



Testing in Regulated Industries:

A Tactical Guide for QA Managers

Index

Introduction	03
Unique Challenges of Testing in Regulated Industries	04
Test “Quality Records”	07
Compliance and Security	12
Environment Test Data Management	17
Disaster Recovery	23
Managing Distributed Teams	29
Self-Assessments for Continuous Improvement	33
About TestRail	37



Introduction



Welcome to “Testing in Regulated Industries: A Tactical Guide for QA Managers.” This essential guide is crafted for QA professionals navigating the complex landscape of regulated sectors, where precision and adherence to strict compliance, security, and data integrity standards are critical to the success of the business.

The regulated industry landscape presents a unique set of challenges for QA teams. QA managers in these environments must grapple with a complex web of compliance, security, and data integrity standards and a significantly lower risk tolerance. These rigorous demands mean that achieving a high degree of quality in these sectors could be much harder to achieve compared to other, more lenient industries.

In our 2023 annual TestRail user survey, about a third of our respondents reported working in a regulated industry such as finance, healthcare, energy, transportation, government, or telecommunications. Within those industries, about half of our respondents work with a QA team of ten dedicated testers or fewer and deploy new releases between one and four times per month.

The priorities and challenges in these regulated industries aren’t so different from any other QA team—respondents reported a need to improve efficiency, reduce bugs in production, and automate more tests, as well as struggles with not having enough time and resources dedicated to QA. However, these familiar challenges are only intensified when combined with the strict compliance requirements inherent to working in such regulated industries.

This condensed guide highlights the importance of maintaining detailed quality records, innovative testing strategies to protect sensitive information, and the criticality of disaster recovery in ensuring the continuous operation of essential services. Moreover, it addresses the complexities of managing distributed QA teams, emphasizing the need for effective coordination and compliance.

This material’s objective is to help QA professionals go beyond meeting regulatory demands and empower teams to drive continuous improvement and achieve excellence in software quality assurance. With that said, let’s dive into this material designed to help you master the unique challenges of QA in regulated industries, equipped with strategies and tools to excel.



SECTION 01

The Unique Challenges of Testing in Regulated Industries

Unique Challenges of Testing in Regulated Industries



Any QA team has plenty of moving parts to keep track of. Still, the compliance, security, and data integrity standards inherent to highly regulated industries pose a unique set of challenges for the QA manager. Below are the most common challenges experienced by QA teams across the various sectors with high levels of regulation:



Maintaining “Quality Records”

Keeping detailed, traceable records of all testing activities and who performed them is an important component of testing across all industries, especially those subject to audit. You must be able to select a random test at any point in time and see why it was run, who ran it, and what requirement it was linked to—and if it failed, why it failed, and its associated defect. For teams of any significant size, maintaining these records is both difficult to achieve and time-consuming without the aid of tools specifically designed for this purpose.



Compliance and Security

When your software handles sensitive data, such as users’ healthcare information or financial details, there is no room for error regarding the security of your application and your testing platform. Additionally, each industry has regulations that must be adhered to, meaning that tools and approaches that work for one company may not apply to another. These strict guidelines can significantly impact a QA team’s productivity and ability to keep pace with the demands of the company and customers.



Test Data Management

Due to such privacy and security regulations, testers within regulated industries often cannot test with actual production data. This requires using approaches such as data de-identification and property-based testing to test in a production-like environment without risking the exposure of sensitive data. This adds an additional layer of complexity for software testers, and any violations of data privacy laws can come with steep fines and legal consequences.

Unique Challenges of Testing in Regulated Industries



Disaster Recovery

Software used in regulated industries often has guaranteed uptime for its end users, who can't risk any downtime in the systems used to run things like healthcare equipment and energy production. This means that it is crucial for QA and Site Reliability Engineers to build and maintain a disaster recovery plan for all possible emergencies. However, building a disaster recovery plan is a manual and time-consuming process, and plans must be validated and brought up-to-date quarterly. QA teams often struggle with efficiently storing and tracking their disaster recovery plans.



Managing Distributed Teams

It's becoming increasingly common for QA teams to work remotely, which comes with its own considerations. Adding the challenges of working in a regulated industry to the mix can make managing a remote QA team even more difficult. Managing team members in different time zones can make tracking test progress challenging, and having testers distributed across international borders can add additional layers of regulatory compliance that your entire team must abide by.

However, these challenges are manageable!

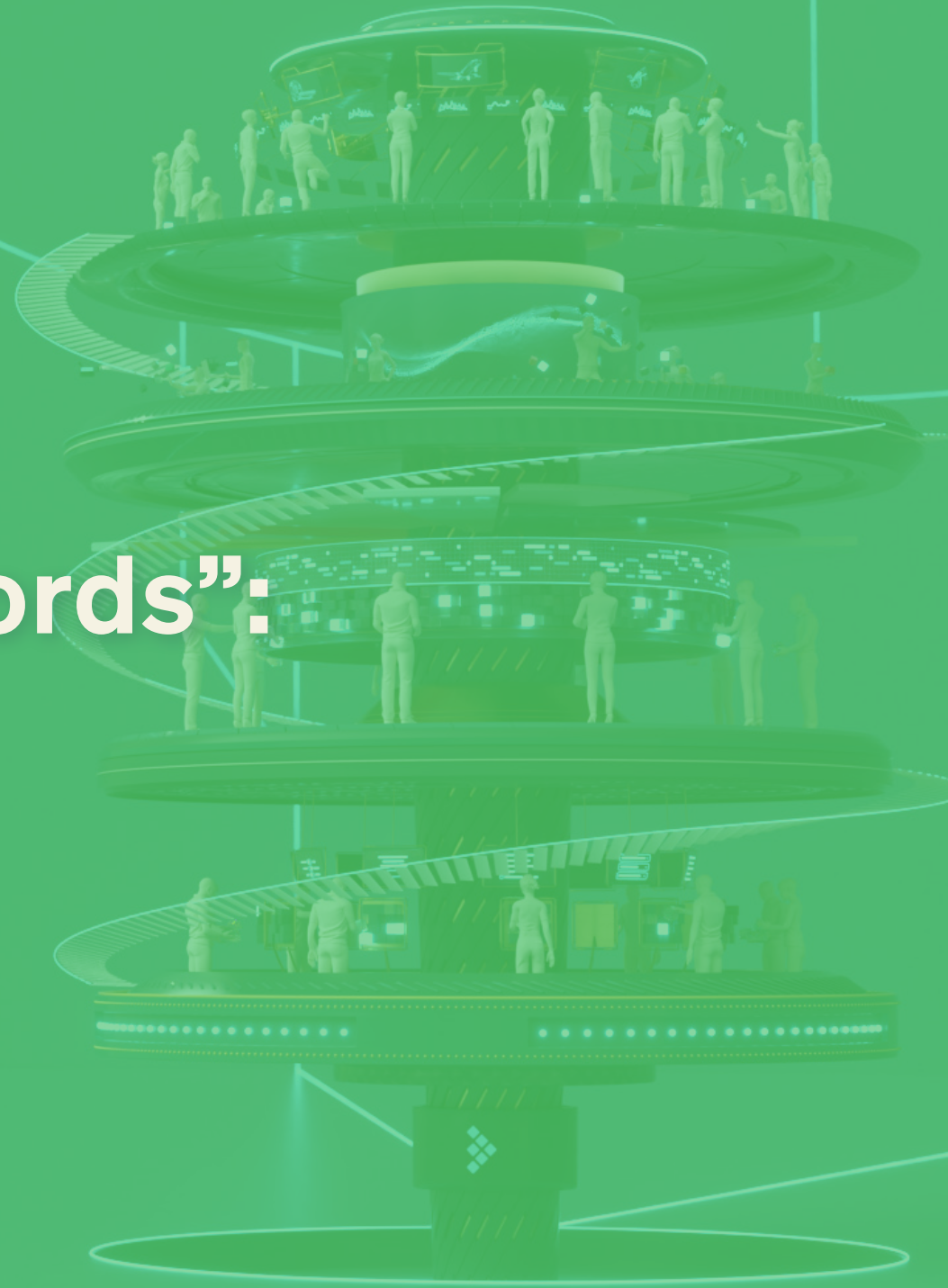
In this ebook, we've outlined some strategies and approaches to conquering testing in highly regulated environments—as well as a plan for performing self-assessments and generating continuous improvement.

This guide addresses these challenges head-on and introduces [TestRail Enterprise](#) as a key tool in the QA manager's arsenal. Designed to meet the specific needs of regulated industries, TestRail Enterprise supports critical functions such as granular user access control, test case review and approval workflows, and customizable audit logging. It facilitates a scalable, audit-ready QA process that not only meets regulatory demands but also drives continuous improvement and excellence in software quality assurance.

SECTION 02

Test “Quality Records”:

Who, What, Where, and Why?





Test “Quality Records”

QA managers in regulated industries must always be ready for an audit. At any given time, you must be able to pull any test from your repository and show when it was executed, who ran it, why it passed or failed, and what software requirements and defects it is linked to. Additionally, you must be able to show which user created, updated, or deleted any given entity within your test management platform at any time. It’s imperative that QA managers maintain complete and comprehensive quality records by intelligently tracking, documenting, and managing all of their testing artifacts. How can you record and maintain all of that information?

User Access Control

If you have tens or even hundreds of engineers in your organization—all with access to your test management systems and the ability to edit test cases and results—you can easily see where this might get messy. Test cases and results start changing for no apparent reason, and you can’t tell who edited them or why. This is a headache for your QA team, and also a liability come audit time.

The best way to start maintaining solid quality records is to implement a “least privileges” approach to user access in your test management system. This means that everyone on your team has the exact privileges they need to do their jobs without access to capabilities, projects, or systems that they do not truly need.

If you use a test management platform such as TestRail, you can edit each user’s access and permissions as well as create custom roles. You can set project-specific roles for each user, ensuring that testers don’t have access to more projects than they need for their unique jobs.

As your QA program scales, you may find it helpful to invest in a solution such as TestRail Enterprise, which allows project-level administration. Project-level administration enables you to grant administrative privileges to users on a project-by-project basis—ensuring that a team member who needs administrative access to Project A doesn’t accidentally edit test results in Project B.

Edit Role

Name *
Manager
Ex: Lead, Tester or Guest

This role is the default role
The default role is the pre-selected role for new users and the fallback role for users when you delete other roles. Only one role can be selected as the default.

Permissions	Add/Edit	Delete	Close	Modify
Attachments	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		
Cases & Sections	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		
Configurations	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		
Milestones	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		
Runs & Plans	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
Runs & Plans (Closed)		<input checked="" type="checkbox"/>		
Reports	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		
Reports (Scheduled)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		
Suites	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		
Test Results	<input checked="" type="checkbox"/>			<input checked="" type="checkbox"/>

Enable project administration for this role
Project administration settings will only apply to users with this role set as their Global Role. Enabling this setting will provide access to a subset of the Administration area. Any additional permissions will apply only to projects which are assigned to each user.

Project Administration Permissions	Add/Edit	Delete
Projects	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Case Fields & Templates	<input checked="" type="checkbox"/>	
User Variables	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Users	<input checked="" type="checkbox"/>	

Test “Quality Records”



Traceability

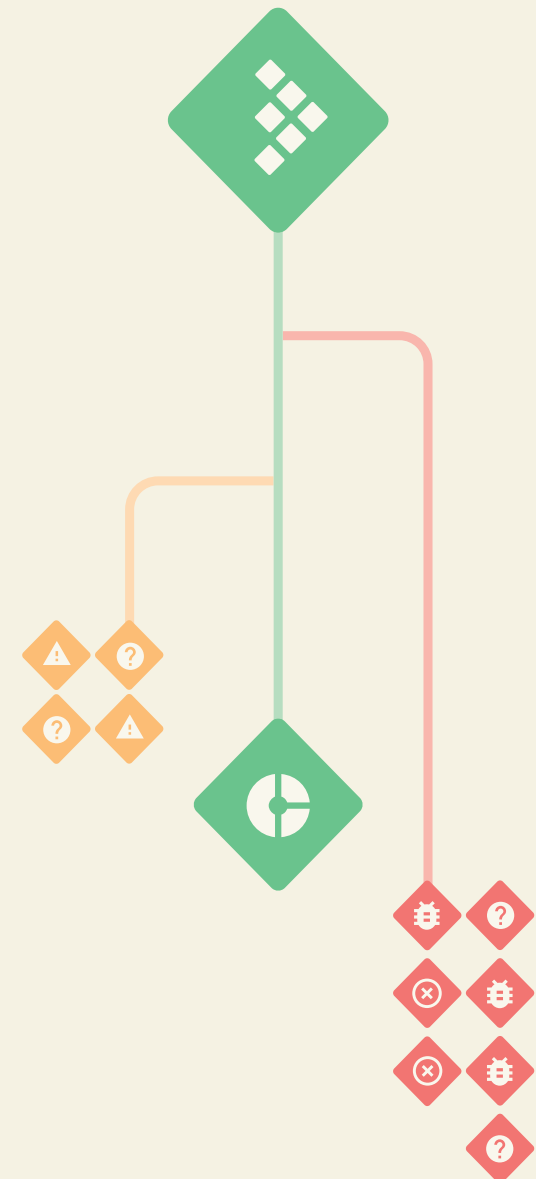
Traceability is vital to the QA process—without linking tests to requirements and requirements to defects, you have no trail demonstrating that all necessary tests have been performed and all bugs have been addressed. It’s also easy to let your traceability efforts slip a little when things get busy. Who has time to link requirements and defects when there are fires to put out? However, traceability is a priority in regulated industries where a bug could be a matter of life or death (or at least a failed audit).

Traceable and timestamped execution data and results should be available for every test in your repository—including past releases and versions. Each failed test should also be linked to a defect artifact in a defect tracking tool like Jira, clearly establishing the trail between requirement -> failed test -> defect -> passed test -> release.

It’s essential that you can pick a random test from your repository at any point and see when it was run, who ran it, what release it’s related to, what requirement it’s testing, why it passed or failed, and what defects are linked to its results. That may seem like a lot of information to record for every test execution, but a centralized test management system such as TestRail does most of the work for you.

TestRail automatically records test execution data (such as the timestamp and associated tester) whenever a test is run. It can integrate with nearly any defect tracking tool to allow testers to link requirements and push bugs without leaving the platform. Some integrations, like Jira, support two-way visibility, enabling your QA and dev teams to see TestRail and Jira data from their respective tools without logging into the other platform.

After your requirements and defects are linked, you can simply run a [Summary for References \(Defects\) report](#) to see the traceability between references, test cases, test results, and defects at a glance.



Test “Quality Records”



User Audit Logging

Just maintaining your test execution data likely won't be enough when audit time comes around—you may also need to log every action taken in your test management system by every user, including changes to settings and the creation or deletion of any entity within the platform. This information should also be traceable to projects, versions, and releases as applicable.

In addition to being another tool in the audit-readiness toolkit, user audit logging makes it easier to keep track of changes, especially across large or distributed QA teams. Keeping historical records of user actions eliminates questions like “Who changed this?” “Who ran this?” “Where did that go?” A detailed user audit log answers those questions quickly, establishes responsibility, and allows your testing to continue unimpeded.

Some test management solutions, especially those geared toward enterprise organizations, offer built-in audit logging capabilities. TestRail Enterprise allows administrators to customize the level of detail captured in their audit logs and specify the maximum number of records and days to store. These logs can then be exported for further retention and analysis.

The screenshot shows the 'Site Settings' page in TestRail, with the 'AUDITING' tab selected. Underneath, the 'LOG' sub-tab is active, displaying a table of audit records. A filter is set to 'Date: Last month'. The table has columns for Date, Project Name, Entity Type, Entity Id, Entity Name, Action, Author, and Mode. Five records are visible, all dated 28th Dec 2023 18:27:43, performed by Chris Faraglia. The actions include 'Delete' for 'Automation Demo - Merge Develop Smoke Tests' and 'Verify TestRail Website Navigation'.

Date	Project Name	Entity Type	Entity Id	Entity Name	Action	Author	Mode
28th Dec 2023 18:27:43	TRPD Test Demo	Test Run	529	Automation Demo - Merge Develop Smoke Tests	Delete	Chris Faraglia	UI
28th Dec 2023 18:27:43	TRPD Test Demo	Test	7988	Verify TestRail Website Navigation	Delete	Chris Faraglia	UI
28th Dec 2023 18:27:40	TRPD Test Demo	Test Run	530	Automation Demo - Merge Develop Smoke Tests	Delete	Chris Faraglia	UI
28th Dec 2023 18:27:40	TRPD Test Demo	Test	7989	Verify TestRail Website Navigation	Delete	Chris Faraglia	UI

[TestRail Enterprise's audit logging system](#) helps administrators track changes across the various entities within their TestRail instance. With audit logging enabled administrators can track every entity in their installation.



Test “Quality Records”

Challenges

- ⚠ Tests are getting changed, moved, or deleted, and I don't know how or why.
- ⚠ I have failed tests that aren't linked to defects.
- ⚠ I don't know who is changing settings, editing test cases, or creating, updating, or deleting entities in our test management system.

Solutions

- ✅ Ensure every testing team member only has access to the tests, projects, and systems they need to do their job. Use a platform like TestRail to customize user access and roles globally or on a project-by-project basis. know how or why.
- ✅ Establish traceability and build an auditable record between requirement and release using a test management platform that integrates with your defect tracking tool. The TestRail-Jira integration provides a simple workflow for testers to push bugs and link defects directly from failed tests.
- ✅ Consider recording a user audit log. Platforms like TestRail Enterprise can log every action taken on any entity in the system, removing the mystery of “Who did that?”

TestRail helps us provide evidence for what we did. The accuracy of the TestRail reports that we give to management is worth a lot. What's more accurate and confident than being able to argue with exact data on what you've done and what you have not?

Torsten Zelger,
Test Manager



SECTION 03

Compliance and Security:

Maintaining Security Posture



Compliance and Security: Maintaining Security Posture



Security is top-of-mind for all organizations, but when your systems are responsible for processing sensitive information, it's necessary to ensure that security mechanisms and access controls are valid and functional.

Access Control

The [Open Worldwide Application Security Project \(OWASP\)](#)—a nonprofit dedicated to improving software security—regularly publishes a report of what they have identified as the top ten most serious threats to software security. This report raises awareness of the vulnerabilities that are occurring most frequently and pose the biggest risk to developers. In OWASP's most recently published Top 10 report, [“Broken Access Control” ranked #1.](#)

Maintaining good access control means ensuring that all users of your software can only do what they are specifically given permission to do. [The OWASP Top 10 2021 states](#), “Failures typically lead to unauthorized information disclosure, modification, or destruction of all data or performing a business function outside the user's limits.”

Broken access control can manifest in scenarios such as:



A user is able to log in with invalid credentials successfully



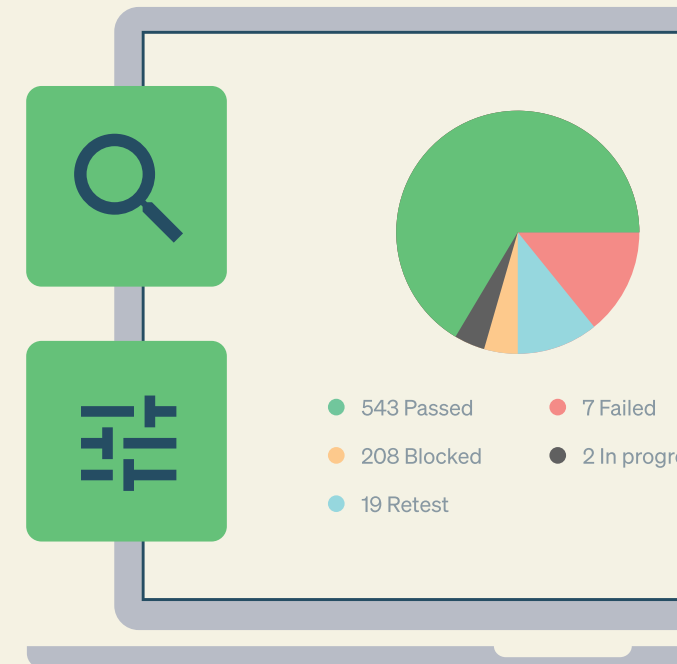
A read-only user is able to edit data



A non-admin user is able to perform admin-only actions



A user is able to access restricted data by bypassing the front end via an API call

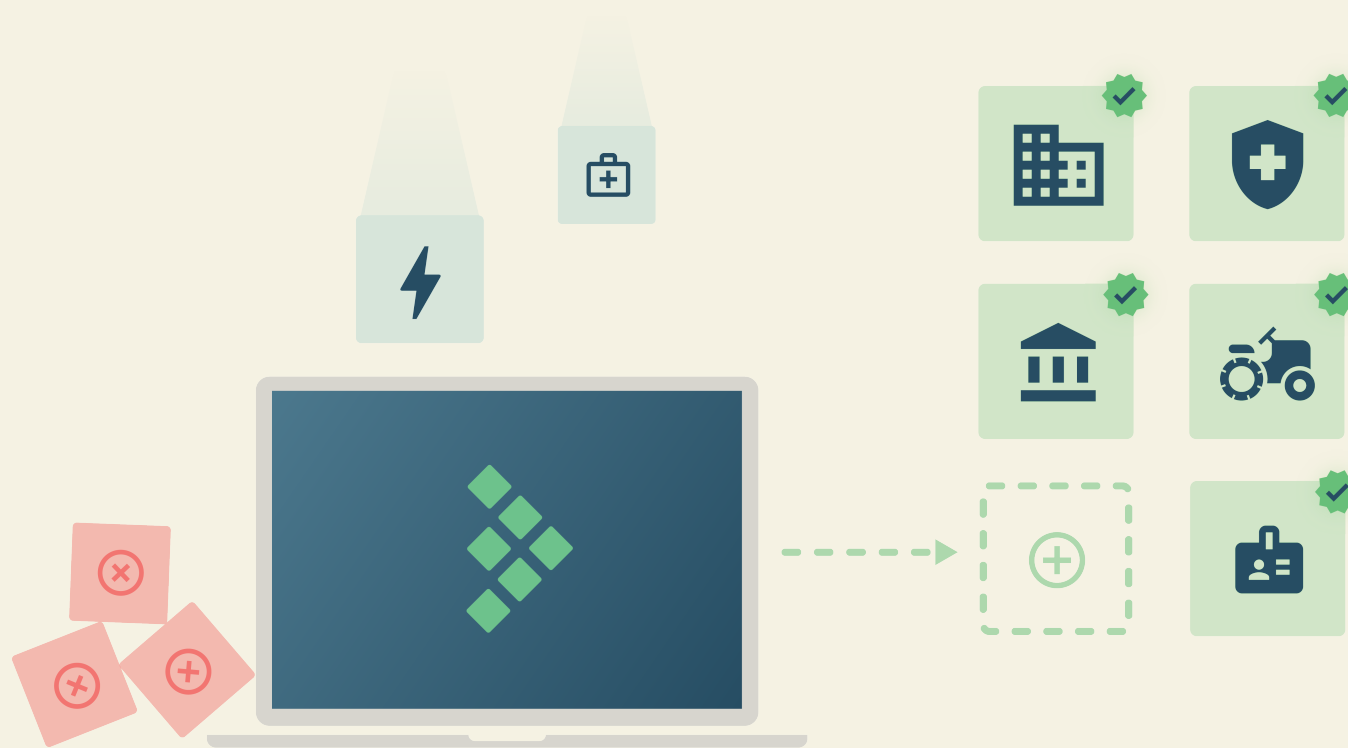


Compliance and Security



It's easy to see how broken access control can quickly compound into a lethal security threat, opening the door for unauthorized users to be able to view, edit, or delete your sensitive data. This is why QA teams must perform functional testing that covers access control across your entire system—both the front end and the back end, along with any cloud infrastructure components.

For example, a testing team may want to validate that invalid login credentials will prevent a successful login. They should test this both via the front-end web app and through a back-end method that does not include the user interface, such as a REST API that submits the user credentials for authentication and authorization. This ensures that an API invocation won't uncover data that the front-end UI was obscuring and validates that your data is secure no matter how a user attempts to access it.



Test Management Platform Security



In addition to security threats to the software under development, QA teams should also be vigilant about security threats to their test management system. Even if you don't test using actual production data, your test management platform still holds a wealth of information on your product and testing efforts that not everyone needs access to.



Role-Based Access

As mentioned above, under “Test quality records,” maintaining a strict “least permissions” policy for users of your test management platform is key to a secure and audit-ready QA team. In addition to making it much easier to track who changed or deleted what, ensuring each user only has the permissions they need to do their job means there are far fewer opportunities for a mishap with your data to occur—whether accidental or malicious.

You should only have users with administrative privileges on any system, and your test management platform is no exception. Admin access doesn't have to give admin users free rein over everything, either—solutions such as TestRail Enterprise offer project-level administration capabilities that allow you granular project-by-project access control capabilities.



IP Whitelisting

Sensitive data can also be exposed when users access your test management platform on an insecure network such as a mobile hotspot or public wi-fi. Such networks make it easier for bad actors to gain access to test management data or credentials and compromise test quality records.

You can mitigate this concern by [enacting IP whitelisting](#) on your test management platform. This will ensure that only devices connected to approved networks—such as your office internet—can access your systems. IP whitelisting prevents authorized users from accessing your data on insecure networks and malicious access attempts from parties outside of your organization.



Identity Authentication

Even once your user access roles are locked down and your approved IPs are whitelisted, there is still a security threat in your user accounts. Someone other than the user in question could get access to an administrator's login credentials, or there could be a data breach that exposes your passwords to anyone who cares to log in and take a look.

An authentication framework, such as SAML, OAuth 2.0, or OpenID Connect, can mitigate this by providing a robust standard that implements authentication and authorization principles that will best ensure the security of your platform.

Not every test management platform will support authentication frameworks—look for enterprise solutions such as TestRail Enterprise, which has built-in support for SAML single sign-on (SSO), OAuth, and OpenID Connect.

Compliance and Security



Challenges

- ⚠️ There are too many users with administrative privileges on my test management platform, making it hard to ensure the security of our data.
- ⚠️ I have testers who work remotely, and I am concerned about the security of the network connections they may be using to access test data.
- ⚠️ I am concerned about the security risks involved with storing and managing the login credentials to my test management system.

Solutions

- ✅ Use your test management tool's permissions and privileges settings to ensure that each tester has only the access they need to do their work. Consider a platform like TestRail Enterprise for more granular control, like project-level administration.
- ✅ Enact IP whitelisting on your test management platform to ensure users can only access your data when connected to secure, pre-approved networks.
- ✅ Consider a solution like TestRail Enterprise that offers support for authentication frameworks such as SAML single sign-on, OAuth 2.0, and OpenID Connect.

We have hundreds of thousands of customers using our invoicing tool. Because we deal with multiple countries, there are multiple sets of laws and legislation. This means that there is an even greater need for organized testing to be able to focus on the legislative differences from country to country.

*Jiří Malý,
Senior Software Tester*



SECTION 04

Environment Test Data Management:


What to do with restricted test data





Environment Test Data Management

It's important to test your software in an environment as close to production as possible—from the operating systems and devices used to run the software to the data being logged in it. However, this isn't always possible in industries where production data is highly sensitive information, such as bank account numbers or medical records. Luckily, there are ways to test systems critical to highly regulated industries without risking the security of sensitive data.

ID Number	First Name	Last Name	SSN
 42315678	John	Smith	908-765-1234
XXXXXXXX	John	Smith	XXX-XXX-XXXX

Data De-Identification and Masking

Data de-identification and data masking are terms used to describe the process of altering data so that it cannot be directly used to identify the individual it is associated with. This involves removing, replacing, or otherwise obscuring certain personal identifiers dictated through your regulatory body or risk-based statistical analysis.

Data de-identification is [expressly governed under HIPAA](#), so many data de-identification tools and resources will be relevant to the healthcare industry—but the benefits apply to any industry that needs to sever their test data from the identities of its original subjects.

Environment Test Data Management



HIPAA describes two different methods of data de-identification: Safe Harbor and Expert Determination.

Safe Harbor requires removing [eighteen specific categories](#) of information from data, including but not limited to names, dates, geographic data, social security numbers, phone numbers, account/license/record numbers, biometric information, and IP addresses.

However, sometimes removing all eighteen Safe Harbor identifiers [renders the dataset useless for its needed purposes](#). If a dataset must include one or more Safe Harbor identifiers, an organization can also employ Expert Determination, in which an expert uses a risk-based approach to certify that the likelihood of a person being identified from the data is sufficiently low.

Many tools and platforms are available to help de-identify data and ensure compliance—[learn more about the landscape of data de-identification tools on G2](#).

Safe Harbor

Name: *****

Date: **-**-****

Geographic Location: *****, **

Social Security: ***-**-****

Phone Numbers: ***-***-****

IP Address: ****-**-*****

Expert Determination

Name: *****

Date: 03-24-1995

Phone Numbers: ***-***-****

Geographic Location: **Seattle, Wa**



Environment Test Data Management

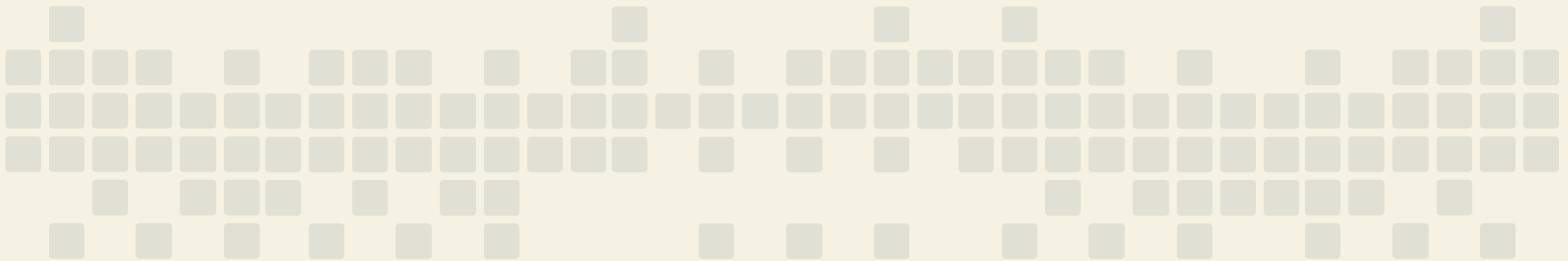


Property-Based Testing

Another method of testing software utilized in highly regulated environments is property-based testing. This method of testing dynamically generates realistic data to test the property of a given field without using or storing any actual production data.

Additionally, property-based testing can [quickly test an immense range of inputs in a single test](#)—making it easy to check for edge cases without having to write a different unit test for every potential input.

There are a variety of code libraries available that allow you to define and generate property-based test data for use in your test scripts. Popular options include [Faker](#) (JavaScript), [Hypothesis](#) (Python), [RapidCheck](#) (C++), and [ScalaCheck](#) (Scala).





Environment Test Data Management

Test Data Parameterization

But what happens if property-based testing isn't suited to your needs and you instead have a library of de-identified data values to test? If creating a separate unit test for each input field and data value seems daunting, test data parameterization can help streamline the process.

Test data parameterization allows you to [run the same test with multiple configurations of different values](#). This type of data-driven testing allows you to test your application's basic logic or functionality with multiple inputs without copying and maintaining duplicate test cases.

TestRail Enterprise has a built-in test data parameterization feature, allowing you to define up to 500 variables (such as input fields) and 100 different datasets for each variable. Variables can then be added to test cases, allowing one test case to cover all potential datasets in that variable.

Test data is specific to each project, allowing you immense flexibility in your test cases and greater coverage with fewer cases to manage. Learn more about how to utilize this feature in the [TestRail Enterprise User Guide](#).

Test Data ← Back

Define and use variables for your Test Cases that you can later pass to your Test Runs. **Choose your way to go:** Import your own CSV ([check out our Test Data Guide here](#)) or build your own dataset from scratch through the table below. You'll be able to upload your CSV to update existing data and add new variables or datasets.

Search by Variable, Dataset name, Value... Edit Delete

	A	B	C	D	E
1	Variable ↑	Default What's this?	portugal_local	poland_local	india_local
2	zip_code	DC 20500	1300-004	03-936	1234567
3	street_address	1602 Pennsylvania Avenue NW,	Praça Afonso de Albuquerque	ul. Łaskarza Andrzeja 135	123123 street avenue
4	phone	+1480-238-8619	+351 231 000 000	+48 72 906 80 43	...
5	name	Matthew Caponigro	José Domingues	Świętosława Zarzecka	...
6	currency	\$	€	zł	...
7	country	US	Portugal	Poland	...

TestRail Enterprise's test data parameterization feature allows you to test the same basic logic or functionality in your application with multiple inputs, without having to copy and maintain duplicate test cases.



Environment Test Data Management

Challenges

- ⚠️ My production data contains highly sensitive information, such as health records or bank account numbers, and I cannot perform tests using it as-is.
- ⚠️ I don't have to test using actual production data, but I need a way to generate large amounts of realistic data to validate proper system functionality.
- ⚠️ I have a huge library of data to test, but don't want to create a separate unit test for every possible data value in every input field.

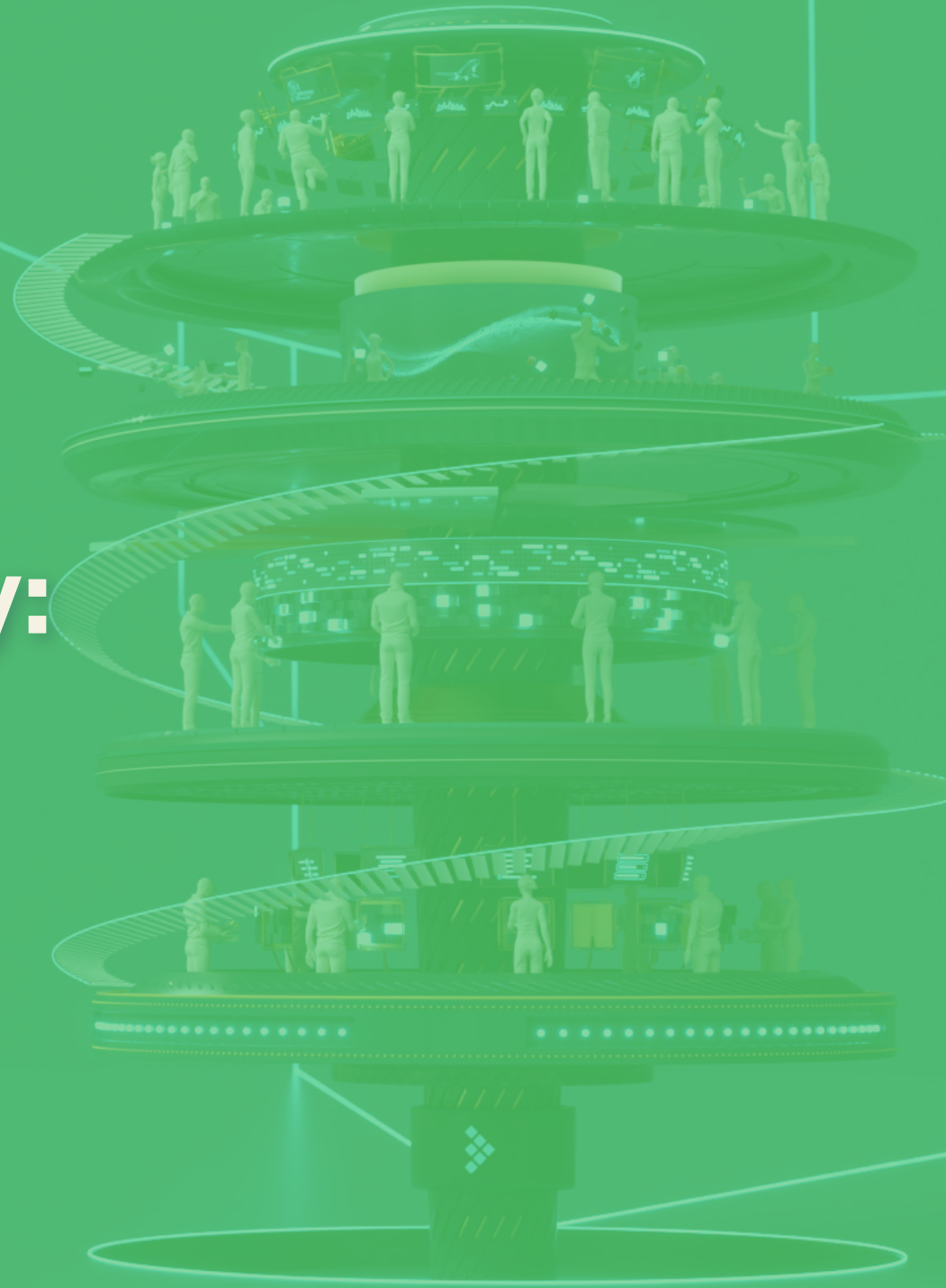
Solutions

- ✅ Consider a method of data de-identification or masking that protects the identities tied to the data while allowing you to test in an environment as close to production as possible. Your industry will have its own guidelines on what needs to be obscured to render the data secure.
- ✅ Consider a property-based testing tool that will allow you to define and generate realistic test data that validates the functionality of given fields.
- ✅ Consider a test management tool supporting test data parameterization, such as TestRail Enterprise. This allows you to create one test case with dynamic test data fields for different configurations of inputs rather than multiple test cases for every possible input.

SECTION 05

Disaster Recovery:

Hope for the best,
but be ready for the worst



Disaster Recovery

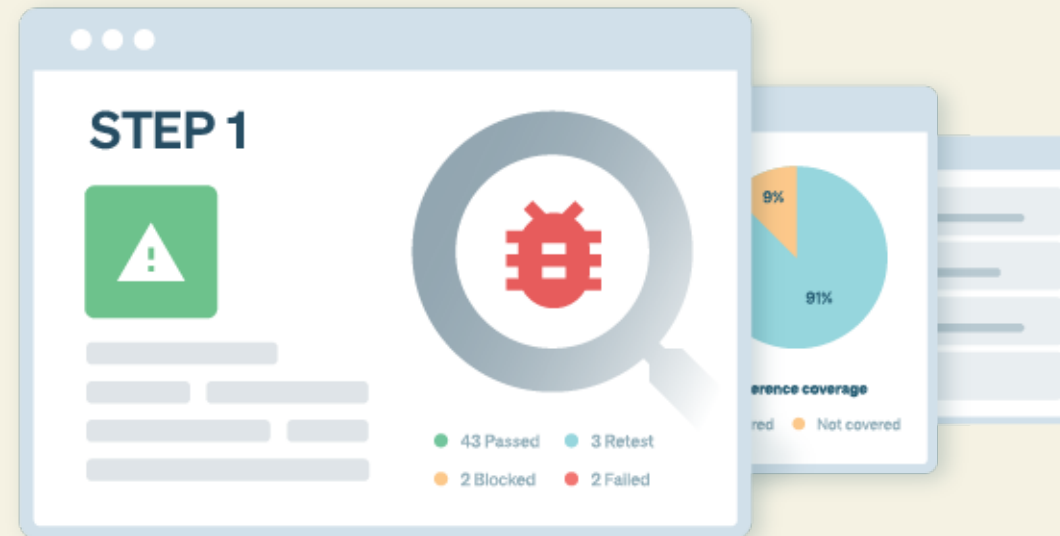


You can't afford downtime when your software is integral to systems that facilitate processes like medical care, banking transactions, or nuclear energy. If you work in such an industry, you likely guarantee your customers 24/7 uptime—and they're counting on that.

So, what do you do when disaster inevitably strikes? You plan for it.

Creating, maintaining, and regularly testing a disaster recovery plan is good practice for everybody, but especially critical to those in regulated industries. However, it can be a manual and time-consuming process, and involves collaboration between QA and Site Reliability Engineers (SRE). Additionally, your disaster recovery plan itself must be regularly tested and updated.

With so many demands on QA, it can be easy to let efforts like disaster recovery planning slip—but by building a thorough plan and utilizing test management tools to help track validation progress, disaster planning doesn't have to be a burden.



Disaster Recovery



How to Create a Disaster Recovery Plan

It's time to break out the pen and paper (or Google Docs) because creating a disaster recovery plan is one of those tasks that can't be automated.

Creating a disaster recovery plan is a collaborative effort between QA and SRE. It can be something as simple as a flowchart, or a comprehensive guide with dozens of different processes, checklists, and procedures. Regardless of how extensive your plan is, it should consist of the following elements to be effective:

- ✓ **Clear Objectives:** Define Recovery Time Objectives (RTO), Recovery Point Objectives (RPO), and their corresponding metrics.
- ✓ **Comprehensive Inventory:** List critical assets such as software, hardware, and data that could be impacted.
- ✓ **Roles and Responsibilities:** Clearly define who is impacted and what their key responsibilities include.
- ✓ **Communication Plan:** Outline who is responsible for keeping internal and external stakeholders informed.
- ✓ **Backup Strategies:** Include details on backup schedules, methods, and testing frequency to ensure data integrity.
- ✓ **Detailed Recovery Procedures:** Define step-by-step recovery actions.
- ✓ **Vendor Information:** Provide a contact list for all vendors.
- ✓ **Validation Schedule:** Ensure plan effectiveness with regular testing and updates.
- ✓ **Employee Training:** Educate the team on their roles.



Disaster Recovery

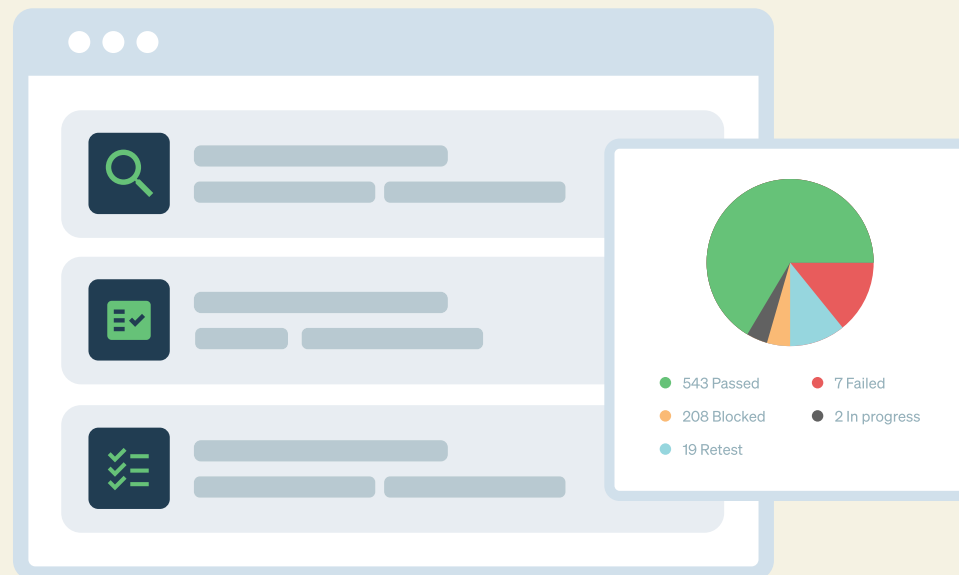
It's important to also plan for failovers, where the data from an unexpectedly downed server is seamlessly transferred to another one. When performed quickly (ideally automatically), this allows services to continue running uninterrupted. For example, if an Amazon Web Services (AWS) user opts to leverage [AWS's Multi-AZ feature](#), their database automatically fails over to a standby instance without manual intervention—ensuring no interruption of service or data loss.

If you write down every possible disaster that could befall your software or systems and create a plan for recovering from them using the “if X happens, we do Y, and validate it using Z” template, that's a solid disaster recovery plan.

Testing a Disaster Recovery Plan

Disaster recovery plans are not just “set it and forget it.” In addition to collaborating on their creation, QA and SRE must regularly come together to validate their disaster recovery and failover plans. Such validation is frequently executed quarterly to ensure adherence across common compliance frameworks and requirements.

This also cannot be automated, as QA and SRE must manually review and validate their disaster plans to ensure they are still functional, reliable, and applicable to the current state of their software and the most up-to-date threats. A test management platform optimized for manual test cases is a great place to track such efforts. You can use the “test plan” feature in TestRail to build disaster recovery plans and track your quarterly validation progress.



Disaster Recovery



Challenges

- ⚠️ My team has no formal plan for addressing and recovering from disasters such as accidental deletion of data or server outages.
- ⚠️ My team made a disaster recovery plan, but we don't know how to maintain it.
- ⚠️ I don't know where to store my team's disaster recovery plan or track efforts to validate and update it.

Solutions

- ✅ Have QA and SRE work together to map out all potential disasters, and create a "if X happens, we do Y, and validate it using Z" style plan of approach for each.
- ✅ Disaster recovery validation is typically performed quarterly. Arrange a standing time for QA and SRE to manually review and validate their disaster plans and make necessary updates.
- ✅ Consider a test management platform that is optimized for manual test cases, like TestRail. The "test plan" feature in TestRail can be used to build disaster recovery plans as well as track validation progress.

The hierarchical structure [in TestRail] helps us highlight where we are missing tests and where to put our attention. It's a quick check to say, 'Have we got all the screens? Have we got all of the workflows? Within the screen, have we got all the right workflows?' It's a useful organization tool for us to see the test coverage that we do have.

*Moira Tuffs,
Software QA Manager*



SECTION 06

Managing Distributed Teams:

Keep your QA team on track and compliant across the globe



Managing Distributed Teams



Some or all of your QA team likely works remotely or from a different time zone—as of 2023, [over 40% of the full-time workforce works under a fully remote or hybrid model](#) (combining home and in-office time).

Additionally, a [2023 study of remote work by Buffer](#) found that 74% of respondents work in a company that operates in multiple time zones, and 62% reported that people on their immediate team were distributed across time zones.

While remote work has many benefits, having a distributed QA team can certainly make it more challenging to keep your testing coordinated and compliant. Everything mentioned in this ebook can be applied to your distributed QA team, but there are also a few special considerations for the remote QA manager in a regulated industry.

Managing Distributed Teams



Visibility, organization, and centralization

When your team members are in different time zones, you can't count on everyone being online simultaneously to answer questions and clear up any confusion. Additionally, distributed teams may have team members who jump in to assist with testing as needed without being full-time members of your project.

This means that your project organization, test documentation, testing status, review and approval workflows, and version history must be meticulously managed and easily accessible to anyone who needs it. A testing “center of excellence”—a centralized, single source of testing truth—is vital to anyone who manages distributed QA teams.

A testing center of excellence often takes the form of a dedicated test management platform like TestRail. Here are some features you can find in TestRail that will help you maintain visibility, clarity, and compliance across your distributed QA team:

- **Test assignments and to-do lists:** Assign test cases to individual testers, ensuring each team member has a personalized to-do list (and no questions about what to do next)
- **Live dashboards:** See the progress of test runs, test suites, milestones, projects, and individual test cases at-a-glance
- **Centralized repository:** Assemble your manual, automated, and non-functional testing in a single source to more easily aggregate testing efforts and ensure everyone has real-time visibility into defects and test failures
- **Defect and requirement integrations:** Integrate with platforms like Jira to link all test cases with their associated requirements and defects for iron-clad traceability
- **Test case review and approvals*:** Enable the built-in test case review and approval workflow to ensure that no test cases get executed or changed without approval
- **Test case versioning*:** See a full version history of all test cases and easily compare changes over time
- **Audit logging*:** Keep a detailed log of every action taken by every TestRail user, eliminating the questions of when and who changed what

* = Available as part of the TestRail Enterprise platform offering

Managing Distributed Teams



Common challenges specific to regulated industries

In addition to an increased need for visibility and collaboration, distributed teams working in regulated industries face unique challenges in upholding security and compliance across time zones and borders.



Diverse compliance standards for international teams

In addition to an increased need for visibility and collaboration, distributed teams working in regulated industries face unique challenges in upholding security and compliance across time zones and borders.



Cloud configurations for disaster recovery

It may become necessary to establish specific cloud configurations for disaster recovery and replication and ensure multiple availability zone coverage for application environments.



Data access restrictions for confidential information

It may be required to implement stringent data access restrictions for team members not domestic to a specific country, particularly concerning confidential data.

Managing Distributed Teams



Challenges

- ⚠️ My team works across multiple time zones, making it difficult to get fast answers and clarity on testing status.
- ⚠️ It's hard for me to track who is creating and editing test cases, which is harming our ability to maintain compliance.

Solutions

- ✅ Consider utilizing a testing center of excellence—a centralized repository for all of your tests and results. A single source of truth that is automatically updated with every test execution will eliminate confusion and make it easy to track progress.
- ✅ Consider a test management solution such as TestRail Enterprise with capabilities like test case review and approval, test case versioning, and audit logging. This will ensure that all actions in your platform are tracked and no test cases are executed without approval.

If a person gets sick prior to a deadline, I am able to ask another QA person to execute those test cases in TestRail, which keeps the project on schedule. If we need testers to quickly swarm on a testing effort for an urgent customer request, TestRail supports that.

*Kelli Jordan,
Director of Quality Assurance*





SECTION 07

Self-Assessments for Continuous Improvement and Audit-Readiness

Self-Assessments for Continuous Improvement

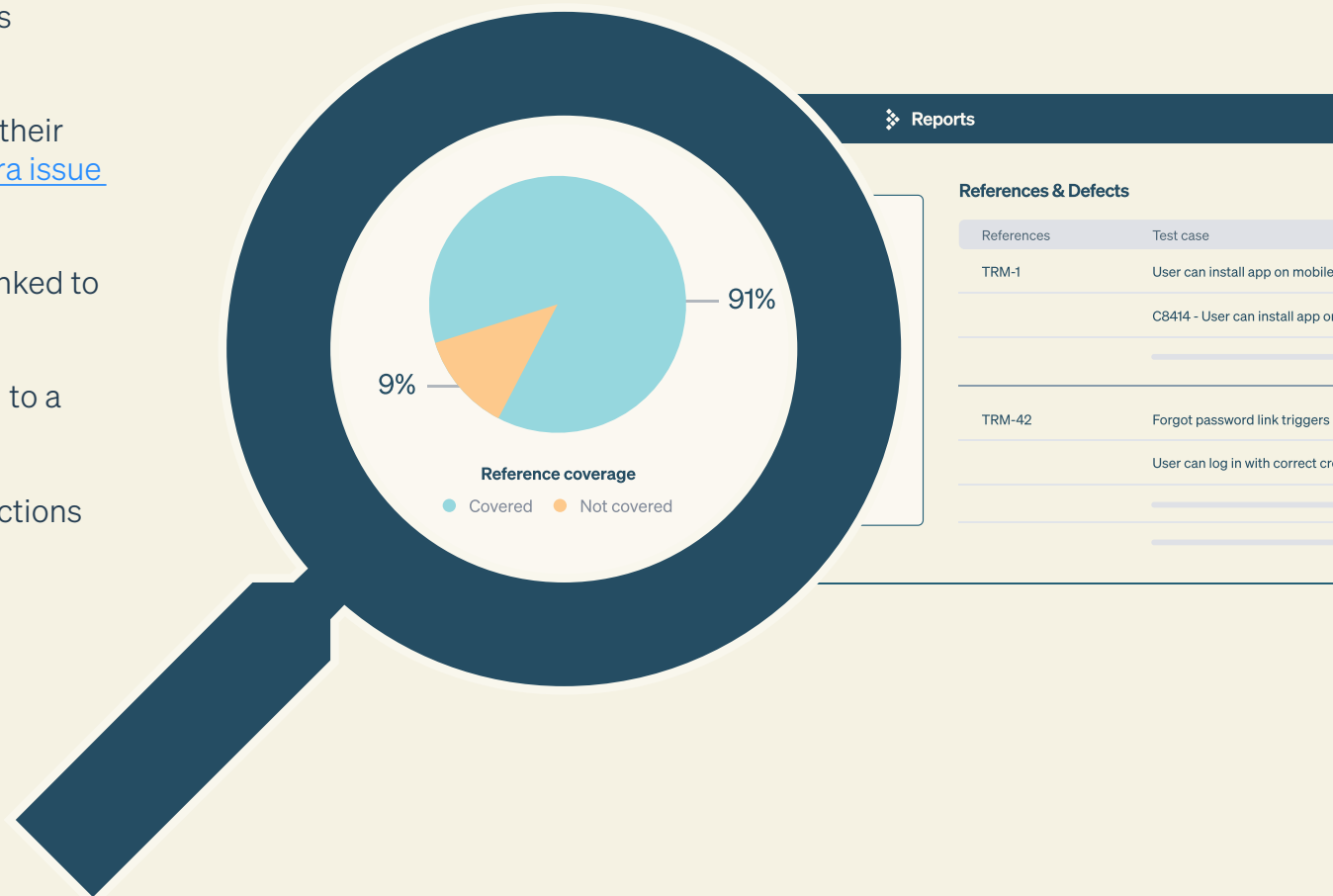


No matter how great your QA team is performing, you can never rest on your laurels. You should lead your team with a passion for continuous improvement and always assess your processes and practices with the aim of making them better.

One of the easiest and most fruitful ways to foster a culture of continuous improvement is to get into the habit of regular self-assessment exercises. Self-assessments can help identify areas for improvement, such as process gaps, and they also ensure that you are audit-ready at any moment.

Critical exercises to perform in your self-assessments include validating that:

- You have [full traceability](#) via linkage of all tests to their corresponding requirements artifact, such as a [Jira issue or defect](#)
- Tests that have been planned and executed are linked to a given release event or designation
- Failed tests for a given release or sprint are linked to a defect artifact, such as a Jira defect
- Your disaster recovery plan is up-to-date and functions as intended



Self-Assessments for Continuous Improvement



Self-assessments

What does a self-assessment look like in practice? Here are some example scenarios:



Self-assessment example #1

Pretend like you're an auditor. Pick a random Jira story and see how easily you can identify the exact amount of testing, where it was tested, who did the testing, and the test results. If the test failed, is there a linked defect? Was the bug fixed and was the fix validated with additional testing before deployment?



Self-assessment example #2

Pull a defect report drill-down. Are there still tests associated with closed defects? Do you have failed tests with no defect linked?



Self-assessment example #3

Check the accessibility of testing and results for each release. For example, let's assume you pushed a release last week. Can you see every test that was run for that release, the status, and when it was executed—including disaster recovery testing and performance testing? Can you see both manual and automated test results? Are they in one spot? Are they in different places, like multiple people's drives or computers? Is everything centralized and easy to find?

Self-Assessments for Continuous Improvement



If you're constantly overwhelmed with just managing the chaos of your testing activities, you'll never have time to assess and improve. Leveraging a centralized test management platform (such as TestRail) for both manual and automated testing with built-in defect integrations and robust reporting can make regular self-assessments a task you look forward to rather than a time-consuming chore.

My recent conversation with an auditor started with him selecting a Jira story ticket and asking me to walk him through our testing process, presenting evidence [from TestRail] along the way... The evidence proved that no new code goes out to production without being subjected to testing in multiple environments and on multiple browser types. The auditor was very happy.

*Kelli Jordan,
Director of Quality Assurance*





Orchestrate Testing. Elevate Quality.

Transform your testing into a centralized, scalable process with the Quality OS by TestRail. It's time to transform your dreams of faster, frictionless releases into reality.

Try a 30-day free trial or register for a demo today:

[Try TestRail](#)

[Get a Demo](#)

Want to learn more? Improve your testing efficiency and earn free certifications at the TestRail Academy. Visit today and explore courses in test automation, agile testing, testing in regulated industries, QA analytics, and more.

[TestRail Academy](#)

