

# The Ultimate QA Testing Handbook

---

Everything you need to know, to test how you want.



[www.globalapptesting.com](http://www.globalapptesting.com)



[info@globalapptesting.com](mailto:info@globalapptesting.com)



UK +44 (0) 330 808 0106

US +1-800-461-2670



[linkedin.com/company/global-app-testing](https://linkedin.com/company/global-app-testing)



[twitter.com/qaops](https://twitter.com/qaops)



[facebook.com/globalapptesting](https://facebook.com/globalapptesting)

# Contents

---

Introduction.....	03
Chapter 1 - QA Testing.....	04
Chapter 2 - Regression Testing.....	10
Chapter 3 - Crowdfunding.....	18
Chapter 4 - Functional Testing.....	26
Chapter 5 - Localization Testing.....	33
Chapter 6 - Exploratory Testing.....	42
Chapter 7 - Manual Testing.....	47
Chapter 8 - Agile Testing.....	55
Chapter 9 - Usability Testing.....	66
Final Thoughts.....	68



# Introduction

Let's start off with the basics.

Unless you know exactly how, when, why and for whom your software testing should be conducted, the results you receive are bound to be lackluster in their impact.

So whether you're after increased customer acquisition, decreased user churn or better app store ratings, building the foundation of an effective testing strategy is paramount.

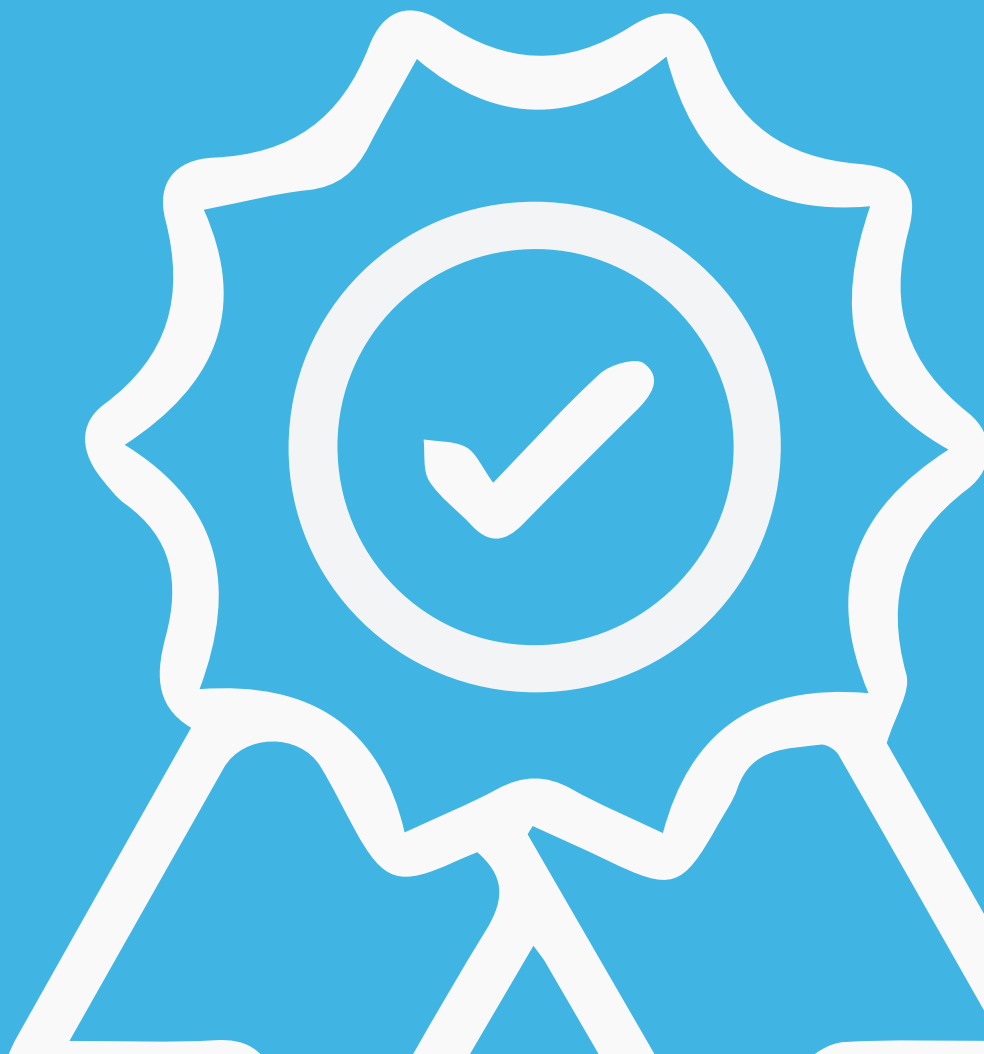
Admittedly, it's easy for anyone to be confused by the sheer number of testing types, how they overlap and what benefits they actually provide. At the end of the day, it's all about proving value; without it, why even bother?

And so our Ultimate QA Testing Handbook enters the ring, battling this confusion and offering a collection of need-to-know insights and best practices for QA teams.

Now let's jump in.

---

# Chapter 1 - QA Testing



# What is Quality Assurance Testing?

Quality Assurance (QA) testing is the process of ensuring that your product is of the highest possible quality for your customers. QA describes simply the techniques used to prevent issues with your software product or service and to ensure a great user experience for your customers. With that said, what are the QA best practices you need in your repertoire?

## Writing good test cases

Should developers write tests? On the one hand, the Agile approach is about ownership; so involving developers in the test case writing process and making QA one of their responsibilities is understandable.

However, on the other hand, developers who create tests might become biased and write code that will pass the test without meeting other quality standards or unconsciously create a test with limited coverage. With this approach, developers often can't see the wood for the trees.

It is for this reason that teams create a test plan but outsource the process to QA engineers skilled in testing and more suited to detect bugs that often crawl through to production.

**With this in mind, here's our four top tips on how to write good test cases:**

1. Write each test case with a narrow focus, but remember to retain cohesion across your entire test case suite. Your test case suite should have a scope that is adapted to the scale of your project.
2. Customize and execute test cases in an environment that is different from the one used for development. Each test should be based on clear expectations and result in a measurable outcome.
3. Break down each test case into a series of concise steps. Taking these steps will tell you whether or not a feature works. You can think of writing a test case as a series of actions associated with a question. When an action is taken, the automated test or human testers should answer a simple question to measure the success of the action.
4. The instructions written for each test case should give testers a clear understanding of what they are expected to do. You can save time and get better results by providing test cases, instructions and tutorials that aren't liable to misinterpretation. There are testing tools available to make this even easier.

## Continuous integration and continuous delivery

Continuous integration (CI) and continuous delivery (CD) are strategies used in software development that complement the Agile methodology. You can incorporate a continuous testing strategy into CI and CD.

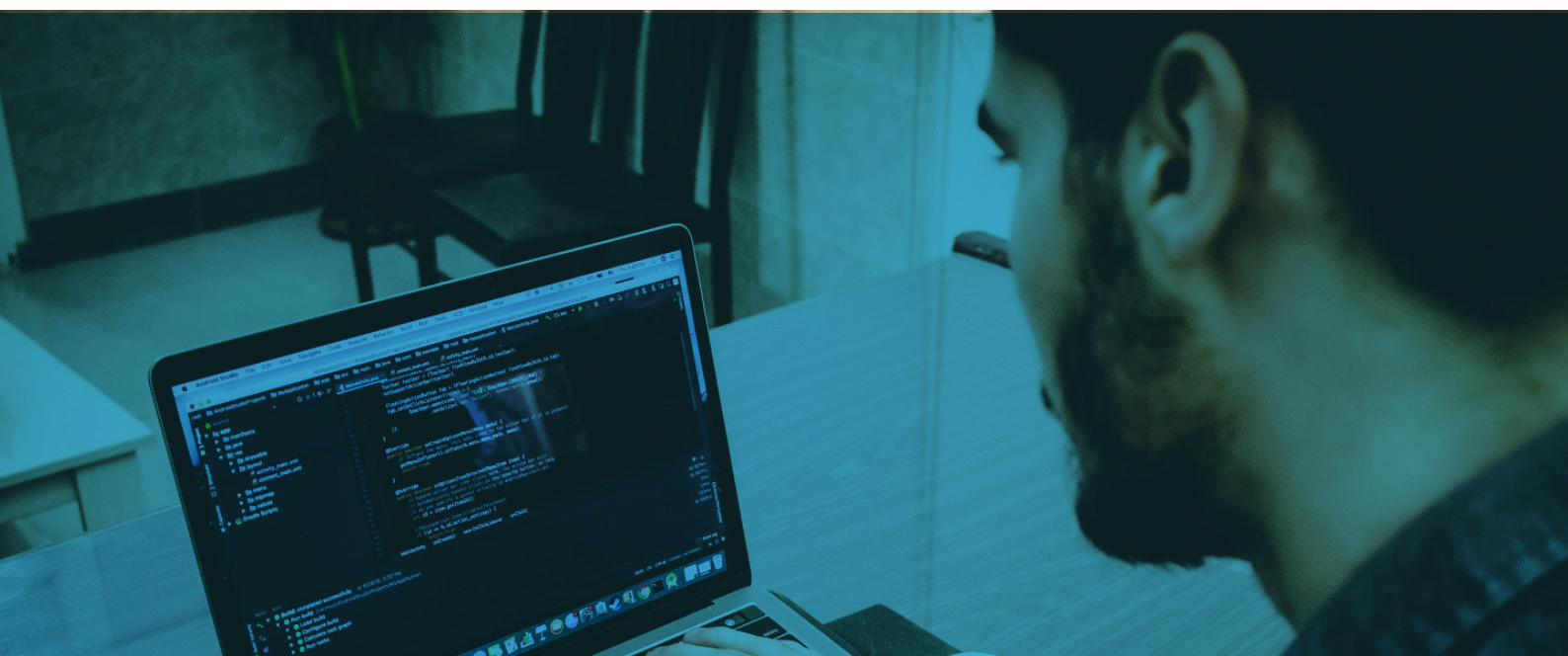
Without CI and CD, developers split up their work and assemble different segments of the code late in the development cycle. This can result in a lack of cohesion, compatibility and issues with how the different segments of the code interact.

With continuous integration, the code is kept in a central repository. Developers work on making small changes to the code and upload small sections of it to the central repository regularly. You can incorporate quality management into this methodology by including a series of tests performed every time the code is updated. The new segments need to be tested, but you should also conduct regression testing to see how changes affect the product's main features.

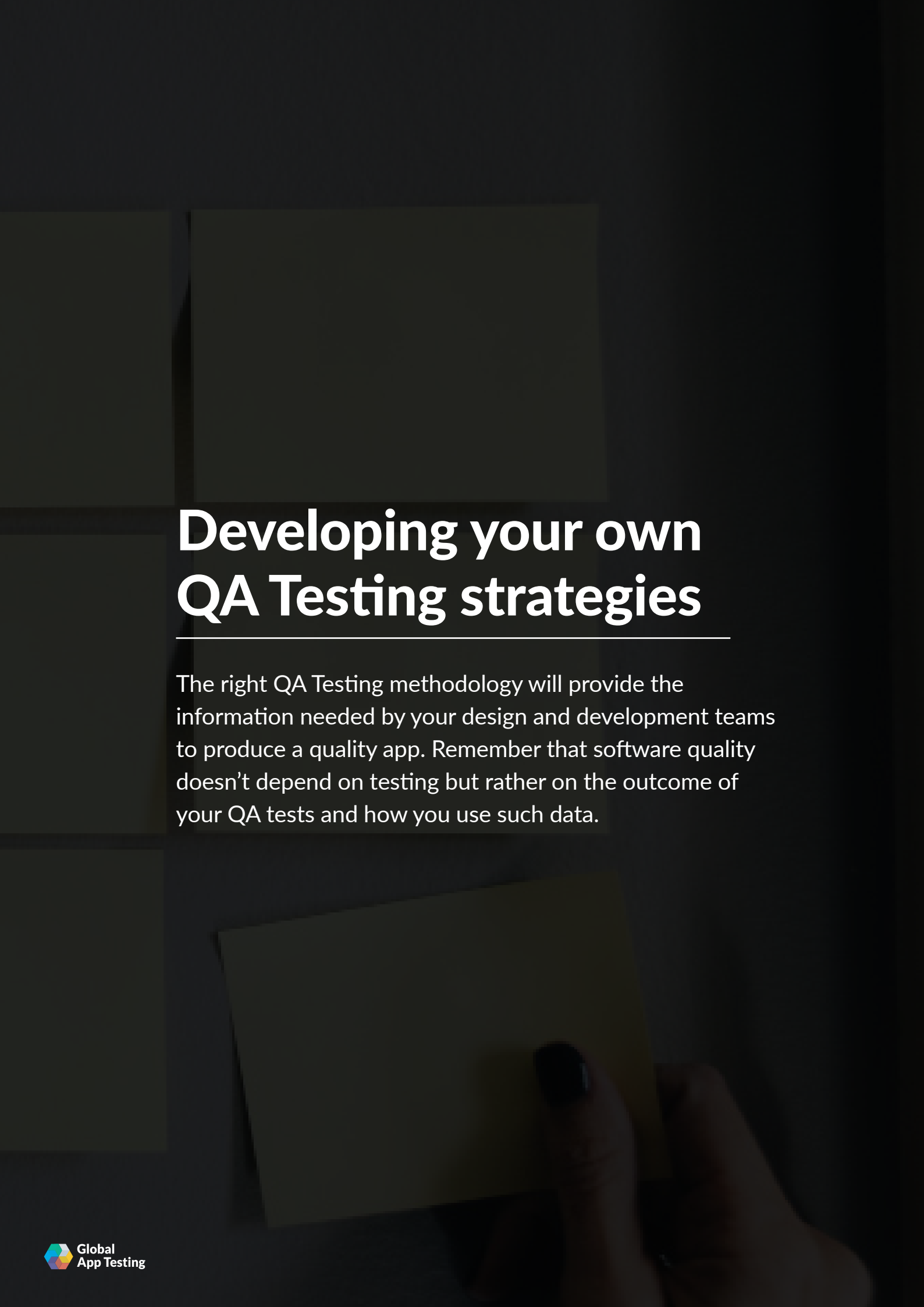
[Continuous delivery](#) allows you to release new iterations of your product on a regular basis. This is a quick and efficient approach to addressing bugs and issues that affect the user experience.

The key is to incorporate user feedback into your CI and CD processes so that issues can be addressed quickly and a new and improved version of your product can be released.

Again, you will have to incorporate testing in your process, for instance, by having crowd testers perform usability tests before a new major version of your product is made available to users.







# Developing your own QA Testing strategies

---

The right QA Testing methodology will provide the information needed by your design and development teams to produce a quality app. Remember that software quality doesn't depend on testing but rather on the outcome of your QA tests and how you use such data.

## Adapt your approach to QA testing to the product you are developing.

1. **Focus.** Establish clear objectives for each test, and test only one component at a time, for instance a specific user interface or security feature.
2. **Familiarize.** Understand the breadth of testing types on offer, how to use each and the insights they provide.
3. **Repeat.** Test main features more than once with [regression tests](#). New additions to the code repository can interfere with features that previously passed tests.
4. **Report.** Determine how bugs will be reported and what kind of data is needed. Will you use an open-source bug tracking tool or build one that's specifically suited to your workflow?
5. **Analyze.** Establish QA metrics early to ensure they're consistently tracked. Keep records of every test conducted, and use this data to determine where bugs are likely to occur. This data will help you to develop new tests that address problem areas.
6. **Localize.** Test in different environments to cover a wide range of scenarios, devices, operating systems and user personas.
7. **Unit and Integration Testing.** While Unit Testing isolates each component of your app, integration tests assess how well each subsystem works. Run unit tests in parallel to save time, but only move onto integration tests once you've ensured individual components work as they should.
8. **Delight customers.** Humans testing end-to-end scenarios through Functional Testing provide user experience insights that other testing types do not. Wait until issues detected from Unit and Integration Testing are resolved before deploying Functional Testing.



## Achieving high quality and speed

Whether you're building a web application, downloadable software or an API, delivering quality and speed are key objectives for a successful release, yet often in battle with each other. To ensure both are achieved, review your QA Testing process regularly as you move through the different cycles of your project.

The code used for automated tests should also be tested, and written tests sent to human testers should be carefully reviewed. Make it easy for everyone involved with your project to report bugs and share feedback.

Maintaining a list of clear quality objectives is crucial for achieving high quality and speed. Align your quality objectives with users' expectations, and use these objectives when writing test cases. You should also take into account the ISO 9000 quality management standard and your stakeholders' needs.



Working with a clear set of quality objectives will help developers, testers and designers better understand what's expected of them and foster an environment where everyone owns quality.

Also remember to focus on efficiency, as this directly impacts your team's day-to-day. Using an off-the-shelf bug tracker like Jira is the best way to keep track of quality issues and ensure they're addressed in a satisfactory and timely manner.

Lastly, your QA strategy should be unique to the product you are developing and its lifecycle. Keep it aligned with the scope of the project, your definition of quality and end users' expectations.

---

# Chapter 2 - Regression Testing



# What is Regression Testing?

What tools are available? How is it performed?

Regression Testing forms an essential component of any strong testing strategy, so this chapter is designed to provide a fundamental understanding of it, regardless of your company size or software offering.

## Automated Regression Testing

When thinking about Regression Testing, it's understandable why you may want to tack "automated" to the beginning of it. Automation is after all the buzzword that keeps on buzzing. Furthermore, Regression Testing is repetitive and manual, so why not automate it?

Regression tests are the perfect candidate to be automated - in certain circumstances.

Automated approaches to Regression Testing make sense when developers are able to write and maintain the scenarios required to complete the testing appropriately. This means that developers are responsible for maintaining proper test scripts for their regression test cases.

In other words, Automated Regression Testing is only as good as your regression test scripts.

## Regression test scripts

Regardless of how you perform Regression Testing (manually or automated), you're going to need a robust set of regression test scripts.

Regression test scripts are the foundation of your regression.

Unfortunately, regression test scripts are quite likely the most time-consuming part of any QA professional's job! Regression test selection can be a tricky process, as due to their critical nature, you must write scripts that cover every possible scenario you can think of. You need to spot any side effects, any impact on dependencies and any problems the new changes might cause.

But what happens if you fail to cover every possible scenario?

Regression test scripts only test what you tell them to - automated or not.

The person who writes the test script must ensure that it considers all variations and flows, which demands a massive amount of work and upkeep.

Any time there is a change in your product or software, checking regression test scripts for accuracy is required. Otherwise, determining whether a failure is because of a bug in your code change or a poorly written test script will be impossible.

## Automated Regression Testing tools for web applications

One of the more interesting developments in recent years has been the proliferation of testing tools designed for web applications to perform Automated Regression Testing.

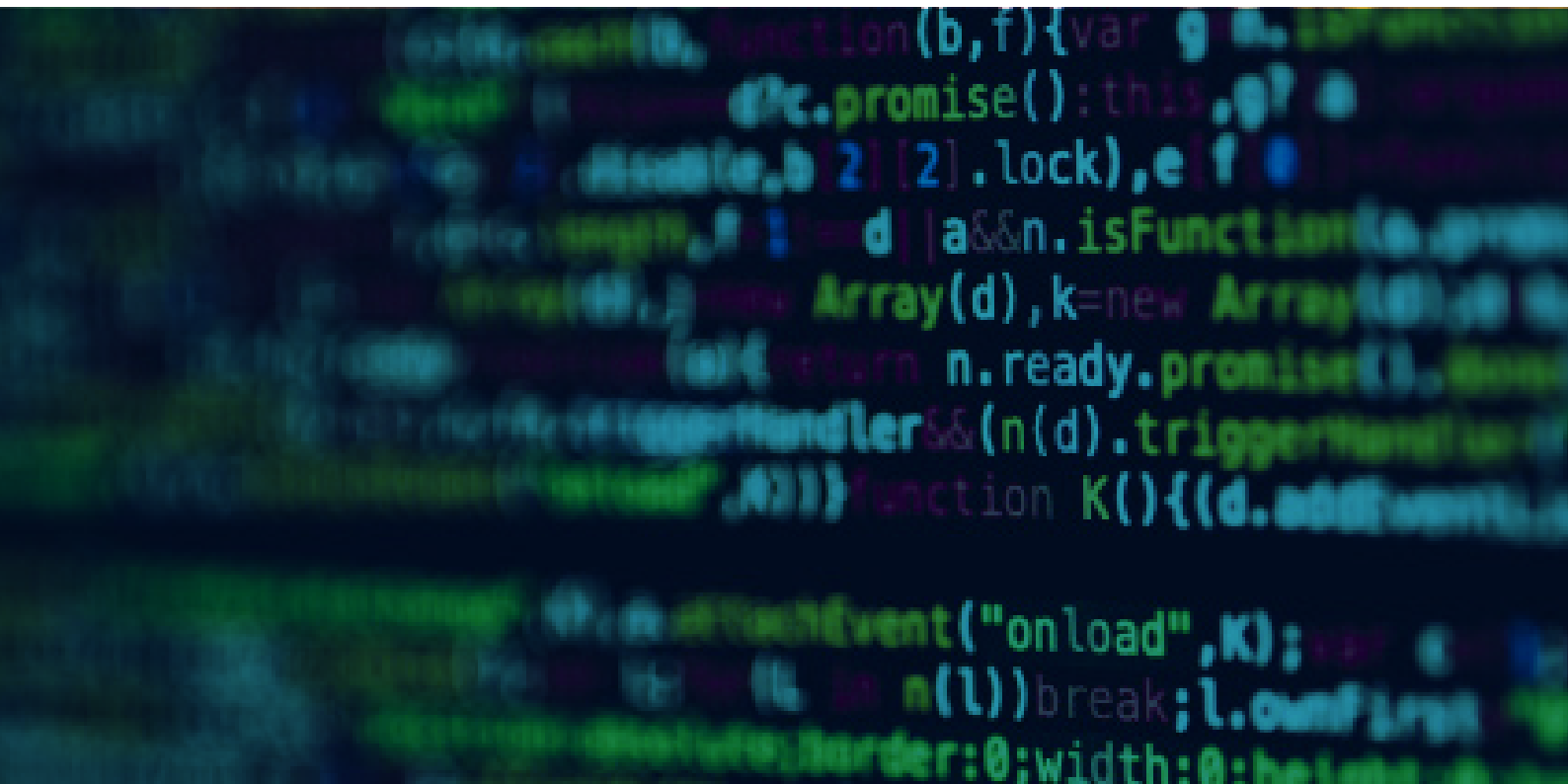
Automated Regression Testing tools for web applications are designed to alleviate as much effort as possible for humans, redistributing such efforts to machines. Sounds great, right?

In theory, Automated Regression Testing tools are ideal, as they free up your team to focus on more important functions. However, the answer is much more complex and nuanced. As discussed above, any type of Regression Testing is only as good as the test scripts used to execute it.

An automated regression test suite for web applications can save significant time, energy and resources as long as you're prepared to put in the effort to maintain it properly. And this effort can be much more of an undertaking than originally anticipated.

While automation has its benefits, the time and energy upkeep makes the case for incorporating Manual Testing into your testing process.

In the near-term future, however, newer tools will solve these issues by building "smart" test scripts requiring much less handholding!



# The age of Agile is upon us!



“Agile Testing leaves very little time for documentation. It relies on quick and innovative test case design rather than elaborate test case documents with detailed steps or results.”

- Nishi Grover

This probably contradicts what we've been talking about so far, with Regression Testing requiring specificity and detail to be effective.

Regression Testing in Agile simply requires that the agility of software development cycles and sprints are taken into account. Modern engineering teams will already be Agile and most likely pursuing some level of DevOps too.

This means that the right systems, processes and procedures must first be in place before Regression Testing in Agile can begin. Prioritization is key - you're still required to build out test scripts and decide on the level of coverage you want to achieve.

This is where a proper Regression Testing strategy for Agile becomes vitally important...



## Developing an Agile Regression Testing strategy

If you're developing with the Agile methodology (and if you aren't, why not?), then developing a robust software testing strategy that encompasses Regression Testing should be at the top of the list.

The level of sophistication of your company and team will impact your Agile Regression Testing strategy over the long term. There are three basic phases of all products:

- Validation: product-market fit
- Predictability: creating a stable infrastructure for scale
- Scaling: minimizing negative impact to unlock growth



“When things begin to break, it's not necessarily a sign that your QA has begun to fail. It could just indicate that your needs have changed, and so you need to change your strategy to succeed.”

- Excerpt From: *Leading Quality*, by Ronald Cummings-John and Owais Peer

Adjust your Regression Testing strategy according to the level of maturity of your product.

Don't simply assume that you should start automating every part of your Regression Testing because you'll waste valuable time that could have been dedicated to the product development itself!

## Regression Testing in a sprint

One of the many benefits of Regression Testing in a sprint is that it can happen as soon as the latest iterations are put into the deployment flow. Depending on your setup, this means that Regression Testing in a sprint can act as your CI/CD tool.

But why does it matter if it's performed in a sprint, anyway?

Well, according to Dan James and Bryan Aho, it could save you a lot of money - upwards of \$5,000.

Ultimately the ability to save time and cash is a major boost for any development team!

During sprints, testing often receives the least amount of time dedicated to it, so maximizing efforts should be a top priority.

## Regression Testing vs. Unit Testing

New to the world of QA? A common question we're asked is to explain the difference between Regression Testing and Unit Testing. So here we go...

Think of Unit Testing as the individual testing of specific components or enhancements in your software or product. It's the smallest amount of testing that can be carried out.

Compare the unit test then to a regression test, and you can see that regression is the sum of all its parts.

Unit tests must always be executed by the developer and should be built into the code itself. Regression tests can be created by anyone and run externally from the codebase.

## Regression Testing vs. Functional Testing

A big debate in the world of software development QA is the role of Regression Testing vs. Functional Testing.

Can you get away with simply running regression tests alone? In our humble opinion, no.

While there are plenty of companies that rely on Regression Testing alone, only Functional Testing can provide an entirely holistic view of how your app actually works for the user.

Sure, Regression Testing will tell you if a certain feature no longer works (like your checkout flow), but it won't tell you if the new feature you've built works as intended.

This is where your Regression Testing strategy comes to the fore.

If you don't know what your entire testing strategy is in the first place, it's going to be impossible to know what kind of tests you need to employ at each stage of the software development life cycle.

You need to use different types of testing at different stages, so it's critical not to write off either Regression or Functional Testing at any point. They each have different, important parts to play.



# Real-world applications of Regression Testing

By now, the role of Regression Testing and its benefits for your company should be clear.

But, just in case it isn't, let's examine some examples where more Regression (or more complete regression) Testing would have revealed holes and avoided calamity.



In January 2012, UK retailer Marks and Spencer accidentally priced TVs at £199 instead of the £1099 for which they should have been listed. Regression test scripts should have been written to check the price against the database, resulting in simple bug fixes rather than potential losses.



In 2016, Nest Thermostat received an upgrade that paralyzed devices due to a bug in the firmware. This left customers out in the cold during one of the coldest weekends of the year! Nest should have regression tested the devices before launching the new update.



In December 2018, O2 released a software update internally that caused 30 million users to lose access to data. Surely a couple of regression tests could have confirmed the new software would have caused this?

These are just a handful of failures that could have been avoided had Regression Testing been better incorporated in the company's testing strategy.

## How important is Regression Testing for your business?

Realistically speaking, Regression Testing is only as important as the level of maturity of your product, as we've previously examined.



“In order to improve testing at Blackboard, Ashley and her team didn't reduce the time it took for the regression tests to execute. Instead, they focused on breaking up the testing into smaller sections so that they were running the most valuable tests at the right time.”

- Excerpt From: *Leading Quality*, by Ronald Cummings-John and Owais Peer

You and your team must first identify your product's maturity level and then develop the appropriate testing strategy to compliment it. Here are some example questions to ask (taken from *Leading Quality*):

- What testing types make the most sense for our product's current stage?
- How can we optimize the feedback loops of our test execution so that our engineers get the most value from them?
- Do we need to reach outside our company to external partners to increase the team's capabilities or capacity?
- Where can we utilize external partners to fill in short-term or long-term gaps?

We hope that this guide has helped to dispel some of the myths surrounding Regression Testing!

Keep in mind that your testing strategy, including Regression, Functional and Unit Testing, should be adjusted and updated constantly. Think of Regression Testing as a single pillar that holds up the overall quality of the software application that you provide to your customers.

Because after all, we're developing software for humans to use and enjoy.

---

# Chapter 3 - Crowdtesting



## What is Crowdtesting?

Crowdtesting is all about harnessing the power of a “crowd” of testers to conduct various types of tests. These can include functional testing, exploratory testing, regression testing, localization and test case execution

Crowdtesting does not replace the entirety of your QA team or overhaul your entire strategy but rather builds upon your existing [QA process](#), scaling it with the help of a pool of testers spread across the globe.

By harnessing the power of the crowd, QA teams access thousands of professional testers without turning to hire internally. The globality of a crowd enables localized testing, providing users with a digital experience that looks and feels as if it were designed exactly for them.

How's that for customer delight? At [Global App Testing](#), our crowd has more than 55,000 testers in 189 countries with access to a broad range of OS and device combinations. Our refined algorithm selects the best testers for the right tests in real time.



# Advantages of Crowdfunding

## 1. Speed

By partnering with a Crowdfunding provider, you gain access to on-demand, remote testing from a team of skilled testers anywhere in the world.

POV: You send off a test on Friday evening just as you close your laptop for the weekend. When returning back to work Monday morning, your results are at your doorstep!

With testers operating in different time zones and working untraditional hours, Crowdfunding provides QA teams with 24/7 support. Feel confident in the knowledge that a tester will always be on hand to ensure your product works exactly as it should.

This expedited feedback loop means your team can work quicker to fix bugs and run more tests. With a streamlined SDLC and QA no longer considered a bottleneck, product releases will not only be faster but also of higher quality.

At [Global App Testing](#), we use a blend of automation and Crowdfunding, so you receive [exploratory test](#) results from the crowd within 48 hours. Test case execution results can be produced in as quickly as 30 minutes. Now how's that for speed?

## 2. Scale

It's safe to say that the ability to effectively scale is a common goal for most companies.

So as your business scales and customer base grows, QA must scale accordingly. A bug that impacts 10% of a 100-strong user base will quickly become a massive issue when user numbers reach 100,000.

Crowdfunding builds upon your QA capabilities and integrates seamlessly into your existing workflow, so your team remains productive and focused on high impact initiatives.

This in turn enables your company to scale and expand testing exponentially - all without the burden of hiring an extra resource. No longer is scaling defined by recruiting new testers and automation engineers.

Reducing the pressures of finding available testers every time you finish or update code will increase your team's capacity to work on bug fixes and strategy.

And when your SDLC works like clockwork, it makes scaling a whole lot easier.

### 3. Localization

Let's say you want to launch your app globally. You've tested your app in the UK, and it works perfectly. So it should work just as well in new, global markets, right?

Not necessarily.

Language barriers, cultural nuances, local devices and network combinations all offer a host of potential roadblocks in your global product launches.

That's where Crowdtesting comes in.

Leverage the crowd to perform [Localization Testing](#) in the countries of your choice, so you can rest assured your app works as effectively globally as it does in the country in which it was created. Moreso, user experience will increase, as the app will feel more native to them; say goodbye to that three star app rating and hello to five!

And... our testers love to test, and we love our testers!

It's a fantastic way to work remotely, allowing freedom of working hours and increased financial security. They encompass one supportive, global community.

### 4. Quality

When testing internally, it's common to be a bit biased or accidentally let a bug slip through the cracks; think of it like grading your own homework.

Crowdtesting provides local context from professional, unbiased testers across the globe who can spot bugs in-house teams might miss. Ensuring your app works as effectively worldwide as it does in the country it was created will undoubtedly improve product quality.

Speed, localization and the ability to scale all lead to one thing - higher quality.

# What's the difference between Crowdtesting and Outsourcing?

---

Great question.

Outsourcing typically involves passing the responsibility of your testing over to an external vendor, however, this doesn't necessarily allow for scalability. You're merely requesting testing resources you need at the time rather than expanding your capacity.

Additionally, the relationship between outsourcing vendors and their clients is often distant; you send off your tests only to receive raw results, with little guidance or suggestion on how to proceed.

Partnering with a Crowdtesting solution doesn't simply mean "borrowing" another company's resources. Global App Testing works within your current QA process, fitting seamlessly into your SDLC and integrating with your existing testing and development tools.

What's more, with Global App Testing, each customer is assigned a dedicated Customer Success Manager to deliver continuous improvement. So, not only can you relish in increased scale, device and test coverage, but you can also reap the benefits of sharing best practices and the latest technologies. Whether that includes monthly business reviews, setting goals or reaching milestones, Global App Testing provides testing solutions as well as a strategic arm.



## 5 advantages Crowdfunding holds over in-house testing

You might be wondering: “Why don’t I just hire internally to scale my testing?”

There’s a misconception that hiring internally solves all your scaling problems, providing you with a larger in-house team and more testing capacity as a result. But internal hiring requires a considerable budget and significant time dedicated to recruiting and training new employees.

And when your user base grows again or if you want to expand to a new country, you’re going to face the same issues you did before. So, what makes Crowdfunding better?

### 1. Crowdfunding saves time

In Europe, the average number of days to fill an IT position is 66. In the US and Canada, the average number of calls or interviews needed to fill an IT position is 15.

Let’s imagine you find the right candidate in 50 days, and it takes 8 interviews. That’s a lot of time spent perfecting an in-house team.

If you look at your calendar right now, it’s likely difficult to even find an hour-long chunk of time to work on that project you’ve been neglecting. Those hours spent reviewing CVs and interviewing candidates probably feel a lot longer now.

But with Crowdfunding, you have immediate access to tens of thousands of testers across the world, all at your fingertips. This saves recruitment time, so you can begin testing immediately - and with faster results.

And the beauty of Crowdfunding is that testers operate in time zones all across the globe, each with working hours that suit your schedule. So while you’re fast asleep, testers are hard at work testing your product. How’s that for a stress reliever?

### 2. Crowdfunding improves quality

We’ve said it before but because QA is our lifeblood, we just have to say it again. Crowdfunding improves quality. And quality improves, well, a lot - ROI, customer experience, scalability, speed to market, confidence in release cycles - the list goes on!

The highest quality apps are tested on hundreds of devices and OS combinations.

But what if you lack them in-house?

The last thing you want is to be scrambling last minute to order the new iPhone, OnePlus or Samsung device. This will inevitably slow down your entire SDLC as you twiddle your thumbs waiting for it to arrive. And while admittedly this is not the most tech-savvy approach, believe us, it does happen!

With Crowdfunding, you have access to testers equipped with a wealth of device/ OS combinations and networks. So now you can test on hundreds of devices without the time or commitment involved in buying them for your in-house testing team.

### 3. Crowdfunding is cost-efficient

According to [Glassdoor](#), the average salary of a QA tester in London is \$35,464 a year. And for a QA automation engineer, that salary is even higher at [\\$70,892](#).

That's a huge number to beat, especially when you need multiple testers working on multiple projects; the costs really do start to rack up.

Crowdfunding, however, enables you to expand your testing capacity exponentially without having to pay for new testers every time.

By leveraging our solution, you receive the best fitting 15-20 professional testers selected from a pool of 55,000 testers based across the world for each test - all with a scalable price solution. With a focus on budget efficiency, you'll be able to run high-quality testing without the associated price tag.

### 4. Crowdfunding delivers fast results

52% of organizations struggle to find time to test mobile apps.

That's because setting up test scripts, sourcing testers, analyzing results and prioritizing bugs all take time - especially in today's fast-paced tech world where team capacity is stretched to the very limit.

With Crowdfunding, there's no need to worry about sourcing testers, setting up complicated software testing tools or ensuring tests are of high quality.

Crowdfunding also works to eliminate unnecessary stress by fitting seamlessly into your SDLC and working with your existing QA strategy - not against it.

### 5. Crowdfunding provides experienced testers

Finding effective testers isn't always easy.

A great tester needs: experience, devices, time, efficiency, analytical skills and the ability to write detailed bug reports. So to be part of a crowd, testers must meet such standards.

Global App Testing employs only professionally vetted testers with considerable testing experience, so you can rest assured that your testers are of the highest quality.

In fact, we've met a considerable number of testers from our crowd all over the world at our Testathon events.

High-quality testers catch high-impact bugs, and it's the high-impact bugs that affect your users the most.

## Is Crowdfunding right for your team?

In today's tech world, the advancement of technology continues to accelerate at a rapid pace. This means your customers demand quality in an over-saturated, competitive market.

To meet these demands, companies need to deliver quality experiences every single time, with new and exciting features their competitors don't possess.

So with companies pressured to develop and release software more rapidly, the last thing they need is QA acting as a bottleneck.

If you want to continue to release software at speed without breaking the bank, Crowdfunding is the ideal solution. You might be thinking, "I don't have the budget for it." We're happy to say that at Global App Testing, our on-demand solution works with any budget. It enables you to scale globally, remain cost-effective and ensure you're delivering a quality product every time.

Crowdfunding fits seamlessly into your SDLC, working with existing processes to provide fast, actionable test results. This means your team can keep working to the best of its ability without being held back by the time-consuming, slow process of traditional QA.

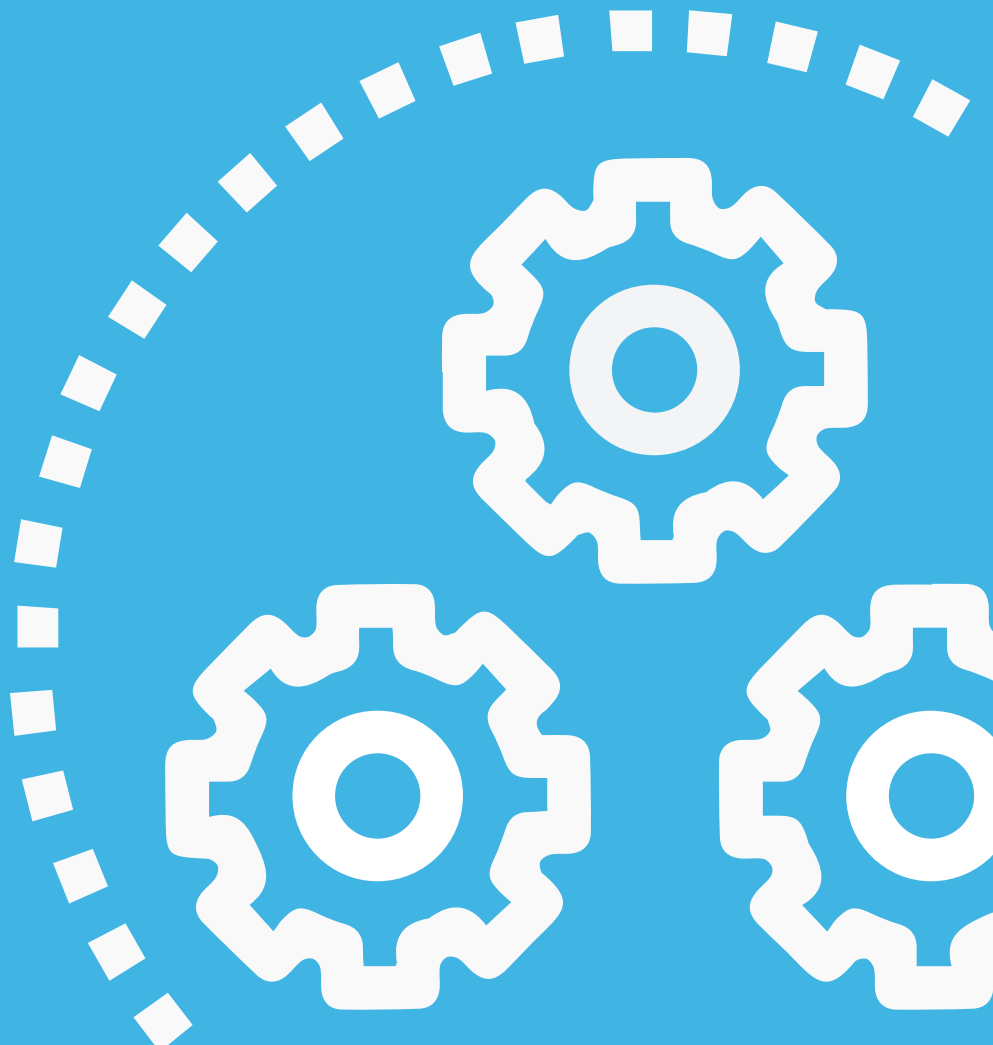
What's more, leveraging testers in almost any country gives you a competitive edge over your competitors, providing experiences that are fully localized and tailored to the individual user. And with churn rates at 71% after three months, this isn't something you want (or can afford) to pass by.

So, if you want to minimize costs and save time while you scale, localize and deliver quality, Crowdfunding is your answer.



---

# Chapter 4 - Functional Testing



## What is Functional Testing?

Functional Testing is a type of Black Box testing. It answers the fundamental questions:

“Does this actually work?”

“Can the user do what we expected?”

This method of software testing assesses the functional requirements of an app, verifying whether the software works as per customer needs. [Functional Testing](#) is not concerned with *how* processing occurs, but rather, *what* the results of processing should be.

### Functional Testing checks:

- **The main functions of a piece of software or a web application.**  
Quite simply, do the features work as intended?
- **How easy is it to access?**  
Can the end-user interact with everything they need to?
- **Usability.**  
The end-user should be able to make use of it without any difficulties.
- **What happens when errors occur?**  
Are there appropriate messages, and is there a way of logging them?



# How to test software functionality

1. Identify and clarify the functions that you expect the software or web application to perform.
2. Create input data based on these functional specifications.
3. Determine the output based on these functional specifications.
4. Write and execute test cases to gather test data (these can be manual or automated tests).
5. Compare the results of actual and expected output.
6. Make changes *if* the results do not match the end user's needs.

## Non-Functional vs. Functional Testing

So what is the difference between Non-Functional and Functional Testing?

The answer is relatively simple: Non-Functional Testing is concerned with the *how*, whereas Functional Testing is concerned with the *what*.

Functional Testing verifies *what* the system should do, and Non-Functional Testing tests how well the system works. Equally, Functional Testing is carried out to verify software actions, and Non-Functional Testing examines the performance of the software.

Another comparison you might see when discussing Non-Functional vs. Functional Testing is Black Box Testing vs. White Box Testing. Black Box Testing looks at the functionality of the software *without* looking at the internal structures. White Box Testing looks at these internal structures.

## Types of Functional Testing

- Unit Testing
- Functions
- Accessibility
- Smoke Testing
- Integration Testing

Many functional tests are designed around given requirement specifications - meeting business requirements is a vital step in designing any application. For instance, a requirement for an eCommerce website is the ability for a user to buy goods.

A practical example of this is the journey of a customer checking out their shopping basket. First, they should be sent to a secure payment page, then directed to bank security verification and then finally receive a confirmation email. Functional Testing verifies that each of these steps works.

## Types of Non-Functional Testing

- Performance testing
- Load Testing
- Reliability
- The readiness of a system
- Usability Testing

A practical example of this is measuring how many people can check out their shopping baskets simultaneously.

Not every type of software test falls neatly into these two categories, though. For instance, Regression Testing could fit into both categories, depending on how the tests are run.



# Top 5 Functional Testing best practices

## 1. Use Exploratory Testing

Exploratory Testing involves a lot of freedom on the tester's part. During [Exploratory Testing](#), the tester investigates an app to try and identify potential bugs. This method is relatively unstructured, and testers have the freedom to run tests when and how they see fit. This simultaneous process of test design and execution is hugely beneficial for Functional Testing.

Since Exploratory Testing focuses on how the app works, rather than how it's put together, testers don't require any particular knowledge of languages like Python or Java - just a familiarity with QA.

Testers put on their investigative hats and explore an app in real-world scenarios on real-world devices. By testing on-demand with minimal preparation required, you receive fast and valuable direction on the bugs in your app or software product.

When leveraging Global App Testing's crowd, for each test you receive 15-20 highly skilled testers, equipped with best practices in locating reproducible bugs within your app. On top of that, bug severity ranked results for exploratory tests are provided within 24-48 hours.

## 2. Automate some (but not all!) of the process

[Test automation](#) can save time and money.

If you find your team repeatedly testing a specific function, it's not sensible to use valuable resources across your DevOps or QA teams to continue to physically test it. By writing test cases for automation, you can run thousands of tests at once, verifying that the specific feature you're targeting works as expected.

All types of Functional Testing can't be automated, though.

[Exploratory Testing](#), for example, requires the creativity of testers to explore the app initially and decide which parts of it to test. System Testing and User Acceptance Testing also require manual efforts. But have no fear automation fans, there are many Functional Testing tools available to make the process easier and more efficient!

To create a high-quality product, you need to implement a blend of manual and automated testing.

Creating a Requirement Traceability Matrix (RTM) is one way to ensure all business requirements are tested, whether by automation tools or your testing team. By ingraining this in your QA strategy, you'll ensure that you discover more software bugs and cover more testing bases.

### 3. Use well-planned test case execution

Test Case Execution is the process of testing specific workflows in your app, a crucial part of Functional Testing. By using [Test Case Execution](#), your testers can work their way through particular functions for your mobile or web app, ensuring that they work as they should within the pre-planned workflow.

Let's say you write a test case for your eCommerce app. You want to ensure that users can search for "red trainers," click on a specific pair, add them to their shopping bag and then finally checkout. Within each of these steps, testers will be testing particular functions like the search bar, card payment and shopping bag to see if their results are as expected. They can then report on the application's conformance to the requirement specifications, and the development team can make the appropriate changes.

By planning test execution rigorously, you can verify that the test case covers all functionalities.

### 4. Test early and test often

It's critical to catch issues with functions early in the software development life cycle (SDLC) to prevent them from having a real impact on your company.

For example, if the checkout function of your shopping app is full of bugs, you'll see a direct effect on revenue. Equally, if the "sign up" functionality on your website isn't working correctly, you'll inevitably suffer a loss of sign-ups. Running functional tests early dramatically reduces the chance of these revenue inhibitors from happening.

You also don't want to wait until the last stages or the UAT (User Acceptance Testing) phase to determine what's wrong!

Implement testing techniques like Unit Testing in your product's design and development stages to avoid functionality issues further down the line.

### 5. Partner with a professional team

By partnering with a company like [Global App Testing](#), you can shift the time spent on Functional Testing to a crowd of testers, so you have more time to focus on test strategy and analysis. This frees up team resources to develop new features and design updates, something only they have the skill to do.

And the cherry on top? Enjoy a newfound competitive edge, as your competitors are stuck in the weeds of tediously testing themselves, neglecting to spend time enhancing their product.

# Make Functional Testing part of your QA strategy

Functional Testing is essential, and when executed correctly, it can reduce the number of software bugs your customers end up discovering, ultimately improving their experience.

As aforementioned, Functional Testing is most effective through a combination of Manual and Automation Testing.

Whether you're designing software, web apps or an API, it's important that it works. Functional Testing is the key to understanding whether it does - and if it doesn't, how to fix it.

If you're open to learning more about how Global App Testing could fit in with your existing processes, [why not speak with one of our QA experts?](#)

---

# Chapter 5 - Localization Testing

It's a competitive world, and any new software product needs to keep up with its rivals and make an immediate impact. Every company wants to bring their creation to market as quickly as possible and expand into new ones. In doing so, it's crucial to strike a balance between speed and quality.

A rigorous testing process must be carried out before any new app or website is launched to ensure a great user experience across all markets. For a business with global reach, Localization Testing is one of the most critical factors in engaging and retaining users.

This chapter will explore what Localization Testing is and how you can harness it most effectively. And in a world where the customer reigns supreme, our top five best practices for Localization Testing will help grow your business by delivering a product that truly delights customers.



## What is Localization Testing?

Localization Testing is a software testing technique that checks whether an app or website offers full functionality and usability in a particular locale. If a product has been customized for a targeted language or region, Localization Testing verifies the accuracy and suitability of the content.

It's not just about translation - the testing process makes sure that the product is adapted appropriately for the end-user's cultural, linguistic and functional requirements. As well as ensuring your app works as effectively in one country as it does in another, a "localized" experience makes the customer feel like the product was developed especially for their use.

### Why is localization so important?

If you plan to expand into a new country, localization is crucial. Ensuring that your product is designed well and works smoothly in a specific region will lend you authority and popularity, leading to increased market share and revenue.

In a fast-moving world, most people don't give second chances. Customers are more likely to stick with a localized product that works for them, and those who encounter linguistic or cultural gaffes will be put off.

In short, better UX equals more sales.

Localization also offers scalability and a reduction in overall costs. Although the roll-out of initial testing might take time, putting this process in place will ensure that future releases and tests can be carried out much faster. This means you'll be ready to move into new markets as soon as they emerge.

**In the next section, we'll take a look at how localization can boost your business by:**

- Embracing global opportunities
- Building scalability
- Improving the testing process
- Engaging with customers

## How can localization boost your business?

Technology has connected the world like never before, and companies must take a global perspective to remain competitive in such an environment. Leveraging Localization Testing is a foolproof way to send the right message to a specific, global audience and generate additional revenue streams.

## Embracing global opportunities

We won't lie - setting up a successful localization process takes [hard work](#), commitment and resources. But rather than thinking of it as a one-off expense, think of it as an investment in opening up your business to global opportunities in the future.

While major global competitors are bound to expand into new markets, local companies in your target region may want a piece of the pie; and they have the advantage of knowing their audience. Localization Testing is vital for maintaining your product's competitive edge.

It's worth taking a moment here to mention Internationalization Testing, which is the process of ensuring a product supports multiple languages and locales (whereas Localization Testing focuses on the validation of one particular locale or language). The two together are referred to as globalization.



# Building Scalability

---

A key benefit of Localization Testing is its scalability. This means your business has more scope for further expansion whenever the time is right for you. By focusing on localization at the development stage, you can build that scalability potential into the process - and ensure that localized versions of your product reach the market as quickly as possible.

It's helpful to make sure your source content and user interface are flexible enough to be adapted for a different locale. For example, you can standardize expressions and phrasing in your original language, make embedded text editable within source graphics and check that subtitles on video tutorials are overlaid rather than burned.



## Improving the testing process

The aim of Localization Testing is to identify errors in translation or issues with cultural differences. But you'll also find that a well-designed localization strategy helps streamline the QA workflow by finding ways to save time and money.

Localization is crucial for your global outlook, but it must work in conjunction with the other areas of the software development life cycle. For instance, there should be overlap with the functional testing team to give the localization team an understanding of the product architecture.

If you've added a new feature or fixed a glitch, Regression Testing should be applied to localization; all teams must check that this hasn't impacted the basic functionality. When testers are scrutinizing your output for language-specific errors, they'll probably spot opportunities for improvement in the original language too!

## Engaging with customers

Localization is the number one way to [build a sustainable rapport](#) with your global audience. As we mentioned earlier, customers who feel the product is both relatable and usable are most likely to embrace it and share their opinions with others. This is key when working to increase your app store rating.

Users genuinely appreciate attention to detail, especially if they're used to dealing with apps or websites that are glitchy in terms of features like translation. Optimizing your product for a specific locale will show customers that you care about them and want to meet their needs.

[Research](#) shows that internet users from non-English-speaking countries prefer to buy products in their native language, so you'd be crazy to release an app with an American keyboard into a market that uses a different alphabet! By taking cultural differences into account, your company will demonstrate sensitivity and awareness.

Customers also want full compatibility with their preferred devices and operating systems. This makes hardware compatibility tests a crucial element, with real users testing products on real devices in the target locale.

# Top 5 best practices for localization

While it has its benefits, localization does bring its challenges, including slower time to market, harder to measure ROI and higher demands on internal resources. We've compiled a few top tips to help you navigate these roadblocks to ensure localization grows - not inhibits - your business.

## 1. Take your time

Rome wasn't built in a day and localization won't happen overnight!

It'll take time to assemble the resources and team necessary to carry out the testing. An additional factor impacting time is the research required to learn about your target audience and the local market conditions.

It's critical to learn everything you can about the target users to tailor the localized experience to them. This is one area where automation can't do the job for you. Human testers need to walk through the customer journey and perform extensive research to create highly specific test cases for the product.

Take time to ensure the accuracy and appropriateness of your translation. For instance, think about when you put text through Google Translate, translate it into a different language and then back to English. It often won't make sense at all. That's because language nuances often don't translate accurately without the help of an expert.

A popular [tonic water](#) company once introduced its product to the Italian market without realizing that its name actually translated to "toilet water" in Italian! This was an expensive blunder, and confidence from local markets went down the drain - no pun intended. This only further demonstrates that some bugs can only be found by native speakers with [local insights](#).

## 2. Test early and often

"Test early, test often" is a common mantra in software testing, and it definitely applies to localization. However, too many companies treat localization as an afterthought, tacking it on to the end of the process when the product is almost ready. This is a mistake.

Let's say you leave Localization Testing until the late stages of your development process. You're confident in your app and ready to release.

You decide to quickly translate it into the target language to make it accessible to other target markets.

Then the panic hits. Suddenly, the text doesn't fit in buttons or boxes, and everything looks out of whack. Your whole design might suffer if you're switching to a language that reads right-to-left, such as Arabic, Hebrew or Urdu.

Your team is now forced to spend a considerable amount of time editing code to make sure everything fits. What's worse? The whole thing could've been avoided had Localization Testing been included earlier on.

As well as carrying out Localization Testing at an early stage, regular testing and retesting is necessary to iron out any bugs or glitches during the development process. Automation can save time, especially if you use a Localization Testing tool such as Selenium WebDriver.

However, due to the nuances we mentioned earlier, you need to combine automated technology with human input. This is where the Agile methodology comes in handy, with its principles of adaptability and continuous feedback.

In an Agile team, the product is thoroughly tested and debugged as it's created, rather than waiting until it's complete. Testers not only find defects earlier on in the process, but they also develop an intricate knowledge of the product.

The Agile approach condenses the development cycle and constantly provides user feedback, ensuring the product adapts to the market during development and reaches customers as soon as possible. This lets the team prioritize product usability, ensuring it suits any market.

### 3. Make sense of the data

Analyzing data and customer feedback is vital for measuring your product's performance and identifying areas for improvement. But here's the thing - it won't work unless you have a sense of what you're actually measuring.

The localization process helps companies think about differences in language and culture to create an appropriate user experience. It stands to reason that the way you measure success and [interpret the data](#) will differ from country to country.

To prove ROI, you need to establish appropriate metrics for the specific region you're targeting. Decide on the number of downloads, purchases and sign-ups you want to measure your success against, and focus your efforts on achieving these goals in a global market.

You could analyze the number of new customers acquired in global markets, the percentage of market share, web traffic by country and language and increases in revenue. Build support throughout your company for the project by delivering on key metrics which support your business.

And if you aren't hitting the right numbers, ask yourself why? Here's an interesting example of just that.

A Swedish tech company found that they weren't delivering on their key performance indicators in Indonesia. They had low numbers of sign-ups and couldn't identify the issue.

After a round of Localization Testing, they realized that it was because their onboarding process required a last name, which just under 40% of people in Indonesia don't have. A large number of Indonesian people were not signing up merely due to the fact that they were unable to do so. Once the company fixed that bug, a whole new market of 262 million people opened up.

#### 4. Be culturally aware

Localization isn't a straightforward translation process. Cultural nuances also don't often translate easily or at all. For example, when a [famous diaper brand](#) tried to expand its reach to Japan, they printed the typical image of a stork carrying a baby onto the packaging. When sales seemed to be dwindling, they gathered some local insights to figure out the cause of the problem.

They quickly discovered that, in Japan, the story goes that giant peaches bring babies to their parents, not storks! The cultural reference was utterly lost, and the company suffered as a result. By implementing Localization Testing, embarrassing or offensive mistakes from cultural differences are entirely avoidable.

Apart from accidentally causing offense, another pitfall is failing to make your product adaptable for all language content. For example, the [user interface](#) should be tested to accommodate text with larger lengths without distorting the alignment and cater to alternate writing directions.

It's not just different alphabets and characters that must be accommodated. Think about spelling variants, numerical systems, currency symbols, hotkeys, HTML and hyperlinks. The date and time format will need to change for different countries, as will phone number formats. Then there are weights and measures, calendar differences and licensing laws particular to the region.

This is an area where human testing needs to be deployed across a wide range of scenarios. Crowdfunding is an effective way to do this, as it provides access to a large number of testers in different countries with different languages.

#### 5. Choose a partner with global reach

Getting localization right can be a challenge, but you don't have to do it alone.

There are plenty of experts who already have the infrastructure in place and a guaranteed ability to test and deploy products in global markets. These companies can help you implement a highly efficient localization testing process, saving you time and valuable resources.

By partnering with a professional Crowdfunding company, you can achieve a global reach without having to hire separate testing teams across the world. [Global App Testing](#) has 55,000 testers across 189 countries on hand 24/7 to test your product. This will ensure that your translations are correct, functionality is to a high standard and that you understand any cultural nuances that may apply to new markets.

[LiveSafe](#), for example, is used by major educational institutions and corporations to allow users to communicate in real-time with their organization's security team. Livesafe relies on Global App Testing to ensure that their apps work globally in highly critical security situations. By testing in over 19 countries with Global App Testing, LiveSafe has confirmed that their app works in the countries they're targeting, guaranteeing global growth and the safety of their users.

# Reaping the rewards of Localization Testing

---

If you intend on becoming a global business, Localization Testing is a must. When done right, you'll open the doors to millions of new customers across the world - and massively improve your earning potential.

Accurate translations and cultural sensitivity are the cornerstones of the process, as well as ensuring your software and content are compatible with a wide range of devices, operating systems and browsers. Such attention to detail will let your customers know that your company cares about their digital experiences.

The importance of researching your target market can't be overstated, and it definitely pays to bring in a linguistics expert or a localization firm with proven expertise. That way, you can avoid any embarrassing and costly misunderstandings.

---

# Chapter 6 - Exploratory Testing



## What is Exploratory Testing?

As the name implies, Exploratory Testing tasks testers with exploring an application to identify and document potential bugs.

Testers embark on a process of investigation and discovery to test a product effectively. During Exploratory Testing, testers are free to run tests how and when they decide. This means there's a simultaneous process of test design and test execution.

**Best practice Exploratory Testing answers the following questions:**

1. Does the app perform the function it was designed for?
2. Does the app work under multiple scenarios?
3. Is app performance good enough?
4. What potential bugs are there?

These four questions are answered through a process of learning, designing and execution.

---

### Top 3 best practices for Exploratory Testing

Best practice Exploratory Testing has three main components:

- Learning
- Designing
- Executing





Let's break them down a bit more...

### Learning

Testers need a comprehensive understanding of the app or website they're testing to test it effectively. By understanding valuable information like industry awareness, company details and competitive benchmark data, the tester will have the context necessary to execute tests successfully. Learning, therefore, is central to the Exploratory Testing process.

### Designing

A big difference between Exploratory Testing and Scripted Testing is in design. In Scripted Testing, the test has specific parameters or rules, whereas Exploratory Testing has no preset path or predetermined conditions. Exploratory testers conduct tests in a way which they deem fit. As a result, tests are designed frequently and freely, making designing a crucial skill for an exploratory tester. Therefore, you need to ensure you have professional testers with a varied skill set and can design effective test cases.

### Executing

Seamless test execution is also a fundamental best practice. In Exploratory Testing, the tester has the freedom to execute a test as soon as they'd like to do so. As soon as the test is written, it can be conducted. This freedom means that nobody's waiting on scripted requirements, and work can be conducted relatively seamlessly.

## Exploratory Testing in the real world

Let's imagine you want to test a mobile gaming app.

If you chose Exploratory Testing, testers worldwide would be given very general scenarios to test. For example, in a gaming app, the user might expect to use tokens or in-game inventory items before a battle. The Exploratory Testing for this may require a tester to verify that this functionality works and the player can complete the battle as expected.

Exploratory Testing, in this case, involves simultaneous test design and test execution, as testers explore the game and the different in-game scenarios that may occur. Although there is a structure to the testing process, with testers planning the general scenarios they need to test, there's an inherent freedom in how and when the testers navigate the app.

These kinds of tests can't be executed via automation. While a script could check that tokens or inventory items are purchasable, it can't emulate a human doing battle! This is the inherent human aspect of Exploratory Testing. Since some tests can't be predetermined (like a battle in a game), the human creativity required for Exploratory Testing is a key factor of its success.

## Difference between Scripted Testing and Exploratory Testing

Unlike traditional, Scripted Testing, Exploratory doesn't restrict testers to a predefined set of instructions. Test cases are not written in advance, enabling the tester to learn and iterate throughout the process. This makes Exploratory Testing unique and extremely valuable, as testers can use their own creativity to discover bugs that the software developer may not have ever anticipated.

## Exploratory Testing in Agile teams

Working in Agile involves adaptive planning and continuous improvement; the aim is to be able to respond to change quickly and efficiently. This means that Exploratory Testing can work exceptionally well with Agile teams. Increased demand for frequent software updates requires faster software development and continuous delivery. Testing methods need to respond to change quickly and seamlessly. Enter Exploratory Testing, where the constant process means bugs can be found faster and releases occur more frequently.

## Advantages of Exploratory Testing

- **Finding more bugs:** A major benefit is that Exploratory Testing finds bugs that automation simply can't. Automated tests are limited to the test cases that are written for them; they won't test any bugs outside of this scope. You can learn more about this on our blog: [QA vs. Humans: Can You Automate Everything?](#)
- **Speed of the test cycles:** Exploratory Testing doesn't require extensive planning. The scope of a test cycle must be clear, but detailed test cases aren't necessary. That's because testers are trusted to test what they believe requires testing. In comparison, automated tests are fast in test execution but can be time-consuming to plan.
- **Idea generation:** The creativity required of testers and the fast pace of Exploratory Testing means that more ideas are created.

## Disadvantages of Exploratory Testing

- **Getting it right is difficult:** As previously mentioned, Exploratory Testing relies heavily on the testers. It can be an expensive and difficult skill to acquire, so you may find it difficult to find exploratory testers.
- **Measuring ROI can be tricky:** Evaluating the success of Exploratory Testing in the short term can prove difficult. Think of running a marathon - you can't train ten hours per day a week before the race and expect to do well. It takes months of regular training to be successful. Exploratory Testing is the same. Long-term consistency will generate better results than short-term intensity.

# When to employ Exploratory Testing

We have discussed what Exploratory Testing is, the advantages and disadvantages of the technique, and how it can be used in Agile, but how do you know *when* you should actually implement it?

## 5 scenarios where Exploratory Testing can augment your testing strategy:

1. You're not sure what tests to run.
2. You want to diversify the testing process after a cycle of scripted testing.
3. You need rapid feedback on a product.
4. You want to understand the ins and outs of a new release quickly.
5. You want to find more bugs!

Exploratory Testing is a creative and diverse testing method. Employing this technique can help you find more bugs and achieve fast results, as testers are free to write and execute test cases seamlessly.

---

# Chapter 7 - Manual Testing



## What is Manual Testing?

Manual Testing occurs when human testers check the quality of a new application without using automation tools or scripting. This type of testing identifies bugs or defects, verifies the product is error-free and confirms it meets specified functional requirements.

The process compares the behavior of a software application (or one of its components or features) with the expected behavior that was defined in the initial phases of the software development life cycle.

Manual testers design test cases or scenarios with 100 percent test coverage and execute them one by one before verifying the results. They ensure that any reported issues are passed to the development team to fix and then tested again.

Automating everything is impossible, so Manual Testing is an essential part of your QA testing strategy.

### Examples of Manual Testing

When assessing usability and accessibility, Manual Testing is especially beneficial. For example, if you were launching an eCommerce website, you would check elements including:

- Optimization for a range of browsers and devices
- Smooth checkout process
- Fast-loading hi-res images
- Links to social media channels

During Manual Testing, testers check the code that drives each of these functions to ensure they work as the client intends. Manual testers are also able to comment on the look and feel of the website, evaluating it from the user's perspective.

### Manual Testing vs. Automation Testing

As its name suggests, Manual Testing requires human involvement, while Automation Testing uses machines to execute test cases automatically. Any type of application can be tested manually, however, Manual Testing is especially suitable for assessing user interfaces (UI) and user experience (UX), and for ad-hoc or Exploratory Testing.

Automation Testing is recommended only for stable systems, which are likely to have fewer bugs, and is most commonly used for Regression Testing and Performance Testing. Testing tools like JMeter and Selenium are commonly employed.

### Advantages and Disadvantages of Manual Testing

Manual Testing is essential in ironing out the highest number of bugs, however, it does have its downsides. We've outlined the primary pros and cons to better inform you of the challenges and rewards associated with this method of testing.

Let's take a look...



## 1. Accurate

Automated tools are smart, but they're not as smart as humans. Certain scenarios require a real person with real-world experience. So when it comes to identifying bugs and glitches in software, Manual Testing is more likely to catch 'em all.



## 2. Human Insight

Manual software testers bring valuable human perspective by focusing on the look and feel of a product. They can evaluate the application's visual components and highlight UI and UX issues by adopting the mindset of the end-user.

# Advantages of Manual Testing



## 3. Adaptable

The manual method is beneficial in ad-hoc testing, as it's easily adaptable when unplanned changes are made to the software. The human input means test cases can be redesigned or tweaked with relative ease. Manual Testing is also agile enough to be performed on a broad range of applications.



## 4. Saves Money

Although Manual Testing requires skilled labor, it can in fact save money, as it doesn't require expensive tools. Automation tools are costly to install and take ample time to set up and learn.



### 1. Resource Heavy

Manual Testing is undeniably more time-consuming than automation, elongating the testing process and sometimes increasing costs. It also requires a large number of human resources, with testers requiring considerable analytical and creative skills.



### 2. Not Always Suitable

Certain types of testing, such as Performance and Load Testing, are not suitable for manual methods. For example, humans can't simulate a large number of users for a performance test in the way a machine could. Large amounts of test data are also more efficiently handled by automation.

## Disadvantages of Manual Testing



### 3. Potential for Error

This is the flipside to #1 on the “pros” list. Humans are smarter than machines in many ways, but they're also prone to human error. Since Manual Testing is repetitive and tedious, it's possible for testers to lose concentration, ultimately leaving an error undetected.



### 4. Not Reusable

As the Manual Testing process can't be recorded, manual tests are therefore not reusable. As a result, separate test cases for each new application must be developed, a process much easier to complete through Automation Testing where the scripts are reusable.



# Types of Manual Testing

Manual Testing varies, with different types suited to different software and environments. Let's take a look at some of the most commonly used techniques.

---

## Acceptance Testing

The client or end-user performs User Acceptance Testing (UAT) to confirm if the software meets the agreed requirements. Sometimes called Pre-Production Testing, it occurs during the final phase before releasing the product to market.

UAT is an example of Functional Testing, and types of Acceptance Testing include Alpha (executed within the organization) and Beta (where the application is released to a limited market to generate user feedback).

## Black Box Testing

Also known as Behavioral Testing, this method analyzes an application's functionality from the end-user's perspective. The internal code structure is not visible during testing (hence the name "Black Box"), so testers are only aware of the inputs and expected outputs of the software.

Black Box Testing has several subdivisions, including Functional Testing for requirement compliance, Smoke Testing to assess basic functionality and partitioning (dividing software into groups that are expected to exhibit similar behavior).

## Integration Testing

Integration Testing is the process of testing an application with two or more integrating components. It's performed once the individual components have been unit-tested and identifies problems with the interfaces and their interactions.

The two main methods are the Bottom-Up approach (moving steadily from the bottom module to the top module) and Top-Down approach (the opposite).

## System Testing

System Testing means testing the system as a whole once all its components have been unit-tested and integrated. It checks that the complete application works as intended by comparing it against the original requirements.

Also called End-To-End Testing, it typically involves Installability Testing (does the software install correctly?) and Recovery Testing (can the application recover from hardware crashes and network failures?).

## Unit Testing

---

This form of testing evaluates the individual units or components of an application's source code to ensure each function performs as intended.

Developers usually conduct Unit Testing rather than engineers, as it requires detailed knowledge of the internal program design and code. Also known as Module Testing or Component Testing, it simplifies the debugging system and helps to detect and protect against bugs in the future.

## White Box Testing

---

Sometimes called Transparent Box Testing or Structural Testing, this method of testing checks an application's internal structures. It's performed by the developer, who checks the software's internal codes before passing it to a test engineer.

White Box Testing focuses on strengthening security and improving the software's design and usability. A combination of both Black Box and White Box Testing is known as Gray Box Testing.



While several different types of Manual Testing exist, the overall testing process is the same in all. Testing can be conducted in-house, however, some businesses select a specialized company to handle their testing.

### There are roughly six stages in performing Manual Testing:

#### Understand requirements

1.

Testers begin Manual Testing by first understanding the project's requirements fully. They consider questions like: What does the client expect from the application? What problem is it aiming to solve for end-users? Testers must analyze all requirement documents to recognize the expected behavior of the software and exactly what needs to be tested.

#### Prepare test cases

2.

Once the requirements are understood, testers can draft test cases to cover various scenarios, such as what happens when a user enters an invalid password or how the software would cope with a crash. These test plans establish a sequence for testing functionality and usability across the entire application.

#### Review test cases

3.

It's helpful to review the draft test cases with team leaders and the client to ensure they cover all bases and make any amendments before commencing the execution. This will save time in the long run.

#### Execute test cases

4.

Manual Testing can now be carried out using any of the techniques listed in the previous section. In addition to finding bugs, the aim is to identify potential pain points for users and loopholes that hackers could exploit. Testers execute test cases one by one, sometimes using bug-tracking tools like Jira.

#### Report bugs

5.

When bugs are identified, the testing team will pass the metrics to the development team in the form of a [test report](#). This contains details on how many defects or bugs were found, how many test cases failed and which need to be re-run.

#### Test again

6.

Once the development team has fixed the bugs, the software is handed back to the testers. They carry out the test cases again to confirm the problem has been resolved.

# Combining Manual and Automated Testing

---

Although automation is a time-saver, Manual Testing remains a vital part of software development. Human testers embody the mindset of the end-user and imagine multiple test scenarios for an application or function.

It's worth remembering that while software testing attempts to find as many bugs as possible, identifying all possible defects is quite literally impossible. Manual testers often spot issues that a machine could overlook, but they're also susceptible to human error.

Using a combination of Manual and Automated Testing is the most effective way to catch the highest number of bugs and defects. Global App Testing offers the best of both worlds by blending human insight with intelligent automation. [Contact our team](#) to learn more from one of QA experts.

---

# Chapter 8 - Agile Testing



## What is Agile software development?

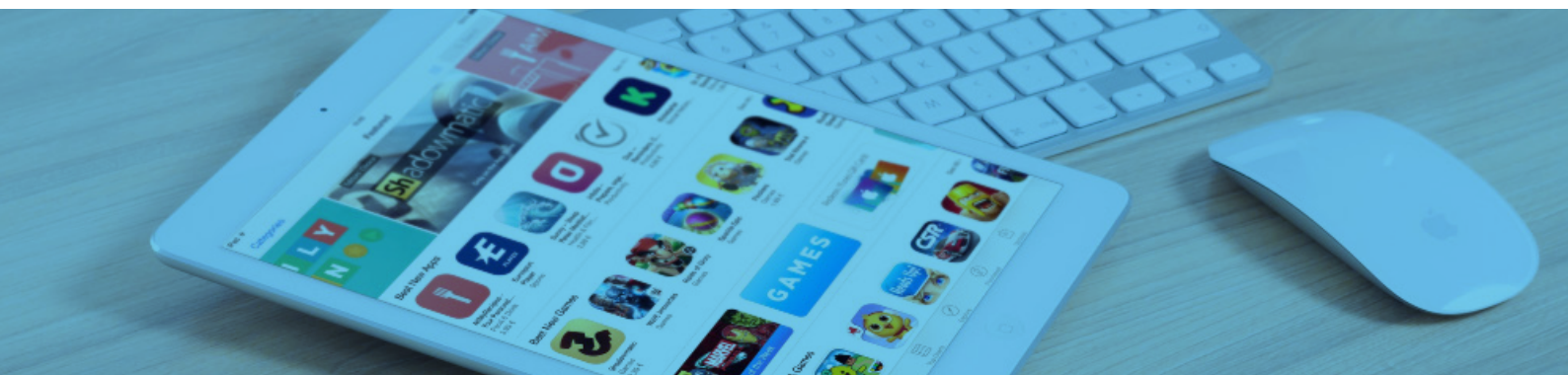
Agile Testing has become a critical part of application lifecycles, significantly impacting software development, testing and quality assurance. It's also gained widespread acceptance as a crucial driver for the delivery of high-quality products. In this chapter, we take a deep dive into the world of Agile Testing to provide a better understanding of how it works and how it can help you.

To understand Agile Testing, it's vital first to understand what the Agile development methodology involves. It's an umbrella term encompassing many practices that are quite different from traditional development techniques.

Let's start by looking at the key principles of Agile software development.

### The four core values are:

- To focus on people rather than processes and tools
- That a working piece of software is more important than detailed documentation
- That ongoing collaboration with customers matters more than a fixed contract
- To be responsive to change, rather than sticking to a plan



As the name implies, an Agile methodology is focused on responding to change. A team might use many frameworks, such as Scrum or Kanban, but all of these center on a collaborative approach.

A traditional development approach might separate team members based on the area they're working on and slowly add pieces together to create a finished product. With Agile, continuous integration is key - the whole team collaborates, and new features are added as they work. It creates an entirely different software development life cycle.



# What is Agile Testing?

---

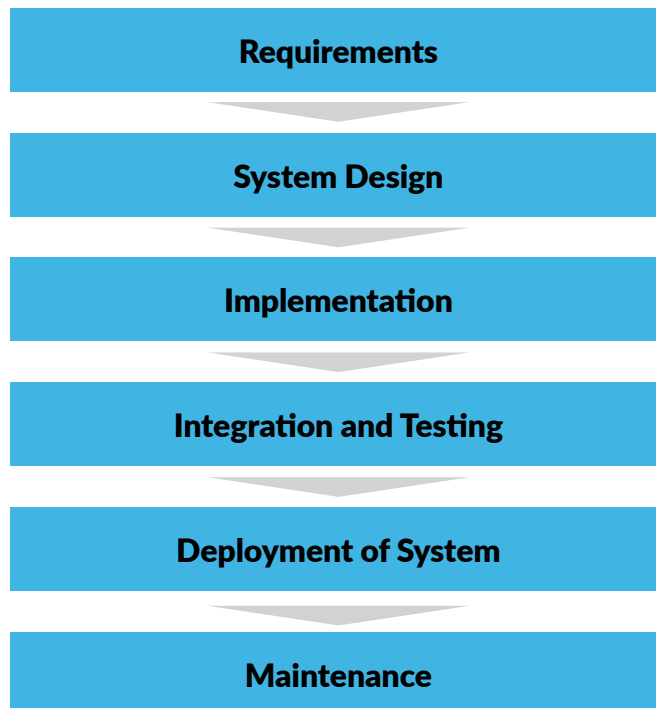
Agile Testing operates under the philosophy that continuous testing is a crucial part of development - on par with coding.

In Agile, testing is integrated directly into the development process so that bugs are discovered as early and as often as possible. As a result, testers can identify problems at every point in the development process, moving the product quickly towards release.



## Traditional testing method vs. Agile Testing

To keep things from breaking in the customers' hands, testers attempt to break it first - and then proceed to have it fixed. In the traditional waterfall method of development, the [sequence of events](#) looks like this:



With this method, the next step doesn't begin until the previous step has been completed fully, so the testing team doesn't receive the product until [late in the development cycle](#). This can be a real challenge for your software testing team, as any bugs they catch at this point will be difficult and costly to eradicate from the product.

Testers who enter the process at this juncture are also unable to ask the right questions and perform the right tests because there's little feedback from other members of the development team (who are sometimes viewed as 'the enemy') or from customers.

Testers are forced simply to wait for the product to come down the assembly line (or waterfall), then use a narrow set of skills to decide whether it should be kicked back to a previous step in the development process.

With Agile, the test plan is in place throughout. Every time a new update is made to the code, the test team gets their hands on it, feeding back directly to the developers. These test cycles can also feature automated tests and a small selection of end-users.

# Key principles of Agile Testing

In the book, [Agile Testing: A Practical Guide for Testers and Agile Teams](#), Lisa Crispin and Janet Gregory distilled Agile testing into [10 principles](#). Since its publication, these 10 principles have been widely accepted as the foundation of Agile Testing. **These demand that you:**

1. **Provide continuous feedback.** Agile testers don't just test constantly. They're also responsible for distributing the results of those tests and facilitating the provision of feedback from customers to developers to create a more robust product.
2. **Deliver value to the customer.** While this is the second principle, it's the most crucial. The end goal of every action taken by an Agile tester is to create the best product possible for the customer.
3. **Enable face-to-face communication.** The Agile tester's role is to reduce confusion and errors by communicating directly with developers and enabling customers to communicate directly with developers.
4. **Have courage.** Developers can be defensive about their work. To meet their goal and provide value to the customer, Agile testers must fight for the changes and fixes that need to be made.
5. **Keep it simple.** Agile testers act on the aphorism that simplicity is the ultimate in sophistication. For testing, that means performing only those tests that are necessary and all tests that are necessary. For the product, that means delivering the simplest possible product that delivers the most possible value.
6. **Practice continuous improvement.** Agile testers are keen learners; they're never done learning how to do their job better.
7. **Respond to change.** Agile testers are adaptable and flexible, keeping up with feedback from user stories and changes in the product and marketplace.
8. **Self-organize.** Instead of waiting at an assigned seat on the assembly line, Agile testers spring into action at every point in the process. They actively seek out problems and bring people together to solve them.
9. **Focus on people.** Agile testers are collaborative, preferring human interaction to technology. Their focus on people enables them to deliver a product that prioritizes usability and utility.
10. **Enjoy.** No one is as successful at meeting their goals as when they enjoy what they're doing. Agile testers who enjoy the work are able to deliver the greatest possible value to the customer.

# The typical profile of an Agile tester

Derived from the above principles, an Agile tester typically possesses:

- Strong communication skills
- A diverse, technical-based skill set
- Familiarity with a variety of testing tools and test automation
- An ability to collaborate with others effortlessly
- A willingness to embrace change
- An ability to liaise with everyone from DevOps to business analysts
- Broad experience in Exploratory Testing
- A results-oriented personality
- A passion for delivering value in business

---

## Types of testing in Agile

A broad range of methodologies has been developed for [Agile Testing processes](#). Below are four of the [most popular Agile Testing methods](#) currently in use. While no single procedure is perfect for a specific product, these frameworks are useful as starting points from which to generate a bespoke approach:

### 1. Acceptance test-driven development

ATDD is a form of TDD (test-driven development). It embraces the collaborative nature of Agile Testing, bringing together customers, developers and testers to create acceptance tests from the customer's point of view. Only once these tests have been created is the corresponding functionality developed.

It's easy to make test cases with this style of workflow. This gives developers direct insight into what customers want and how the product will be used, removing ambiguity from the process and reducing the chances of significant errors being made.

### 2. Behavior-driven development

BDD is based on and enhances test-driven development and acceptance test-driven development (ATDD). Using their structure adds the identification of correct business outcomes and performs tests based on those preferred outcomes.

BDD has five steps:

1. Describe the behavior.
2. Write the step definition.
3. Run and fail.
4. Write code to make the step pass.
5. Run and pass.

### 3. Exploratory Testing in Agile

Exploratory Testing is a cyclical method, progressing from test design > test execution > analysis > learning, before beginning the loop again. The tests themselves are not scripted; instead, they're generated by Agile testers as the product is explored, requiring the tester to make full use of their unique skill set.

Exploratory Testing is the closest testers get to interacting with a product precisely as it will appear "in the wild." It's a great way to find out quickly if you have working software, and it allows testers to identify bugs that wouldn't be found through other testing methodologies.

### 4. Session-based testing

Like BDD does for ATDD, session-based testing builds on and refines Exploratory Testing.

The strength of Exploratory Testing - the creativity of the people who do it - can also be its greatest weakness. Session-based testing attempts to remedy this by adding structure. First, before a test session begins, a charter is created. Second, uninterrupted testing sessions take place, focusing mainly on a single charter. The entire session is then reported on, and the manager is debriefed after the test. The additional structure ensures that all product areas are thoroughly tested and avoids backlogs building in any particular area.


---

## Agile Testing quadrants

With these and other testing methodologies, it can be difficult to assess *which* type of test should be run, how often it should be run, when it should be run, and who it should be run by. There are so many types of tests - acceptance testing, regression testing, unit testing, and more. There's also the question of whether manual or automated testing is better suited for the current iteration of the product. Gregory and Crispin created the concept of Agile Testing quadrants, which provide a taxonomy for tests.

[According to Crispin](#), the two left-hand quadrants help teams know which code to write and determine when they're done writing it. The two right-hand quadrants help teams learn more about the code they've written, providing feedback to the left-hand quadrants.

- **Q1 - The Automated Quadrant** contains tests designed to improve the code of the product being created; they're performed to help the team create a better product.
- **Q2 - The Automated & Manual Quadrant** contains tests designed to improve the business outcomes of the product being created; they're performed to help the team develop a product that drives value for the business and customers.
- **Q3 - The Manual Quadrant** contains tests to provide feedback for tests in quadrants 1 and 2 by testing the product and user experience to ensure business outcomes.
- **Q4 - The Tools Quadrant** contains tests that use technology to ensure the code fulfills all nonfunctional requirements such as security and compatibility.

A high-angle, top-down photograph of a diverse group of about eight people sitting around a large wooden table in a meeting. They are looking at documents and talking. The image is dark and semi-transparent, serving as a background for the text. On the table, there are laptops, mugs, and papers.

There are three simple benefits to adopting Agile Testing: a happier team, a higher quality product and faster delivery. Achieving this trifecta is worth the effort put into developing an effective Agile Testing framework.

# Advantages of Agile Testing

## 1. A higher-quality product

---

Agile enables testers to detect more defects earlier in the development process.

One of the Agile principles is “continuous feedback.” The doctrine of starting testing concurrently with development means bugs can be eliminated soon after they appear. Each iteration of the product is tested thoroughly and debugged as it’s created, rather than waiting until it’s finished. Testing also involves every member of the development team, so the skills of developers and testers are leveraged to produce a perfect product.

Another outcome of continuous feedback combined with early and frequent testing is testers developing an intricate product knowledge. Depending on the methodology of testing used, they can combine that knowledge with customer input to help developers create a superior product.

## 2. Fast delivery

---

With Waterfall Testing, the initial stages of development and eventual release into the market are separated by months, if not years. As a result, features or the entire product can be completely irrelevant by the time it reaches customers.

Agile Testing methodology both compresses the development cycle and constantly provides customer feedback, ensuring the product adapts to the market during development and reaches customers as soon as possible.

## 3. A happier team

---

The last principle on the Agile Testing list is no mistake: enjoyment. Agile Testing necessitates close interaction between all team members, creating a happier, more enjoyable and more productive workplace. Developers, testers and customers work side by side to create the best product and the most value possible.

**Crispin and Gregory say it best:**



“A team that guides itself with Agile values and principles will have higher team morale and better velocity than a poorly functioning team of talented individuals.”

## Disadvantages of Agile Testing

No system is perfect. Improperly implemented, Agile Testing can weaken team structure and product development, preventing a viable product from ever being released. Even when properly used, all Agile methodologies have their weaknesses. For example, Exploratory Testing can lack the structure necessary to ensure a product is tested comprehensively; ATDD accounts for customer feedback but not for business outcomes.

The emphasis Agile Testing places on people can also be its downfall. If Agile testers are excluded from the team they need to be closely integrated with, they're rendered useless. If a single skilled Agile tester leaves, it can prove to be a major setback for the development of the product.

Finally, since everyone in the team performs testing, the muddled hierarchy could lead to confusion and conflict. Methodologies like Scrum attempt to circumvent this by having "scrum masters," but this has the potential to fall back into a more traditional method rather than staying truly Agile.

---

### Agile testing strategy

With dedication, each of these pitfalls can be overcome and the three powerful benefits experienced. The first step towards successful Agile Testing is determining when Agile Testing shouldn't be used. Blind adoption of Agile Testing can result in a weak, crash-prone product.

Here are a few cases in which Agile may not be the best way to test:

1. When the project scope is crystal clear and very unlikely to change
2. When the project is governed by a single product owner or stakeholder with minimal requirements
3. When the people on your team lack the skills necessary to perform Agile Testing
4. When the customer insists on using a traditional waterfall approach to testing

Once you've determined that Agile Testing will benefit your team, your product and your customers, you should spend as much time as necessary to pick the right methodology and create a process for testing using the four-quadrant model.



To counteract the possibility of testers' exclusion, testers should work in as close physical proximity to the developers as possible. They should meet with them often to see what they're currently working on and to give them a chance to review the tests that have been developed. Taking an iterative approach here and in the testing process itself can help connect the teams early and help with later collaboration.

Testers can open doors for themselves by providing helpful feedback based on interactions with both developers and customers. In short, they should make themselves indispensable to developers to be able to perform their job well.

### **Make Agile Testing successful**

The greatest thing that can be done to guarantee the success of Agile Testing for a product is to hire people who have the essential characteristics of an Agile tester and build a culture of self-organization and independent thinking in the entire organization.

That environment will naturally result in "stable infra" without sacrificing speed, resulting in happier workers delivering a better, more valuable product - faster - to a satisfied customer.



---

# Chapter 9 - Usability Testing



## Integrate Usability Testing into your SDLC

As part of Functional Testing or conducted standalone, Usability Testing provides valuable real world feedback from real world users on real devices - whether localized or non-localized.

### Ensure better engagement and adoption through Usability Testing

Experienced testers walk in the footsteps of your users, providing feedback that can then be translated into future product improvements. Elements including linguistics, content and functionality are reviewed, delivering a holistic analysis of your app.

Usability Testing is conducted with the following in mind:

- **Support feature development.** Confirm that new features feel intuitive and are engaging for customers before launch.
- **New market localization.** Determine cultural nuances of new markets.
- **Remove language barriers.** Verify content and app flows for local users with real, local users.
- **Tailored flexibility.** Receive feedback from multiple markets or even just one.



---

# Final Thoughts



## Developing your own QA testing strategies

An efficient and effective QA testing strategy will readily provide the information needed by your design and development teams to produce a quality app. Remember that software quality doesn't depend on testing but instead on the outcome of your QA tests and how you use this data.

When designing your approach to QA testing, remember to adapt it to the product you're developing.

### QA testing best practices:

- 1. Test one thing at a time:** Tests should maintain clear objectives, focus on features or examine your interface or security.
- 2. Understand the types of testing on offer:** Remember to learn the nuances between different types of tests to understand their business impact.
- 3. Use Regression Tests:** Testing a main feature once isn't enough. New additions to the code repository can interfere with features that previously passed tests.
- 4. Report and track bugs:** Determine how bugs will be reported and what kind of data is needed. Will you use an open-source bug tracking tool or build one that's specifically suited to your workflow?
- 5. Leverage analytics:** Determine which QA metrics to track. Keep records of every test conducted and use this data to determine where bugs are likely to occur. This data will help you to develop new tests that address problem areas.
- 6. Choose the right environment for tests:** Try covering a wide range of scenarios, including different devices, OS, and user profiles.
- 7. Use unit and integration tests:** Unit Testing isolates each component of your app, while Integration Tests assess how well each subsystem works. Run unit tests in parallel to save time, but don't move onto integration tests until you've ensured individual components work as they should.
- 8. Don't neglect the UI:** Leverage functional tests performed by human testers to perform end-to-end scenarios and establish a feel for the UI of the app. It might be best to wait until issues detected during unit and integration tests are resolved.



## Achieving high quality and speed

Whether you're building a web application, downloadable software or an API, high quality and speed should remain at the top of your priority list. It's essential to regularly review your QA testing process for efficiency as you move through the different cycles of your project.

Remember to align quality objectives with user expectations and employ them when writing test cases. Working with a clear set of quality objectives will help developers, testers and designers understand what's expected of them better and foster an environment where everyone owns quality.

Creating such a culture of quality across your business will not only increase confidence in upcoming releases and drive more return on investment, but it will also inevitably alleviate a few headaches in the process.





# Global App Testing



[www.globalapptesting.com](http://www.globalapptesting.com)



[info@globalapptesting.com](mailto:info@globalapptesting.com)



UK +44 (0) 330 808 0106

US +1-800-461-2670



[linkedin.com/company/global-app-testing](https://linkedin.com/company/global-app-testing)



[twitter.com/qaops](https://twitter.com/qaops)



[facebook.com/globalapptesting](https://facebook.com/globalapptesting)