shipped to me.
Feature	Payment workflow
Feature	Shopping Cart
Feature	Inventory of Items
Epic	In order to call a taxi I must reserve based on time and location in the mo...

User stories

User stories are children of Features and grand-children of Epics. User Stories short requirements or requests written from the perspective of end user (this can be many perspectives, for example a customer, student, school administrator, business owner, etc.). The user story backlog should represent what needs to be built in the form of flexible individual parts (the user stories). The backlog is often prioritized and groomed to make sure it reflects feedback as the application is built. User stories are built from the business side and based on priority, they are put into sprints for developers to work on (the main person using your user story will be developers).

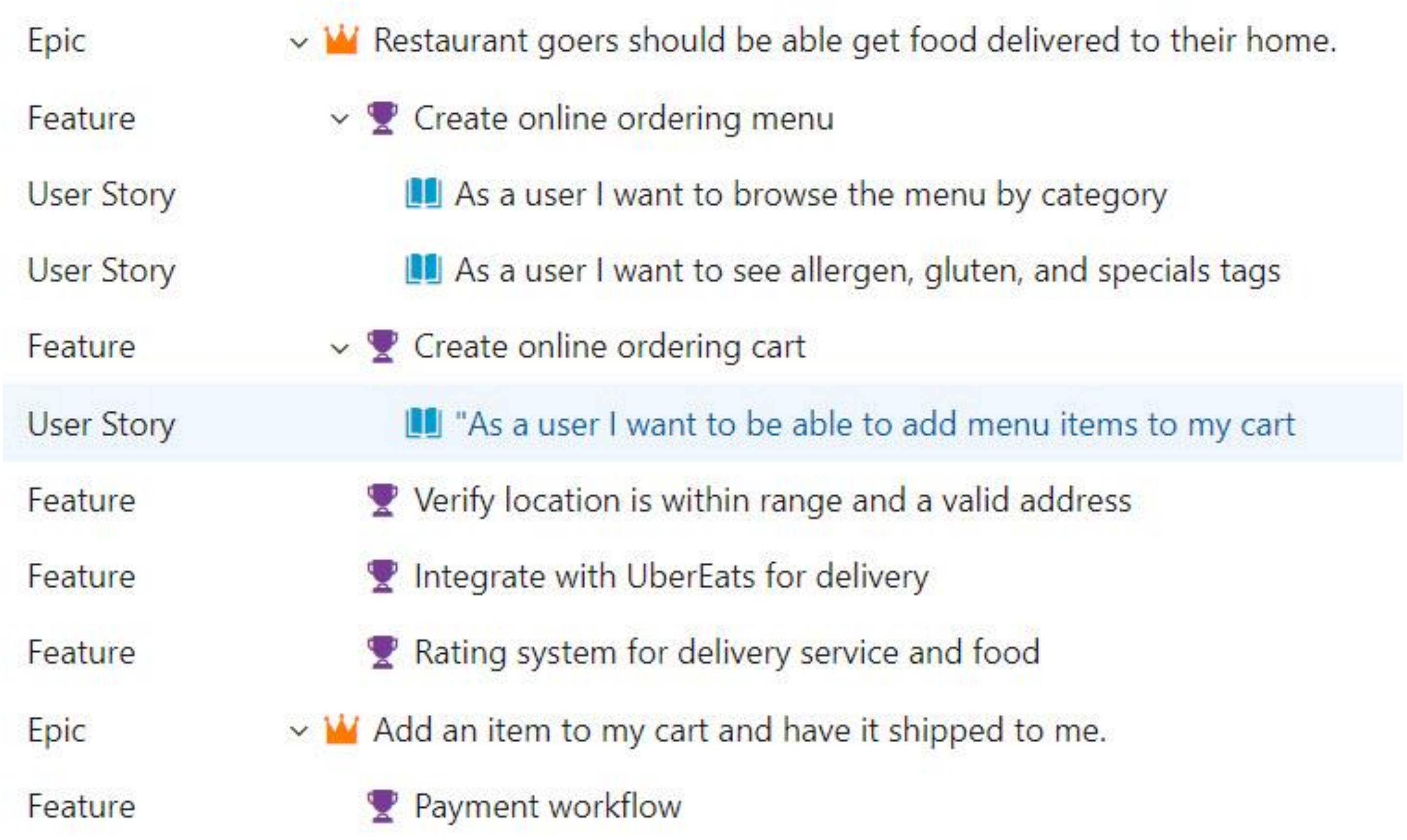
While not required, user stories are typically written in the following format: **“As a [persona, user type], I [want to], so that [goal]”**

Azure DevOps Workshop

By Crystal Tenn

1.24.19

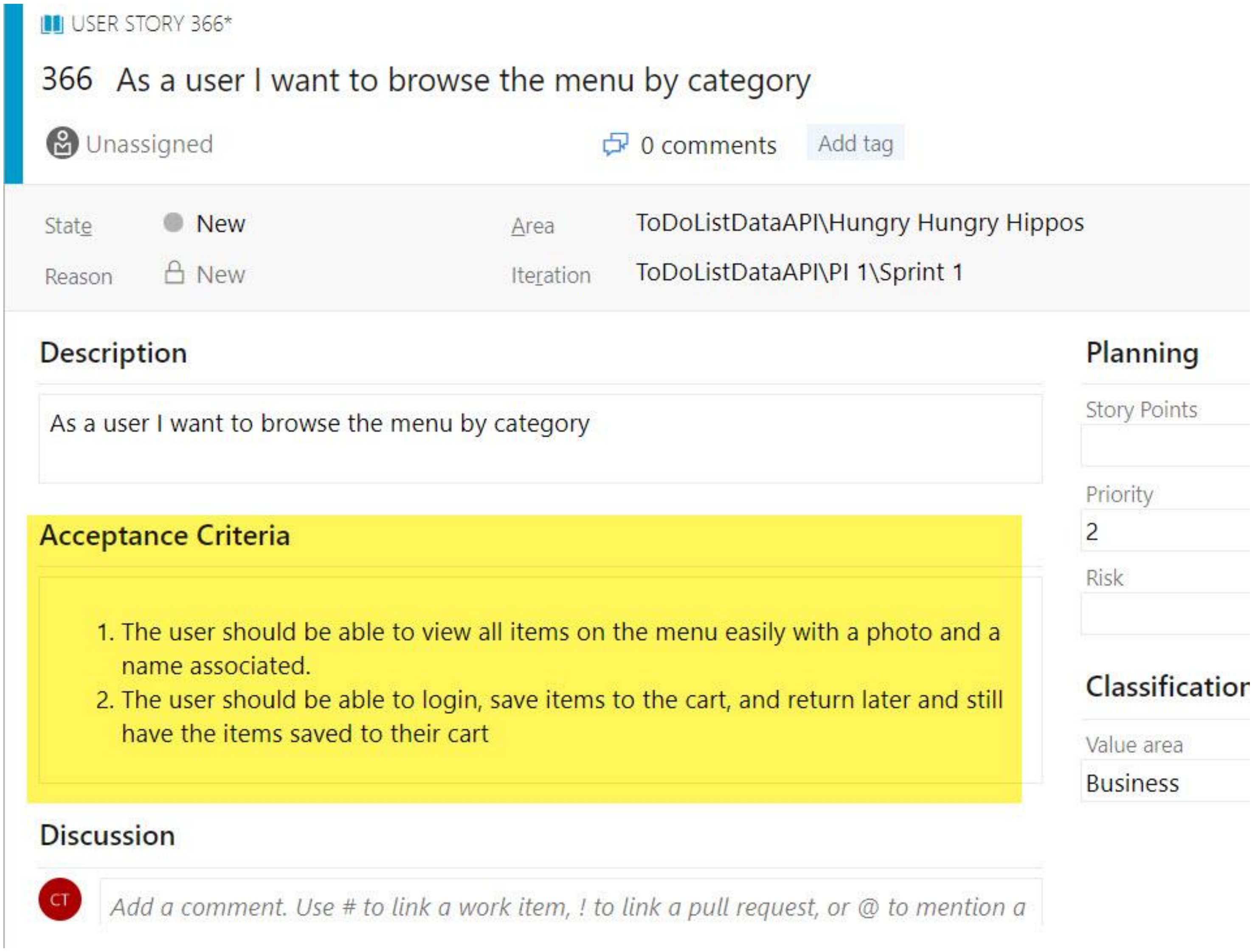
User Story Examples



Epic	Restaurant goers should be able get food delivered to their home.
Feature	Create online ordering menu
User Story	As a user I want to browse the menu by category
User Story	As a user I want to see allergen, gluten, and specials tags
Feature	Create online ordering cart
User Story	"As a user I want to be able to add menu items to my cart
Feature	Verify location is within range and a valid address
Feature	Integrate with UberEats for delivery
Feature	Rating system for delivery service and food
Epic	Add an item to my cart and have it shipped to me.
Feature	Payment workflow

It is also important to include Acceptance Criteria with the user story. The story is generally "done" when the user can complete the outlined task. A developer will create the functionality outlined in your user story, then they will check that the application meets all of you acceptance criteria. Once they feel it meets your acceptance criteria, they will send the code to QA for testing.

Examples of Acceptance criteria:



USER STORY 366*

366 As a user I want to browse the menu by category

Unassigned   0 comments   Add tag

State: New   Area: ToDoListDataAPI\Hungry Hungry Hippos
Reason: New   Iteration: ToDoListDataAPI\PI 1\Sprint 1

Description

As a user I want to browse the menu by category

Acceptance Criteria

1. The user should be able to view all items on the menu easily with a photo and a name associated.
2. The user should be able to login, save items to the cart, and return later and still have the items saved to their cart

Discussion

Add a comment. Use # to link a work item, ! to link a pull request, or @ to mention a

Planning
Story Points:
Priority: 2
Risk:

Classificatior
Value area: Business

Bugs

Bugs are used to track code defects and are created by QA while the test code.

Tasks

Tasks are the breakdown of work to be completed (for the given user story or bug). At the beginning of a spring during sprint planning, developers will create specific tasks to complete your user stories that are usually technical in nature and QA will create tasks for testing. The combined hours of dev and QA in tasks will determine the actual number of story points the User Story will take them to complete. The developers and QA will use these tasks for the sprint burn down hours, presentation in daily stand up meetings, and to track their work.

Note - A feature describes a specific solution that may address multiple user stories whereas epics and user stories should be solution neutral

Who is responsible for what?

- *Product owners (PO)* - The product owner is typically a project's key stakeholder. Responsible for prioritization and final decisions on questions around work items. The Product Owner has a significant role in quality control and is the only team member empowered to accept stories as done. Click here for more details.
- *Business analysts (BA)* - Are responsible for refining the backlog (Epics, Features, and User Stories) and making sure that everything is up to date based on meetings and feedback. Also, they are responsible for breaking new requirements (processes, pro-types, etc) down into well written user stories. User stories should have a parent/grandparent (Feature and Epic) that they belong to, description in a “As a --, I want --” format, in depth Acceptance Criteria for how it should function, and a list of dependencies/relations to other user stories. User Stories should be small enough that many of them can be completed during a sprint. Analysts may need to answer questions from developers about more details on a User Story and be able get answers from the business side.
- *Developers* - Write the code based on user stories from the business analysts. They create tasks for their work. They work on bugs from QA.
- *Build and Release engineers* - They create tasks that will automatically build code from the developers and deploy code to environments for dev/qa/prod.

- *Quality Assurance (QA)* - They will test the QA environment code, create bugs for developers to fix, create test cases and manage their test scenarios. They ensure with regression suites that the entire site functionality works with new code, and that new features that are built are free of bugs and work correctly.
- *Test engineers / QA engineers / Test automation* - They create automated tests by writing code. The automated tests can be easily run by software to complete simple tests to reduce manual testing done by QA so QA has more time for exploratory testing.

What is Agile vs. Scrum vs. Kanban vs. Waterfall?

- **Agile** - iterative development methodology that values human communication and feedback, adapting to changes, and producing working results. tl;dr: build a little bit at a time, get feedback, adapt the backlog as needed. More generic method of doing 2-4 weeks sprints that is adaptable and not requiring specific terminology. You ask for a car, we will build the wheels and see how you like it. Then build the engine next sprint, etc.. easy to modify wheels if feedback is given in another sprint.
  - **Scrum** is a framework for implementing Agile. Scrum has a very specific set of terminology and method of doing things in 2-4 week sprints. Teams are self sufficient.
  - **Kanban** is a framework for implementing Agile. Kanban workflow is a big to-do list of items and work gets completed when there is capacity. Teams may not have all members needed and borrow skills from members not on the team. No time frame / time limit, likely an infinite to-do list.
- **Waterfall** (older method of building software) - linear and sequential method of building software. tl;dr: plan the whole thing, build the whole thing in one go, test it all in one go, and hand it off. No iterative feedback along the way, you asked for a car, here is your entire car, too bad if you don't like pieces of it.

Most places, realistically, are hybrids of Agile/Scrum or hybrids of Agile/Scrum/Waterfall/Kanban and the world is rarely cut and dry.