

Agile Testing

Agile testing is a software testing practice that follows the Agile software development methodology. In Agile development, projects tend to evolve during each sprint among collaborators and shareholders. Agile testing focuses on ensuring quality throughout the Agile software development process.

Key idea is that testing isn't a separate phase — it's part of the development process. Agile Testing is all about speed, quality, and flexibility. It focuses on delivering small, working features fast, testing constantly, and working together as a team.

Key Characteristics of Agile Testing

1. **Continuous Integration:** In continuous integration, developers integrate their code changes into a shared mainline several times a day. Testing happens alongside development, not after it.
2. **Continuous Delivery:** In continuous delivery, every change that passes all tests is automatically released into production.
3. **Collaborative:** The goal is to deliver value to the customer with every iteration.
4. **Adaptive:** Requirements can change; testing adapts with them.

Types of Agile Testing

Type	Description	Example
Acceptance Testing	Validates that the feature meets requirements.	"Track map updates every 10 seconds"
Exploratory Testing	Tester uses the feature without scripts to find unexpected issues.	"What happens if I switch network mid-delivery?"
Automated Regression Testing	Runs previously written tests to ensure nothing broke.	Re-checking login, cart, or payment flow after a delivery feature is added
Unit Testing	Developers test small pieces of code.	Testing the function that calculates ETA
Integration Testing	Tests combined parts of the system.	Does the map correctly get GPS data from the server?

User Story

User stories are short, simple descriptions of a feature or functionality written from the perspective of the **end user**. They are a key part of Agile development and help teams understand what the user wants and why.

A user story typically follows this template:

As a [type of user],
I want [some goal],
so that [some reason].

Example 1: Food Delivery App

As a customer,
I want to see my order status in real-time,
so that I know when to expect my food.

Example 2: E-commerce Website

As a registered user,
I want to save items to a wishlist,
so that I can buy them later.

A good user story should be:

- I** - Independent - Self-Contained
- N** - Negotiable — Can be discussed and changed
- V** - Valuable — Delivers value to users
- E** - Estimable — Can be sized or estimated
- S** - Small — Can be done in a sprint
- T** - Testable — Can be verified with acceptance tests

Acceptance Criteria

These are conditions that must be met for the story to be considered “done.”

For example:

User Story:

As a customer, I want to reset my password, so that I can regain access to my account.

Acceptance Criteria:

- User can request a password reset via email.
- A secure token is sent to their email.
- The link expires in 30 minutes.
- Password must meet strength requirements.

How is agile testing done?

Development teams can conduct Agile testing in several ways. The most common way is for Agile teams to integrate their code changes into a shared mainline several times a day. This allows for increased collaboration to point out project flaws and improve them quickly.

Another way to perform agile testing is through test-driven development (TDD). In TDD, developers write unit tests before they write the code for a new feature. These unit tests define the requirements for the new feature.

Once the developers write the code, they'll perform unit tests to make sure that everything works as expected.

Example: Food Delivery App - Track Delivery in Real-Time

User Story Created

- “As a user, I want to track my delivery in real-time so I can estimate when it arrives.”

Acceptance Criteria Defined

- Map shows driver's location
- Updates every 10 seconds
- Shows estimated time of arrival

Testers Join Sprint Planning

- They help write **Acceptance Tests** (criteria for success).

Test-Driven Development (TDD) or Behavior-Driven Development (BDD)

Developers and testers may write tests before or during coding.


Continuous Testing

- As the feature is being coded, automated and manual tests are run.
- Bugs are fixed *immediately*, not after the sprint.

Demo and Feedback

- At sprint end, team demos the feature.
- Stakeholders give feedback that may lead to new user stories.

Imagine a **2-week sprint** to build a “Rate Delivery Experience” feature.

Day	Activity
Day 1	Product owner and tester write user stories & acceptance criteria
Day 2-4	Developer writes code; tester writes automation scripts
Day 5	Automated tests are run; bug found and fixed on the same day
Day 6-8	More testing, including edge cases and mobile versions
Day 9	Exploratory testing reveals a UI bug
Day 10	Sprint review with a working, tested rating feature 

Challenges during agile testing

Agile teams move quickly, and they are constantly making changes. This means that testers need to be adaptable, and they need to be able to change their approach on the fly.

Agile teams typically have a lot of stakeholders involved. Testers need to be able to communicate effectively with all team members, as well as with the customer.

Agile development environments often have tight deadlines. Testers need to be able to work well under pressure and they need to be able to deliver quality results quickly