EBOOK

Planning and Executing Integrated End-to-End Tests

5 Strategies to help teams effectively test user experiences

End-to-end testing (E2E) isn't just another test to automate. Instead, it's a method to test your product in ways that approximate real-world use cases from the perspective of your users and customers. In short, E2E testing isn't just a testing framework—it's a customer experience framework.

Building an E2E testing strategy can enrich your organization in many ways. Not only can it improve existing customer experiences, but it helps your organization adapt to new use cases as your products and applications evolve. A thoughtful E2E testing strategy will also loop in your team and stakeholders, allowing you to connect testing results to an improved CX for your entire organization.

CHAPTERS

Why is E2E Testing critical to CX?

First, customer experience is critical to your organization. End-to-end testing is one of just a few ways quality teams can reliably - and quantifiably - improve CX. Adding features that customers request will improve their satisfaction in the short-term, but increasing the quality of these features will make your customers loyal in the long run.

The fact of the matter is that customers will vote with their feet if quality isn't what it needs to be:

86% of purchasers are willing to pay more for a better customer experience.

PWC, "Experience is everything: Here's how to get it right," 2018

65% of consumers say a positive experience is more influential than advertising.

PWC, "Experience is everything: Here's how to get it right," 2018

81% of companies compete on customer experience alone.

Gartner, "Key Findings From the Gartner Customer Experience Survey," March 2018

8% increase in revenue when an organization focuses on customer experience.

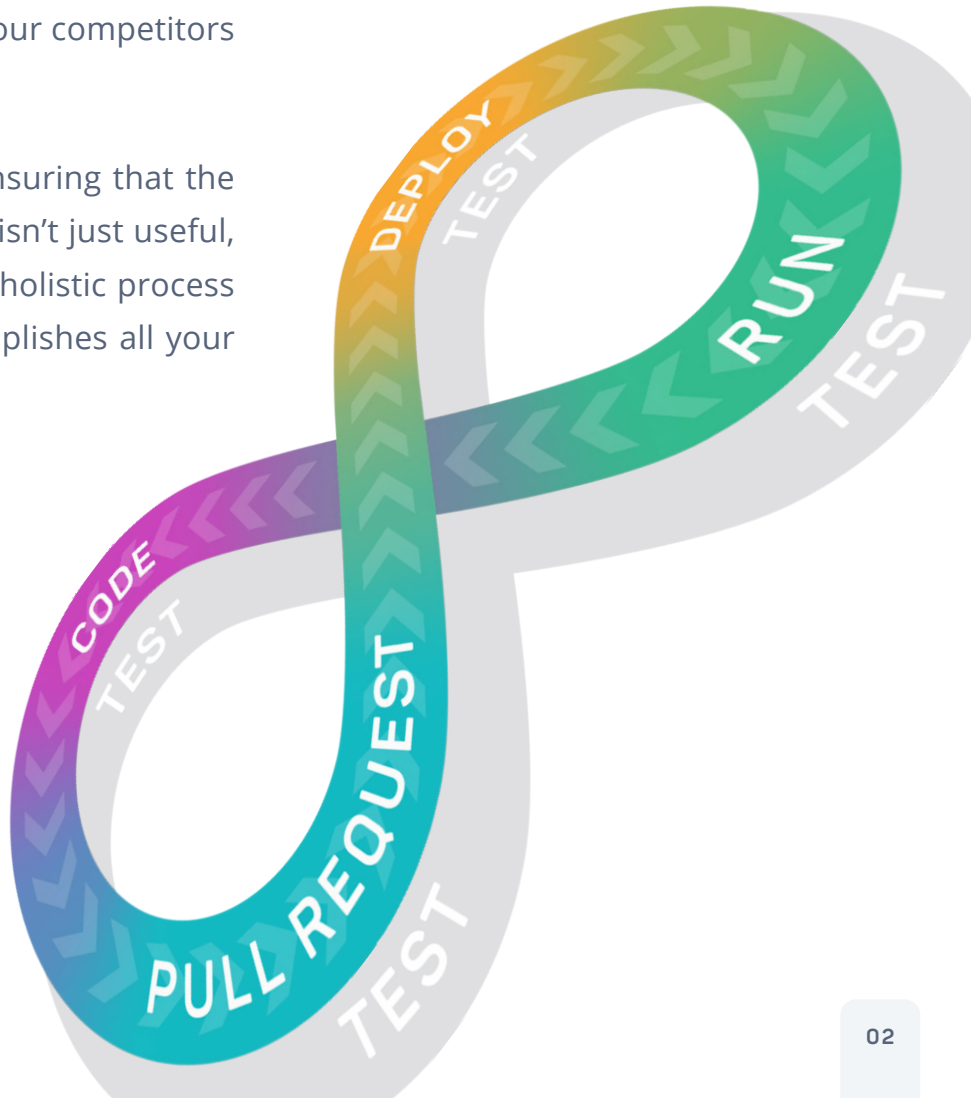
Bain & Company, "The Five Disciplines of Customer Experience Leaders," 2015

51% of employees are unhappy at work because of the software they use.

G2, "State of Software Happiness Report," 2019

E2E testing is so closely linked with customer experience because of the prominence of DevOps and CI/CD—philosophies that emphasize the importance of "ship fast and break things." As a result, many DevOps teams are forced to rely on shipping fixes or rolling back releases to negate defective software, both of which risk customers discovering bugs. Any new release has the potential to dramatically change the way that people use your application, which might adversely affect the usability, accessibility, or even the regulatory compliance of your application. Even if you don't want to use CI/CD due to this risk, there are few alternatives—all your competitors are moving at the same breakneck speed.

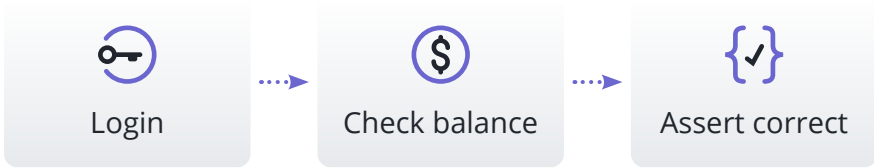
E2E testing helps mitigate the possibility of bugs in production, ensuring that the customer journey remains present and ensures that your product isn't just useful, but also functional and enjoyable. This is because E2E testing is a holistic process that tests the entire flow of an application, ensuring that it accomplishes all your end users' needs.



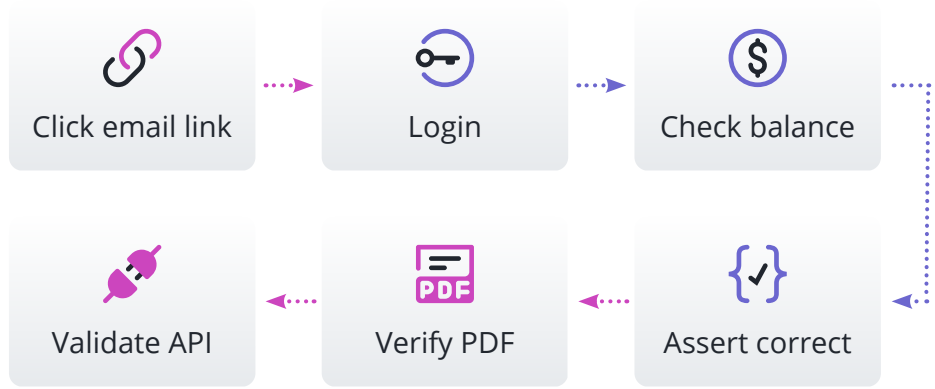
Why Aren't We Implementing More E2E Testing, Then?

Clearly, E2E tests are a powerful tool to help us achieve the best possible customer experience. So why haven't we implemented more E2E tests in our applications? In short, applications are becoming more complex, making our tests more complex. Not to mention that spinning up variations of the test in every browser, frequently changing UIs, and difficulty debugging make it difficult to see a return on those end-to-end testing efforts. If that wasn't enough, it's easy to over-test your application without proper buy-in from your team.

Take a look at a traditional testing workflow. An E2E test runs in the browser where your users also interface with your application. The test might cover the following scenario:



But, an E2E testing workflow should cover all the touchpoints surrounding the workflow.



By expanding test coverage beyond simple functionality, you can test many use cases: new scenarios, new device and browser combinations, and complex multi-device customer journeys. All of this helps catch bugs preproduction by testing under conditions that more closely simulate the real world.

E2E testing is not a panacea—it's possible to do it incorrectly. But with proper planning and alignment with your team, the benefits of E2E testing significantly outweigh the costs.

5 Strategies To Help You Plan and Execute an E2E Testing Strategy

1 • APPROACHING AN END-TO-END TESTING STRATEGY

Due to the many factors involved, planning an E2E testing strategy may take longer than implementation. But there are significant benefits to a carefully planned strategy. Just as it's difficult to fix a bug that's found in production, it's also difficult to alter a testing strategy mid-flight. Therefore it's very important to build a technical and organizational process for E2E tests.

E2E testing is better when it's automated, so your first step is to understand the capabilities and limitations of your automation tools. Your testers will inevitably experience pain points and bottlenecks even in a well-optimized QA department, and it's important to eliminate these obstacles before evolving an E2E strategy—otherwise it will be hobbled by the same constraints.

Solving problems with your current automation solutions and existing test strategy means bringing in your stakeholders. In general, your stakeholders are people who are accountable for the quality of software products in your organization, which may include you, your CTO, your developers, and more. Each of these individuals need to be made aware of the effort to create an E2E testing strategy.

Stakeholders also need goals and benchmarks to work against. How will you define and measure success? You'll need metrics to understand whether you're moving in the right direction and that each aspect of your strategy is being implemented correctly, or if you need to pivot. Once you can establish goals across your entire team, you can document your answers to those questions. You'll need these in order to repeat your process with other projects and team members.

GOALS

- What are the objectives of our automation efforts?
- How do we define and measure success?
- How can we best align with organizational and company objectives?

ROLES

- Who is responsible for all of the aspects of quality within your team?
- Who defines test cases?
- Who updates out-of-date tests?
- Who triages failures?
- Who determines when tests should run?

PROCESS

- How will you communicate?
- How will you manage the tests, plans, etc.?
- How will you monitor progress, and iterate to improve over time?
- How will you roll out this plan?

Finally, you need to choose the right pre-test framework, which will connect your E2E testing strategy to real-world execution. Choose a framework that's robust enough for your use case, as well as one that fits the skill set of your entire team. For best results, don't go all-in at once. Instead, pilot E2E testing in conjunction with a single new feature release or a smaller team, which will allow you to adapt your strategy as needed, perfect implementation, and have realistic expectations for widespread adoption.

Keys to Approaching Your Test Automation Strategy

- Take the time to set your strategy first. You'll be more successful in the long run.
- Looking to change your process? Try it with one new feature release, or a small team.
- Getting pushback? Point to quality metrics like bugs in production - and tie these to business goals - to make a case for change.

2 • BE THOUGHTFUL ABOUT TEST COVERAGE

Any investment in testing or testing frameworks will require time, effort, and money. How can you budget and invest in the right areas?

The good news is that your team already has a wide range of data from the user's perspective. You know what they experience, you know the flows they use most frequently, and you know how they're interacting with your application. With this diverse information set, it's very easy to understand where your users might be experiencing frustration. Your analytics tools and customer data platforms will be able to tell you where your users get stuck and where they abandon.

Here, you can also see how your users are entering your application. Do they use the browser on their mobile device, the browser in their desktop, or a browser that's emulated from inside their email client? It's important to compare this information to your acceptance criteria to avoid scope creep and maintain a close connection between testing and the customer experience.

There's a strong temptation to test too much, but some tests are best covered at lower levels. Boundary, input validation, performance tests and others are best performed at the unit or integration level. It's not necessary to achieve 100% E2E test coverage, and in fact this can be undesirable. The more you add to your test cases, the more brittle they become. Segment your tests instead and mitigate the potential for them to break. Your focus should always be on impact and effectiveness over sheer volume.

In addition, bear in mind that E2E testing is expensive. Like any other test, the value of E2E testing is dependent on being applied to the right cases, and overuse will only waste your resources and strain your ability to test in other areas. Utilise your user data to find the most common paths that your users take and test those using E2E. Data-driven testing that's reflective of real-world use is essential to delivering quality in DevOps.

The corollary to this is that user journeys change over time. While you shouldn't test every possible user journey, you should frequently review to make sure that tests evolve in conjunction with your users' behavior. You should also make sure that you're aligning with business objectives. If you're a growth-stage startup, you may be more focused on testing the user onboarding flows. If you're an established enterprise, you may be more focused on testing for regulatory compliance or information security. As owners of quality for the entire enterprise, QA teams need to be able to continuously update testing strategy to support your product and business goals.

Keys to Being Thoughtful About Your Approach

- App usage data should drive decisions around what cases require E2E tests.
- Re-evaluate your user journeys on a regular basis.
- Understand how and what users experience in your application.
- Resist the urge to test lower-level logic.

3 • TAKE A WHOLE TEAM APPROACH

One way to ensure the success of E2E testing is to make sure that everyone is responsible for quality.

A good example is the shift-left approach that adds software developers to the QA process. With the right tools and workflows, developers can automatically run tests as soon as they hit “save.” These might be the same tests that you’re running against your regression suite, which can help your team save a considerable amount of time. If a new feature passes tests during development, it will also pass in a pull request. But if a defect is detected, it’s much easier to address and resolve the bug earlier in the development cycle when there’s less of a domino effect that can impact other aspects of the product. Ultimately, this means that you catch a lot more bugs a lot earlier in the development lifecycle, where they’re much easier to solve.

Although developers will be performing these tests, QA can still own them by acting as a quality coach. You can give developers the ability to create tests or change them alongside their code with QA providing feedback and best practices. By sharing their expertise, QA can support developers through tricky areas within the testing platform and share tips usually relegated to power users. This makes developers even more effective at testing, creating a force multiplier for the entire QA department and integrating quality into the entire SDLC.

Ultimately, this improves the quality of the organization at large. You aren't just creating more tests—you're building out a better testing process. This success redounds to the benefit of your entire organization. Developers share in the success because this means that they spend more time on features and less time fixing bugs, giving them more time to build new features and products. Your support team shares in the success when there are fewer overall bugs and tickets, letting them spend more time per issue and improving customer satisfaction. Even the sales team can enjoy the benefits because they're less likely to encounter bugs during demonstrations—and successful demos can increase revenue for the whole company. As an organization, everyone benefits from a renewed focus on innovation - letting you stay ahead of the competition without sacrificing quality.

Keys to Taking a Whole-Team Approach

- Include quality metrics in team goals.
- Lower the barrier to automated test creation by providing resources to your development team on testing best practices.
- Integrate testing into the team's workflow.

4 • HOW TO UNDERSTAND QUALITY SIGNALS

Your application is always changing. Separate from developer updates, the application absorbs more information based on user input, which should affect the way your team conducts testing and validation.

New users may behave differently—if you see a sudden influx of more technical users, for example, these users will have different flows. Test the most high-impact areas to get the biggest return and ensure that your quality strategy evolves with your company's growth.

Testers should always use data to drive their decisions. Keep regular tabs on product and usage data. As new features are released, many customers will undertake new journeys—but new features can also break existing tests. It's important to get ahead of this by validating your tests ahead of any major product updates.

As always, reduce testing in areas that no longer have relevance. More testing does not necessarily equal better testing. Strive to reduce complexity and brittleness, which will reduce noise that can disguise adverse changes in your application or testing suite.

Lastly, look for qualitative signals outside traditional data sources. Watch for an increase in support tickets, especially in areas of your product that you aren't testing. Talk to your sales engineers and look for areas that they try to avoid during demos, which can help you understand and avoid red flags. If you've created a connected and collaborative testing organization that incorporates people in addition to processes, you'll be able to sound the alarm and get ahead of these issues in a timely manner.

Keys to Understanding Quality Signals

- Find the right cadence - based on your release cycle - to re-evaluate E2E tests.
- Keep a close eye on test failures and bugs in production.
- Communicate problem areas to the broader team to determine if additional tests are required.
- Enable Automated End-to-End Testing with intelligent, low-code Technology

5 • ENABLE AUTOMATED END-TO-END TESTING WITH INTELLIGENT, LOW-CODE TECHNOLOGY

End-to-end testing is one of the most important new quality engineering processes. It can radically improve the customer experience, which, all things being equal, can help you rise to the top of your field and outshine your competitors. Strategizing and implementing E2E testing can be a serious drain on resources, but with low-code test automation solutions like mabl, you'll be able to take a more efficient and effective approach—implementing E2E and capturing a loyal customer base before your competitors can think about doing the same.

Customers say that mabl acts as a force multiplier for the QA team, enabling quality leaders to strategize and implement an automated E2E testing practice. Not only will you be able to create tests more rapidly, you'll also be more confident in your approach. mabl's low-code frontend helps you expand testing throughout the organization, rapidly expanding testing to match the cadence of your developers. What's more, mabl itself can adapt to the changing web application, flagging visual changes and automatically scoping them into new tests.

While end-to-end testing may have seemed like an untenable task in the past, with the help of mabl's low-code test automation you can use real-user behavior to help drive your testing strategy and build high quality applications. Get started today with your free trial of mabl.

ABOUT MABL

Mabl is the leading intelligent test automation platform built for CI/CD. It's the only SaaS solution that tightly integrates automated end-to-end testing into the entire development lifecycle. Mabl's native auto-heal capability evolves tests as the application UI evolves with development; and the comprehensive test results help users quickly and easily resolve bugs before they reach production.

Creating, executing, and maintaining reliable tests has never been easier. Mabl enables software teams to increase test coverage, speed up development and improve application quality - empowering everyone on the team with the ability to ensure the quality of the applications at every stage.

Learn more at mabl.com