



What is Testing?



Purpose of Testing

A primary **purpose of testing** is to detect software failures so that defects may be discovered and corrected.

Another purpose of testing is to ensure the quality of the product.



Types of testing

1. Functional Testing
2. Non-functional testing



1. Functional Testing

The testing of the functionality of the software.



2. Non-Functional Testing

It refers to performance and security testing



2. Non-functional testing

Non-functional testing answers the following questions:

How well?

How fast?

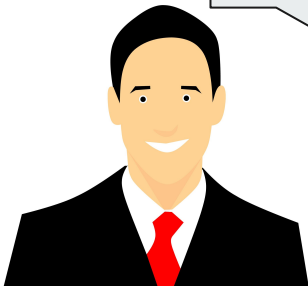
How stable?

How usable?

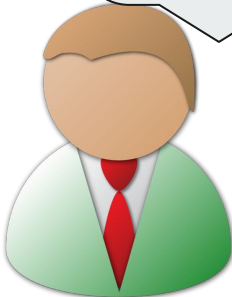
How secure?



Example



Does this system work properly
for you?



Yes... But it's really slow and
unintuitive.

We've been hacked yesterday.



What is a requirement?

The software requirements are description of features and functionalities of the target system. Requirements convey the expectations of users from the software product.



What is a test case?

A **test case** is a set of actions performed on a system to determine if it satisfies software requirements and functions correctly



How a test case should look?

A test case should contain:

1. Summary (Title)
2. Description (Optional)
3. Preconditions (Optional)
4. Test steps (Action and Expected result)
5. Attachment - optional
6. Priority - Low/Medium/High



Example of a test case

Summary: Pen checking	
Description: This test checks that the pen is working accordingly	
Preconditions: pen, ink, paper	
Action	Expected results
1. Check that pen form is according to specification.	1. The pen has the form as per specification.
2. Check the color of the ink.	2. The ink is of blue color
3. Take a paper sheet and check if the pen can write.	3. The pen is writing without interruptions and without spilling ink.



What is a User Story?

User stories are intentionally short and simple, focused on one particular feature written from the perspective of the user of a system rather than the system's developers.

They're different from traditional software system requirements in that they bring end users to the center of the conversation, adopting a more informal language to provide context for the developers of the system.

A user story thus constitutes an end goal, a desired functionality from the perspective of the user, not necessarily articulated in technical terms and agnostic to the engineering efforts that it entails



User Stories Guidelines

A well formulated user story is:

- Simple and precise (can be programmed, tested, and integrated within a single sprint)
- Focused on a specific user goal

User stories are often expressed as *persona + need + purpose*. It is then typically written in a simple sentence that follows the structure:

"As a [persona], I [want to ...], so that [...]"

Example:


"As a user,
I want to be able to securely log in to the system
so that my information can only be accessed by me."



User story - acceptance criteria

The acceptance criteria determine the specific conditions that the software product must satisfy to be accepted by and meet the expectations of the user. It also forms the basis for the acceptance testing stage.

- Each acceptance criterion should be independently testable
- Each acceptance criterion test should have a clear pass/fail result
- Acceptance criteria should be focused on the end result (functionality), not the mechanism through which it is achieved



An acceptance criterion is often expressed as a sentence following the structure:

"Given [precondition], when I [do some action] then I expect [result]".

User Story	Acceptance Criteria
As a new user, I want to register by creating a username and password so that the system can remember me and my data.	Given that I am a new user, when I go to the sign up page and enter an username and password and click on sign up, then I am successfully registered and able to log in with my chosen credentials.
As a registered user, I want to log in with my username and password so that the system can authenticate me and I can trust it.	<ol style="list-style-type: none">1. Given that I am a registered user and logged out, if I go to the log in page and enter my username and password and click on Log in, then the data associated to my user should be accessible.2. Given that I am a registered user and logged out, if I go to the log in page and enter my username but an incorrect password and click on Log in, then log in fails with an error message that specifies that the username or password was wrong.



Adding Non - functional Requirements to US

A good practice in defining acceptance criteria is to also include non-functional requirements, i.e., requirements related to the system's qualities and attributes that are not necessarily directly related to the functionality, but are crucial in meeting the user's expectations in regards to the system's behavior.

For example, building on the list of acceptance criteria for the user story:

"As a registered user, I want to log in with my username and password so that the system can authenticate me and I can trust it."

We can add as a criterion:

"Given that I am a registered user and logged out, if I go to the login page and enter my username and password and click on Log in, then my user login session is loaded in less than eight seconds."



Creating acceptance criteria

1. Writing criteria is the first step to working on the story
2. Agree on criteria before coding begins (testers + devs)
3. Use criteria as the basis for test cases



Splitting stories - large stories

Sometimes stories get to big for a person to handle it.

Large stories disadvantages:

- tough to estimate
- Work is not distributed well
- Bigger misses



Small Stories

Advantages of Small Stories

- more flexibility in release planning
- easier negotiation with product owner
- can save development work - very small and insignificant stories usually are never implemented



When we know that a story is too big?

- When the team become uncomfortable when giving estimates
- Teams will have less confidence as storis grow
- Where does the team become less accurate?



Epics

EPIC = a big unit of work that has one common objective.

An epic is a feature, customer request or business requirement.

Epic example: Manage student enrollment

Manage Enrollment: Example with Context



Every quarter, the Registrar, prepares for the beginning of the next academic quarter.



Six weeks before the beginning of the next quarter, classes are opened to enrollment.



During this time, students enroll, the cashier manages payments, the enrollment supervisor audits enrollment activity, and the Registrar tracks the performance of cashier, student services and enrollment operations through a variety of reports and tracking mechanisms.

“Manage Enrollment” with Its User Stories



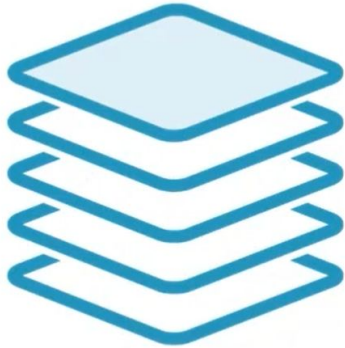
As a <student> I can <search for, find, & select classes> so that <I can enroll>

As a <student> I can <view my currently enrolled classes> so that <I can drop a class>

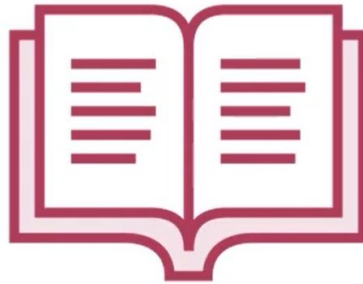
As an <administrator> I can <open a class> so that <students can search & sign up>

As <department staff> I can <view the class roster> so that <I can review class enrollment attributes <discounts, dates, etc.>

Coding User Stories with Tasks



Epic or Feature



User Story

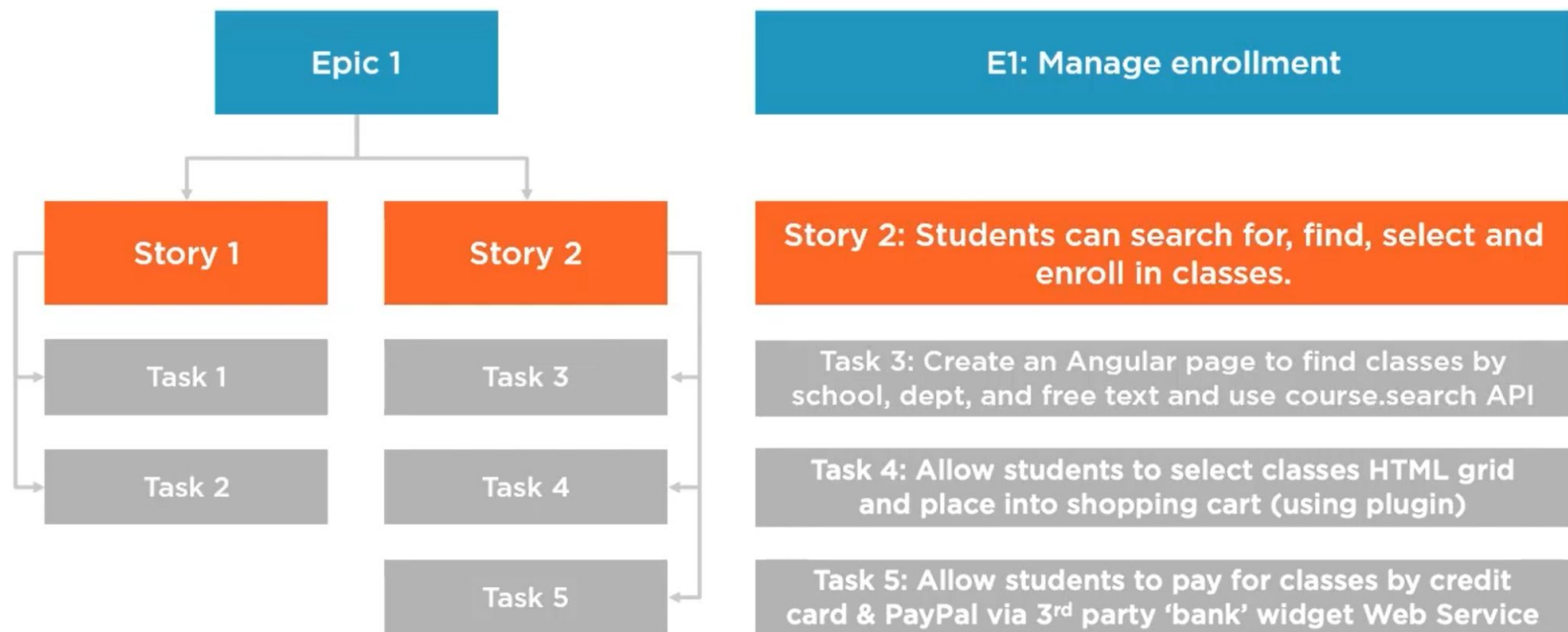


Task

Written in
business/user terms

Written in developer
terms

Example: Epics, Stories, and Tasks





Constructive Bug Report

At least answers What? When? How?

- Software version tested
- Link to relevant requirements
- Short description
- Concrete steps taken to reproduce the failure
- Expected results:
- Actual Results:
- Screenshots, logs or other applicable useful information
- Reproduction rate



Constructive Bug Report

Read your own report

No criticism of any kind

Only information pertinent to understanding, reproducing and fixing the issue